UCOT API

by UCOT

Package
ucot

UCOT package, contains general classes for all modules.

ucot
Class Messages

```
java.lang.Object
   |
   +-ucot.Messages
```

public class **Messages**
extends java.lang.Object

This class contains interface to the localised message strings.

Strings are readed using ResourceBundle that uses ucot/message.properties file (or its localized version).
Author:
pajumasu

Fields

BUNDLE_NAME

`private static final java.lang.String BUNDLE_NAME`

Constant value: **ucot.messages**

RESOURCE_BUNDLE

`private static final java.util.ResourceBundle RESOURCE_BUNDLE`

Constructors

Messages

`public Messages()`

Methods

getString

`public static java.lang.String getString(java.lang.String key)`

Returns localized string for the given key. If no string is found then !key! is returned (where key is the key-string that was given as parameter)

getString("FILE_NOT_FOUND"); would return some string definded for example in properties file or string "!FILE_NOT_FOUND!" if no matching key is located.

Parameters:

(continued on next page)

(continued from last page)

`key` - Key for the string required.

Returns:
localized string for the given key.

formatMessage

```
public static java.lang.String formatMessage(java.lang.String messageString,
        java.lang.Object[] args)
```

Method formats messages with parameters to user readable format. MessageString is normal string that marks the places for the dynamic information with {}-marks and number in between them.

For example: formatMessage("File {0} not found.", new File("file.txt")); would return string: "File file.txt not found.".

To use with getString method you can do like this: System.out.println(formatMessage(getString("FILE_NOT_FOUND"), new File("file.txt")));. If there is property file that contains line: FILE_NOT_FOUND=File {0} not found. then it would return the same string: "File file.txt not found.".

Parameters:
`messageString` - Message format string.
`args` - Arguments for the string.

Returns:
Returns string that is builded using the arguments.

getFormatedMessage

```
public static java.lang.String getFormatedMessage(java.lang.String key,
        java.lang.Object[] args)
```

Gets string localized string and format the arguments using it.

Returns:
The formated and localized string.

See Also:
`formatMessage(String, Object[])`
`getString(String)`

ucot
Class ModuleProperties

```
java.lang.Object
  |
  +-ucot.ModuleProperties
```

All Implemented Interfaces:
ModulePropertyInterface

Direct Known Subclasses:
DummyUI, DummyParser, SimpleParser, GXLOutput, DummyInput, ProcessMLInputAdapter, SimpleInputAdapter, AbbottsHeuristic, DummyHeuristic, Core

---

public abstract class **ModuleProperties**
extends java.lang.Object
implements ModulePropertyInterface

This class implements abstracts methods for handling, saving and loading of module's properties. It is recommended that all UCOT modules extend from this class because all the interfaces require these methods to be implemented anyway and these methods implemented here are sufficient for practically all other modules besides the core.
Author:
tujupien

Fields

propertiesURL

`protected java.net.URL **propertiesURL**`

---

properties

`protected java.util.Properties **properties**`

Constructors

ModuleProperties

`public **ModuleProperties**()`

Methods

getPropertiesURL

`private java.net.URL **getPropertiesURL**()`

(continued from last page)

Method for creating the URL from the properties file, which is the same as the class name with an .xml extension.

Returns:
URL to the properties file.

getProperties

```
public java.util.Properties getProperties()
```

See Also:
ModulePropertyInterface.getProperties()

setProperties

```
public void setProperties(java.util.Properties properties)
```

See Also:
ModulePropertyInterface.setProperties(Properties)

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

See Also:
ModulePropertyInterface.applyProperties()

saveProperties

```
public void saveProperties()
  throws java.io.IOException
```

See Also:
ModulePropertyInterface.saveProperties()

loadProperties

```
public void loadProperties()
  throws java.io.IOException
```

See Also:
ModulePropertyInterface.loadProperties()

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

(continued from last page)

See Also:

`ModulePropertyInterface.loadDefaultProperties()`

ucot
Interface ModulePropertyInterface

All Subinterfaces:
UIInterface, ParserInterface, OutputInterface, InputInterface, HeuristicInterface, ControlInterface

All Known Implementing Classes:
GraphicalUI, ModuleProperties

---

public interface **ModulePropertyInterface**
extends

This interface defines required methods for another interfaces to support properties in a way that the module implementing this interface would be compatible with UCOT core.
Author:
tujupien

Methods

getProperties

```
public java.util.Properties getProperties()
```

Returns module's properties.

Returns:
Module's properties.

---

setProperties

```
public void setProperties(java.util.Properties properties)
```

Sets options for the adapter.

Notice that this does not need to be an perfect set of properties for this module because these properties should be merged to the current properties. So it is possible to change only one property value by giving a new property object with the new value for the given key.

Parameters:
`properties` - Properties for the adapter.

---

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

Applies current properties for the module.

(continued from last page)

Throws:

`BadPropertyValueException` - In this case exception is thrown only if either the given parser or heuristic does not exist.

saveProperties

```
public void saveProperties()
  throws java.io.IOException
```

Saves current properties to the properties XML file.

Throws:

`IOException` - Exception is thrown if something went wrong.

loadProperties

```
public void loadProperties()
  throws java.io.IOException
```

Loads settings from the current properties XML file.

Throws:

`IOException` - Exception is thrown if something went wrong.

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

Method which returns the factory default properties for the module.

Returns:

Default properties.

Package
ucot.core

ControlInterface and Core class, these control the program flow.

ucot.core
Class AnalyzeModelLogger

```
java.lang.Object
   |
   +-ucot.core.AnalyzeModelLogger
```

All Implemented Interfaces:
java.util.Observer

public class **AnalyzeModelLogger**
extends java.lang.Object
implements java.util.Observer

Class for logging changes in current analyze model. The changes are observed and the log is updated every time the analyze model notifies all observers about its changes.
Author:
pajumasu

Fields

logger

`private java.util.logging.Logger logger`

updations

`private java.util.List updations`

Constructors

AnalyzeModelLogger

`public AnalyzeModelLogger()`

Methods

update

```
public void update(java.util.Observable observable,
         java.lang.Object arg)
```

This method updates the logger when something changes in the analyze model.

Parameters:
`observable` - Observable object, in this case the analyze model.

(continued from last page)

`arg` - Arguments for this method. If this is not an instance of Updation, then this method will do nothing.

ucot.core
Interface ControlInterface

All Superinterfaces:
  ModulePropertyInterface

All Known Implementing Classes:
  Core

public interface **ControlInterface**
extends ModulePropertyInterface

This interface controls the basics of the UCOT-program (starting, parsing, shutting down etc). UCOT core implements this interface and it is designed in a way that the user interface using the core can control the execution of the program effectively.
Author:
  vevijopi, tujupien

Fields

PROPERTY_CURRENT_PARSER

```
public static final java.lang.String PROPERTY_CURRENT_PARSER
```

Constant value: **`CURRENT_PARSER`**

PROPERTY_CURRENT_HEURISTIC

```
public static final java.lang.String PROPERTY_CURRENT_HEURISTIC
```

Constant value: **`CURRENT_HEURISTIC`**

Methods

shutdown

```
public void shutdown()
```

Shuts down the program. The shutdown routine triggers all possible autosave actions and after that the core gets rid of all its modules and prepares itself for getting automatically junkbusted.

getRootDir

```
public java.lang.String getRootDir()
```

Returns the UCOT root directory. That is either the system directory where the class files are stored under a hierarchical directory structure based on the package definitions or the system directory where the UCOT JAR distribution package is stored. The result depends solely on the fact which distribution is currently used.

(continued from last page)

Returns:
The path to the UCOT root directory as a String.

clearAnalyzeModel

```
public void clearAnalyzeModel()
```

Clears the whole current analyze model. Basically this is similiar to creating a whole new empty analyze model. Notice that the old analyze model is not saved anywhere and the restoration of the old model is impossible without manually backing up the old analyze model before using this function.

loadUseCases

```
public void loadUseCases(java.net.URL url)
```

Loads use cases from the given file.

Parameters:
`url` - URL to the use case file.

reloadUseCases

```
public void reloadUseCases(java.net.URL url)
```

Reloads use cases from file. First core should remove all use cases that are loaded from the given url, and then it should (re)read the use cases from the given file.

Parameters:
`url` - URL of the file to be reloaded.

loadAnalyzeModel

```
public void loadAnalyzeModel(java.net.URL url)
  throws java.io.IOException
```

Loads serialized analyze model from the given file.

Parameters:
`url` - URL to the file containing a serialized analyze model.

Throws:
`IOException` - If something goes wrong while loading the analyze model from given URL.

saveAnalyzeModel

```
public void saveAnalyzeModel(java.net.URL url)
```

(continued from last page)

Saves the current analyze model to a java serialization file which location is pointed by the given URL.

Parameters:
`url` - URL to the file where the current analyze model should be serialized and saved.

getUseCaseCollection

```
public UseCaseCollection getUseCaseCollection()
```

Returns all currently loaded use cases in a UseCaseCollection.

Returns:
UseCaseColletion containing all currently loaded use cases.

addToAnalyzeModel

```
public void addToAnalyzeModel(java.util.Vector useCases)
```

Requests core to parse given use cases, perform heuristic on them and add then them to current analyze model.

Default parser and heuristic are used for this operation.

Parameters:
`useCases` - Use cases to work magic on.

addToAnalyzeModel

```
public void addToAnalyzeModel(java.util.Vector useCases,
        ParserInterface parser,
        HeuristicInterface heuristic)
```

Requests core to parse use cases, perform heuristic on them and add them to current analyze model.

If either the given parser or heuristic is unknown to the UCOT core, then nothing is done.

Parameters:
`useCases` - Use cases to work magic on.
`parser` - Parser to use.
`heuristic` - Heuristic to use.

getAnalyzeModel

```
public AnalyzeModel getAnalyzeModel()
```

Method for getting a pointer to the current analyze model being handled in the core.

Notice that all editing to the analyze model should be done through the AnalyzeModelEditor which can be easily acquired with the `getEditor()` method from the `AnalyzeModel` itself.

(continued from last page)

Returns:
Current analyze model.

getParsers

`public java.util.Vector **getParsers**()`

Returns a vector containing all the parsers currently available to the UCOT core.

Returns:
Vector containing all known parser adapters as `ParserInterfaces`.

getHeuristics

`public java.util.Vector **getHeuristics**()`

Returns a vector containing all the heuristics currently available to the UCOT core.

Returns:
Vector containing all known heuristic adapters as `HeuristicInterfaces`.

getOutputs

`public java.util.Vector **getOutputs**()`

Returns a vector containing all the output adapters currently available to the UCOT core.

Returns:
Vector containing all known output adapters as `OutputInterfaces`.

getInputs

`public InputCollection **getInputs**()`

Returns all the input adapters currently available to UCOT core.

Returns:
InputCollection which contains all known input adapters as `InputInterface`.

setCurrentParser

`public void **setCurrentParser**(ParserInterface parser)`

Sets the default parser to be used in parsing progresses.

Parameters:
`parser` - Parser to be used by default.

(continued from last page)

setCurrentHeuristic

```
public void setCurrentHeuristic(HeuristicInterface heuristic)
```

Sets the default heuristic to used in analyzation progresses.

Parameters:
`heuristic` - Default heuristic to be used by default.

getCurrentParser

```
public ParserInterface getCurrentParser()
```

Returns the current default parser.

Returns:
Default parser as a `ParserInterface`.

getCurrentHeuristic

```
public HeuristicInterface getCurrentHeuristic()
```

Returns the current heuristic.

Returns:
Current heuristic as a `HeuristicInterface`.

output

```
public void output(java.net.URL url,
                   OutputInterface output,
                   AnalyzeModel analyzeModel)
   throws java.lang.Exception
```

Exports given analyze model to given URL using given output adapter.

Parameters:
`url` - Destination URL.
`output` - `OutputInterface` of the output adapter to be used.
`analyzeModel` - Analyze model to export.

Throws:
`If` - something goes wrong with the output operation, then an exception is thrown.

ucot.core
Class Core

```
java.lang.Object
  |
  +-ucot.ModuleProperties
      |
      +-ucot.core.Core
```

All Implemented Interfaces:
ControlInterface, ModulePropertyInterface

public class **Core**
extends ModuleProperties
implements ModulePropertyInterface, ControlInterface

This is the implementation of the core of the UCOT program, the controller unit. The core administrates the whole program by handling the analyze model and keeping track and controlling the usage of all possible modules loaded to the program.

The core component offers a `ControlInterface` for the other components to request different kinds of operations from the core component.
Author:
UCOT

Fields

FILEFORMAT_NOT_SUPPORTED_ERROR

`private static final java.lang.String` **`FILEFORMAT_NOT_SUPPORTED_ERROR`**

CANNOT_LOAD_FILES_ERROR

`private static final java.lang.String` **`CANNOT_LOAD_FILES_ERROR`**

CANNOT_ADD_SELECTED_USE_CASE_ERROR

`private static final java.lang.String` **`CANNOT_ADD_SELECTED_USE_CASE_ERROR`**

FILE_NOT_SAVED_ERROR

`private static final java.lang.String` **`FILE_NOT_SAVED_ERROR`**

FILE_NOT_WRITABLE_ERROR

`private static final java.lang.String` **`FILE_NOT_WRITABLE_ERROR`**

(continued from last page)

PLUGIN_NOT_COMPATIBLE_ERROR

`private static final java.lang.String` **`PLUGIN_NOT_COMPATIBLE_ERROR`**

FILE_ALREADY_LOADED

`private static final java.lang.String` **`FILE_ALREADY_LOADED`**

rootDir

`private java.lang.String` **`rootDir`**

inputs

`private ucot.input.InputCollection` **`inputs`**

useCases

`private ucot.input.UseCaseCollection` **`useCases`**

analyzeModel

`private ucot.model.AnalyzeModel` **`analyzeModel`**

heuristics

`private java.util.Vector` **`heuristics`**

parsers

`private java.util.Vector` **`parsers`**

outputs

`private java.util.Vector` **`outputs`**

(continued from last page)

userInterface

`private ucot.ui.UIInterface` **`userInterface`**

currentParser

`private ucot.parser.ParserInterface` **`currentParser`**

currentHeuristic

`private ucot.heuristic.HeuristicInterface` **`currentHeuristic`**

progressBar

`private ucot.ui.ProgressBarInterface` **`progressBar`**

logger

`private java.util.logging.Logger` **`logger`**

RUNNING

`private static boolean` **`RUNNING`**

OUTPUT_JOB_STACK

`private static java.util.Stack` **`OUTPUT_JOB_STACK`**

EXECUTING_OUTPUT_THREAD

`private static java.lang.Thread` **`EXECUTING_OUTPUT_THREAD`**

parsingThreads

`protected static java.util.Vector` **`parsingThreads`**

Constructors

(continued from last page)

Core

`public Core()`

Default constructor for UCOT core component. This initializes the core component by loading all available modules.

Methods

getFileNotFoundMessage

`public static java.lang.String getFileNotFoundMessage(java.net.URL file)`

Returns the localized 'file not found' message.

Parameters:
`file` - URL to the file that is not found (to be part of the message).

Returns:
Localized text as a String saying File *given file* not found!.

---

main

`public static void main(java.lang.String[] args)`

Main method that starts the UCOT core.

Parameters:
`args` - Command line arguments.

---

setRootDir

`private void setRootDir()`

Parses the current location of the Core.class file and updates the class variable to match it.

---

getRootDir

`public java.lang.String getRootDir()`

---

shutdown

`public void shutdown()`

See Also:
`ControlInterface.shutdown()`

loadUseCases

```
public void loadUseCases(java.net.URL url)
```

See Also:
`ControlInterface.loadUseCases(URL)`

loadAnalyzeModel

```
public void loadAnalyzeModel(java.net.URL url)
  throws java.io.IOException
```

See Also:
`ControlInterface.loadAnalyzeModel(URL)`

getUseCaseCollection

```
public UseCaseCollection getUseCaseCollection()
```

See Also:
`ControlInterface.getUseCaseCollection()`

clearAnalyzeModel

```
public void clearAnalyzeModel()
```

See Also:
`ControlInterface.clearAnalyzeModel()`

runParserAndHeuristic

```
private void runParserAndHeuristic(java.lang.Runnable runnable,
         java.util.Vector useCases,
         ParserInterface parser,
         HeuristicInterface heuristic)
```

This method runs the parser and heuristic in a single separate thread. The thread waits first for other parsing and analyzation threads invoked earlier to finish and then starts the real execution.

Parameters:
`runnable` - Runnable where this thread is running.
`useCases` - Use cases to be parsed.
`parser` - Parser to use.
`heuristic` - Heuristic to use.

(continued from last page)

addToAnalyzeModel

```
public void addToAnalyzeModel(java.util.Vector useCases,
        ParserInterface parser,
        HeuristicInterface heuristic)
```

addToAnalyzeModel

```
public void addToAnalyzeModel(java.util.Vector useCases)
```

getAnalyzeModel

```
public AnalyzeModel getAnalyzeModel()
```

See Also:
`ControlInterface.getAnalyzeModel()`

getParsers

```
public java.util.Vector getParsers()
```

See Also:
`ControlInterface.getParsers()`

getHeuristics

```
public java.util.Vector getHeuristics()
```

See Also:
`ControlInterface.getHeuristics()`

getOutputs

```
public java.util.Vector getOutputs()
```

See Also:
`ControlInterface.getOutputs()`

getInputs

```
public InputCollection getInputs()
```

See Also:
`ControlInterface.getInputs()`

setCurrentParser

```
public void setCurrentParser(ParserInterface parser)
```

See Also:
`ControlInterface.setCurrentParser(ParserInterface)`

setCurrentHeuristic

```
public void setCurrentHeuristic(HeuristicInterface heuristic)
```

See Also:
`ControlInterface.setCurrentHeuristic(HeuristicInterface)`

getCurrentParser

```
public ParserInterface getCurrentParser()
```

See Also:
`ControlInterface.getCurrentParser()`

getCurrentHeuristic

```
public HeuristicInterface getCurrentHeuristic()
```

See Also:
`ControlInterface.getCurrentHeuristic()`

finalize

```
protected void finalize()
  throws java.lang.Throwable
```

See Also:
`Object.finalize()`

executeOutput

```
private void executeOutput()
```

This method executes the output operations in a separate thread. This thread keeps on running once started and ends execution when core's finalize method has been called.

(continued from last page)

output

```
public void output(java.net.URL url,
         OutputInterface output,
         AnalyzeModel model)
  throws java.lang.Exception
```

See Also:
`ControlInterface.output(URL, OutputInterface, AnalyzeModel)`

saveAnalyzeModel

```
public void saveAnalyzeModel(java.net.URL url)
```

See Also:
`ControlInterface.saveAnalyzeModel(URL)`

reloadUseCases

```
public void reloadUseCases(java.net.URL url)
```

See Also:
`ControlInterface.reloadUseCases(URL)`

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

See Also:
`ModulePropertyInterface.applyProperties()`

findParser

```
private ParserInterface findParser(java.lang.String name)
```

Helper method for applyProperties to find parsers by name.

Parameters:
`name` - Parser to search.

Returns:
Found parser or null if no parser with given name existed.

findHeuristic

```
private HeuristicInterface findHeuristic(java.lang.String name)
```

Helper method for applyProperties to find heuristics by name.

(continued from last page)

Parameters:
`name` - Heuristic to search.

Returns:
Found heuristic or null if no heuristic with given name existed.

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

See Also:
`ModulePropertyInterface.loadDefaultProperties()`

saveProperties

```
public void saveProperties()
  throws java.io.IOException
```

See Also:
`ModulePropertyInterface.saveProperties()`

loadProperties

```
public void loadProperties()
  throws java.io.IOException
```

See Also:
`ModulePropertyInterface.loadProperties()`

ucot.core
Class Core.OutputJob

```
java.lang.Object
  |
  +-ucot.core.Core.OutputJob
```

private class **Core.OutputJob**
extends java.lang.Object

This class is a simple container for single output jobs. An instance of `OutputJob` is used to store the necessary data for one output operation.

Necessary data equals to the `OutputInterface` of the output adapter to be used for output, `URL` to the target file and `AnalyzeModel` that is being saved.

This class is required to be able to stack these operations for the separate output thread running on the background.
Author:
tujupien

Fields

output

`private ucot.output.OutputInterface` **`output`**

url

`private java.net.URL` **`url`**

analyzeModel

`private ucot.model.AnalyzeModel` **`analyzeModel`**

Constructors

Core.OutputJob

`private` **`Core.OutputJob`**`()`

ucot.core
Class DummyProgressBar

```
java.lang.Object
   |
   +-ucot.core.DummyProgressBar
```

All Implemented Interfaces:
ProgressBarInterface

public class **DummyProgressBar**
extends java.lang.Object
implements ProgressBarInterface

DummyProgressBar which does absolutely nothing but helps avoiding null values in ProgressBar variables.
Author:
tujupien

Constructors

DummyProgressBar

`public DummyProgressBar()`

Methods

getMaximum

`public int getMaximum()`

getMinimum

`public int getMinimum()`

getPercentageComplete

`public double getPercentageComplete()`

getString

`public java.lang.String getString()`

getValue

```
public int getValue()
```

setMaximum

```
public void setMaximum(int maximum)
```

setMinimum

```
public void setMinimum(int minimum)
```

setString

```
public void setString(java.lang.String string)
```

setValue

```
public void setValue(int value)
```

setVisible

```
public void setVisible(boolean visible)
```

ucot.core
Class PluginClassLoader

```
java.lang.Object
  |
  +-java.lang.ClassLoader
      |
      +-ucot.core.PluginClassLoader
```

public class **PluginClassLoader**
extends java.lang.ClassLoader

This is class loader for UCOT-programs plugins. It is used to load plugins from spesific directory. Currently it does not handle jar files properly so only plain class files can be loaded.

This class loader delegates class loading normally to its parents. If the classes are not found on general classpaths then classes are looked under the directory set at the construction time. Classes are located normal way: packages are directories and classes are files ending with `.class`-extension.
See Also:
  `PluginLoader`

Fields

pluginDir

```
java.io.File pluginDir
```

jarFilter

```
private java.io.FileFilter jarFilter
```

This is file filter for listin jar files.

Constructors

PluginClassLoader

```
public PluginClassLoader(java.io.File dir)
```

Costructs the classloader for spesific directory.

Parameters:
  `dir` - The directory for the class loader.

Methods

findClass

```
protected java.lang.Class findClass(java.lang.String name)
  throws java.lang.ClassNotFoundException
```

loadClassData

```
protected byte[] loadClassData(java.lang.String name)
  throws java.lang.ClassNotFoundException
```

Loads class data under the directory defined for this classloader. It modifies the class name so that `package.package1.ClassABCD` means file `package/package1/ClassABCD.class`. Then the file is located and if its is not found then `ClassNotFoundException` is thrown.

Parameters:
`name` - The class name we need to load.

Returns:
The data for the class.

Throws:
`ClassNotFoundException` - Thrown if file for the class is not found.

ucot.core
Class PluginLoader

```
java.lang.Object
  |
  +-ucot.core.PluginLoader
```

public class **PluginLoader**
extends java.lang.Object

Handles plugin loading. This class finds all the subdirectories under spesific direcotyr. Those spesific subdirectories are considered plugin directories. Very simple configuration file is expected to be found under the plugin directory. This file is called `plugin.properties` and it is normal java properties file.

Here is example of plugin.properties file that defines plugin called "Adapter". Property called class defines the plugins main class that is loaded under this plugin loader.

```
name=Adapter
 class=package1.AdapterClass
```

To use loaded plugins the program asks loaded class with `getClasses()` or `getClasses(Class)`

Fields

DEFAULT_PLUGIN_DIR_NAME

`private static final java.lang.String` **`DEFAULT_PLUGIN_DIR_NAME`**

Constant value: **`plugins`**

DEFAULT_PLUGIN_PROPERTIES_FILE

`private static final java.lang.String` **`DEFAULT_PLUGIN_PROPERTIES_FILE`**

Constant value: **`plugin.properties`**

PLUGIN_DIR_NOT_EXIST_LOGMESSAGE

`private static final java.lang.String` **`PLUGIN_DIR_NOT_EXIST_LOGMESSAGE`**

Constant value: **`PluginLoader.PLUGIN_DIR_NOT_EXIST_LOGMESSAGE`**

PLUGIN_DIR_IS_NOT_DIR_LOGMESSAGE

`private static final java.lang.String` **`PLUGIN_DIR_IS_NOT_DIR_LOGMESSAGE`**

(continued from last page)

Constant value: **`PluginLoader.PLUGIN_DIR_IS_NOT_DIR_LOGMESSAGE`**

LOADER_CHECKING_DIR_LOGMESSAGE

`private static final java.lang.String` **`LOADER_CHECKING_DIR_LOGMESSAGE`**

Constant value: **`PluginLoader.LOADER_CHECKING_DIR_LOGMESSAGE`**

FOUND_MAIN_DIR_LOGMESSAGE

`private static final java.lang.String` **`FOUND_MAIN_DIR_LOGMESSAGE`**

Constant value: **`PluginLoader.FOUND_MAIN_DIR_LOGMESSAGE`**

NO_PLUGIN_PROPERTIES_FOUND_LOGMESSAGE

`private static final java.lang.String` **`NO_PLUGIN_PROPERTIES_FOUND_LOGMESSAGE`**

Constant value: **`PluginLoader.NO_PLUGIN_PROPERTIES_FOUND_LOGMESSAGE`**

ERROR_READING_FILE_LOGMESSAGE

`private static final java.lang.String` **`ERROR_READING_FILE_LOGMESSAGE`**

Constant value: **`PluginLoader.ERROR_READING_FILE_LOGMESSAGE`**

PLUGIN_CLASS_NOT_DEFINED_LOGMESSAGE

`private static final java.lang.String` **`PLUGIN_CLASS_NOT_DEFINED_LOGMESSAGE`**

Constant value: **`PluginLoader.PLUGIN_CLASS_NOT_DEFINED_LOGMESSAGE`**

PLUGIN_NAME_NOT_DEFINED_LOGMESSAGE

`private static final java.lang.String` **`PLUGIN_NAME_NOT_DEFINED_LOGMESSAGE`**

Constant value: **`PluginLoader.PLUGIN_NAME_NOT_DEFINED_LOGMESSAGE`**

LOADING_PLUGIN_LOGMESSAGE

`private static final java.lang.String` **`LOADING_PLUGIN_LOGMESSAGE`**

Constant value: **`PluginLoader.LOADING_PLUGIN_LOGMESSAGE`**

PLUGIN_CLASS_NOT_FOUND_LOGMESSAGE

`private static final java.lang.String` **`PLUGIN_CLASS_NOT_FOUND_LOGMESSAGE`**

Constant value: **`PluginLoader.PLUGIN_CLASS_NOT_FOUND_LOGMESSAGE`**

(continued from last page)

LOADED_PLUGIN_LOGMESSAGE

`private static final java.lang.String` **`LOADED_PLUGIN_LOGMESSAGE`**

Constant value: **`PluginLoader.LOADED_PLUGIN_LOGMESSAGE`**

CLASS_PROPERTY_KEY

`private static final java.lang.String` **`CLASS_PROPERTY_KEY`**

Constant value: **`class`**

NAME_PROPERTY_KEY

`private static final java.lang.String` **`NAME_PROPERTY_KEY`**

Constant value: **`name`**

---

pluginMainDir

`java.io.File` **`pluginMainDir`**

---

logger

`java.util.logging.Logger` **`logger`**

---

pluginClasses

`java.util.Map` **`pluginClasses`**

Constructors

PluginLoader

`public` **`PluginLoader`**`(Core core)`

Methods

getPluginDir

`public java.io.File` **`getPluginDir`**`()`

---

getClasses

`public java.util.Collection` **`getClasses`**`()`

(continued from last page)

getClasses

`public java.util.Collection` **`getClasses`**`(java.lang.Class reqApi)`

getDirNameForPlugin

`public java.lang.String` **`getDirNameForPlugin`**`(java.lang.Class klass)`

Package ucot.exceptions

Exceptions specified for UCOT.

ucot.exceptions
Class BadPropertyValueException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-ucot.exceptions.BadPropertyValueException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **BadPropertyValueException**
extends java.lang.Exception

Thrown when UCOT application tries to apply properties for adapter and fails to do so because of illegal value of property.

Author:
tujupien

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **-2418630707378890457**

Constructors

BadPropertyValueException

`public BadPropertyValueException()`

BadPropertyValueException

`public BadPropertyValueException(java.lang.String message)`

ucot.exceptions
Class CannotLoadUseCasesException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-ucot.exceptions.UseCaseException
              |
              +-ucot.exceptions.CannotLoadUseCasesException
```

All Implemented Interfaces:
java.io.Serializable

public class **CannotLoadUseCasesException**
extends UseCaseException

Thrown when `UseCaseCollection` fails to get `UseCases` from given source.
Author:
tujupien

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **8870077222554560312**

Constructors

CannotLoadUseCasesException

`public` **`CannotLoadUseCasesException`**`()`

CannotLoadUseCasesException

`public` **`CannotLoadUseCasesException`**`(java.lang.String message)`

ucot.exceptions
Class FileFormatNotSupportedException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.io.IOException
            |
            +-ucot.exceptions.FileFormatNotSupportedException
```

All Implemented Interfaces:
java.io.Serializable

public class **FileFormatNotSupportedException**
extends java.io.IOException

Exception thrown when fileformat is not supported.

`Core` throws this when it tries to load serialized `AnalyzeModel` or use cases and the fileformat of the source is not supported.
Author:
tujupien

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **-9168394555942361427**

Constructors

FileFormatNotSupportedException

`public` **`FileFormatNotSupportedException`**`()`

FileFormatNotSupportedException

`public` **`FileFormatNotSupportedException`**`(java.lang.String message)`

ucot.exceptions
Class UseCaseException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-ucot.exceptions.UseCaseException
```

All Implemented Interfaces:
java.io.Serializable

Direct Known Subclasses:
CannotLoadUseCasesException

public class **UseCaseException**
extends java.lang.Exception

General exception for usecases.
Author:
tujupien

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **994398655500521030**

Constructors

UseCaseException

`public UseCaseException()`

UseCaseException

`public UseCaseException(java.lang.String s)`

Package
ucot.heuristic

Heuristic interface and related classes.

ucot.heuristic
Class AbbottsHeuristic

```
java.lang.Object
   |
   +-ucot.ModuleProperties
      |
      +-ucot.heuristic.AbbottsHeuristic
```

All Implemented Interfaces:
HeuristicInterface, ModulePropertyInterface

public class **AbbottsHeuristic**
extends ModuleProperties
implements ModulePropertyInterface, HeuristicInterface

Abbott's Heuristic is a way to etsimate which parts of the speech are relevant and in which way when doing object analyze.
Author:
UCOT

Fields

name

```
public static final java.lang.String name
```

Constant value: **`Abbott's heuristic`**

logger

```
private java.util.logging.Logger logger
```

Constructors

AbbottsHeuristic

```
public AbbottsHeuristic()
```

Methods

doHeuristic

```
public void doHeuristic(ParsedData data,
         AnalyzeModel analyzeModel)
  throws java.lang.Exception
```

(continued from last page)

getName

```
public java.lang.String getName()
```

toString

```
public java.lang.String toString()
```

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

ucot.heuristic
Class DummyHeuristic

```
java.lang.Object
  |
  +-ucot.ModuleProperties
      |
      +-ucot.heuristic.DummyHeuristic
```

All Implemented Interfaces:
HeuristicInterface, ModulePropertyInterface

public class **DummyHeuristic**
extends ModuleProperties
implements ModulePropertyInterface, HeuristicInterface

DummyHeuristic which does absolutely nothing but helps avoiding null values in heuristic variables.
Author:
UCOT

Constructors

DummyHeuristic

```
public DummyHeuristic()
```

Methods

doHeuristic

```
public void doHeuristic(ParsedData data,
         AnalyzeModel model)
  throws java.lang.Exception
```

getName

```
public java.lang.String getName()
```

ucot.heuristic
Interface HeuristicInterface

All Superinterfaces:
    ModulePropertyInterface

All Known Implementing Classes:
    AbbottsHeuristic, DummyHeuristic

---

public interface **HeuristicInterface**
extends ModulePropertyInterface

Interface for all heuristics used by UCOT. Takes ParsedData and stores analyzemodel to given model. Appends if possible.
Author:
    UCOT

Methods

doHeuristic

```
public void doHeuristic(ParsedData data,
                        AnalyzeModel analyzeModel)
  throws java.lang.Exception
```

Performs heuristic on given ParsedData object returns.

Parameters:
    `data` - Parsed Use case for the heuristic.
    `analyzeModel` - Analyze model where parsed data is added to.

---

getName

```
public java.lang.String getName()
```

Returns adapter's name.

Returns:
    Adapter's name.

Package
ucot.input

Classes related to reading usecases from a file.

ucot.input
Class DummyInput

```
java.lang.Object
   |
   +-ucot.ModuleProperties
        |
        +-ucot.input.DummyInput
```

All Implemented Interfaces:
InputInterface, ModulePropertyInterface

public class **DummyInput**
extends ModuleProperties
implements ModulePropertyInterface, InputInterface

DummyInput which does absolutely nothing but helps avoiding null values in input variables.
Author:
tujupien

Constructors

DummyInput

`public DummyInput()`

Methods

read

```
public UseCaseCollection read(java.net.URL url)
  throws java.lang.Exception
```

See Also:
`InputInterface.read(URL)`

canRead

`public boolean canRead(java.net.URL url)`

See Also:
`InputInterface.canRead(URL)`

ucot.input
Class InputCollection

```
java.lang.Object
  |
  +-ucot.input.InputCollection
```

public class **InputCollection**
extends java.lang.Object

Contains input handlers.
Author:
UCOT

Fields

inputs

```
private java.util.Vector inputs
```

Constructors

InputCollection

```
public InputCollection()
```

Creates input collection.

Methods

add

```
public void add(InputInterface inf)
```

Adds given input handler to the collection.

Parameters:
`inf`

getInputForUrl

```
public InputInterface getInputForUrl(java.net.URL url)
  throws java.io.FileNotFoundException
```

Returns input that can read file on given url

(continued from last page)

Parameters:

`url` - of the file we want to read

Returns:

InputInterface that can read file or null if none can

getInputCount

```
public int getInputCount()
```

Returns:

How many different handlers there are.

getInput

```
public InputInterface getInput(int index)
```

Returns handler by its index

Parameters:

`index` - Index of the wanted handler.

Returns:

The handler at that index.

ucot.input
Interface InputInterface

All Superinterfaces:
ModulePropertyInterface

All Known Implementing Classes:
DummyInput, ProcessMLInputAdapter, SimpleInputAdapter

public interface **InputInterface**
extends ModulePropertyInterface

Interface for inputs used by UCOT core. Inputs are the modules that handle reading data from given url and parsing usecases from it. User should first check that some URL is readable by this reader by invoking `canRead(URL)` method. It is up to the Input to set each loaded usecase's url correctly.

Methods

read

```
public UseCaseCollection read(java.net.URL url)
  throws java.lang.Exception
```

Reads usecase collection from URL.

Parameters:
`url` - URL where input is loaded from

Returns:
Steps read from the input

canRead

```
public boolean canRead(java.net.URL url)
```

Test if this input can read the data in specified url.

Parameters:
`url` - The url.

Returns:
true If input can read the contents of the url.

toString

```
public java.lang.String toString()
```

Returns inputs name as a String

(continued from last page)

Returns:
name of this input

ucot.input
Class ProcessMLInputAdapter

```
java.lang.Object
  |
  +-ucot.ModuleProperties
      |
      +-ucot.input.ProcessMLInputAdapter
```

All Implemented Interfaces:
InputInterface, ModulePropertyInterface

public class **ProcessMLInputAdapter**
extends ModuleProperties
implements ModulePropertyInterface, InputInterface

This InputInterface reads ProcessMl-files and parses usecases from them. References to sub-usecase is stored to usecase's steps. If sub-usecase has no references to it, it is currently discarded.
Author:
vevijopi

Fields

logger

`private java.util.logging.Logger` **`logger`**

Constructors

ProcessMLInputAdapter

`public` **`ProcessMLInputAdapter`**`()`

Methods

read

```
public UseCaseCollection read(java.net.URL url)
  throws java.io.IOException
```

getCorrectAbstraction

`private org.w3c.dom.Element` **`getCorrectAbstraction`**`(org.w3c.dom.NodeList abstractions)`

(continued from last page)

Goes throguh a nodelist containing "abstraction"-named elements and returns the one which has level 0

Parameters:
`abstractions` - list of elements with the name abstraction (from processml)

Returns:
abstraction-element with level 0, or null if none found

ParseSteps

```
private boolean ParseSteps(UseCase usecase,
         org.w3c.dom.NodeList steps)
```

Parses steps from given nodelist, stores them to given usecase

Parameters:
`usecase` - where steps are stored
`steps` - nodelist containing the "step"-elements

Returns:
true if everything went ok

ParseStep

```
private UseCaseStep ParseStep(org.w3c.dom.Node node)
```

Parses usecase step from given node and returns it, or null if acceptable one wasn't found

Parameters:
`node` - xml-element that contains a step

Returns:
parsed usecasestep or null

ParseInstanceDetails

```
private void ParseInstanceDetails(org.w3c.dom.Element processInstance,
         UseCase usecase)
```

Parses processInstance's id and adds it to usecase, also checks if this usecase is a subusecase

Parameters:
`processInstance` - xml-element to parse the details from
`usecase` - usecase that was created from processInstance attribute

canRead

```
public boolean canRead(java.net.URL url)
```

Tests if this adapter can read the file. For now, only test is that the file ends with ".xml" If we're adding more xml based inputs, we could verify that the file matches processml.dtd. I didn't implement this, because it could slow down loading (other types of) files a bit.

Parameters:

`url` - url of the file to test

Returns:

true if this adapter can read the file

toString

```
public java.lang.String toString()
```

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

ucot.input
Class SimpleInputAdapter

```
java.lang.Object
   |
   +-ucot.ModuleProperties
        |
        +-ucot.input.SimpleInputAdapter
```

All Implemented Interfaces:
InputInterface, ModulePropertyInterface

---

public class **SimpleInputAdapter**
extends ModuleProperties
implements ModulePropertyInterface, InputInterface

Input adapter for the simple input format. Reads usecases that are stored in this format:

```
 [name]
 Name of the usecase is here
 [id]
 Id of the usecase (must be unique within file, it is discarded after loading)
 [steps]
 Steps separated with linebreak. Step can have a sub-usecase, it is marked with (usecaseid)
after the steps description and "."-character.
 [end]
 Use case ends with [end] tag, another usecase can begin with [name] element now
```

Author:
pajumasu & vevijopi

Fields

name

`public static final java.lang.String` **`name`**

Constant value: **`Simple input adapter`**

Constructors

SimpleInputAdapter

`public` **`SimpleInputAdapter`**`()`

Methods

(continued from last page)

getName

```
public java.lang.String getName()
```

Returns the name of this adapter

Returns:
name of this adapter

read

```
public UseCaseCollection read(java.net.URL url)
  throws java.io.IOException
```

readUseCases

```
public void readUseCases(java.io.BufferedReader reader,
         UseCaseCollection collection,
         java.net.URL url)
  throws java.io.IOException
```

Reads usecases from given reader, stores them to given collection and sets their url to given one.

Parameters:
`reader` - reader for the inputstream
`collection` - where all found usecases are stored
`url` - url for the usecase

Throws:
`IOException`

readUseCase

```
public UseCase readUseCase(java.io.BufferedReader reader,
         java.net.URL url)
  throws java.io.IOException
```

Reads a single usecase from given reader, sets it's url to given one

Parameters:
`reader` - reader to read usecases with
`url` - url for the usecases

Returns:
read new usecase or null if error encountered

Throws:
`IOException`

(continued from last page)

parseStep

```
private UseCaseStep parseStep(java.lang.String line)
```

Parses usecase step from given line. Also stores relation id

Parameters:
`line`

Returns:
parsed usecasestep

canRead

```
public boolean canRead(java.net.URL url)
```

Tests if this adapter can read the file. For now, only test is that the file ends with ".txt"

Parameters:
`url` - url of the file to test

Returns:
true if this adapter can read the file

toString

```
public java.lang.String toString()
```

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

ucot.input
Class UseCase

```
java.lang.Object
   |
   +-ucot.input.UseCase
```

All Implemented Interfaces:
java.lang.Iterable

---

public class **UseCase**
extends java.lang.Object
implements java.lang.Iterable

UseCase class. Contains UseCaseSteps related to this usecase.
Author:
vevijopi

Fields

useCaseSteps

`private java.util.Vector useCaseSteps`

Vector that contains all usecase steps

---

url

`private java.net.URL url`

---

name

`private java.lang.String name`

---

isUseCaseAnalyzed

`private boolean isUseCaseAnalyzed`

Is usecase added to current analyze model.

---

subUseCase

`private boolean subUseCase`

(continued from last page)

Is this a sub-usecase, mainly used when connecting sub-usecases to usecase steps

model

`private ucot.model.AnalyzeModel **model**`

When heuristic was ran on this usecase, this model was created. It was later added to the main usecase, but a copy was left. This copy is used for highlighting.

heuristic

`private ucot.heuristic.HeuristicInterface **heuristic**`

Which heuristicInterface was used on this usecase

parser

`private ucot.parser.ParserInterface **parser**`

Which parserInterface was used on this usecase

useCaseId

`private java.lang.String **useCaseId**`

UseCase's id loaded from processml files or from simple usecase format Only used for connecting usecase steps to sub-usecases

Constructors

UseCase

`public **UseCase**()`

Methods

setParser

`public void **setParser**(ParserInterface parser)`

Set the parser that was used to parse this usecase

Parameters:

`parser` - parser that was used

setHeuristic

`public void **setHeuristic**(HeuristicInterface heuristic)`

Set the heuristic that was used to this usecase

Parameters:
`heuristic` - heuristic

getParser

`public ParserInterface **getParser**()`

Get the parser that was used on this usecase

Returns:
parser that was used

getHeuristic

`public HeuristicInterface **getHeuristic**()`

Get the heuristic that was used on this usecase

Returns:
heuristic that was used

getParserName

`public java.lang.String **getParserName**()`

Returns name of the parser that was used in creating this usecase

Returns:
parser's name

getHeuristicName

`public java.lang.String **getHeuristicName**()`

Returns name of the heuristic that was used in creating this usecase

Returns:
name of the heuristic

getAnalyzeModel

`public AnalyzeModel **getAnalyzeModel**()`

(continued from last page)

Returns the (mini) analyzemodel that was created from this (and only this) usecase

Returns:
mini analyzemodel

setAnalyzeModel

`public void` **`setAnalyzeModel`**`(AnalyzeModel model)`

Sets the (mini) analyzemodel that was created from this (and only this) usecase

Parameters:
`model` - this usecase was created from

setAsSubUseCase

`public void` **`setAsSubUseCase`**`(boolean sub)`

Mark this usecase as a sub usecase

Parameters:
`sub` - true if this usecase is a sub-usecase

isSubUseCase

`public boolean` **`isSubUseCase`**`()`

Is this usecase a sub-usecaes

Returns:
true if it is

isUseCaseAnalyzed

`public boolean` **`isUseCaseAnalyzed`**`()`

Is this usecase analyzed and added to main analyzemodel

Returns:
true if it is

setUseCaseAnalyzed

`public void` **`setUseCaseAnalyzed`**`(boolean analyzed)`

Set wheiter this usecase is analyzed.

(continued from last page)

Parameters:
`analyzed` - is the usecase analyzed

setUseCaseAnalyzed

`public void **setUseCaseAnalyzed**()`

Mark this usecase as analyzed

iterator

`public java.util.Iterator **iterator**()`

Iterator for the usecase steps

setId

`public void **setId**(java.lang.String relationId)`

Set relation id for this usecase. Only used for connecting UseCaseSteps to subusecases

Parameters:
`relationId` - this usecase's id loaded from file

getUseCaseId

`public java.lang.String **getUseCaseId**()`

Returns this usecase's id.

Returns:
usecase id

getStepCount

`public int **getStepCount**()`

Returns the count of steps this usecase has

Returns:
step count

getStep

`public UseCaseStep **getStep**(int index)`

Returns step with given index

(continued from last page)

Parameters:
`index` - index of the step

Returns:
UseCaseStep or null if index was incorrect

clear

```
public void clear()
```

Removes all usecase's steps

addStep

```
public void addStep(java.lang.String step,
        UseCase subUseCase)
```

Adds a new step to this usecase

Parameters:
`step` - step's description
`subUseCase` - reference to sub usecase

addStep

```
public void addStep(UseCaseStep step)
```

Adds a new UseCaseStep to this usecase

Parameters:
`step` - UseCaseStep object to add

getUrl

```
public java.net.URL getUrl()
```

Returns the url where this usecase was loaded from

Returns:
url where this usecase was loaded from

setUrl

```
public void setUrl(java.net.URL url)
```

Sets the url where this usecase was loaded from

Parameters:

(continued from last page)

`url`

getName

`public java.lang.String **getName**()`

Returns this usecase's name

Returns:
name

setName

`public void **setName**(java.lang.String name)`

Sets a new name for this usecase

Parameters:
`name` - new name

toString

`public java.lang.String **toString**()`

Returns this usecase as a string with this format:

```
Name
 ---
 (step index). (step)
 (step index). (step)
 ...
```

equals

`public boolean **equals**(java.lang.Object obj)`

Equals method for usecases.

hashCode

`public int **hashCode**()`

ucot.input
Class UseCaseCollection

```
java.lang.Object
   |
   +-java.util.Observable
      |
      +-ucot.input.UseCaseCollection
```

public class **UseCaseCollection**
extends java.util.Observable

Collection of usecases. Handles finds and merges for them.
Author:
vevijopi

Fields

CANNOT_LOAD_USE_CASES_ERROR

`protected static java.lang.String` **`CANNOT_LOAD_USE_CASES_ERROR`**

useCases

`private java.util.Vector` **`useCases`**

Constructors

UseCaseCollection

`public` **`UseCaseCollection`**`()`

Methods

merge

```
public void merge(UseCaseCollection collection)
  throws CannotLoadUseCasesException
```

Merges given UseCaseCollection to this

Parameters:
`collection` - UseCaseCollection to merge

exists

```
public boolean exists(UseCase usecase)
```

(continued from last page)

Checks if given UseCase exists in this collection

Parameters:
`usecase` - UseCase to test

Returns:
true if UseCase exists in this collection

getUseCaseCount

```
public int getUseCaseCount()
```

Returns the count of use cases inside this collection.

Returns:
The count of use cases.

addUseCase

```
public boolean addUseCase(UseCase usecase)
```

Add usecase to collection and notify observers

Parameters:
`usecase` - UseCase to add

Returns:
true if usecase was added

addUseCase

```
public boolean addUseCase(UseCase usecase,
          boolean notify)
```

Add usecase to collection and notify observers

Parameters:
`usecase` - UseCase to add
`notify` - should we notifyobservers and set usecase as changed

Returns:
true if usecase was added

find

```
public UseCase find(java.lang.String id)
```

Finds usecase that has the given id

Parameters:
`id` - id to find

Returns:
usecase with given id, -1 if none found

getUseCase

```
public UseCase getUseCase(int index)
```

(continued from last page)

Returns UseCase from index

Parameters:
`index`

Returns:
UseCase

removeFromUrl

```
public void removeFromUrl(java.net.URL url)
```

Removes all usecases that are loaded from given url

Parameters:
`url` - usecases that are loaded from this url are removed

getURLs

```
public java.util.List getURLs()
```

Returns list of source URLs

Returns:
list of source URLs

getUseCasesFromURL

```
public java.util.List getUseCasesFromURL(java.net.URL url)
```

Returns list of usecases from source described in url.

Parameters:
`url`

Returns:
list of usecases from source described in url

clear

```
public void clear()
```

Clears loaded usecases.

remove

```
public void remove(int index)
```

Removes usecase with given index

Parameters:
`index` - index of the usecase to remove

removeAllSubUseCase

```
public void removeAllSubUseCase()
```

Removes all usecases that have been marked as subusecase

markAllUnanalyzed

```
public void markAllUnanalyzed()
```

Marks all usecases within this collection as unanalyzed

markAllUnanalyzed

```
private void markAllUnanalyzed(UseCase usecase)
```

Marks given usecase and it's sub usecases as unanalyzed

Parameters:
`usecase` - usecase to be marked

toString

```
public java.lang.String toString()
```

Stores all usecases within this collection to a string and returns it

resolveSubUseCases

```
public void resolveSubUseCases(boolean markAsSubUseCases)
```

Resolves step and sub-usecase relations, removes sub-usecases from usecasecollection and stores their reference to the corresponding usecase step

Parameters:
`markAsSubUseCases`

resolveSubUseCases

```
public void resolveSubUseCases(UseCase usecase,
         boolean markAsSubUseCases)
```

Resolves given usecases step's sub-usecases and marks them as sub-usecases if necessary

Parameters:
`markAsSubUseCases`

ucot.input
Class UseCaseStep

```
java.lang.Object
  |
  +-ucot.input.UseCaseStep
```

public class **UseCaseStep**
extends java.lang.Object

UseCase step class. Contains step's text and relation to another usecase (sub-usecase)
Author:
vevijopi

Fields

step

```
private java.lang.String step
```

Step description

subUseCase

```
private ucot.input.UseCase subUseCase
```

Sub usecase for this step

subUseCaseId

```
private java.lang.String subUseCaseId
```

Used to connect steps to sub-usecases. id of another usecase (loaded from file). It is used to connect these two together.

Constructors

UseCaseStep

```
public UseCaseStep(java.lang.String step,
                   UseCase a)
```

Methods

setSubUseCaseId

```
public void setSubUseCaseId(java.lang.String id)
```

Set sub usecase's id

Parameters:
`id` - id of a subusecase for this step

getSubUseCaseId

`public java.lang.String **getSubUseCaseId**()`

Returns sub usecase's id

Returns:
usecase id

setStep

`public void **setStep**(java.lang.String step)`

Sets step's description

Parameters:
`step` - description

getStep

`public java.lang.String **getStep**()`

Returns step's description

Returns:
description

getStep

`public java.lang.String **getStep**(boolean showSubUsecaseName)`

Returns step's description

Parameters:
`showSubUsecaseName` - should the subusecase's name be shown aswell

Returns:
description

getSubUseCase

`public UseCase **getSubUseCase**()`

returns sub usecase

Returns:
sub usecase

setSubUseCase

`public void **setSubUseCase**(UseCase usecase)`

Sets sub usecase

Parameters:
`usecase`

equals

`public boolean **equals**(java.lang.Object o)`

toString

`public java.lang.String **toString**()`

returns this step as a string aswell as subusecase if one exists

hashCode

`public int **hashCode**()`

Package ucot.model

Classes related to analyzemodel and its editing.

ucot.model
Interface AnalyzeModel

All Known Implementing Classes:
ObjectAnalyzeModel

public interface **AnalyzeModel**
extends

Interface for analyze models.

Analyze model is class that represents the result of UCOT-program. it is the internal representation of analyzed use case.

Basically analyze model is builded from *entities* that contains:

- Methods
- Attributes
- Parents
- Type

Methods are basically named relations that point to some entity. Method can point back to its owner. There can be only one method-relation that has same name and point to same entity. It is also possibly that method dont point to any entity. This just means that the method exists, but it has no influence to other entity in the analzyed model.

For example this interface has four methods. These can be represented as method-relations:

Attributes are relations between entities which caries cardinal information. For example the implementation of this class could have attribute that has attribute-relation one-to-many to Observers.

Parents are the "superentities" of the entity. These are same kind of consept like superclasses in java.

Type just a string that tells the type of the entity.
Author:
pajumasu

Methods

getEditor

`public ModelEditor getEditor()`

Returns editor for this analyze model.

Returns:
The ModelEditor that can be used to edit this model.

addObserver

`public void addObserver(java.util.Observer observer)`

(continued from last page)

Adds observer for this analyzemodel

Parameters:

`observer` - The observer to be added.

ucot.model
Class AnalyzeModelException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-ucot.model.AnalyzeModelException
```

All Implemented Interfaces:
java.io.Serializable

Direct Known Subclasses:
NoSuchAttributeException, NoSuchChildException, NoSuchEntityException, NoSuchMethodException, NoSuchParentException

---

public class **AnalyzeModelException**
extends java.lang.Exception

General exception used by analyzemodel
Author:
tujupien

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **1**

Constructors

AnalyzeModelException

`public` **`AnalyzeModelException`**`()`

AnalyzeModelException

`public` **`AnalyzeModelException`**`(java.lang.String message)`

AnalyzeModelException

```
public AnalyzeModelException(java.lang.String message,
                             java.lang.Throwable cause)
```

(continued from last page)

AnalyzeModelException

`public` **`AnalyzeModelException`**`(java.lang.Throwable cause)`

ucot.model
Interface ModelEditor

All Known Implementing Classes:
    ObjectAnalyzeModelEditor

---

public interface **ModelEditor**
extends

ModelEditor is a interface to edit analyze models.

Edition is made trough ModelEditor enable modifications tracking and to protect the internal workings of the concrete implementation of the analyze model.

Every change to the AnalyzeModel should notify the observers of the AnalyzeModel with appropriate Updation object. See Updation object for additional information about how to inform changes.
See Also:
    `for structure of information passed about modifications`, `for detailed explanation about the items found and editable in the model.`
Author:
    pajumasu

Methods

clearModel

```
public void clearModel()
```

Clears model and makes it empty.

If cleared MUST send Updation message:
 Type: UpdationType.CLEAR

See Also:
    `Updation.UpdationType.CLEAR`

---

getEntityNames

```
public java.util.Set getEntityNames()
```

Returns the names of the entities that the model contains.

Returns:
    the names of the entities that the model contains.

---

addEntity

```
public void addEntity(java.lang.String name)
```

(continued from last page)

Adds an entity to the model. If entity exists nothing is done.

If addition occurs MUST send Updation message:
Type: UpdationType.ADD
DataType: UpdationDataType.ENTITY
Parameters: name of the entity.

Parameters:
`name` - The name of the entity.

See Also:
`Updation.UpdationDataType.ENTITY`

removeEntity

```
public void removeEntity(java.lang.String name)
```

Removes entity from model. If no such entity is found nothing is done.

If removing occurs MUST send Updation message:
Type: UpdationType.DELETE
DataType: UpdationDataType.ENTITY
Parameters: name of the entity.

Parameters:
`name` - The name of the entity.

Throws:
`NoSuchEntityException` - If given entity is not found.

See Also:
`Updation.UpdationDataType.ENTITY`

getEntityType

```
public java.lang.String getEntityType(java.lang.String entity)
```

Returns the type of the entity.

Parameters:
`entity` - The name of the entity.

Returns:
The type of the entity.

Throws:
`NoSuchEntityException` - If given entity is not found.

setEntityType

```
public void setEntityType(java.lang.String entity,
         java.lang.String entityType)
```

(continued from last page)

Sets the type of the entity.

If type is set MUST send Updation message:
 Type: UpdationType.MODIFY
 DataType: UpdationDataType.ENTITY_TYPE
 Parameters: name of the entity, the type string

Parameters:
`entity` - The name of the entity.
`entityType` - The type

Throws:
`NoSuchEntityException` - If given entity is not found.

See Also:
`Updation.UpdationType.MODIFY`
`Updation.UpdationDataType.ENTITY_TYPE`

changeEntityName

```
public void changeEntityName(java.lang.String oldName,
          java.lang.String newName)
  throws NoSuchEntityException
```

Changes the name of the entity.

If change occurs MUST send Updation message:
 Type: UpdationType.MODIFY
 DataType: UpdationDataType.ENTITY
 Parameters: old name of the entity, new name of the entity

Parameters:
`oldName` - The entity name that is going to be changed.
`newName` - The new name for that entity.

Throws:
`NoSuchEntityException` - If given entity is not found.

See Also:
`Updation.UpdationType.MODIFY`
`Updation.UpdationDataType.ENTITY`

containsEntity

```
public boolean containsEntity(java.lang.String entityName)
```

Does the model contain the entity?

Parameters:
`entityName` - The name of the entity which existence is checked.

Returns:
true if the entity exists. False otherwise.

(continued from last page)

addParent

```
public void addParent(java.lang.String entityName,
         java.lang.String parentEntity)
  throws NoSuchEntityException
```

Adds parent to the entity. If entity is not found

if adding occurs MUST send Updation message:
 Type: UpdationType.ADD
 DataType: UpdationDataType.ENTITY_PARENT
 Parameters: name of the entity, the name of the parent

Parameters:

`entityName` - The name of the child entity.
`parentEntity` - The name of the parent entity.

Throws:

`NoSuchEntityException` - If given entity is not found.

See Also:

`Updation.UpdationDataType.ENTITY_PARENT`

removeParent

```
public void removeParent(java.lang.String entityName,
         java.lang.String parentEntity)
  throws AnalyzeModelException
```

Removes given parent from the entity.

If removing occurs MUST send Updation message:
 Type: UpdationType.DELETE
 DataType: UpdationDataType.ENTITY_PARENT
 Parameters: name of the entity, the name of the parent

Parameters:

`entityName` - The name of the child entity.
`parentEntity` - The name of the parent entity.

Throws:

`NoSuchEntityException` - If given entity is not found.

See Also:

`Updation.UpdationDataType.ENTITY_PARENT`

removeAllParents

```
public void removeAllParents(java.lang.String entityName)
  throws NoSuchEntityException
```

Removes all the parents of the entity. This does same thing than calling removeParent(String,String) for every parent.

Parameters:

`entityName` - The name of the entity which parents are cleared.

(continued from last page)

Throws:
`NoSuchEntityException` - If given entity is not found.

getParents

```
public java.util.Set getParents(java.lang.String entityName)
```

Returns the parents of the entity.

Parameters:
`entityName` - The name of the entity.

Returns:
The parents of the given entity.

Throws:
`NoSuchEntityException` - If given entity is not found.

addChild

```
public void addChild(java.lang.String entityName,
         java.lang.String childEntity)
  throws NoSuchEntityException
```

Adds child to the entity. This is same than calling addParent(childEntity, entityName) and same kind of updation message is expected.

if adding occurs MUST send Updation message:
 Type: UpdationType.ADD
 DataType: UpdationDataType.ENTITY_PARENT
 Parameters: name of the entity, the name of the parent

Parameters:
`entityName` - The name of parent entity.
`childEntity` - The name of the child.

Throws:
`NoSuchEntityException` - If given entity is not found.

See Also:
`Updation.UpdationDataType.ENTITY_PARENT`

removeChild

```
public void removeChild(java.lang.String entityName,
         java.lang.String childEntity)
  throws AnalyzeModelException
```

Removes given child from the entity. This is same than calling removeParent(childEntity, entityName) and same kind of updation message is expected.

If removing occurs MUST send Updation message:
 Type: UpdationType.DELETE
 DataType: UpdationDataType.ENTITY_PARENT
 Parameters: name of the entity, the name of the parent

(continued from last page)

Parameters:
`entityName` - The entity.
`childEntity` - The child to be removed.

Throws:
`NoSuchEntityException` - If given entity is not found.

See Also:
`Updation.UpdationDataType.ENTITY_PARENT`

removeAllChildren

```
public void removeAllChildren(java.lang.String entityName)
  throws AnalyzeModelException
```

Removes all the children from the entity. This is same than calling removeChild(String, String) for all the childs.

Parameters:
`entityName` - The name of the entity which childs are cleared.

Throws:
`NoSuchEntityException` - If given entity is not found.

getChildren

```
public java.util.Set getChildren(java.lang.String entityName)
```

Returns the names of the entity's children.

Parameters:
`entityName` - The name of the entity.

Returns:
The child of the given entity.

Throws:
`NoSuchEntityException` - If given entity is not found.

addMethod

```
public void addMethod(java.lang.String entityName,
         java.lang.String methodName)
  throws NoSuchEntityException
```

Add method to the entity.

If adding occurs MUST send Updation message:
Type: UpdationType.ADD
DataType: UpdationDataType.METHOD
Parameters: name of the entity, the name of the method

Parameters:
`entityName` - The name of the entity.

(continued from last page)

`methodName` - The name of the method.

Throws:

`NoSuchEntityException` - If given entity is not found.

See Also:

`Updation.UpdationDataType.METHOD`

removeMethod

```
public void removeMethod(java.lang.String entityName,
         java.lang.String methodName)
  throws NoSuchEntityException
```

Removes method from the entity.

If remove occurs MUST send Updation message:
Type: UpdationType.REMOVE
DataType: UpdationDataType.METHOD
Parameters: name of the entity, the name of the method

Parameters:

`entityName` - The entity which owns the method.
`methodName` - The name of the method.

Throws:

`NoSuchEntityException` - If given entity is not found.

See Also:

`Updation.UpdationDataType.METHOD`

changeMethodName

```
public void changeMethodName(java.lang.String entityName,
         java.lang.String methodOldName,
         java.lang.String methodNewName)
  throws AnalyzeModelException
```

Changes method name.

If change occurs MUST send Updation message:
Type: UpdationType.MODIFY
DataType: UpdationDataType.METHOD
Parameters: name of the entity, the old name of the method, the new name of the method.

Parameters:

`entityName` - The name of the entity.
`methodOldName`
`methodNewName`

Throws:

`NoSuchEntityException` - If given entity is not found.
`NoSuchMethodException` - If given method is not found.

See Also:

`Updation.UpdationType.MODIFY`

(continued from last page)

Updation.UpdationDataType.METHOD

containsMethod

```
public boolean containsMethod(java.lang.String entityName,
        java.lang.String methodName)
```

Checks if the given method exists in the entity.

Parameters:

`entityName` - The name of the entity
`methodName` - The name of the method.

Returns:

True if the entity has given method.

Throws:

`NoSuchEntityException` - If given entity is not found.

getMethodNames

```
public java.util.Set getMethodNames(java.lang.String entityName)
```

Returns the names of the entity's methods.

Parameters:

`entityName` - The name of the entity

Returns:

The entity's methods' names.

Throws:

`NoSuchEntityException` - If given entity is not found.

getEntitiesInfluencedByMethod

```
public java.util.Set getEntitiesInfluencedByMethod(java.lang.String entityName,
        java.lang.String methodName)
```

Returns set of entities that are refered by methods in the model.

Parameters:

`entityName` - The name of the entity that owns the method.
`methodName` - The methods name.

Returns:

Set of entity names that are refered by the method in some way.

Throws:

`NoSuchEntityException` - If given entity is not found.
`NoSuchMethodException` - If given method is not found.

(continued from last page)

addEntityInfluenceByMethod

```
public void addEntityInfluenceByMethod(java.lang.String entityName,
         java.lang.String methodName,
         java.lang.String influencedEntity)
  throws AnalyzeModelException
```

Adds an influence between entity's method and the given entity.

If addition occurs MUST send Updation message:
 Type: UpdationType.ADD
 DataType: UpdationDataType.METHOD_INFLUENCE
 Parameters: name of the entity, the name of the method, the name of entity influenced.

Parameters:

`entityName` - The name of the entity that owns the method.
`methodName` - The methods name.
`influencedEntity`

Throws:

`NoSuchEntityException` - If given entity is not found.
`NoSuchMethodException` - If given method is not found.

See Also:

`Updation.UpdationDataType.METHOD_INFLUENCE`

removeEntityInfluenceByMethod

```
public void removeEntityInfluenceByMethod(java.lang.String entityName,
         java.lang.String methodName,
         java.lang.String influencedEntity)
  throws AnalyzeModelException
```

Removes influnce between entity's method and the given entity.

If deletion occurs MUST send Updation message:
 Type: UpdationType.DELETE
 DataType: UpdationDataType.METHOD_INFLUENCE
 Parameters: name of the entity, the name of the method, the name of entity influenced.

Parameters:

`entityName` - The name of the entity that owns the method.
`methodName` - The methods name.
`influencedEntity` - The name of the method that is influenced by the given method.

Throws:

`NoSuchEntityException` - If given entity is not found.
`NoSuchMethodException` - If given method is not found.

See Also:

`Updation.UpdationDataType.METHOD_INFLUENCE`

addAttribute

```
public void addAttribute(java.lang.String entityName,
         java.lang.String attributeName)
  throws NoSuchEntityException
```

(continued from last page)

Adds attribute to the entity.

If addition occurs MUST send Updation message:
 Type: UpdationType.ADD
 DataType: UpdationDataType.ATTRIBUTE
 Parameters: name of the entity, the name of the attribute

Parameters:

`entityName` - The name of the entity.
`attributeName` - The name of the attribute.

Throws:

`NoSuchEntityException` - If given entity is not found.

See Also:

`Updation.UpdationDataType.ATTRIBUTE`

removeAttribute

```
public void removeAttribute(java.lang.String entityName,
         java.lang.String attributeName)
  throws AnalyzeModelException
```

Removes attribute from the entity.

If deletion occurs MUST send Updation message:
 Type: UpdationType.DELETE
 DataType: UpdationDataType.ATTRIBUTE
 Parameters: name of the entity, the name of the attribute

Parameters:

`entityName` - The name of the entity.
`attributeName` - The name of the attribute.

Throws:

`NoSuchEntityException` - If given entity is not found.
`NoSuchAttributeException` - If given attribute is not found.

See Also:

`Updation.UpdationDataType.ATTRIBUTE`

containsAttribute

```
public boolean containsAttribute(java.lang.String entityName,
         java.lang.String attributeName)
  throws NoSuchEntityException
```

Checks wheter the entity contains the attribute or not.

Parameters:

`entityName` - The name of the entity.
`attributeName` - The name of the attribute.

Returns:

True if contains.

(continued from last page)

Throws:

`NoSuchEntityException` - If given entity is not found.

getAttributeFromCardinal

```
public java.lang.String getAttributeFromCardinal(java.lang.String entityName,
         java.lang.String attributeName)
  throws AnalyzeModelException
```

Gets from part of the cardinality of the attribute relation. From part means the cardinality on the entity's side that owns the attribute. For example in one-to-many relation the 'one' is *from cardinality*.

Parameters:

`entityName` - The name of the entity.

`attributeName` - The name of the attibute.

Returns:

The cardinality value of the from part of the cardinal relation.

getAttributeToCardinal

```
public java.lang.String getAttributeToCardinal(java.lang.String entityName,
         java.lang.String attributeName)
  throws AnalyzeModelException
```

Gets to part of the cardinality of the attribute relation. To part means the cardinality on the attributes side. For example in one-to-many relation the 'many' is *to cardinality*.

Parameters:

`entityName` - The name of the entity.

`attributeName` - The name of the attibute.

Returns:

The cardinality value of the to part of the cardinal relation.

setAttributeFromCardinal

```
public void setAttributeFromCardinal(java.lang.String entityName,
         java.lang.String attributeName,
         java.lang.String cardinal)
  throws AnalyzeModelException
```

Sets the cardinality on the entitys side.

If change occurs MUST send Updation message:
 Type: UpdationType.MODIFY
 DataType: UpdationDataType.ATTRIBUTE_FROM_CARDINALITY
 Parameters: name of the entity, the name of the attribute, the from cardinality

Parameters:

`entityName` - The name of the entity.

`attributeName` - The name of the attibute.

`cardinal` - The cardinality value.

Throws:

(continued from last page)

NoSuchEntityException - If given entity is not found.
NoSuchAttributeException - If given attribute is not found.

See Also:
Updation.UpdationType.MODIFY
Updation.UpdationDataType.ATTRIBUTE_FROM_CARDINALITY

setAttributeToCardinal

```
public void setAttributeToCardinal(java.lang.String entityName,
         java.lang.String attributeName,
         java.lang.String cardinal)
  throws AnalyzeModelException
```

Sets the cardinality on attributes side.

If change occurs MUST send Updation message:
Type: UpdationType.MODIFY
DataType: UpdationDataType.ATTRIBUTE_TO_CARDINALITY
Parameters: name of the entity, the name of the attribute, the to cardinality

Parameters:
`entityName` - The name of the entity.
`attributeName` - The name of the attibute.
`cardinal` - The cardinality value.

Throws:
NoSuchEntityException - If given entity is not found.
NoSuchAttributeException - If given attribute is not found.

See Also:
Updation.UpdationType.MODIFY
Updation.UpdationDataType.ATTRIBUTE_TO_CARDINALITY

getAttributeNames

```
public java.util.Set getAttributeNames(java.lang.String entityName)
```

Returns the attributes of the entity.

Parameters:
`entityName` - The name of the entity.

Returns:
The attributes.

Throws:
NoSuchEntityException - If given entity is not found.

merge

```
public void merge(AnalyzeModel model)
  throws AnalyzeModelException
```

(continued from last page)

Merges given AnalyzeModel to the editors model. Merge only adds things that do not yet exist in the current model. Nothing is delted or modified. For example attribute's cardinalities are not modified even if there is same attribute with different cardinalities.

All proper updation messages should be sended for every action made. It might be good idea to send AnalydeModel.signalModificationStarted() when starting modifications and AnalzydeMode.signalReady() when done.

Parameters:

`model` - The analyze model beign merged.

getUpdations

```
public java.util.List getUpdations()
```

Get updations done to this analyzemodel.

Returns:

List of Updations or null if this feature is not supported.

mergeEntity

```
public void mergeEntity(java.lang.String targetEntityName,
        java.util.Set mergeSet)
```

Merges entities to other entity. If given target entity (called targetEntityName) does not exists it is created.

All the methods, attributes, and parents are added to one entity and the sources are removed afterwards. If there are same attributes it depens on the underlaying implementation which one of them will remain in the final entity.

Parameters:

`targetEntityName` - The name of the entity after merge.
`mergeSet` - The set of entity names that are going to be merged.

stepBack

```
public void stepBack(int steps)
```

Undoes some edition steps.

Parameters:

`steps` - to undo

execute

```
public boolean execute(Updation updation)
  throws AnalyzeModelException
```

Executes action defined by updaton object.

(continued from last page)

Returns:
true if success, false if did not.

readySignal

`public void readySignal()`

Signals all observers that this model is ready.

Sends observers Updation-message which type is Updation.UpdationType.READY.

See Also:
`updationStartedSignal()`

updationStartedSignal

`public void updationStartedSignal()`

(continued from last page)

Signals all observers that this model is beign modified.

This method should be called before the model is going to trough lots of changes. All observers receive updation signal that is from this model and the argument is instance of Updation which type is Updation.UpdationType.MODIFICATION_STARTED.

Example of Observer listening the model:

```
 new Observable(){
                void update(Observable o, Object arg){
                        // Check that we know how to handle the parameters.
                        if (! (o instanceof AnalyzeModel)) return;
                        if (! (arg instanceof Updation)) return;

                        Updation updation = (Updation) arg;

                        // Check updation type
                        switch(updation.getType()){
                                case Update.MODIFICATION_STARTED:
                                        drawUpdates = false;    // Dont draw updates.
                                break;
                                case Update.READY:

                                        drawUpdates = true;             // Start drawing
updates.
                                break;
                        }
                        // Draw updates if we are not in midle of updation.
                        if (drawUpdates){
                                doDrawUpdates();
                        }
                }
 }
```

saveUpdationsToFile

```
public void saveUpdationsToFile(java.io.File target)
  throws java.io.IOException
```

Saves the updations (modification log) to the given file.

Parameters:
`target` - Defines the target filename for the modification log file.

ucot.model
Class NoSuchAttributeException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-ucot.model.AnalyzeModelException
              |
              +-ucot.model.NoSuchAttributeException
```

All Implemented Interfaces:
java.io.Serializable

public class **NoSuchAttributeException**
extends AnalyzeModelException

General exception used when attribute isn't found
Author:
tujupien

Fields

serialVersionUID

```
private static final long serialVersionUID
```

Constant value: **8915129262644339201**

entityName

```
java.lang.String entityName
```

The name of the entity.

attributeName

```
java.lang.String attributeName
```

The name of the method

Constructors

NoSuchAttributeException

```
public NoSuchAttributeException(java.lang.String entityName,
                                java.lang.String attributeName)
```

(continued from last page)

NoSuchAttributeException

```
public NoSuchAttributeException(java.lang.String entityName,
                                java.lang.String attributeName,
                                java.lang.String message)
```

NoSuchAttributeException

```
public NoSuchAttributeException(java.lang.String entityName,
                                java.lang.String attributeName,
                                java.lang.Throwable cause)
```

NoSuchAttributeException

```
public NoSuchAttributeException(java.lang.String entityName,
                                java.lang.String attributeName,
                                java.lang.String message,
                                java.lang.Throwable cause)
```

Methods

buildMessage

```
private static java.lang.String buildMessage(java.lang.String entityName,
         java.lang.String attributeName)
```

Builds message for the error.

Parameters:
`entityName` - The entity name used in the error.

Returns:
The error message string.

---

setInternals

```
private void setInternals(java.lang.String entityName,
         java.lang.String attributeName)
```

ucot.model
Class NoSuchChildException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-ucot.model.AnalyzeModelException
              |
              +-ucot.model.NoSuchChildException
```

All Implemented Interfaces:
java.io.Serializable

public class **NoSuchChildException**
extends AnalyzeModelException

Error which is thrown if no entity of given name is found.
Author:
pajumasu

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **1419331743786346560**

entityName

`java.lang.String entityName`

The name of the entity.

Constructors

NoSuchChildException

`public NoSuchChildException(java.lang.String entityName)`

NoSuchChildException

```
public NoSuchChildException(java.lang.String entityName,
                            java.lang.String message)
```

(continued from last page)

NoSuchChildException

```
public NoSuchChildException(java.lang.String entityName,
                            java.lang.Throwable cause)
```

NoSuchChildException

```
public NoSuchChildException(java.lang.String entityName,
                            java.lang.String message,
                            java.lang.Throwable cause)
```

Methods

buildMessage

```
protected static java.lang.String buildMessage(java.lang.String entityName)
```

Builds message for the error.

Parameters:
`entityName` - The entity name used in the error.

Returns:
The error message string.

ucot.model
Class NoSuchEntityException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-ucot.model.AnalyzeModelException
              |
              +-ucot.model.NoSuchEntityException
```

All Implemented Interfaces:
java.io.Serializable

public class **NoSuchEntityException**
extends AnalyzeModelException

Error which is thrown if no entity of given name is found.
Author:
pajumasu

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **3810508935149009434**

entityName

`java.lang.String entityName`

The name of the entity.

Constructors

NoSuchEntityException

```
public NoSuchEntityException(java.lang.String entityName)
```

NoSuchEntityException

```
public NoSuchEntityException(java.lang.String entityName,
                             java.lang.String message)
```

(continued from last page)

NoSuchEntityException

```
public NoSuchEntityException(java.lang.String entityName,
                             java.lang.Throwable cause)
```

NoSuchEntityException

```
public NoSuchEntityException(java.lang.String entityName,
                             java.lang.String message,
                             java.lang.Throwable cause)
```

Methods

buildMessage

```
protected static java.lang.String buildMessage(java.lang.String entityName)
```

Builds message for the error.

Parameters:
`entityName` - The entity name used in the error.

Returns:
The error message string.

ucot.model
Class NoSuchMethodException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-ucot.model.AnalyzeModelException
            |
            +-ucot.model.NoSuchMethodException
```

All Implemented Interfaces:
java.io.Serializable

public class **NoSuchMethodException**
extends AnalyzeModelException

Error which is thrown when requested method is not found.
Author:
pajumasu

Fields

serialVersionUID

```
private static final long serialVersionUID
```

Constant value: **733527374956747393**

entityName

```
java.lang.String entityName
```

The name of the entity.

methodName

```
java.lang.String methodName
```

The name of the method

Constructors

NoSuchMethodException

```
public NoSuchMethodException(java.lang.String entityName,
                             java.lang.String methodName)
```

(continued from last page)

NoSuchMethodException

```
public NoSuchMethodException(java.lang.String entityName,
                             java.lang.String methodName,
                             java.lang.String message)
```

NoSuchMethodException

```
public NoSuchMethodException(java.lang.String entityName,
                             java.lang.String methodName,
                             java.lang.Throwable cause)
```

NoSuchMethodException

```
public NoSuchMethodException(java.lang.String entityName,
                             java.lang.String methodName,
                             java.lang.String message,
                             java.lang.Throwable cause)
```

Methods

buildMessage

```
private static java.lang.String buildMessage(java.lang.String entityName,
         java.lang.String methodName)
```

Builds message for the error.

Parameters:

`entityName` - The entity name used in the error.

Returns:

The error message string.

---

setInternals

```
private void setInternals(java.lang.String entityName,
         java.lang.String methodName)
```

ucot.model
Class NoSuchParentException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-ucot.model.AnalyzeModelException
              |
              +-ucot.model.NoSuchParentException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **NoSuchParentException**
extends AnalyzeModelException

Error which is thrown if no entity of given name is found.

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **1419331743786346560**

---

entityName

`java.lang.String entityName`

The name of the entity.

Constructors

NoSuchParentException

`public NoSuchParentException(java.lang.String entityName)`

Constructs `NoSuchParentException` with entity name that is missing.

Parameters:
`entityName` - The name of the entity that is not found.

NoSuchParentException

```
public NoSuchParentException(java.lang.String entityName,
                             java.lang.String message)
```

(continued from last page)

Constructs `NoSuchParentException` with entity name that is missing and message.

Parameters:
`entityName` - The name of the entity that is not found.
`message` - The message.

NoSuchParentException

```
public NoSuchParentException(java.lang.String entityName,
                             java.lang.Throwable cause)
```

Constructs `NoSuchParentException` with entity name that is missing and the cause.

Parameters:
`entityName` - The name of the entity that is not found.
`cause` - The cause.

NoSuchParentException

```
public NoSuchParentException(java.lang.String entityName,
                             java.lang.String message,
                             java.lang.Throwable cause)
```

Constructs `NoSuchParentException` with entity name that is missing, message containing some explanation and the cause.

Parameters:
`entityName` - The name of the entity that is not found.
`message` - The message.
`cause` - The cause.

Methods

buildMessage

```
protected static java.lang.String buildMessage(java.lang.String entityName)
```

Builds message for the error. Message looks like this:

```
No such entity called 'entityName'.
```

Parameters:
`entityName` - The entity name used in the error.

(continued from last page)

Returns:
The error message string.

ucot.model
Class Updation

```
java.lang.Object
   |
   +-ucot.model.Updation
```

All Implemented Interfaces:
java.io.Serializable

public class **Updation**
extends java.lang.Object
implements java.io.Serializable

Class contains updation information about the model. It contains the type of change and information about the change.

For example deletion of method would look like this:

```
new Updatition(UpdationType.DELETION, UpdatioDatatype.METHOD, "entity", "method");
```

It can be used to signal that the "method" of "entity" is deleted.
Author:
pajumasu

Fields

serialVersionUID

`public static final long` **`serialVersionUID`**

Constant value: **121289714289124**

type

`ucot.model.Updation.UpdationType` **`type`**

dataType

`ucot.model.Updation.UpdationDataType` **`dataType`**

references

`java.lang.String` **`references`**

(continued from last page)

Constructors

Updation

```
public Updation()
```

Initializes empty instance `Updation`. `UpdationType` of this is "no operation" and this does not carry any data.

Updation

```
public Updation(Updation.UpdationType type,
                Updation.UpdationDataType dataType)
```

Initializes new instance of `Updation` which has the given `UpdationType` and `UpdationDataType` as values.

Parameters:

`type` - `UpdationType` of this `Updation`

`dataType` - `UpdationDataType` of this `Updation`

Updation

```
public Updation(Updation.UpdationType type,
                Updation.UpdationDataType dataType,
                java.lang.String[] references)
```

Initializes new instance of `Updation` which has the given `UpdationType`, `UpdationDataType` and references as values.

Parameters:

`type` - `UpdationType` of this `Updation`

`dataType` - `UpdationDataType` of this `Updation`

`references` - References of this `Updation`. The references that should be carried depend on the used `UpdationDataType`.

See Also:

`Updation.UpdationDataType`

Updation

```
public Updation(Updation.UpdationType type)
```

Initializes new instance of `Updation` which has the given `UpdationType` as value.

Parameters:

`type` - `UpdationType` of this `Updation`

Methods

(continued from last page)

getDataType

`public Updation.UpdationDataType **getDataType**()`

Returns the `UpdationDataType` of this instance of `updation`.

Returns:

`UpdationDataType` of this `Updation`.

getReferences

`public java.lang.String[] **getReferences**()`

Returns the references carried by this instance of `updation`.

What the references are can be deciphered by checking the `UpdationDataType` of this `Updation`.

Returns:

References carried by `Updation`.

See Also:

`Updation.UpdationDataType`
`getDataType()`

getType

`public Updation.UpdationType **getType**()`

Returns the `UpdationType` of this instance of `Updation`.

Returns:

`UpdationType` of this `Updation`.

ucot.model
Class Updation.UpdationType

```
java.lang.Object
  |
  +-java.lang.Enum
      |
      +-ucot.model.Updation.UpdationType
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

public static final class **Updation.UpdationType**
extends java.lang.Enum

Contains information about the updation type.
Author:
pajumasu

Fields

NOP

`public static final ucot.model.Updation.UpdationType` **`NOP`**

No operation.

MODIFY

`public static final ucot.model.Updation.UpdationType` **`MODIFY`**

Modification occurred.

NEW

`public static final ucot.model.Updation.UpdationType` **`NEW`**

Cration ocurred.

DELETION

`public static final ucot.model.Updation.UpdationType` **`DELETION`**

Deletion ocurred.

CLEAR

`public static final ucot.model.Updation.UpdationType` **`CLEAR`**

Model has just been completely wiped out.

READY

`public static final ucot.model.Updation.UpdationType` **`READY`**

(continued from last page)

Big chunk of modification is done.

MODIFICATION_STARTED

```
public static final ucot.model.Updation.UpdationType MODIFICATION_STARTED
```

Big chunk of modification started. Time intensive updations should be cased and wait for ready signal.

Constructors

Updation.UpdationType

```
private Updation.UpdationType()
```

Methods

values

```
public final static Updation.UpdationType[] values()
```

valueOf

```
public static Updation.UpdationType valueOf(java.lang.String name)
```

ucot.model
Class Updation.UpdationDataType

```
java.lang.Object
   |
   +-java.lang.Enum
      |
      +-ucot.model.Updation.UpdationDataType
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

public static final class **Updation.UpdationDataType**
extends java.lang.Enum

Contains information about the type beign changed.
Author:
pajumasu

Fields

ENTITY

`public static final ucot.model.Updation.UpdationDataType` **`ENTITY`**

Entity is beign altered. Reference contains the name of the entity.

METHOD

`public static final ucot.model.Updation.UpdationDataType` **`METHOD`**

Method is beign altered. Reference contains the name of the entity and the name of the method.

ATTRIBUTE

`public static final ucot.model.Updation.UpdationDataType` **`ATTRIBUTE`**

Attribute is beign altered. Reference contains the name of the entity and the name of the attribute.

ATTRIBUTE_TO_CARDINALITY

`public static final ucot.model.Updation.UpdationDataType` **`ATTRIBUTE_TO_CARDINALITY`**

Attribute to cardinality is beign altered. Reference contains the name of the entity and the name of the attribute and the to cardinality.

ATTRIBUTE_FROM_CARDINALITY

`public static final ucot.model.Updation.UpdationDataType` **`ATTRIBUTE_FROM_CARDINALITY`**

Attribute to cardinality is beign altered. Reference contains the name of the entity and the name of the attribute and the from cardinality.

(continued from last page)

METHOD_INFLUENCE

`public static final ucot.model.Updation.UpdationDataType` **`METHOD_INFLUENCE`**

Method influence is beign altered. Reference contains the name of the entity and the name of the methdod and the name of the entity that is influenced.

ENTITY_PARENT

`public static final ucot.model.Updation.UpdationDataType` **`ENTITY_PARENT`**

Parent relation is beign altered. Reference contains the name of the entity which parent is changed and the entity that is set to the first ones parent.

ENTITY_TYPE

`public static final ucot.model.Updation.UpdationDataType` **`ENTITY_TYPE`**

The type is beign altered Reference contains the name of the entity which type is changed and the type.

EMPTY

`public static final ucot.model.Updation.UpdationDataType` **`EMPTY`**

No data caried.

Constructors

Updation.UpdationDataType

`private` **`Updation.UpdationDataType`**`()`

Methods

values

`public final static Updation.UpdationDataType[]` **`values`**`()`

---

valueOf

`public static Updation.UpdationDataType` **`valueOf`**`(java.lang.String name)`

Package
ucot.model.object

Objects used by analyzemodel.

ucot.model.object
Class Attribute

```
java.lang.Object
  |
  +-ucot.model.object.Attribute
```

All Implemented Interfaces:
java.io.Serializable

---

public class **Attribute**
extends java.lang.Object
implements java.io.Serializable

This is attribute of an entity. It is basically direct link with cardinal information.
Author:
pajumasu

Fields

serialVersionUID

`public static final long serialVersionUID`

Constant value: **82650222828672626**

---

fromCardinal

`java.lang.String fromCardinal`

---

toCardinal

`java.lang.String toCardinal`

---

entity

`ucot.model.object.Entity entity`

Constructors

Attribute

`public Attribute(Entity entity)`

Creates new Attribute reference to given entity.

Parameters:

(continued from last page)

`entity` - The entity this attribute refers to.

Methods

getEntity

```
public Entity getEntity()
```

Get the entity this attribute refers to.

Returns:
The entity referred.

setEntity

```
public void setEntity(Entity entity)
```

Sets the entity this attribute refers to.

Parameters:
`entity` - The entity to be refered.

getName

```
public java.lang.String getName()
```

Returns the name of the attribute.

Returns:
name of the attribute

ucot.model.object
Class Entity

```
java.lang.Object
   |
   +-ucot.model.object.Entity
```

All Implemented Interfaces:
java.io.Serializable

public class **Entity**
extends java.lang.Object
implements java.io.Serializable

This class represents entity in analyze model.
Author:
pajumasu

Fields

serialVersionUID

`public static final long serialVersionUID`

Constant value: **2368572046870269076**

attributes

`private java.util.Map attributes`

Maps attribute names and attributes. Attribute names are not in use currently.

methods

`private java.util.Map methods`

Maps method names and methods.

childEntities

`private java.util.Set childEntities`

The child entities.

parentEntities

`private java.util.Set parentEntities`

The parent entities.

(continued from last page)

name

`private java.lang.String` **`name`**

The name of this entity.

---

type

`private java.lang.String` **`type`**

The type of the entity.

---

deleted

`boolean` **`deleted`**

Marks if this entity is deleted or not. Because there are many references to entities inside the model it is easier to mark entity deletet when its is removed from the model and the unnessesary references are removed when they are needed. This way we dont need to know all the places for the references and remove them on site when entity is deleted.

Constructors

Entity

`public` **`Entity`**`(java.lang.String name)`

Creates the entity.

The name is modified so that the first letter is always an uppercase letter.

Parameters:
`name` - Name of this entity.

Methods

setName

`protected void` **`setName`**`(java.lang.String name)`

Sets the name of the entity.

The name is modified so that the first letter is always an uppercase letter.

Parameters:
`name` - Name of this entity.

---

updateAttributes

`private void` **`updateAttributes`**`()`

Removes deleted entities from the attribute list.

(continued from last page)

addAttribute

```
public void addAttribute(java.lang.String name,
        Attribute attribute)
```

Adds attribute for this entity

Parameters:
`name` - Name of the `Attribute`.
`attribute` - The `Attribute` to add.

getAttribute

```
public Attribute getAttribute(java.lang.String name)
```

Returns attribute called name.

Parameters:
`name` - Name of the `Attribute` to return.

Returns:
The `Attribute` called 'name'.

getAttributes

```
public java.util.Set getAttributes()
```

Returns all the attributes.

Returns:
`Attributes` of this entity in a Set.

removeAttribute

```
public void removeAttribute(java.lang.String name)
```

Removes attribute called name.

Parameters:
`name` - Name of the attribute to remove.

removeAttribute

```
public void removeAttribute(Attribute toBeRemoved)
```

Removes given attribute.

Parameters:
`toBeRemoved` - `Attribute` that should be removed.

addAttribute

`public void addAttribute(Attribute attribute)`

Adds attribute.

Attributes name is set based on the name of the entity defined in attribute.

Parameters:
`attribute` - `Attribute` to add.

getName

`public java.lang.String getName()`

Returns the name of this entity.

Returns:
Name of this entity.

checkEntityCollection

`private void checkEntityCollection(java.util.Collection col)`

Removes deleted entites from given collection.

Parameters:
`col` - Collection of entities.

getParents

`public java.util.Set getParents()`

Returns the parent entities of this entity.

Returns:
Parent entities of this entity.

getChildren

`public java.util.Set getChildren()`

Returns the children of this entity.

Returns:
Set of children of this entity.

addParent

`public void addParent(Entity entity)`

(continued from last page)

Adds parent for this `Entity`. Also adds this entity as a child of the given entity if nessesary.

Parameters:
`entity` - `Entity` that should be added as parent.

---

addChild

`public void **addChild**(Entity entity)`

Adds child for this `Entity`. Also adds this entity as a parent of the given entity if nessesary

Parameters:
`entity` - `Entity` that should be added as child.

---

removeParent

`public void removeParent(Entity entity)`

Removes parent of this entity and also removes this entity from the child list of a given entity if nessesary.

Parameters:
`entity` - `Entity` that should be removed from this entitys parents.

---

removeChild

`public void removeChild(Entity entity)`

Removes child of this entity and also removes this entity from the parent list of a given entity if nessesary.

Parameters:
`entity` - `Entity` that should be removed from this entitys children.

---

getMethod

`public Method getMethod(java.lang.String methodName)`

Returns `Method` of this `Entity` that is called 'name'.

Parameters:
`methodName` - Name of the `Method`.

Returns:
method `Method` called 'name'.

---

addMethod

`public void addMethod(java.lang.String methodName)`

(continued from last page)

Creates and adds `Method` for this `Entity` using given name.

Parameters:
`methodName` - Name of the `Method` to add.

addMethod

`public void **addMethod**(Method method)`

Adds method for this entity.

Parameters:
`method` - `Method` to add.

removeMethod

`public void **removeMethod**(java.lang.String methodName)`

Removes method called 'name'.

Parameters:
`methodName` - Name of the method to remove.

getMethodNames

`public java.util.Set **getMethodNames**()`

Returns the names of the methods.

Returns:
Set of method names of this `Entity`.

getType

`public java.lang.String **getType**()`

Return the type of this entity.

Returns:
The type of this entity.

setType

`public void **setType**(java.lang.String type)`

Sets the type for this entity.

(continued from last page)

Parameters:
`type` - The type

getMethods

`public java.util.Set **getMethods**()`

Get all the methods of this `Entity`.

Returns:
Set of `Methods` of this `Entity`.

ucot.model.object
Class Method

```
java.lang.Object
   |
   +-ucot.model.object.Method
```

All Implemented Interfaces:
java.io.Serializable

public class **Method**
extends java.lang.Object
implements java.io.Serializable

This class represents method of an entity.
Author:
pajumasu

Fields

serialVersionUID

`public static final long serialVersionUID`

Constant value: **4346347435879039046**

name

`private java.lang.String name`

The name of the method.

influenced

`private java.util.Set influenced`

The list of entities that can be indluenced by this method.

Constructors

Method

`public Method(java.lang.String name)`

Creates new method called name. Methods name is converted to lowercase.

Parameters:
`name`

Methods

(continued from last page)

setName

`protected void **setName**(java.lang.String name)`

Sets the name of the method. Methods name is converted to lowercase.

Parameters:
`name`

getInfluenced

`public java.util.Set **getInfluenced**()`

Returns the set of entities that are influenced by this method.

Returns:
set of entities that are influenced by this method

addInfluence

`public void **addInfluence**(Entity entity)`

Adds influence to this method.

Parameters:
`entity`

removeInfluence

`public void **removeInfluence**(Entity entity)`

Removes influence from this method.

Parameters:
`entity`

getName

`public java.lang.String **getName**()`

Returns the name of the method.

Returns:
name of the method

ucot.model.object
Class ObjectAnalyzeModel

```
java.lang.Object
   |
   +-java.util.Observable
      |
      +-ucot.model.object.ObjectAnalyzeModel
```

All Implemented Interfaces:
java.io.Serializable, AnalyzeModel

public class **ObjectAnalyzeModel**
extends java.util.Observable
implements AnalyzeModel, java.io.Serializable

Model that contains representation of the AnalyzeModel. This implementation uses objects to contain the information of the model.
Author:
pajumasu

Fields

serialVersionUID

`public static final long serialVersionUID`

Constant value: **8923590437239046**

editor

`private ucot.model.object.ObjectAnalyzeModelEditor editor`

The editor that edits this model

entities

`private java.util.Map entities`

The entities that are containde by this model.

Constructors

ObjectAnalyzeModel

`public ObjectAnalyzeModel()`

Creates empty model.

Methods

(continued from last page)

getEntity

`public Entity **getEntity**(java.lang.String name)`

Returns entity.

Parameters:
`name`

Returns:
entity

removeEntity

`public void removeEntity(java.lang.String name)`

Removes entity.

Parameters:
`name` - The name of the entity.

removeEntity

`public void removeEntity(Entity removeEntity)`

Removes entity.

Parameters:
`removeEntity` - The entity to be removoed

getEntityNames

`public java.util.Set getEntityNames()`

Returns the names of the entities.

Returns:
names of the entities

addEntity

`public void addEntity(Entity entity)`

Adds an entity to the model.

Parameters:
`entity`

clear

`public void clear()`

getEditor

`public ModelEditor getEditor()`

(continued from last page)

setChanged

```
protected void setChanged()
```

ucot.model.object
Class ObjectAnalyzeModelEditor

```
java.lang.Object
   |
   +-ucot.model.object.ObjectAnalyzeModelEditor
```

All Implemented Interfaces:
java.io.Serializable, ModelEditor

---

public class **ObjectAnalyzeModelEditor**
extends java.lang.Object
implements ModelEditor, java.io.Serializable

Editor for the ObjectAnalyzeModel.
Author:
pajumasu

Fields

serialVersionUID

`public static final long serialVersionUID`

Constant value: **892359043723230577**

---

model

`ucot.model.object.ObjectAnalyzeModel model`

---

updations

`java.util.List updations`

Constructors

ObjectAnalyzeModelEditor

`public ObjectAnalyzeModelEditor(ObjectAnalyzeModel model)`

Methods

sendUpdation

`private void sendUpdation(Updation updation)`

clearModel

```
public void clearModel()
```

getEntityNames

```
public java.util.Set getEntityNames()
```

getEntityType

```
public java.lang.String getEntityType(java.lang.String name)
```

setEntityType

```
public void setEntityType(java.lang.String name,
         java.lang.String entityType)
```

addEntity

```
public void addEntity(java.lang.String name)
```

removeEntity

```
public void removeEntity(java.lang.String name)
```

changeEntityName

```
public void changeEntityName(java.lang.String oldName,
         java.lang.String newName)
  throws NoSuchEntityException
```

containsEntity

```
public boolean containsEntity(java.lang.String entityName)
```

addParent

```
public void addParent(java.lang.String entityName,
         java.lang.String parentEntity)
  throws NoSuchEntityException
```

removeParent

```
public void removeParent(java.lang.String entityName,
         java.lang.String parentEntity)
  throws AnalyzeModelException
```

removeAllParents

```
public void removeAllParents(java.lang.String entityName)
  throws NoSuchEntityException
```

getParents

```
public java.util.Set getParents(java.lang.String entityName)
```

addChild

```
public void addChild(java.lang.String entityName,
         java.lang.String childEntity)
  throws NoSuchEntityException
```

removeChild

```
public void removeChild(java.lang.String entityName,
         java.lang.String childEntity)
  throws AnalyzeModelException
```

removeAllChildren

```
public void removeAllChildren(java.lang.String entityName)
  throws AnalyzeModelException
```

getChildren

```
public java.util.Set getChildren(java.lang.String entityName)
```

addMethod

```
public void addMethod(java.lang.String entityName,
         java.lang.String methodName)
  throws NoSuchEntityException
```

(continued from last page)

removeMethod

```
public void removeMethod(java.lang.String entityName,
         java.lang.String methodName)
  throws NoSuchEntityException
```

changeMethodName

```
public void changeMethodName(java.lang.String entityName,
         java.lang.String methodOldName,
         java.lang.String methodNewName)
  throws AnalyzeModelException
```

containsMethod

```
public boolean containsMethod(java.lang.String entityName,
         java.lang.String methodName)
```

getMethodNames

```
public java.util.Set getMethodNames(java.lang.String entityName)
```

getEntitiesInfluencedByMethod

```
public java.util.Set getEntitiesInfluencedByMethod(java.lang.String entityName,
         java.lang.String methodName)
```

addEntityInfluenceByMethod

```
public void addEntityInfluenceByMethod(java.lang.String entityName,
         java.lang.String methodName,
         java.lang.String influencedEntity)
  throws AnalyzeModelException
```

removeEntityInfluenceByMethod

```
public void removeEntityInfluenceByMethod(java.lang.String entityName,
         java.lang.String methodName,
         java.lang.String influencedEntity)
  throws AnalyzeModelException
```

addAttribute

```
public void addAttribute(java.lang.String entityName,
         java.lang.String attributeName)
  throws NoSuchEntityException
```

(continued from last page)

removeAttribute

```
public void removeAttribute(java.lang.String entityName,
         java.lang.String attributeName)
  throws AnalyzeModelException
```

containsAttribute

```
public boolean containsAttribute(java.lang.String entityName,
         java.lang.String attributeName)
  throws NoSuchEntityException
```

getAttributeFromCardinal

```
public java.lang.String getAttributeFromCardinal(java.lang.String entityName,
         java.lang.String attributeName)
  throws AnalyzeModelException
```

getAttributeToCardinal

```
public java.lang.String getAttributeToCardinal(java.lang.String entityName,
         java.lang.String attributeName)
  throws AnalyzeModelException
```

setAttributeFromCardinal

```
public void setAttributeFromCardinal(java.lang.String entityName,
         java.lang.String attributeName,
         java.lang.String cardinal)
  throws AnalyzeModelException
```

setAttributeToCardinal

```
public void setAttributeToCardinal(java.lang.String entityName,
         java.lang.String attributeName,
         java.lang.String cardinal)
  throws AnalyzeModelException
```

getAttributeNames

```
public java.util.Set getAttributeNames(java.lang.String entityName)
```

(continued from last page)

merge

```
public void merge(AnalyzeModel fromModel)
  throws AnalyzeModelException
```

getUpdations

```
public java.util.List getUpdations()
```

stepBack

```
public void stepBack(int steps)
```

execute

```
public boolean execute(Updation updation)
  throws AnalyzeModelException
```

mergeEntity

```
public void mergeEntity(java.lang.String targetEntityName,
         java.util.Set mergeSet)
```

readySignal

```
public void readySignal()
```

updationStartedSignal

```
public void updationStartedSignal()
```

saveUpdationsToFile

```
public void saveUpdationsToFile(java.io.File target)
  throws java.io.IOException
```

Package
ucot.output

Classes related to outputting (e.g. saving to a file) AnalyzeModel.

ucot.output
Class DummyOutput

```
java.lang.Object
  |
  +-ucot.output.DummyOutput
```

All Implemented Interfaces:
OutputInterface

public class **DummyOutput**
extends java.lang.Object
implements OutputInterface

Dummy Output that does nothing
Author:
tujupien

Constructors

DummyOutput

```
public DummyOutput()
```

Methods

output

```
public void output(AnalyzeModel analyzeModel,
                   java.net.URL url)
    throws java.io.IOException
```

getName

```
public java.lang.String getName()
```

getProperties

```
public java.util.Properties getProperties()
```

setProperties

```
public void setProperties(java.util.Properties properties)
```

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

saveProperties

```
public void saveProperties()
  throws java.io.IOException
```

loadProperties

```
public void loadProperties()
  throws java.io.IOException
```

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

getFileExtension

```
public java.lang.String getFileExtension()
```

ucot.output
Class GXLOutput

```
java.lang.Object
   |
   +-ucot.ModuleProperties
      |
      +-ucot.output.GXLOutput
```

All Implemented Interfaces:
OutputInterface, ModulePropertyInterface

public class **GXLOutput**
extends ModuleProperties
implements ModulePropertyInterface, OutputInterface

GXL output adapter
Author:
pajumasu

Fields

name

`public static final java.lang.String name`

Constant value: **`GXL`**

DEFAULT_FILE_EXTENSION

`public static final java.lang.String DEFAULT_FILE_EXTENSION`

Constant value: **`gxl`**

ATTRIBUTE_TYPE

`public static final java.lang.String ATTRIBUTE_TYPE`

Constant value: **`type`**

ATTRIBUTE_NAME

`public static final java.lang.String ATTRIBUTE_NAME`

Constant value: **`name`**

ATTRIBUTE_TYPE_NAME

`public static final java.lang.String ATTRIBUTE_TYPE_NAME`

Constant value: **`attribute`**

METHOD_TYPE_NAME

```
public static final java.lang.String METHOD_TYPE_NAME
```

Constant value: **method**

PARENT_TYPE_NAME

```
public static final java.lang.String PARENT_TYPE_NAME
```

Constant value: **parent**

logger

```
private java.util.logging.Logger logger
```

Constructors

GXLOutput

```
public GXLOutput()
```

Methods

output

```
public void output(AnalyzeModel analyzeModel,
         java.net.URL url)
  throws java.io.IOException
```

getName

```
public java.lang.String getName()
```

main

```
public static void main(java.lang.String[] args)
  throws java.lang.Throwable
```

Test program used while programming this class

Parameters:
`args`

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

(continued from last page)

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

getFileExtension

```
public java.lang.String getFileExtension()
```

ucot.output
Interface OutputInterface

All Superinterfaces:
    ModulePropertyInterface

All Known Implementing Classes:
    DummyOutput, GXLOutput

---

public interface **OutputInterface**
extends ModulePropertyInterface

Interface for outputs used by UCOT. Outputs store analyzemodel to a given url. Each Output component stores analyze model in a different way/format.
Author:
    vevijopi

Methods

output

```
public void output(AnalyzeModel analyzeModel,
          java.net.URL url)
  throws java.io.IOException
```

Outputs analyze model.

Parameters:
`analyzeModel` - Analyze model to output.

---

getName

```
public java.lang.String getName()
```

Returns adapter's name.

Returns:
Adapter's name.

---

getFileExtension

```
public java.lang.String getFileExtension()
```

Returns the output's valid file extension without the leading dot.

Returns:
Adapter's accepted file extension.

Package
ucot.parser

Classes related to parsing read usecases.

ucot.parser
Class DummyParser

```
java.lang.Object
  |
  +-ucot.ModuleProperties
      |
      +-ucot.parser.DummyParser
```

All Implemented Interfaces:
ParserInterface, ModulePropertyInterface

public class **DummyParser**
extends ModuleProperties
implements ModulePropertyInterface, ParserInterface

DummyParser which does absolutely nothing but helps avoiding null values in parser variables.
Author:
UCOT

Constructors

DummyParser

```
public DummyParser()
```

Methods

parse

```
public ParsedData parse(UseCase useCase)
  throws java.lang.Exception
```

getName

```
public java.lang.String getName()
```

ucot.parser Class Link

```
java.lang.Object
   |
   +-ucot.parser.Link
```

public class **Link**
extends java.lang.Object

Named link that points to some word. This class is used to mark subjects and objects and other structural information in sentence.
See Also:
`Word`, `Sentence`

Fields

name

```
private java.lang.String name
```

The name of the link.

See Also:
`ParserInterface.SUBJECT`
`ParserInterface.OBJECT`

to

```
private ucot.parser.Word to
```

The target word this class poitns to.

Constructors

Link

```
public Link(java.lang.String name,
            Word to)
```

Creates new named link that points to given word.

Parameters:
`name` - The name of the link.
`to` - The target word.

Methods

getName

```
public java.lang.String getName()
```

Returns the name of the link.

(continued from last page)

Returns:
The name of the link.

setName

`public void setName(java.lang.String name)`

Sets the name of the link.

Parameters:
`name`

getTo

`public Word getTo()`

Returns the target of the link.

Returns:
The target of the link.

setTo

`public void setTo(Word to)`

Sets the target of the link.

Parameters:
`to` - The target for the link.

toString

`public java.lang.String toString()`

ucot.parser
Class ParsedData

```
java.lang.Object
  |
  +-ucot.parser.ParsedData
```

public class **ParsedData**
extends java.lang.Object

Parsed data is returned by parser. This contains a list of sentences (containing words) that have been parsed from usecase.
Author:
pajumasu

Fields

sentences

`private java.util.List sentences`

Sentences parsed from usecase

usecase

`private ucot.input.UseCase usecase`

Usecase this ParsedData was created from

Constructors

ParsedData

`public ParsedData()`

Methods

setUseCase

`public void setUseCase(UseCase a)`

getUseCase

`public UseCase getUseCase()`

return the usecase that this parsedData was created from

Returns:
the usecase that this parsedData was created from

addSentence

`public void **addSentence**(Sentence sentence)`

Adds a new sentence

Parameters:
`sentence` - sentence to add

getSentences

`public java.util.List getSentences()`

Returns a list of sentences this parsedData contains

Returns:
a list of sentences this parsedData contains

toString

`public java.lang.String toString()`

getOriginal

`public java.lang.String getOriginal()`

Returns the text of the original use case.

Returns:
text of the original use case

ucot.parser
Interface ParserInterface

All Superinterfaces:
ModulePropertyInterface

All Known Implementing Classes:
DummyParser, SimpleParser

---

public interface **ParserInterface**
extends ModulePropertyInterface

Interface for parsers within this program. Parsers are given usecase and they return ParsedData

---

Fields

NOUN

```
public static final java.lang.String NOUN
```

Constant value: **`noun`**

VERB

```
public static final java.lang.String VERB
```

Constant value: **`verb`**

SUBJECT

```
public static final java.lang.String SUBJECT
```

Constant value: **`subject`**

OBJECT

```
public static final java.lang.String OBJECT
```

Constant value: **`object`**

Methods

parse

```
public ParsedData parse(UseCase useCase)
  throws java.lang.Exception
```

Parses given usecase and returns parsed data as a ParsedData object.

Parameters:
`useCase` - Use case to parse.

(continued from last page)

Returns:
Parsed data as a ParsedData object.

getName

```
public java.lang.String getName()
```

Returns adapter's name.

Returns:
Adapter's name.

ucot.parser
Class Sentence

```
java.lang.Object
  |
  +-ucot.parser.Sentence
```

public class **Sentence**
extends java.lang.Object

Contains information and words of one sentence. Sentence is builded from a list of words and words cary informaton about their part in the sentence and what part of the speech they represents etc.

Author:
panu

Fields

words

`private java.util.List words`

Constructors

Sentence

`public Sentence()`

Methods

addWord

`public void addWord(Word word)`

Adds the given `Word` into this `Sentence`.

Parameters:
`word` - `Word` to add.

getWords

`public java.util.List getWords()`

Returns all `Words` held by this `Sentence` in a `List`.

Returns:
`List` of `Words` of this `Sentence`.

toString

`public java.lang.String` **`toString`**`()`

Returns textual representation of this `Sentence`.

The `String` returned looks like:

```
{SENTENCE: (word 1)(word 2)(word 3)...}
```

ucot.parser
Class SimpleParser

```
java.lang.Object
  |
  +-ucot.ModuleProperties
      |
      +-ucot.parser.SimpleParser
```

All Implemented Interfaces:
ParserInterface, ModulePropertyInterface

public class **SimpleParser**
extends ModuleProperties
implements ModulePropertyInterface, ParserInterface

Simple parser. Expects that the usecase steps consists of three parts separated with ",". First part is noun, second is verb and the third is noun.
Author:
pajumasu

Constructors

SimpleParser

`public SimpleParser()`

Methods

parse

`public ParsedData parse(UseCase useCase)`

getName

`public java.lang.String getName()`

toString

`public java.lang.String toString()`

loadDefaultProperties

`public java.util.Properties loadDefaultProperties()`

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

ucot.parser
Class Word

```
java.lang.Object
   |
   +-ucot.parser.Word
```

public class **Word**
extends java.lang.Object

"Word" of a sentence. It does not nessesary contain a single word but can also contain for example an noun phrase (blak car). `Word` can contain links to other `Word` and they are used to model depencies between different parts of sentence.

See Also:
`Sentence`

Fields

original

```
private java.lang.String original
```

Original form of the word this object is carrying.

basicForm

```
private java.lang.String basicForm
```

Basic form of the word this object is carrying.

wordClass

```
private java.lang.String wordClass
```

Class of the word.

See Also:
`ParserInterface.NOUN`
`ParserInterface.VERB`

links

```
private java.util.List links
```

The links from this `Word` to other words.

Constructors

Word

```
public Word(java.lang.String basicForm,
            java.lang.String wordClass)
```

(continued from last page)

Constructs the word object.

Parameters:
`basicForm` - The basic form of the word.
`wordClass` - The class of the word.

Methods

addLink

```
public void addLink(Link l)
```

Adds link..

Parameters:
`l` - The link to be added.

getLinks

```
public java.util.List getLinks()
```

Returns all the links. Modifying the link list does not change the true linkages.

Returns:
List of links.

getLinks

```
public java.util.List getLinks(java.lang.String name)
```

Returns links of spesific name. Modification of the returned list does not change the true linkages.

Parameters:
`name` - The link name we are interested in.

Returns:
The list of the links called by the given name.

getLink

```
public Link getLink(java.lang.String name)
```

Returns the first link called by spesific name-

Parameters:
`name` - The name

Returns:
The first link called by the given name.

getLinked

```
public Word getLinked(java.lang.String name)
```

Returns the word that is refered by the first link called by the given name.

Parameters:
`name` - The name.

Returns:
The word of refered from the first link.

RemoveLink

```
public boolean RemoveLink(Link l)
```

Removes link from this word.

Parameters:
`l` - The link to be removed.

Returns:
True if the link was contained by this word.

getBasicForm

```
public java.lang.String getBasicForm()
```

Returns the basic form of the word.

Returns:
the basic form of the word.

setBasicForm

```
public void setBasicForm(java.lang.String basicForm)
```

Sets the basic form of the word.

Parameters:
`basicForm` - The basic form.

getWordClass

```
public java.lang.String getWordClass()
```

Returns the word's class (part of the speech: noun, verb and so on).

(continued from last page)

Returns:
The word's class.

setWordClass

`public void **setWordClass**(java.lang.String wordClass)`

Sets the words class (part of the speech: noun, verb and so on).

Parameters:
`wordClass` - The word's class.

toString

`public java.lang.String **toString**()`

getOriginal

`public java.lang.String **getOriginal**()`

Returns the original form of the word.

Returns:
The original form of the word.

setOriginal

`public void **setOriginal**(java.lang.String original)`

Sets the original form of the word.*

Package
ucot.ui

Userinterface.

ucot.ui
Class DummyUI

```
java.lang.Object
  |
  +-ucot.ModuleProperties
      |
      +-ucot.ui.DummyUI
```

All Implemented Interfaces:
UIInterface, ModulePropertyInterface

public class **DummyUI**
extends ModuleProperties
implements ModulePropertyInterface, UIInterface

DummyUI which does absolutely nothing but helps avoiding null values in UI variables. Errors and warnings do get logged.

Author:
tujupien

Fields

DEFAULT_ERROR_TITLE

`public static final java.lang.String` **`DEFAULT_ERROR_TITLE`**

DEFAULT_WARNING_TITLE

`public static final java.lang.String` **`DEFAULT_WARNING_TITLE`**

logger

`private java.util.logging.Logger` **`logger`**

Constructors

DummyUI

`public` **`DummyUI`**`()`

Methods

(continued from last page)

exportDone

```
public void exportDone()
```

analyzeModelLoaded

```
public void analyzeModelLoaded()
```

useCasesLoaded

```
public void useCasesLoaded()
```

useCaseAdded

```
public void useCaseAdded(int foundEntities,
         int addedEntities)
```

setControlInterface

```
public void setControlInterface(ControlInterface a)
```

printError

```
public void printError(java.lang.String errorMessage,
         java.lang.String errorTitle)
```

printError

```
public void printError(java.lang.String errorMessage)
```

printWarning

```
public void printWarning(java.lang.String warningMessage,
         java.lang.String warningTitle)
```

printWarning

```
public void printWarning(java.lang.String warningMessage)
```

(continued from last page)

getProgressBar

```
public ProgressBarInterface getProgressBar()
```

ucot.ui
Interface ProgressBarInterface

All Known Implementing Classes:
ProgressBarDialog, DummyProgressBar

public interface **ProgressBarInterface**
extends

Interface for a progress bar used in core component to indicate the status of parsing and running the heuristic.

This interface defines all methods required for UCOT core to show progress bars for different slow operations. It is up to the user interface itself to decide whether the showing of the progress bar should disable other functionality of user interface or not, but in either case the progress bar is always spawned from a separate thread in the UCOT core. This way the total jamming of the user interface during slow operations is avoided.

Author:
tujupien

Methods

getMaximum

`public int getMaximum()`

Gets the maximum value for the progress bar.

Returns:
The maximum value for the progress bar.

getMinimum

`public int getMinimum()`

Gets the minimum value for the progress bar.

Returns:
The minimum value for the progress bar.

getPercentageComplete

`public double getPercentageComplete()`

Gets the current percentage completed. Which is `getValue() / (getMaximum() - getMinimum())`.

Returns:
Current percentage completed.

getString

`public java.lang.String getString()`

Gets the current action description.

Returns:
Action description.

getValue

`public int getValue()`

Gets the current value of the progress bar.

Returns:
Current value.

setMaximum

`public void setMaximum(int maximum)`

Sets the maximum value of the progress bar.

Parameters:
`maximum` - Maximum value of the progress bar.

setMinimum

`public void setMinimum(int minimum)`

Sets the minimum value of the progress bar.

Parameters:
`minimum` - Minimum value of the progress bar.

setString

`public void setString(java.lang.String string)`

Sets the action description string.

Parameters:
`string` - Current action description.

setValue

`public void setValue(int value)`

(continued from last page)

Sets the current value of the progress bar.

Parameters:
`value` - Current value.

setVisible

```
public void setVisible(boolean visible)
```

Method for setting the progress bar (dialog) visible or hiding it.

Parameters:
`visible` - Indicates wether the progress bar is visible or invisible.

ucot.ui
Interface UIInterface

All Superinterfaces:
ModulePropertyInterface

All Known Implementing Classes:
GraphicalUI, DummyUI

public interface **UIInterface**
extends ModulePropertyInterface

Interface for the UI. A means for the core component to notify UI that certain actions have been taken. For example that file has been loaded, use cases have been loaded or the `ControlInterface` has been changed.

In practice any user interface designed for UCOT program should implement this interface because these methods are used for the interclass communication and the core component uses these methods to control the user interface on some level and to inform user about certain things happening on a lower level of the program.

Author:
vevijopi, tujupien

Methods

exportDone

```
public void exportDone()
```

Method for signaling the user interface that the analyze model has been (successfully) exported.

analyzeModelLoaded

```
public void analyzeModelLoaded()
```

Method for signaling the user interface that the analyze model has been (successfully) loaded.

useCasesLoaded

```
public void useCasesLoaded()
```

Method for signaling user interface that use cases have been (successfully) loaded from file.

useCaseAdded

```
public void useCaseAdded(int foundEntities,
        int addedEntities)
```

(continued from last page)

Core signals user interface that usecases have been parsed, ran heuristic on and been added to given analyze model.

Parameters:
`foundEntities` - How many entities the parser found.
`addedEntities` - How many entities were added.

setControlInterface

```
public void setControlInterface(ControlInterface a)
```

Set a new control interface for the user interface to use.

Parameters:
`a` - `ControlInterface` to the UCOT core.

printError

```
public void printError(java.lang.String errorMessage,
         java.lang.String errorTitle)
```

Prints an error message to the screen.

Parameters:
`errorMessage` - Description of the error.
`errorTitle` - Title of the dialog.

printError

```
public void printError(java.lang.String errorMessage)
```

printWarning

```
public void printWarning(java.lang.String warningMessage,
         java.lang.String warningTitle)
```

Prints a warning to the screen.

Parameters:
`warningMessage` - Description of the warning.
`warningTitle` - Title of the dialog.

printWarning

```
public void printWarning(java.lang.String warningMessage)
```

(continued from last page)

getProgressBar

`public ProgressBarInterface getProgressBar()`

Method for getting a new progressbar for showing the current progress status to the user and halting all other usage of the model editor.

Returns:
ProgressBar interface to the progress bar.

ucot.ui
Interface UseCasePanelInterface

All Known Implementing Classes:
SimpleUseCasePanel

public interface **UseCasePanelInterface**
extends

Interface for the component which shows use case steps from given `UseCase`. User interface should use this interface to control the view of the currently selected use case.
Author:
ilanliuk

Methods

showUseCase

`public void **showUseCase**(UseCase usecase)`

Method for giving the panel `UseCase` to show.

Parameters:
`usecase` - `UseCase` to show.

Clear

`public void **Clear**()`

Clears the use case from the panel

refresh

`public void **refresh**()`

Call to notify panel to refresh itself.

Package
ucot.ui.gui

Graphical userInterface and classes related to it.

ucot.ui.gui
Class ChoosedFile

```
java.lang.Object
   |
   +-ucot.ui.gui.ChoosedFile
```

public class **ChoosedFile**
extends java.lang.Object

This class is used to transfer information from file dialog.
Author:
pajumasu

Fields

filefilter

```
public ucot.utils.CustomFileFilter filefilter
```

url

```
public java.net.URL url
```

Constructors

ChoosedFile

```
public ChoosedFile(java.net.URL url,
                   CustomFileFilter filefilter)
```

ucot.ui.gui
Class GraphicalUI

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-java.awt.Window
            |
            +-java.awt.Frame
               |
               +-javax.swing.JFrame
                  |
                  +-ucot.ui.gui.GraphicalUI
```

All Implemented Interfaces:
ModulePropertyInterface, java.util.Observer, UIInterface, java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, java.awt.MenuContainer, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

public class **GraphicalUI**
extends javax.swing.JFrame
implements javax.swing.WindowConstants, javax.accessibility.Accessible, javax.swing.RootPaneContainer, java.awt.MenuContainer, javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, UIInterface, java.util.Observer, ModulePropertyInterface

Simple graphical user interface for UCOT program which uses dot (in DotPanel) to draw graphics. This class is inherited from JFrame and implements the UIInterface defined in ucot.ui package.
Author:
ilanliuk, tujupien, pajumasu

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`-4246720431258449398`**

core

`private ucot.core.ControlInterface` **`core`**

menuListener

`private java.awt.event.ActionListener` **`menuListener`**

windowListener

`private java.awt.event.WindowListener` **`windowListener`**

analyzeModelTree

`private ucot.ui.gui.tree.analyzemodeltree.AnalyzeModelTree` **`analyzeModelTree`**

modelTreeScrollPane

`private javax.swing.JScrollPane` **`modelTreeScrollPane`**

useCaseTree

`private ucot.ui.gui.tree.usecasetree.UseCaseTree` **`useCaseTree`**

useCaseTreeScrollPane

`private javax.swing.JScrollPane` **`useCaseTreeScrollPane`**

menu

`private javax.swing.JMenuBar` **`menu`**

dotPanel

`private ucot.ui.gui.dot.DotPanel` **`dotPanel`**

dotScrollPane

`private javax.swing.JScrollPane` **`dotScrollPane`**

dotTimer

`private javax.swing.JLabel` **`dotTimer`**

useCasePanel

`private ucot.ui.UseCasePanelInterface` **`useCasePanel`**

(continued from last page)

useCasePanelScrollPane

`private javax.swing.JScrollPane` **`useCasePanelScrollPane`**

statusbar

`private ucot.ui.gui.Statusbar` **`statusbar`**

entityPropertiesDialog

`private ucot.ui.gui.dialog.EntityPropertiesDialog` **`entityPropertiesDialog`**

progressBarDialog

`private ucot.ui.gui.dialog.ProgressBarDialog` **`progressBarDialog`**

currentFile

`private java.net.URL` **`currentFile`**

changesMade

`private boolean` **`changesMade`**

changesIndicator

`private javax.swing.JLabel` **`changesIndicator`**

logger

`private java.util.logging.Logger` **`logger`**

propertiesURL

`protected java.net.URL` **`propertiesURL`**

(continued from last page)

properties

`protected java.util.Properties` **`properties`**

entityTypes

`public java.util.Set` **`entityTypes`**

SERIALIZATION_DESCRIPTION

`protected static java.lang.String` **`SERIALIZATION_DESCRIPTION`**

SERIALIZATION_EXTENSIONS

`protected static java.lang.String` **`SERIALIZATION_EXTENSIONS`**

PROGRAM_TITLE

`protected static java.lang.String` **`PROGRAM_TITLE`**

DEFAULT_ERROR_TITLE

`protected static final java.lang.String` **`DEFAULT_ERROR_TITLE`**

DEFAULT_QUESTION_TITLE

`protected static final java.lang.String` **`DEFAULT_QUESTION_TITLE`**

DEFAULT_WARNING_TITLE

`protected static final java.lang.String` **`DEFAULT_WARNING_TITLE`**

EXPORT_ALL_GRAPH_FILENAME

`public static final java.lang.String` **`EXPORT_ALL_GRAPH_FILENAME`**

Constant value: **`graph`**

EXPORT_ALL_DATA_FILENAME

`public static final java.lang.String` **`EXPORT_ALL_DATA_FILENAME`**

(continued from last page)

Constant value: **data**

EXPORT_ALL_SERIALIZATION_FILENAME

`public static final java.lang.String` **`EXPORT_ALL_SERIALIZATION_FILENAME`**

Constant value: **analyze_model**

EXPORT_ALL_LOG_FILENAME

`public static final java.lang.String` **`EXPORT_ALL_LOG_FILENAME`**

Constant value: **modification_log**

EXPORT_ALL_DOT_FILENAME

`public static final java.lang.String` **`EXPORT_ALL_DOT_FILENAME`**

Constant value: **graph.dot**

PROPERTY_DOT_PATH

`public static final java.lang.String` **`PROPERTY_DOT_PATH`**

Constant value: **DOT_PATH**

PROPERTY_EPS_TO_PDF_PATH

`public static final java.lang.String` **`PROPERTY_EPS_TO_PDF_PATH`**

Constant value: **EPS_TO_PDF_PATH**

PROPERTY_TEMP_INPUT_FILE

`public static final java.lang.String` **`PROPERTY_TEMP_INPUT_FILE`**

Constant value: **TEMP_INPUT_FILE**

PROPERTY_TEMP_OUTPUT_FILE

`public static final java.lang.String` **`PROPERTY_TEMP_OUTPUT_FILE`**

Constant value: **TEMP_OUTPUT_FILE**

PROPERTY_TEMP_EPS_FILE

`public static final java.lang.String` **`PROPERTY_TEMP_EPS_FILE`**

Constant value: **TEMP_EPS_FILE**

PROPERTY_DOT_HIGHLIGHT_COLOR

`public static final java.lang.String` **`PROPERTY_DOT_HIGHLIGHT_COLOR`**

Constant value: **`DOT_HIGHLIGHT_COLOR`**

PROPERTY_DOT_COLOR

`public static final java.lang.String` **`PROPERTY_DOT_COLOR`**

Constant value: **`DOT_COLOR`**

PROPERTY_ENTITY_TYPES

`public static final java.lang.String` **`PROPERTY_ENTITY_TYPES`**

Constant value: **`ENTITY_TYPES`**

PROPERTY_GRAPH_FONT_SIZE

`public static final java.lang.String` **`PROPERTY_GRAPH_FONT_SIZE`**

Constant value: **`GRAPH_FONT_SIZE`**

CLEAR_MODEL_QUESTION_TITLE

`protected static java.lang.String` **`CLEAR_MODEL_QUESTION_TITLE`**

CLEAR_MODEL_QUESTION

`protected static java.lang.String` **`CLEAR_MODEL_QUESTION`**

NEW_MODEL_QUESTION_TITLE

`protected static java.lang.String` **`NEW_MODEL_QUESTION_TITLE`**

NEW_MODEL_QUESTION

`protected static java.lang.String` **`NEW_MODEL_QUESTION`**

QUIT_QUESTION_TITLE

`protected static java.lang.String` **`QUIT_QUESTION_TITLE`**

QUIT_QUESTION

`protected static java.lang.String` **`QUIT_QUESTION`**

SAVE_BEFORE_QUIT_QUESTION_TITLE

`protected static java.lang.String` **`SAVE_BEFORE_QUIT_QUESTION_TITLE`**

SAVE_BEFORE_QUIT_QUESTION

`protected static java.lang.String` **`SAVE_BEFORE_QUIT_QUESTION`**

LOG_FILE_LOAD_TRIGGERED

`protected static java.lang.String` **`LOG_FILE_LOAD_TRIGGERED`**

LOG_QUIT_PROGRAM_TRIGGERED

`protected static java.lang.String` **`LOG_QUIT_PROGRAM_TRIGGERED`**

LOG_QUIT_PROGRAM

`protected static java.lang.String` **`LOG_QUIT_PROGRAM`**

LOG_CLEAR_MODEL

`protected static java.lang.String` **`LOG_CLEAR_MODEL`**

LOG_SETTINGS_TRIGGERED

`protected static java.lang.String` **`LOG_SETTINGS_TRIGGERED`**

LOG_EXPORT_AS_IMAGE_TRIGGERED

`protected static java.lang.String` **`LOG_EXPORT_AS_IMAGE_TRIGGERED`**

LOG_EXPORT_DONE_MESSAGE

`protected static java.lang.String` **`LOG_EXPORT_DONE_MESSAGE`**

(continued from last page)

LOG_ANALYZE_MODEL_LOADED

`protected static java.lang.String` **`LOG_ANALYZE_MODEL_LOADED`**

LOG_USE_CASES_LOADED

`protected static java.lang.String` **`LOG_USE_CASES_LOADED`**

LOG_USE_CASE_LOADING_EXCEPTION

`protected static java.lang.String` **`LOG_USE_CASE_LOADING_EXCEPTION`**

LOG_USE_CASES_ADDED

`protected static java.lang.String` **`LOG_USE_CASES_ADDED`**

FILE_MENU_CAPTION

`protected static java.lang.String` **`FILE_MENU_CAPTION`**

NEW_MENU_CAPTION

`protected static java.lang.String` **`NEW_MENU_CAPTION`**

OPEN_MENU_CAPTION

`protected static java.lang.String` **`OPEN_MENU_CAPTION`**

LOAD_MENU_CAPTION

`protected static java.lang.String` **`LOAD_MENU_CAPTION`**

SAVE_MENU_CAPTION

`protected static java.lang.String` **`SAVE_MENU_CAPTION`**

(continued from last page)

MODIFICATION_LOG_MENU_CAPTION

`protected static java.lang.String` **`MODIFICATION_LOG_MENU_CAPTION`**

SAVE_AS_MENU_CAPTION

`protected static java.lang.String` **`SAVE_AS_MENU_CAPTION`**

EXPORT_MENU_CAPTION

`protected static java.lang.String` **`EXPORT_MENU_CAPTION`**

EXPORT_AS_IMAGE_MENU_CAPTION

`protected static java.lang.String` **`EXPORT_AS_IMAGE_MENU_CAPTION`**

EXPORT_ALL_MENU_CAPTION

`protected static java.lang.String` **`EXPORT_ALL_MENU_CAPTION`**

CLEAR_MODEL_MENU_CAPTION

`protected static java.lang.String` **`CLEAR_MODEL_MENU_CAPTION`**

QUIT_MENU_CAPTION

`protected static java.lang.String` **`QUIT_MENU_CAPTION`**

PROGRAM_MENU_CAPTION

`protected static java.lang.String` **`PROGRAM_MENU_CAPTION`**

SETTINGS_MENU_CAPTION

`protected static java.lang.String` **`SETTINGS_MENU_CAPTION`**

DOT_USE_HORIZONGAL_LAYOUTING_CAPTION

`protected static java.lang.String` **`DOT_USE_HORIZONGAL_LAYOUTING_CAPTION`**

(continued from last page)

HELP_MENU_CAPTION

`protected static java.lang.String` **`HELP_MENU_CAPTION`**

ABOUT_MENU_CAPTION

`protected static java.lang.String` **`ABOUT_MENU_CAPTION`**

PROGRESS_BAR_DIALOG_TITLE

`protected static java.lang.String` **`PROGRESS_BAR_DIALOG_TITLE`**

MODIFIED_STRING

`protected static java.lang.String` **`MODIFIED_STRING`**

UNMODIFIED_STRING

`protected static java.lang.String` **`UNMODIFIED_STRING`**

NEW_FILE_MENU_ACTION

`protected static java.lang.String` **`NEW_FILE_MENU_ACTION`**

MODIFICATION_LOG_MENU_ACTION

`protected static java.lang.String` **`MODIFICATION_LOG_MENU_ACTION`**

OPEN_FILE_MENU_ACTION

`protected static java.lang.String` **`OPEN_FILE_MENU_ACTION`**

LOAD_FILE_MENU_ACTION

`protected static java.lang.String` **`LOAD_FILE_MENU_ACTION`**

(continued from last page)

SAVE_FILE_MENU_ACTION

`protected static java.lang.String` **`SAVE_FILE_MENU_ACTION`**

SAVE_AS_FILE_MENU_ACTION

`protected static java.lang.String` **`SAVE_AS_FILE_MENU_ACTION`**

EXPORT_MENU_ACTION

`protected static java.lang.String` **`EXPORT_MENU_ACTION`**

EXPORT_ALL_MENU_ACTION

`protected static java.lang.String` **`EXPORT_ALL_MENU_ACTION`**

QUIT_MENU_ACTION

`protected static java.lang.String` **`QUIT_MENU_ACTION`**

CLEAR_MODEL_MENU_ACTION

`protected static java.lang.String` **`CLEAR_MODEL_MENU_ACTION`**

EXPORT_AS_IMAGE_MENU_ACTION

`protected static java.lang.String` **`EXPORT_AS_IMAGE_MENU_ACTION`**

SETTINGS_MENU_ACTION

`protected static java.lang.String` **`SETTINGS_MENU_ACTION`**

CHANGE_DOT_LAYOUTING_ACTION

`protected static java.lang.String` **`CHANGE_DOT_LAYOUTING_ACTION`**

ABOUT_MENU_ACTION

`protected static java.lang.String` **`ABOUT_MENU_ACTION`**

(continued from last page)

Constructors

GraphicalUI

`public GraphicalUI(ControlInterface controlInterface)`

Constuctor for Graphical UI. This constructor sets up the whole user interface and spawns it to the screen.

Methods

setChanged

`public void setChanged(boolean changesMade)`

Method for changing the changes made status. Practically anything that modifies the analyze model should call this method with parameter true, and only the save and new operations should call this method with parameter false.

Parameters:

`changesMade` - New status which indicates whether or not any changes have been made to the current analyze model since last save operation..

---

menuQuit

`private void menuQuit(boolean quitWithoutPrompting)`

Action performed: Quit is chosen from the File menu.

Parameters:

`quitWithoutPrompting` - Indicates wether the user really wants to quit without prompting another question to confirm that.

---

menuExportAll

`private void menuExportAll()`

Action performed: User clicks the export all menu item.

---

menuClearModel

`private void menuClearModel(boolean clearWithoutPrompting)`

Action performed: Clear model action is chosen from the File menu.

Parameters:

`clearWithoutPrompting`

menuNewModel

`private void menuNewModel()`

Action performed: User selects the new model option from the menu.

menuLoadUseCaseFile

`private void menuLoadUseCaseFile()`

Action performed: Open file is chosen from the File menu.

menuExport

`private void menuExport()`

Action performed: Export is chosen from the File menu.

menuModificationLog

`private void menuModificationLog()`

Action performed: Export is chosen from the File menu.

menuSettings

`private void menuSettings()`

Action performed: User wants to change program settings.

menuExportAsImage

`private void menuExportAsImage()`

Action performed: User wants to save picture of the model.

changeDotLayouting

`private void changeDotLayouting()`

Action performed: User changes the status of the horizontal layouting checkbox.

(continued from last page)

createWindowListener

```
private void createWindowListener()
```

This method creates and initializes all required window listeners for this graphical user interface.

createMenuListener

```
private void createMenuListener()
```

This method creates and initializes the menu listener for the program menu bar.

showAboutDialog

```
private void showAboutDialog()
```

Method for spawning the about UCOT dialog.

createMenu

```
private javax.swing.JMenuBar createMenu()
```

Method for creating a menu bar to the GUI.

Returns:
Program menu bar as a JMenuBar object.

exportDone

```
public void exportDone()
```

analyzeModelLoaded

```
public void analyzeModelLoaded()
```

useCasesLoaded

```
public void useCasesLoaded()
```

useCaseAdded

```
public void useCaseAdded(int foundEntities,
         int addedEntities)
```

setControlInterface

```
public void setControlInterface(ControlInterface a)
```

update

```
public void update(java.util.Observable o,
         java.lang.Object arg)
```

saveModel

```
public boolean saveModel(java.net.URL target)
```

This method serializes the current status of the project.

Parameters:
`target` - Path to the file used for saving. If this is null, then a save file dialog will be spawned.

Returns:
true if model was saved, false otherwhise.

parseEntityTypes

```
private java.util.Set parseEntityTypes()
```

This function parses the entity types from the property value and splits the separate types into a String vector.

Returns:
Vector containing the available entity types.

saveModel

```
public void saveModel()
```

Default save method. The previous file will be overwritten.

loadModel

```
public void loadModel()
```

Default method for loading an analyze model. The serialization file is defined by the user who gets an file open dialog on his face before loading takes place.

(continued from last page)

updateTypes

```
private void updateTypes()
```

---

getProgressBar

```
public ProgressBarInterface getProgressBar()
```

---

printError

```
public void printError(java.lang.String errorMessage)
```

---

printWarning

```
public void printWarning(java.lang.String warningMessage,
         java.lang.String warningTitle)
```

---

printWarning

```
public void printWarning(java.lang.String warningMessage)
```

---

printError

```
public void printError(java.lang.String errorMessage,
         java.lang.String errorTitle)
```

---

setDisabled

```
public void setDisabled(boolean disabled)
```

Sets the GUI's menus and other vital elements disabled. This is currently necessary because otherwice the user could screw up the whole model or the program by performing unexpected actions while analyzing for previous use cases or something similiar.

Parameters:

`disabled` - True if GUI should be disabled, false if GUI should be enabled.

---

getPropertiesURL

```
private java.net.URL getPropertiesURL()
```

Method for creating the URL from the properties file, which is the same as the class name with an .XML extension.

(continued from last page)

Returns:
URL to the properties file.

getProperties

```
public java.util.Properties getProperties()
```

setProperties

```
public void setProperties(java.util.Properties properties)
```

applyProperties

```
public void applyProperties()
  throws BadPropertyValueException
```

updateProperties

```
private void updateProperties()
```

Updates the properties base don the internal state of this object.

saveProperties

```
public void saveProperties()
  throws java.io.IOException
```

loadProperties

```
public void loadProperties()
  throws java.io.IOException
```

loadDefaultProperties

```
public java.util.Properties loadDefaultProperties()
```

getControlInterface

```
public ControlInterface getControlInterface()
```

Returns the `ControlInterface` this UI uses to control the program.

Returns:
The control interface.

getColorTheme

```
public DotColorTheme getColorTheme()
```

Returns dot panel's color theme.

Returns:
Dot panel's color theme

ucot.ui.gui
Class GUIUtils

```
java.lang.Object
  |
  +-ucot.ui.gui.GUIUtils
```

public class **GUIUtils**
extends java.lang.Object

Miscallaneous utils used by the grapical userinterface.
Author:
ilanliuk

Fields

DEFAULT_ERROR_HEADER

`protected static final java.lang.String DEFAULT_ERROR_HEADER`

Constant value: **Error**

DEFAULT_QUESTION_HEADER

`protected static final java.lang.String DEFAULT_QUESTION_HEADER`

Constant value: **Question**

DEFAULT_WARNING_HEADER

`protected static final java.lang.String DEFAULT_WARNING_HEADER`

Constant value: **Warning**

OK_CAPTION

`protected static final java.lang.String OK_CAPTION`

Constant value: **OK**

Constructors

GUIUtils

`public GUIUtils()`

Methods

(continued from last page)

showOpenFileDialog

```
public static java.net.URL showOpenFileDialog(java.awt.Component owner,
        java.util.Vector fileFilters,
        boolean acceptAllFileFiltersUsed)
```

Method for spawning a file chooser dialog.

Parameters:

`owner` - Parent of this dialog.

`fileFilters` - Allowed file extensions.

`acceptAllFileFiltersUsed` - Defines whether all file types are allowed or not when any file filters are manually defined.

Returns:

The selected file as an URL or null if no file was selected.

showOpenFileDialog

```
public static java.net.URL showOpenFileDialog(java.awt.Component owner,
        java.util.Vector fileFilters)
```

Default open dialog spawner which assumes that all file types are not required to be shown when any file filters are manually defined.

Parameters:

`owner` - Parent of this dialog.

`fileFilters` - Allowed file extensions.

Returns:

The selected file as an URL or null if no file was selected.

showSaveFileDialog

```
public static ChoosedFile showSaveFileDialog(java.awt.Component owner,
        java.util.Vector fileFilters,
        int fileSelectionMode)
```

Method for spawning a file save dialog.

Parameters:

`owner` - Parent of this dialog.

`fileFilters` - Allowed file extensions.

`fileSelectionMode` - defines wether the user can select files, directories or both.

Returns:

The object containing selections.

showSaveFileDialog

```
public static ChoosedFile showSaveFileDialog(java.awt.Component owner,
        java.util.Vector fileFilters)
```

(continued from last page)

Method for spawning a save file chooser dialog.

Parameters:

`owner` - Parent of this dialog.
`fileFilters` - Allowed file extensions.

Returns:

The selected file as an URL or null if no file was selected.

---

showDialog

```
public static int showDialog(java.awt.Window parent,
         int messageType,
         java.lang.String message,
         java.lang.String header,
         int options)
```

Dialog spawner for warnings, errors and questions.

Parameters:

`parent` - Parent `Window` for this dialog.
`messageType` - the type of message to be displayed: `ERROR_MESSAGE`, `INFORMATION_MESSAGE`, `WARNING_MESSAGE`, `QUESTION_MESSAGE`, or `PLAIN_MESSAGE`.
`message` - Message itself.
`header` - the `String` to display in the dialog's title bar
`options` - the options to display in the pane: `DEFAULT_OPTION`, `YES_NO_OPTION`, `YES_NO_CANCEL_OPTION, or` OK_CANCEL_OPTION.

Returns:

User's choice.

---

centerDialog

```
public static void centerDialog(javax.swing.JDialog dialog)
```

This method centers a given dialog to the center of the screen.

Parameters:

`dialog` - `JDialog` to be centered.

---

questionDialog

```
public static int questionDialog(java.awt.Window win,
         java.lang.String question)
```

Method for spawning question dialog.

Parameters:

`win` - Parent `Window` for this dialog.
`question` - The question `String` to show.

Returns:

User's choice.

printWarning

```
public static void printWarning(java.awt.Window win,
         java.lang.String warning,
         java.lang.String header)
```

Method for spawning warning dialog.

Parameters:

`win` - Parent `Window` for this dialog.
`warning` - The warning `String` to show.
`header` - the `String` to display in the dialog's title bar

printInfo

```
public static void printInfo(java.awt.Window win,
         java.lang.String info,
         java.lang.String header)
```

Method for spawning info dialog.

Parameters:

`win` - Parent `Window` for this dialog.
`info` - The info `String` to show.
`header` - the `String` to display in the dialog's title bar

printWarning

```
public static void printWarning(java.awt.Window win,
         java.lang.String warning)
```

Method for spawning warning dialog with dafault header `String`.

Parameters:

`win` - Parent `Window` for this dialog.
`warning` - The warning `String` to show.

printError

```
public static void printError(java.awt.Window win,
         java.lang.String errorMessage,
         java.lang.String header)
```

Method for spawning error dialog.

Parameters:

`win` - Parent `Window` for this dialog.
`errorMessage` - The error `String` to show.
`header` - the `String` to display in the dialog's title bar

(continued from last page)

printError

```
public static void printError(java.awt.Window win,
         java.lang.String errorMessage)
```

Method for spawning error dialog with default header `String`

Parameters:
`win` - Parent `Window` for this dialog.
`errorMessage` - The error `String` to show.

questionDialog

```
public static int questionDialog(java.awt.Window win,
         java.lang.String question,
         java.lang.String header,
         int options)
```

Method for spawning question dialog with wanted options.

Options available are: `DEFAULT_OPTION`, `YES_NO_OPTION`, `YES_NO_CANCEL_OPTION`, `or` OK_CANCEL_OPTION. Options are defined in `JOptionPane`-class.

Parameters:
`win` - Parent `Window` for this dialog.
`question` - The question `String` to show.
`header` - the `String` to display in the dialog's title bar
`options` - The options to display in the pane.

Returns:
User's choise.

See Also:
`javax.swing.JOptionPane`

questionDialog

```
public static int questionDialog(java.awt.Window win,
         java.lang.String question,
         java.lang.String header)
```

Method for spawning question dialog with `YES_NO_OPTION`-option.

Parameters:
`win` - Parent `Window` for this dialog.
`question` - The question `String` to show.
`header` - the `String` to display in the dialog's title bar

Returns:
User's choise.

(continued from last page)

createComboBoxCellEditor

```
public static javax.swing.DefaultCellEditor
createComboBoxCellEditor(java.lang.String[] values,
         java.lang.String self,
         java.lang.String selfPointer,
         boolean allowSelf,
         boolean addEmpty)
```

This method creates a new JComboBox Cell Editor for a JTable element from the given String array. Duplicate entries from given array will be filtered and the array is sorted to alphabetical order.

Parameters:

`values` - Available options in the JComboBox.
`allowSelf` - Allow self pointer in the list.
`self` - The name of the self object.
`selfPointer` - The name of the pointer used to point the self object.
`addEmpty` - Create an empty item to the beginning of the list.

Returns:

JComboBox table cell editor as a DefaultCellEditor.

ucot.ui.gui
Class ModificationLogWindow

```
java.lang.Object
  |
  +-java.awt.Component
      |
      +-java.awt.Container
          |
          +-java.awt.Window
              |
              +-java.awt.Frame
                  |
                  +-javax.swing.JFrame
                      |
                      +-ucot.ui.gui.ModificationLogWindow
```

All Implemented Interfaces:

java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, java.awt.MenuContainer, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

---

public class **ModificationLogWindow**
extends javax.swing.JFrame

This is simple window to show modification log for an analyzemodel.

Author:
pajumasu

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **-711720267848958080**

---

model

`ucot.model.AnalyzeModel model`

The model that we are interested in.

---

text

`javax.swing.JTextArea text`

Text area for textual information about the modifications.

---

last

`int last`

The last index of the updation element added to the text field. Using this we dont rewrite the whole text again when asked to update.

(continued from last page)

CLOSE_CAPTION

`static final java.lang.String` **`CLOSE_CAPTION`**

Constant value: **`Close`**

SAVE_AS_CAPTION

`static final java.lang.String` **`SAVE_AS_CAPTION`**

Constant value: **`Save as..`**

UPDATE_CAPTION

`static final java.lang.String` **`UPDATE_CAPTION`**

Constant value: **`Update`**

CLOSE_ACTION

`static final java.lang.String` **`CLOSE_ACTION`**

Constant value: **`CLOSE`**

SAVE_AS_ACTION

`static final java.lang.String` **`SAVE_AS_ACTION`**

Constant value: **`SAVE_AS`**

UPDATE_ACTION

`static final java.lang.String` **`UPDATE_ACTION`**

Constant value: **`UPDATE`**

TEXT_DESCRIPTION

`static final java.lang.String` **`TEXT_DESCRIPTION`**

Constant value: **`Text file`**

TEXT_EXTENSIONS

`static final java.lang.String` **`TEXT_EXTENSIONS`**

fileChooser

`javax.swing.JFileChooser` **`fileChooser`**

(continued from last page)

actionListener

`java.awt.event.ActionListener` **`actionListener`**

Constructors

ModificationLogWindow

`public` **`ModificationLogWindow`**`(AnalyzeModel model)`

Methods

update

`public void` **`update`**`()`

Updates the view using the current model.

update

`public void` **`update`**`(AnalyzeModel model)`

Updates the view.

Parameters:

`model` - The model whitch modifications we like to see.

close

`public void` **`close`**`()`

saveAs

`public boolean` **`saveAs`**`()`

ucot.ui.gui
Class SimpleUseCasePanel

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.SimpleUseCasePanel
```

All Implemented Interfaces:
UseCasePanelInterface, java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

public class **SimpleUseCasePanel**
extends javax.swing.JPanel
implements javax.accessibility.Accessible, java.io.Serializable, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, UseCasePanelInterface

Simple panel for showing usecase steps
Author:
ilanliuk

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`-6222831904122402064`**

usecase

`private ucot.input.UseCase` **`usecase`**

Constructors

SimpleUseCasePanel

`public` **`SimpleUseCasePanel`**`()`

Methods

(continued from last page)

showUseCase

`public void` **`showUseCase`**`(UseCase usecase)`

Sets the usecase for display.

Clear

`public void` **`Clear`**`()`

clears usecase from panel.

paintComponent

`protected void` **`paintComponent`**`(java.awt.Graphics g)`

refresh

`public void` **`refresh`**`()`

Repaints the panel to refresh view.

ucot.ui.gui
Class Statusbar

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.Statusbar
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

public class **Statusbar**
extends javax.swing.JPanel

Statusbar component for Swing. This class is inherited from JPanel and simply uses flow layout to insert components to the bar.
Author:
UCOT

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **-7570976997422611985**

Constructors

Statusbar

`public Statusbar()`

Default constructor for statusbar. This method initializes the statusbar with the correct layout.

Methods

addColumn

`public void addColumn(javax.swing.JComponent component)`

Method for adding columns to the statusbar.

Parameters:
`component` - Component to be added to the statusbar.

Package
ucot.ui.gui.dialog

Dialogs used by the gui.

ucot.ui.gui.dialog
Class AboutDialog

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-java.awt.Window
            |
            +-java.awt.Dialog
               |
               +-javax.swing.JDialog
                  |
                  +-ucot.ui.gui.dialog.AboutDialog
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

---

public class **AboutDialog**
extends javax.swing.JDialog

About dialog for UCOT program. This dialog is inherited from the Swing's JDialog class and can be constructed and spawned to the screen with a static method: showDialog(JFrame owner) This dialog is modal and will halt all other execution until it's closed.
Author:
tujupien

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **-6739939273105582391**

DIALOG_HEADER

`protected static java.lang.String` **`DIALOG_HEADER`**

TEXT

`protected static java.lang.String` **`TEXT`**

CLOSE_BUTTON_CAPTION

`protected static java.lang.String` **`CLOSE_BUTTON_CAPTION`**

CLOSE_BUTTON_ACTION

`protected static java.lang.String` **`CLOSE_BUTTON_ACTION`**

buttonListener

`protected java.awt.event.ActionListener` **`buttonListener`**

Action listener for all buttons on this about dialog.

Constructors

AboutDialog

`public` **`AboutDialog`**`(javax.swing.JFrame owner)`

Constructor for about dialog. This constructor initializes the whole dialog but does not set it visible.

Parameters:
`owner` - Owner of this about dialog.

Methods

showDialog

`public static void` **`showDialog`**`(javax.swing.JFrame owner)`

Static method for initializing an about dialog and spawning it to the screen.

Parameters:
`owner` - Owner of this about dialog.

ucot.ui.gui.dialog
Class AddToModelWithDialog

```
java.lang.Object
  |
  +-java.awt.Component
      |
      +-java.awt.Container
          |
          +-java.awt.Window
              |
              +-java.awt.Dialog
                  |
                  +-javax.swing.JDialog
                      |
                      +-ucot.ui.gui.dialog.AddToModelWithDialog
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

---

public class **AddToModelWithDialog**
extends javax.swing.JDialog

Dialog extending `JDialog` for asking from user wich `ParserInterface` and `HeuristicInterface` he/she wants to use.

Selected `ParserInterface` and `HeuristicInterface` can be resolved with static methods `getParser()` and `getHeuristic()`. If user closed dialog without selecting `ParserInterface` and `HeuristicInterface` methods `getParser()` and `getHeuristic()` return `null`.

Author:
ilanliuk

See Also:
`javax.swing.JDialog`

Fields

core

`private ucot.core.ControlInterface` **`core`**

---

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`8969289172710583832`**

---

selectedParser

`private static ucot.parser.ParserInterface` **`selectedParser`**

selectedHeuristic

`private static ucot.heuristic.HeuristicInterface` **`selectedHeuristic`**

parserLabel

`private javax.swing.JLabel` **`parserLabel`**

parserComboBox

`private javax.swing.JComboBox` **`parserComboBox`**

heuristicLabel

`private javax.swing.JLabel` **`heuristicLabel`**

heuristicComboBox

`private javax.swing.JComboBox` **`heuristicComboBox`**

dialogListener

`private java.awt.event.WindowListener` **`dialogListener`**

Custom WindowListener to override windowClosing-event. Action in windowClosing-event is same as when user presses Cancel-button.

Constructors

AddToModelWithDialog

```
public AddToModelWithDialog(java.awt.Frame owner,
                            ControlInterface core)
```

Default constructor for `AddToModelWithDialog`.

Parameters:

`owner` - the `Frame` from which the dialog is displayed.
`core` - `ControllInterface`.

Throws:

`HeadlessException` - if GraphicsEnvironment.isHeadless() returns true.

Methods

(continued from last page)

buttonOKClicked

`private void` **`buttonOKClicked`**`()`

Method sets values of `selectedParser` and `selectedHeuristic` same that are in `parserComboBox` and `heuristicComboBox` and closes dialog.

buttonCancelClicked

`private void` **`buttonCancelClicked`**`()`

Method sets values of `selectedParser` and `selectedHeuristic` to `null` and closes dialog.

InitializeComboBoxes

`private void` **`InitializeComboBoxes`**`()`

Sets up `parserComboBox` and `heuristicComboBox`. Clears items from them and adds new items.

showDialog

`public static boolean` **`showDialog`**`(ControlInterface core)`

Static method to create and show dialog. Returns true if user clicked OK button in dialog else returns false.

Parameters:
`core` - `ControlInterface`

Returns:
`true` if user clicked OK-button in dialog, else returns `false`.

getHeuristic

`public static HeuristicInterface` **`getHeuristic`**`()`

Returns `HeuristicInterface` that user selected, or `null` if no `HeuristicInterface` was selected.

Returns:
`HeuristicInterface` user selected from dialog.

getParser

`public static ParserInterface` **`getParser`**`()`

Returns `ParserInterface` that user selected, or `null` if no `ParserInterface` was selected.

(continued from last page)

Returns:

`ParserInterface` user selected from dialog.

ucot.ui.gui.dialog
Class EntityPropertiesDialog

```
java.lang.Object
    |
    +-java.awt.Component
        |
        +-java.awt.Container
            |
            +-java.awt.Window
                |
                +-java.awt.Dialog
                    |
                    +-javax.swing.JDialog
                        |
                        +-ucot.ui.gui.dialog.EntityPropertiesDialog
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

public class **EntityPropertiesDialog**
extends javax.swing.JDialog

Dialog for modifying a single entity's properties in the analyze model. User can modify entity's name, its methods, attributes and parents with this dialog and when modifications are done, the modified analyze model is returned.

```
Usage:
  - first initialize an EntityPropertiesDialog objecs as any other object.
  - then call method:
      modifyEntityProperties(Entity to edit, AnalyzeModel)
    which returns the modified analyze model.
```

Author:
ilanliuk, tujupien, pajumasu.

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **-3386954136246879640**

owner

`private ucot.ui.gui.GraphicalUI` **`owner`**

(continued from last page)

selectedEntityName

`private java.lang.String` **`selectedEntityName`**

entityPropertiesTabbedPane

`private javax.swing.JTabbedPane` **`entityPropertiesTabbedPane`**

analyzeModel

`private ucot.model.AnalyzeModel` **`analyzeModel`**

entityName

`private javax.swing.JTextField` **`entityName`**

The entity name text field.

entityType

`private javax.swing.JComboBox` **`entityType`**

The entity type combobox.

ENTITY_NAME

`private static final java.lang.String` **`ENTITY_NAME`**

ENTITY_TYPE

`private static final java.lang.String` **`ENTITY_TYPE`**

ENTITY_NO_TYPE_DESCRIPTION

`private static final java.lang.String` **`ENTITY_NO_TYPE_DESCRIPTION`**

DIALOG_HEADER

`protected static final java.lang.String` **`DIALOG_HEADER`**

The header for this dialog.

ENTITY_NAME_CHANGED_LOG_MESSAGE

`protected static final java.lang.String` **`ENTITY_NAME_CHANGED_LOG_MESSAGE`**

(continued from last page)

Log message for informing entitys name change.

MAIN_OK_BUTTON

`protected static java.lang.String` **`MAIN_OK_BUTTON`**

Text for the ok button.

MAIN_CANCEL_BUTTON

`protected static java.lang.String` **`MAIN_CANCEL_BUTTON`**

Text for the cancel button.

DELETE_ENTITY_BUTTON

`protected static java.lang.String` **`DELETE_ENTITY_BUTTON`**

Text for the delete entity button.

MAIN_PROPERTIES_HEADER

`protected static final java.lang.String` **`MAIN_PROPERTIES_HEADER`**

Header for the main properties.

METHODS_TAB_KEY

`public static final java.lang.String` **`METHODS_TAB_KEY`**

Constant value: **`METHODS`**

PARENTS_TAB_KEY

`public static final java.lang.String` **`PARENTS_TAB_KEY`**

Constant value: **`PARENTS`**

CHILDREN_TAB_KEY

`public static final java.lang.String` **`CHILDREN_TAB_KEY`**

Constant value: **`CHILDREN`**

ATTRIBUTES_TAB_KEY

`public static final java.lang.String` **`ATTRIBUTES_TAB_KEY`**

Constant value: **`ATTRIBUTES`**

MAIN_OK_BUTTON_ACTION

`protected static final java.lang.String` **`MAIN_OK_BUTTON_ACTION`**

(continued from last page)

Constant value: **MAIN_OK**

MAIN_CANCEL_BUTTON_ACTION

`protected static final java.lang.String` **`MAIN_CANCEL_BUTTON_ACTION`**

Constant value: **MAIN_CANCEL**

DELETE_ENTITY_BUTTON_ACTION

`protected static final java.lang.String` **`DELETE_ENTITY_BUTTON_ACTION`**

Constant value: **DELETE_ENTITY**

tabs

`private java.lang.Object` **`tabs`**

tabKeys

`private java.lang.String` **`tabKeys`**

tabMap

`private java.util.Map` **`tabMap`**

logger

`private static final java.util.logging.Logger` **`logger`**

entityTabs

`private java.util.Collection` **`entityTabs`**

buttonListener

`private java.awt.event.ActionListener` **`buttonListener`**

ActionListener for all button events within the dialog.

Constructors

EntityPropertiesDialog

`public` **`EntityPropertiesDialog`**`(GraphicalUI owner)`

(continued from last page)

`EntityPropertiesDialog` constructor. This creates the whole dialog, but the actual contents of the all fields will be set later in the initialization method.

Parameters:
`owner` - This dialog's owner component.

Throws:
`HeadlessException` - Exception is thrown if the superclass initialization goes wrong.

Methods

modifyEntityProperties

```
public AnalyzeModel modifyEntityProperties(java.lang.String entityName,
        AnalyzeModel model)
```

Method to spawn the `EntityPropertiesDialog` and stay modal until user closes it. Modified analyze model is returned.

Parameters:
`entityName` - name of the entity to edit.
`model` - `AnalyzeModel` to edit.

Returns:
Modified analyze model as an `AnalyzeModel` object.

initializeDialog

```
private void initializeDialog()
```

This method initializes the dialog and all its dynamic components.

showDialog

```
public static AnalyzeModel showDialog(GraphicalUI owner,
        java.lang.String entityName,
        AnalyzeModel model)
```

Constructs, initializes and spawns an entity properties dialog for the given entity in the given analyze model. The given model is updated with the user's modifications and then returned.

Parameters:
`owner` - Owner component of the to generated entity properties dialog.
`entityName` - The name of the entity to be edited.
`model` - The analyze model which is being modified.

Returns:
Modified analyze model as an `AnalyzeModel` object.

showTabFor

```
public void showTabFor(java.lang.String key)
```

(continued from last page)

Shows the current tab.

Parameters:
`key` - The key for the current tab.

ucot.ui.gui.dialog
Class MergeEntitiesDialog

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-java.awt.Window
            |
            +-java.awt.Dialog
               |
               +-javax.swing.JDialog
                  |
                  +-ucot.ui.gui.dialog.MergeEntitiesDialog
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

---

public class **MergeEntitiesDialog**
extends javax.swing.JDialog

Dialog extending `JDialog` component for user to select wich entities he/she wants to merge.

If user selects the entity to merge with, dialog merges entity given in constructor with the selected entity. Name of the merged entity is name of the entity user selected from combobox

Author:
ilanliuk

See Also:
`javax.swing.JDialog`

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`7039192783402548895`**

---

core

`private ucot.core.ControlInterface` **`core`**

---

entity

`private java.lang.String` **`entity`**

(continued from last page)

label

`javax.swing.JLabel **label**`

---

selector

`javax.swing.JComboBox **selector**`

---

okButton

`javax.swing.JButton **okButton**`

---

cancelButton

`javax.swing.JButton **cancelButton**`

OK_BUTTON_ACTION

`private static final java.lang.String **OK_BUTTON_ACTION**`

Constant value: **OK_BUTTON**

CANCEL_BUTTON_ACTION

`private static final java.lang.String **CANCEL_BUTTON_ACTION**`

Constant value: **CANCEL_BUTTON**

---

buttonListener

`private java.awt.event.ActionListener **buttonListener**`

Constructors

MergeEntitiesDialog

```
public MergeEntitiesDialog(java.awt.Frame owner,
                           ControlInterface core,
                           java.lang.String entity)
```

Default constructor for MergeEntitiesDialog

Parameters:

`owner` - the `Frame` from which the dialog is displayed.
`core` - `ControllInterface`.

(continued from last page)

`entity` - Name of the entity to merge.

Throws:

`HeadlessException` - if GraphicsEnvironment.isHeadless() returns true.

Methods

getComboBox

```
private javax.swing.JComboBox getComboBox()
```

Returns `JComboBox` with entities from `AnalyzeModel` of given `ControlInterface` as selections.

The entity to merge is not in selections.

Returns:

`JComboBox`

getSelected

```
private java.lang.String getSelected()
```

Returns selected item from selector `JComboBox`.

Returns:

Selected item from selector `JComboBox`.

showDialog

```
public static void showDialog(ControlInterface core,
         java.lang.String entity)
```

Initializes new instance of `MergeEntitiesDialog` and shows it.

Parameters:

`core` - `ControlInterface`
`entity` - Name of the entity to merge.

ucot.ui.gui.dialog
Class ProgressBarDialog

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-java.awt.Window
            |
            +-java.awt.Dialog
               |
               +-javax.swing.JDialog
                  |
                  +-ucot.ui.gui.dialog.ProgressBarDialog
```

All Implemented Interfaces:
ProgressBarInterface, java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

public class **ProgressBarDialog**
extends javax.swing.JDialog
implements javax.swing.WindowConstants, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, ProgressBarInterface

Progressbar dialog class.
Author:
tujupien

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **-3394097553175022298**

progressBar

`private javax.swing.JProgressBar progressBar`

description

`private javax.swing.JLabel description`

(continued from last page)

owner

```
private ucot.ui.gui.GraphicalUI owner
```

Constructors

ProgressBarDialog

```
public ProgressBarDialog(GraphicalUI owner,
                         java.lang.String title)
```

Default constructor for ProgressBarDialog.

Parameters:

`owner` - Owner of this dialog.
`title` - Title of this this dialog.

Methods

getMaximum

```
public int getMaximum()
```

---

getMinimum

```
public int getMinimum()
```

---

getPercentageComplete

```
public double getPercentageComplete()
```

---

getString

```
public java.lang.String getString()
```

---

getValue

```
public int getValue()
```

---

setMaximum

```
public void setMaximum(int maximum)
```

(continued from last page)

setMinimum

```
public void setMinimum(int minimum)
```

setString

```
public void setString(java.lang.String string)
```

setValue

```
public void setValue(int value)
```

setVisible

```
public void setVisible(boolean visible)
```

ucot.ui.gui.dialog
Class SettingsDialog

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-java.awt.Window
            |
            +-java.awt.Dialog
               |
               +-javax.swing.JDialog
                  |
                  +-ucot.ui.gui.dialog.SettingsDialog
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.accessibility.Accessible, javax.swing.WindowConstants

---

public class **SettingsDialog**
extends javax.swing.JDialog

Settings dialog for UCOT program. There are different kinds of settings available for modification through UCOT (G)UI, and this dialog allows user to change the values for those settings.

This settings dialog uses the ModulePropertyInterface offered by the UCOT modules and after changing the values each component's applyProperties method is called.

Author:
ilanliuk, tujupien

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **3186714875647715870**

---

owner

`private ucot.ui.gui.GraphicalUI owner`

---

logger

`private java.util.logging.Logger logger`

---

buttonListener

`private java.awt.event.ActionListener buttonListener`

(continued from last page)

result

`private java.lang.String` **`result`**

parserComboBox

`private javax.swing.JComboBox` **`parserComboBox`**

heuristicComboBox

`private javax.swing.JComboBox` **`heuristicComboBox`**

typeList

`private javax.swing.JList` **`typeList`**

typeListModel

`private javax.swing.DefaultListModel` **`typeListModel`**

fileTextFields

`private java.util.Vector` **`fileTextFields`**

fileBrowserButtons

`private java.util.Vector` **`fileBrowserButtons`**

dotPathIndex

`private int` **`dotPathIndex`**

epsToPDFpathIndex

`private int` **`epsToPDFpathIndex`**

(continued from last page)

graphComboBoxes

`private java.util.Vector` **`graphComboBoxes`**

---

normalColorIndex

`private int` **`normalColorIndex`**

---

highlightColorIndex

`private int` **`highlightColorIndex`**

MAIN_OK_BUTTON_ACTION

`public static final java.lang.String` **`MAIN_OK_BUTTON_ACTION`**

Constant value: **`BUTTON_OK_CLICKED`**

MAIN_CANCEL_BUTTON_ACTION

`public static final java.lang.String` **`MAIN_CANCEL_BUTTON_ACTION`**

Constant value: **`BUTTON_CANCEL_CLICKED`**

BROWSE_BUTTON_CLICKED

`private static final java.lang.String` **`BROWSE_BUTTON_CLICKED`**

Constant value: **`BUTTON_BROWSE_CLICKED`**

HORIZONTAL_GAP

`private static final int` **`HORIZONTAL_GAP`**

Constant value: **`10`**

VERTICAL_GAP

`private static final int` **`VERTICAL_GAP`**

Constant value: **`5`**

DIALOG_TITLE

`public static final java.lang.String` **`DIALOG_TITLE`**

MAIN_OK_BUTTON

`public static final java.lang.String` **`MAIN_OK_BUTTON`**

MAIN_CANCEL_BUTTON

`public static final java.lang.String` **`MAIN_CANCEL_BUTTON`**

BROWSE_BUTTON_TITLE

`public static final java.lang.String` **`BROWSE_BUTTON_TITLE`**

DOT_PATH_LABEL

`public static final java.lang.String` **`DOT_PATH_LABEL`**

EPS_TO_PDF_PATH_LABEL

`public static final java.lang.String` **`EPS_TO_PDF_PATH_LABEL`**

EXTERNAL_FILES_TITLE

`public static final java.lang.String` **`EXTERNAL_FILES_TITLE`**

GENERAL_SETTINGS_TITLE

`public static final java.lang.String` **`GENERAL_SETTINGS_TITLE`**

GRAPH_SETTINGS_TITLE

`public static final java.lang.String` **`GRAPH_SETTINGS_TITLE`**

ENTITY_TYPE_TITLE

`public static final java.lang.String` **`ENTITY_TYPE_TITLE`**

DEFAULT_PARSER_LABEL

`public static final java.lang.String` **`DEFAULT_PARSER_LABEL`**

(continued from last page)

DEFAULT_HEURISTIC_LABEL

`public static final java.lang.String` **`DEFAULT_HEURISTIC_LABEL`**

INCORRECT_PATH_TO_FILE_QUESTION

`public static final java.lang.String` **`INCORRECT_PATH_TO_FILE_QUESTION`**

DOT_COLOR_LABEL

`public static final java.lang.String` **`DOT_COLOR_LABEL`**

DOT_HIGHLIGHT_COLOR_LABEL

`public static final java.lang.String` **`DOT_HIGHLIGHT_COLOR_LABEL`**

ENTITY_TYPES_GOING_TO_BE_REMOVED_QUESTION

`public static final java.lang.String` **`ENTITY_TYPES_GOING_TO_BE_REMOVED_QUESTION`**

Constructors

SettingsDialog

`public` **`SettingsDialog`**`(GraphicalUI owner)`

Default constructor for SettingsDialog. This constructor initializes the whole dialog, creates the layout and makes all defined properties available for modification.

Parameters:
`owner` - GraphicalUI that owns this dialog.

Throws:
`If` - something with the initialization of the (super) class goes wrong, a HeadlessException is thrown.

Methods

createGraphSettingsPanel

`private javax.swing.JPanel` **`createGraphSettingsPanel`**`()`

createExternalFilesPanel

`private javax.swing.JPanel` **`createExternalFilesPanel`**`()`

This method creates a panel containing text boxes and browse buttons for modifying the path all external files that are associated with this program.

createGeneralPanel

`private javax.swing.JPanel` **`createGeneralPanel`**`()`

Initialize panel for general settings, like default parser and heuristics etc.

createEntityTypesPanel

`private javax.swing.JPanel` **`createEntityTypesPanel`**`()`

Initialize `JPanel` for entity types.

Returns:
`JPanel`

updateEntityTypes

`private void` **`updateEntityTypes`**`()`

buttonBrowseClicked

`private void` **`buttonBrowseClicked`**`(java.awt.event.ActionEvent e)`

Method for browsing files and putting the selected file to the correct text box.

mainCancelButtonClicked

`private void` **`mainCancelButtonClicked`**`()`

User just clicked cancel button and we just need close the dialog without saving anything.

mainOKButtonClicked

`private void` **`mainOKButtonClicked`**`()`

User clicked Apply and we just save the settings and vanish.

createButtonListener

`private void createButtonListener()`

This method initializes ActionListener for all buttons.

getResult

`public java.lang.String getResult()`

Method for figuring out how the user exited the dialog.

Returns:

Returns either SettingsDialog.MAIN_OK_BUTTON_ACTION or SettingsDialog.MAIN_CANCEL_BUTTON_ACTION depending on how the user closed the dialog.

showDialog

`public static java.lang.String showDialog(GraphicalUI owner)`

Static method to create and show SettingsDialog.

Parameters:

`owner` - Owner of the dialog to be shown.

Package ucot.ui.gui.dialog.entitytab

Classes related to the Dialog that edits entity's properties.

ucot.ui.gui.dialog.entitytab
Class AttributesPanel

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.dialog.entitytab.JTableAndButtonsPanel
                  |
                  +-ucot.ui.gui.dialog.entitytab.JTableEntityPropertiesTab
                     |
                     +-ucot.ui.gui.dialog.entitytab.AttributesPanel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, EntityPropertiesEditor

public class **AttributesPanel**
extends JTableEntityPropertiesTab

This panel allows user to edit entitys attributes.

Fields

logger

`java.util.logging.Logger` **`logger`**

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`1980835276224106020`**

NEW_ATTRIBUTES_LOG_MESSAGE

`protected static final java.lang.String` **`NEW_ATTRIBUTES_LOG_MESSAGE`**

Log message for informing creation of the new attributs.

DELETED_ATTRIBUTES_LOG_MESSAGE

`protected static final java.lang.String` **`DELETED_ATTRIBUTES_LOG_MESSAGE`**

Log message for informing deletion of removed attributes.

(continued from last page)

CHANGED_ATTRIBUTES_LOG_MESSAGES

`protected static final java.lang.String` **`CHANGED_ATTRIBUTES_LOG_MESSAGES`**

Log message for informing change of attributes.

ATTRIBUTES_TAB_HEADER

`protected static final java.lang.String` **`ATTRIBUTES_TAB_HEADER`**

Header for the attributes tab.

NEW_ATTRIBUTE_BUTTON

`protected static java.lang.String` **`NEW_ATTRIBUTE_BUTTON`**

Text for new attribute button.

DELETE_ATTRIBUTE_BUTTON

`protected static java.lang.String` **`DELETE_ATTRIBUTE_BUTTON`**

Text for delete attribute button.

NEW_ATTRIBUTE_BUTTON_ACTION

`protected static final java.lang.String` **`NEW_ATTRIBUTE_BUTTON_ACTION`**

Constant value: **`NEW_ATTRIBUTE`**

DELETE_ATTRIBUTE_BUTTON_ACTION

`protected static final java.lang.String` **`DELETE_ATTRIBUTE_BUTTON_ACTION`**

Constant value: **`DELETE_ATTRIBUTE`**

ATTRIBUTES_TABLE_COLUMNS

`protected static final java.lang.String` **`ATTRIBUTES_TABLE_COLUMNS`**

Headers for the attributes table.

Constructors

AttributesPanel

`public` **`AttributesPanel`**`()`

Creates the panel using the localized strings readed from the `Messages` object.

Methods

(continued from last page)

newAttributeButtonClicked

```
private void newAttributeButtonClicked()
```

Action performed: User clicked the 'new attribute' button.

deleteAttributeButtonClicked

```
private void deleteAttributeButtonClicked()
```

Action performed: User clicked the 'delete attribute' button.

action

```
public void action(java.lang.String cmd)
```

This method is called when an action is performed and it calls the corresponding methods to handle the action.

Parameters:

`cmd` - action command

load

```
public void load(AnalyzeModel analyzeModel,
         java.lang.String entityName)
```

save

```
public void save(AnalyzeModel model,
         java.lang.String saveEntityName)
```

ucot.ui.gui.dialog.entitytab
Class ChildrenPanel

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.dialog.entitytab.JTableAndButtonsPanel
                  |
                  +-ucot.ui.gui.dialog.entitytab.JTableEntityPropertiesTab
                     |
                     +-ucot.ui.gui.dialog.entitytab.ChildrenPanel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, EntityPropertiesEditor

---

public class **ChildrenPanel**
extends JTableEntityPropertiesTab

This panel allows user to edit entitys childs.

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **4303304569865873821**

NEW_CHILDREN_LOG_MESSAGE

`protected static final java.lang.String NEW_CHILDREN_LOG_MESSAGE`

Log message for informing creation of new children.

DELETED_CHILDREN_LOG_MESSAGE

`protected static final java.lang.String DELETED_CHILDREN_LOG_MESSAGE`

Log message for informing deletion of removed children.

NEW_CHILD_BUTTON

`protected static java.lang.String NEW_CHILD_BUTTON`

Text for new child button.

(continued from last page)

DELETE_CHILD_BUTTON

`protected static java.lang.String` **`DELETE_CHILD_BUTTON`**

Text for delete child button.

NEW_CHILD_BUTTON_ACTION

`protected static final java.lang.String` **`NEW_CHILD_BUTTON_ACTION`**

Constant value: **`NEW_CHILD`**

DELETE_CHILD_BUTTON_ACTION

`protected static final java.lang.String` **`DELETE_CHILD_BUTTON_ACTION`**

Constant value: **`DELETE_CHILD`**

CHILDREN_TABLE_COLUMNS

`protected static final java.lang.String` **`CHILDREN_TABLE_COLUMNS`**

Headers for the method table.

CHILDREN_TAB_HEADER

`protected static java.lang.String` **`CHILDREN_TAB_HEADER`**

Header for the children tab.

---

logger

`private static final java.util.logging.Logger` **`logger`**

Constructors

ChildrenPanel

`public` **`ChildrenPanel`**`()`

Creates the panel using the localized strings readed from the `Messages` object.

Methods

newParentButtonClicked

`private void` **`newParentButtonClicked`**`()`

Action performed: User clicked the 'new child' button.

(continued from last page)

deleteParentButtonClicked

```
private void deleteParentButtonClicked()
```

Action performed: User clicked the 'delete child' button.

action

```
public void action(java.lang.String cmd)
```

This method is called when an action is performed and it calls the corresponding methods to handle the action.

save

```
public void save(AnalyzeModel analyzeModel,
         java.lang.String saveEntityName)
```

load

```
public void load(AnalyzeModel analyzeModel,
         java.lang.String loadEntityName)
```

ucot.ui.gui.dialog.entitytab
Interface EntityPropertiesEditor

All Known Implementing Classes:

JTableEntityPropertiesTab

---

public interface **EntityPropertiesEditor**
extends

This class represents interface for general entity's properties modification component. When the `EntityPropertiesEditor` is created it can be used to show the information of the wanted entity using `load` method. After the user interaction is done, for example "OK" is pressed in some upeprlevel dialog, the `EntityPropertiesEditor` can be asked to save the modifications user did using `save` method.

Fields

SELF_POINTER_NAME

```
public static final java.lang.String SELF_POINTER_NAME
```

Self pointer string. This string is used to when we need to represent relation that points to the entity beign edited.

Methods

clear

```
public void clear()
```

Clears the panel.

---

load

```
public void load(AnalyzeModel model,
         java.lang.String entityName)
```

Loads information to the panel and shows it. This does not clear the view. It only adds the information to the view. Use `clear` to clear the view.

Parameters:

`model` - The `AnalyzeModel` which contains the entity.
`entityName` - The name of the entity.

See Also:

`clear()`

---

save

```
public void save(AnalyzeModel model,
         java.lang.String entityName)
```

(continued from last page)

Informs the panel that it should update the given model based on the panel's information. Nothing should happen if `save` is called after `load` without user interaction in the panel (or some modifications done in the model).

Parameters:

`model` - The `AnalyzeModel` which contains the entity.

`entityName` - The name of the entity.

getTabName

`public java.lang.String` **`getTabName`**`()`

Returns the table name that should be printed in the tab selection menu.

Returns:

table name

getComponent

`public javax.swing.JComponent` **`getComponent`**`()`

Returns the component to be shown to the user to allow editions.

Returns:

The component for this tab.

ucot.ui.gui.dialog.entitytab Class JTableAndButtonsPanel

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.dialog.entitytab.JTableAndButtonsPanel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

Direct Known Subclasses:
JTableEntityPropertiesTab

---

public class **JTableAndButtonsPanel**
extends javax.swing.JPanel

This is basic panel that contains table and panel for buttons. It is generalized interface for making basic modifications in some data. This should be extended for specialized purposes, but it is not nessesary.

To use this class by extending it only action method should be overrided. It can be used to react the user interaction with buttons.

```
class SomeExtenderClass extends JTableAndButtonsPanel{
  public void action(String cmd) {
                if ( BUTTON_1_ACTIONequals(cmd) ) {
                        doAction1(); return;
                }

                if ( BUTTON_2_ACTION.equals(cmd) ) {
                        doAction2(); return;
                }
         }
        public SomeExtenderClass(){
                super("Header",new String[]{"Column1","Columnt"});
                // Create some buttons
                addButton(BUTTON_1_NAME,BUTTON_1_ACTION);
                addButton(BUTTON_2_NAME,BUTTON_2_ACTION);
        }
 }
```

If this is used as direct component added buttons should contain proper `ActionListener` the code to react their events.

```
class SomeClass{
  public MethodsPanel(){
      JTableAndButtonsPanel panel =
                        new JTableAndButtonsPanel("Header",new
String[]{"Column1","Columnt"});
                // Create some buttons
                panel.addButton(BUTTON_1_NAME,BUTTON_1_ACTION).addActionListener(
                                        new ActionListener(){
                                                public void actionPerformed(ActionEvent e) {
                                                        doAction1;
                                }
                                        });
                panel.addButton(BUTTON_1_NAME,BUTTON_1_ACTION).addActionListener(
                                        new ActionListener(){
                                                public void actionPerformed(ActionEvent e) {
                                                        doAction2;
                                }
                                        });
        }
```

Fields

buttons

java.util.Map **buttons**

serialVersionUID

private static final long **serialVersionUID**

Constant value: **-3785955762470238856**

buttonsPanel

private javax.swing.JPanel **buttonsPanel**

table

private javax.swing.JTable **table**

containerPanel

private javax.swing.JPanel **containerPanel**

(continued from last page)

scrollPane

```
private javax.swing.JScrollPane scrollPane
```

actionListener

```
private java.awt.event.ActionListener actionListener
```

Constructors

JTableAndButtonsPanel

```
public JTableAndButtonsPanel()
```

Creates the panel with empty table and no buttons.

Methods

setTable

```
public void setTable(javax.swing.JTable newTable)
```

getTable

```
public javax.swing.JTable getTable()
```

addButton

```
public javax.swing.JButton addButton(java.lang.String label,
         java.lang.String actionCommand)
```

action

```
protected void action(java.lang.String actionCmd)
```

ucot.ui.gui.dialog.entitytab
Class JTableEntityPropertiesTab

```
java.lang.Object
  |
  +-java.awt.Component
      |
      +-java.awt.Container
          |
          +-javax.swing.JComponent
              |
              +-javax.swing.JPanel
                  |
                  +-ucot.ui.gui.dialog.entitytab.JTableAndButtonsPanel
                      |
                      +-ucot.ui.gui.dialog.entitytab.JTableEntityPropertiesTab
```

All Implemented Interfaces:
EntityPropertiesEditor, java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

Direct Known Subclasses:
AttributesPanel, ChildrenPanel, MethodsPanel, ParentsPanel

---

public abstract class **JTableEntityPropertiesTab**
extends JTableAndButtonsPanel
implements javax.accessibility.Accessible, java.io.Serializable, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, EntityPropertiesEditor

---

Fields

tableModel

`javax.swing.table.DefaultTableModel` **`tableModel`**

---

columnNames

`java.lang.String` **`columnNames`**

---

tabName

`java.lang.String` **`tabName`**

Constructors

(continued from last page)

JTableEntityPropertiesTab

```
public JTableEntityPropertiesTab(java.lang.String name,
                                 java.lang.String[] columnNames)
```

Methods

getTabName

```
public java.lang.String getTabName()
```

getModel

```
public javax.swing.table.DefaultTableModel getModel()
```

Returns:
model

clear

```
public void clear()
```

updateCellEditor

```
public void updateCellEditor(AnalyzeModel analyzeModel,
         java.lang.String entityName,
         int column_index,
         boolean allowSelf)
```

Parameters:
`analyzeModel` - analyze model to use
`entityName` - entity to
`column_index`
`allowSelf`

setColumns

```
public void setColumns(java.lang.String[] columnNames)
```

Sets the column names

Parameters:
`columnNames` - array of column names to set

load

```
public void load(AnalyzeModel analyzeModel,
         java.lang.String entityName)
```

(continued from last page)

save

```
public void save(AnalyzeModel model,
         java.lang.String entityName)
```

getComponent

```
public javax.swing.JComponent getComponent()
```

ucot.ui.gui.dialog.entitytab
Class MethodsPanel

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.dialog.entitytab.JTableAndButtonsPanel
                  |
                  +-ucot.ui.gui.dialog.entitytab.JTableEntityPropertiesTab
                     |
                     +-ucot.ui.gui.dialog.entitytab.MethodsPanel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, EntityPropertiesEditor

---

public class **MethodsPanel**
extends JTableEntityPropertiesTab

This panel allows user to edit entitys methods.

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`-5071995337843998163`**

NEW_METHODS_LOG_MESSAGE

`protected static final java.lang.String` **`NEW_METHODS_LOG_MESSAGE`**

Log message for informing creation of the new methods.

DELETED_METHODS_LOG_MESSAGE

`protected static final java.lang.String` **`DELETED_METHODS_LOG_MESSAGE`**

Log message for informing deletion of removed methods.

CHANGED_METHODS_LOG_MESSAGE

`protected static final java.lang.String` **`CHANGED_METHODS_LOG_MESSAGE`**

Log message for informing change of methods.

(continued from last page)

NEW_METHOD_BUTTON

`protected static java.lang.String` **`NEW_METHOD_BUTTON`**

Text for new method button.

DELETE_METHOD_BUTTON

`protected static java.lang.String` **`DELETE_METHOD_BUTTON`**

Text for delete method button.

NEW_METHOD_BUTTON_ACTION

`protected static final java.lang.String` **`NEW_METHOD_BUTTON_ACTION`**

Constant value: **`NEW_METHOD`**

DELETE_METHOD_BUTTON_ACTION

`protected static final java.lang.String` **`DELETE_METHOD_BUTTON_ACTION`**

Constant value: **`DELETE_METHOD`**

METHODS_TAB_HEADER

`protected static java.lang.String` **`METHODS_TAB_HEADER`**

Header for the methods tab.

METHODS_TABLE_COLUMNS

`protected static final java.lang.String` **`METHODS_TABLE_COLUMNS`**

Headers for the method table.

logger

`private static final java.util.logging.Logger` **`logger`**

Constructors

MethodsPanel

`public` **`MethodsPanel`**`()`

Creates the panel using the localized strings readed from the `Messages` object.

Methods

(continued from last page)

newMethodButtonClicked

```
private void newMethodButtonClicked()
```

Action performed: User clicked the 'new method' button.

deleteMethodButtonClicked

```
private void deleteMethodButtonClicked()
```

Action performed: User clicked the 'delete method' button.

action

```
public void action(java.lang.String cmd)
```

This method is called when an action is performed and it calls the corresponding methods to handle the action.

load

```
public void load(AnalyzeModel analyzeModel,
         java.lang.String loadEntityName)
```

save

```
public void save(AnalyzeModel analyzeModel,
         java.lang.String saveEntityName)
```

ucot.ui.gui.dialog.entitytab
Class ParentsPanel

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.dialog.entitytab.JTableAndButtonsPanel
                  |
                  +-ucot.ui.gui.dialog.entitytab.JTableEntityPropertiesTab
                     |
                     +-ucot.ui.gui.dialog.entitytab.ParentsPanel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, EntityPropertiesEditor

public class **ParentsPanel**
extends JTableEntityPropertiesTab

This panel allows user to edit entitys parents.

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **4325462850141268992**

NEW_PARENTS_LOG_MESSAGE

`protected static final java.lang.String` **`NEW_PARENTS_LOG_MESSAGE`**

Log message for informing creation of new parents.

DELETED_PARENTS_LOG_MESSAGE

`protected static final java.lang.String` **`DELETED_PARENTS_LOG_MESSAGE`**

Log message for informing deletion of removed parents.

NEW_PARENT_BUTTON

`protected static java.lang.String` **`NEW_PARENT_BUTTON`**

Text for new parent button.

(continued from last page)

DELETE_PARENT_BUTTON

`protected static java.lang.String` **`DELETE_PARENT_BUTTON`**

Text for delete parent button.

NEW_PARENT_BUTTON_ACTION

`protected static final java.lang.String` **`NEW_PARENT_BUTTON_ACTION`**

Constant value: **`NEW_PARENT`**

DELETE_PARENT_BUTTON_ACTION

`protected static final java.lang.String` **`DELETE_PARENT_BUTTON_ACTION`**

Constant value: **`DELETE_PARENT`**

PARENTS_TABLE_COLUMNS

`protected static final java.lang.String` **`PARENTS_TABLE_COLUMNS`**

Headers for the method table.

PARENTS_TAB_HEADER

`protected static java.lang.String` **`PARENTS_TAB_HEADER`**

Header for the parents tab.

logger

`private static final java.util.logging.Logger` **`logger`**

Constructors

ParentsPanel

`public` **`ParentsPanel`**`()`

Creates the panel using the localized strings readed from the `Messages` object.

Methods

newParentButtonClicked

`private void` **`newParentButtonClicked`**`()`

Action performed: User clicked the 'new parent' button.

(continued from last page)

deleteParentButtonClicked

```
private void deleteParentButtonClicked()
```

Action performed: User clicked the 'delete parent' button.

action

```
public void action(java.lang.String cmd)
```

This method is called when an action is performed and it calls the corresponding methods to handle the action.

save

```
public void save(AnalyzeModel analyzeModel,
         java.lang.String saveEntityName)
```

load

```
public void load(AnalyzeModel analyzeModel,
         java.lang.String loadEntityName)
```

Package ucot.ui.gui.dot

Classes for displaying graphs generated by Dot program.

ucot.ui.gui.dot
Class DotColorModel

```
java.lang.Object
  |
  +-ucot.ui.gui.dot.DotColorModel
```

public class **DotColorModel**
extends java.lang.Object

Class that implements a color model used in dot graph drawing environment. This model can be used for both main colors and highlights.
Author:
UCOT

Fields

DEFAULT_COLOR

`private static final ucot.ui.gui.dot.DotColorModel.ColorModel` **`DEFAULT_COLOR`**

edgeColor

`private java.lang.String` **`edgeColor`**

fillColor

`private java.lang.String` **`fillColor`**

fontColor

`private java.lang.String` **`fontColor`**

model

`private ucot.ui.gui.dot.DotColorModel.ColorModel` **`model`**

Constructors

(continued from last page)

DotColorModel

```
private DotColorModel(java.lang.String edgeColor,
                      java.lang.String fillColor,
                      java.lang.String fontColor)
```

Constructor which sets all the colors as specified.

Parameters:
`edgeColor` - `Color` of edges in the graph.
`fillColor` - `Color` for filling the elements on the graph. Can also be used for graph background.
`fontColor` - Text color.

DotColorModel

```
private DotColorModel(java.lang.String drawColor,
                      java.lang.String fillColor)
```

Constructor for quick color model initialization.

Parameters:
`drawColor` - `Color` for all edges and text in the graph.
`fillColor` - Background color for the whole graph and all elements.

Methods

getEdgeColor

```
public java.lang.String getEdgeColor()
```

Method for acquiring the edge color.

Returns:
Edge color.

getFillColor

```
public java.lang.String getFillColor()
```

Method for acquiring the fill color.

Returns:
Fill color.

getFontColor

```
public java.lang.String getFontColor()
```

(continued from last page)

Method for acquiring the font color.

Returns:
Font color.

getModel

```
public java.lang.String getModel()
```

Method for acquiring the name of the current model.

Returns:
Color model's name.

blue

```
private static DotColorModel blue()
```

This static method creates a blue dot color model.

Returns:
Blue dot color model.

green

```
private static DotColorModel green()
```

This static method creates a green dot color model.

Returns:
Green dot color model.

blackOnWhite

```
private static DotColorModel blackOnWhite()
```

This static method creates a black text on white background dot color model.

Returns:
Black text on white background dot color model.

whiteOnBlack

```
private static DotColorModel whiteOnBlack()
```

This static method creates a white text on black background dot color model.

Returns:

(continued from last page)

White text on black background dot color model.

red

`private static DotColorModel red()`

This static method creates a red dot color model.

Returns:
Red dot color model.

color

`public static DotColorModel color(DotColorModel.ColorModel colorModel)`

Static method for creating a color model from available sets.

color

`public static DotColorModel color(java.lang.String colorModel)`

Static method for creating a color model from available sets.

ucot.ui.gui.dot
Class DotColorModel.ColorModel

```
java.lang.Object
   |
   +-java.lang.Enum
      |
      +-ucot.ui.gui.dot.DotColorModel.ColorModel
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

public static final class **DotColorModel.ColorModel**
extends java.lang.Enum

ColorModel sets available for using with Dot
Author:
tujupien

Fields

BLACK_ON_WHITE

`public static final ucot.ui.gui.dot.DotColorModel.ColorModel` **`BLACK_ON_WHITE`**

WHITE_ON_BLACK

`public static final ucot.ui.gui.dot.DotColorModel.ColorModel` **`WHITE_ON_BLACK`**

GREEN

`public static final ucot.ui.gui.dot.DotColorModel.ColorModel` **`GREEN`**

BLUE

`public static final ucot.ui.gui.dot.DotColorModel.ColorModel` **`BLUE`**

RED

`public static final ucot.ui.gui.dot.DotColorModel.ColorModel` **`RED`**

Constructors

(continued from last page)

DotColorModel.ColorModel

`private` **`DotColorModel.ColorModel`**`()`

Methods

values

`public final static DotColorModel.ColorModel[]` **`values`**`()`

valueOf

`public static DotColorModel.ColorModel` **`valueOf`**`(java.lang.String name)`

ucot.ui.gui.dot
Class DotColorTheme

```
java.lang.Object
  |
  +-ucot.ui.gui.dot.DotColorTheme
```

public class **DotColorTheme**
extends java.lang.Object

This class implements a color theme for dot markup language, which is useful for `DotPanel` when figuring out color strings. It is easy to change the color theme using the method 'changeToColorTheme' and giving it the identifier of the preferred color theme.

All themes available currently have to be hard coded here because of the way how Dot understands colors.TODO: Make color themes more dynamic? TODO: Add more color themes.
Author:
tujupien

Fields

normal

`private ucot.ui.gui.dot.DotColorModel normal`

highlight

`private ucot.ui.gui.dot.DotColorModel highlight`

DEFAULT_HIGHLIGHT

`private static final ucot.ui.gui.dot.DotColorModel.ColorModel DEFAULT_HIGHLIGHT`

DEFAULT_COLOR

`private static final ucot.ui.gui.dot.DotColorModel.ColorModel DEFAULT_COLOR`

Constructors

DotColorTheme

```
public DotColorTheme(DotColorModel.ColorModel color,
                     DotColorModel.ColorModel highlight)
```

(continued from last page)

Constructor for DotColorTheme class.

Parameters:
`color` - Normal color of the graph.
`highlight` - Color of the highlighted elements.

DotColorTheme

`public DotColorTheme(DotColorModel.ColorModel color)`

Constructor for DotColorTheme class.

Parameters:
`color` - Normal color of the graph.

DotColorTheme

`public DotColorTheme()`

Default constructor for DotColorTheme class which initially uses the default color theme.

Methods

changeColor

`public void changeColor(java.lang.String color)`

Normal color changer.

Parameters:
`color` - New normal color.

---

changeColor

`public void changeColor(DotColorModel.ColorModel color)`

Normal color changer.

Parameters:
`color` - New normal color.

---

changeHighlight

`public void changeHighlight(java.lang.String highlight)`

Highlight color changer.

(continued from last page)

Parameters:
`highlight` - New highlight color.

changeHighlight

`public void **changeHighlight**(DotColorModel.ColorModel highlight)`

Highlight color changer.

Parameters:
`highlight` - New highlight color.

getHighlightedColorString

`public java.lang.String **getHighlightedColorString**()`

Method which formats highlighted item's color attributes into dot's syntax.

Returns:
Highlight nodes' or edges' string in dot's syntax.

getBackgroundColorString

`public java.lang.String **getBackgroundColorString**()`

Method for getting the background color of the whole graph in dot's syntax.

Returns:
Background color string in dot's syntax.

getNormalColorString

`public java.lang.String **getNormalColorString**()`

Method which formats normal item's color attributes into dot's syntax.

Returns:
Normal nodes' or edges' string in dot's syntax.

getColorString

`public java.lang.String **getColorString**(boolean isHighlighted)`

Method for getting the appropriate color string for an entity based on its highlight status.

Parameters:
`isHighlighted` - Defines wether or not the returned color string is supposed to be for an highlighted entity or a normal entity.

Returns:

(continued from last page)

Returns appropriate color string in dot's syntax.

getColor

`public java.lang.String **getColor**()`

Method for acquiring the name of the current color model.

Returns:
Name of the color model.

getHighlight

`public java.lang.String **getHighlight**()`

Method for acquiring the name of the current highlight model.

Returns:
Name of the highlight model.

getBackgroundColorAsJavaObject

`public java.awt.Color **getBackgroundColorAsJavaObject**()`

Method for getting the background color of the graph as a java object. This helps to figure out the background color of the DotPanel.

Returns:
Graph background color as a Java object.

ucot.ui.gui.dot
Class DotPanel

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JPanel
               |
               +-ucot.ui.gui.dot.DotPanel
```

All Implemented Interfaces:
java.util.Observer, java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible

---

public class **DotPanel**
extends javax.swing.JPanel
implements javax.accessibility.Accessible, java.io.Serializable, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.Observer

A dot panel class inherited from `JPanel` class which is used for drawing dot graphs from the given analyze model.

This class implements the `Observer` interface which allows this panel to keep track of the current status of analyze model. Panel automatically updates the graph when the model changes

`DotPanel` contains also a method `highlight` for highlighting any submodels from the given `Analyzemodel`.
Author:
ilanliuk, tujupien, vevijopi.
See Also:
`AnalyzeModel`

Fields

serialVersionUID

`public static final long **serialVersionUID**`

Constant value: **3899637467435823526**

THREAD_RUNNING

`private static boolean **THREAD_RUNNING**`

DOT_JOB_STACK

`private static java.util.Stack **DOT_JOB_STACK**`

(continued from last page)

EXECUTING_DOT_THREAD

`private static java.lang.Thread` **`EXECUTING_DOT_THREAD`**

EXPORT_JOB_STACK

`private static java.util.Stack` **`EXPORT_JOB_STACK`**

EXECUTING_EXPORT_THREAD

`private static java.lang.Thread` **`EXECUTING_EXPORT_THREAD`**

logger

`private java.util.logging.Logger` **`logger`**

DEFAULT_EXPORT_IMAGE_TYPE

`public static final ucot.ui.gui.dot.DotPanel.ExportImageType` **`DEFAULT_EXPORT_IMAGE_TYPE`**

timeElapsed

`protected long` **`timeElapsed`**

timerComponent

`protected javax.swing.JLabel` **`timerComponent`**

fontSize

`protected int` **`fontSize`**

colorTheme

`public ucot.ui.gui.dot.DotColorTheme` **`colorTheme`**

(continued from last page)

SCROLL_SPEED

`public static final int` **`SCROLL_SPEED`**

Constant value: **`20`**

---

modelImage

`private java.awt.image.BufferedImage` **`modelImage`**

---

owner

`protected ucot.ui.gui.GraphicalUI` **`owner`**

---

horizontalLayout

`private boolean` **`horizontalLayout`**

---

isUpdating

`private boolean` **`isUpdating`**

Constructors

DotPanel

`public` **`DotPanel`**`(GraphicalUI owner)`

Constructor for `DotPanel`. This constructor adds an observer to the analyze model and sets its own status to updating.TODO fix this param, it was analyzeModel before

Parameters:
`owner` - `AnalyzeModel` which this `DotPanel` should draw.

Methods

setHorizontalLayout

`public void` **`setHorizontalLayout`**`(boolean horizontalLayout)`

Sets wether or not the dot should use the horizontal layout for the entities and their relationships.

Parameters:
`horizontalLayout` - New value for horizontal layout, true means yay for horizontal layouting.

(continued from last page)

getHorizontalLayout

```
public boolean getHorizontalLayout()
```

Returns the current value of horizontal layouting.

Returns:
`Boolean` `true` if horizontal layout is in use, else `false`.

update

```
public void update(java.util.Observable observableObject,
         java.lang.Object updationArg)
```

saveImage

```
private void saveImage(java.awt.image.BufferedImage graph,
         java.io.File target,
         DotPanel.ExportImageType imageType)
  throws java.io.IOException
```

This method writes the graph given as a buffered image to the given file in the format specified by imageType.

Parameters:
`graph` - Graph as `BufferedImage` to be written.
`target` - `File` where the graph should be saved.
`imageType` - File format to use when saving as `ExportImageType`.

Throws:
`IOException` - If something goes wrong with the writing.

saveExport

```
private void saveExport(java.io.File target,
         DotPanel.ExportImageType imageType)
  throws java.io.IOException
```

This method generates a new graph from the analyze model and exports it to the given target file in a format specified by imageType.

Parameters:
`target` - `File` where the graph should be saved.
`imageType` - File format to use when saving as `ExportImageType`.

Throws:
`IOException` - If something goes wrong with the writing.

executeExport

```
private void executeExport()
```

(continued from last page)

This method should be used inside a new thread. Once started, this method will keep on running until the `DotPanel` gets finalized and while running, this method will perform all graph exporting scheduled for `DotPanel`.

exportImage

```
public void exportImage(java.io.File target,
         DotPanel.ExportImageType imageType)
```

This method starts a new thread for export actions if one has not been started yet. In each case a new export job is pushed to the top of the export job stack and the thread is notified about this action, which wakes the thread and it will start doing the topmost job from the stack when it wakes up.

Parameters:

`target` - `File` where the image should be saved.
`imageType` - Format for the image to be saved as `ExportImageType`.

exportImage

```
public void exportImage(java.io.File target)
```

Saves current image to given file in format that is tried to guess from filename.

Parameters:

`target` - `File` pointing the saving destination.

convertEPSToPDF

```
private void convertEPSToPDF(java.io.File epsImage,
         java.io.File target)
  throws java.io.IOException
```

Converts eps file to pdf using epstopdf program.

Parameters:

`epsImage` - `File` pointing eps file to convert.
`target` - `File` where the converted pdf file should be saved.

Throws:

`IOException` - if something went wrong when accessing the files.

runDot

```
private java.io.File runDot(java.io.File dotInputFile,
         java.io.File outputFile,
         java.lang.String args)
```

This method runs the dot executable with the given dot input file and returns the location of the image file.

(continued from last page)

Parameters:

`dotInputFile` - `File` which should be executed with Dot.
`outputFile` - Path to the output file.
`args` - Command line arguments for dot.

Returns:

File where the image is located. `Null` is returned if something went wrog.

runDot

```
private java.io.File runDot(java.io.File dotInputFile,
         java.lang.String args)
```

Overloaded `runDot` method which uses the default temporary output file as a target for the file.

Parameters:

`dotInputFile` - `File` which should be executed with Dot.
`args` - Command line arguments for dot.

Returns:

`File` where the image is located. `Null` is returned if something went wrog.

See Also:

`runDot(File, File, String)`

runDot

```
private java.io.File runDot(java.io.File dotInputFile)
```

Overloaded `runDot` method. This will create the dot graph with default settings (PNG image with the paths defined in settings XML).

Parameters:

`dotInputFile` - `File` which should be executed with Dot.

Returns:

`File` where the image is located. `Null` is returned if something went wrog.

See Also:

`runDot(File, String)`

writeEntityMethods

```
private void writeEntityMethods(DotPanel.DotJob job,
         java.io.BufferedWriter writer,
         java.lang.String entity)
  throws java.io.IOException
```

Method for writing the methods of one entity to the dot file.

Parameters:

`job` - `DotJob` that contains the `AnalyzeModel` where the entity we are accessing is located.

(continued from last page)

`writer` - `BufferedWriter` to use for writing.
`entity` - Name of the entity which methods are being written.

Throws:
`IOException` - If something goes wrong when accessing the file.

writeEntityAttributes

```
private void writeEntityAttributes(DotPanel.DotJob job,
         java.io.BufferedWriter writer,
         java.lang.String entity)
  throws java.io.IOException
```

Method for writing the attributes of one entity to the dot file. Method also highlights the attribute relation if it's needed.

Parameters:
`job` - `DotJob` that contains the `AnalyzeModel` where the entity which attributes we are accessing is located.
`writer` - BufferedWriter to use for writing.
`entity` - Name of the entity whose attributes are being written.

Throws:
`IOException` - If something goes wrong when accessing the file.

writeEntityParents

```
private void writeEntityParents(DotPanel.DotJob job,
         java.io.BufferedWriter writer,
         java.lang.String entity)
  throws java.io.IOException
```

Method for writing the parents of one entity to the dot file. Method also highlights the parent-child relation if it's needed.

Parameters:
`job` - `DotJob` that contains the `AnalyzeModel` where the entity which parents we are accessing is located.
`writer` - `BufferedWriter` to use for writing.
`entity` - Name of the entity whose parents are being written.

Throws:
`IOException` - If something goes wrong when accessing the file.

writeEntities

```
private void writeEntities(DotPanel.DotJob job,
         java.io.BufferedWriter writer)
  throws java.io.IOException
```

Method for writing all entities to the dot file with all their attribute, method and parent relationships.

Parameters:
`job` - `DotJob` that contains the `AnalyzeModel` where are the entities we need to write.
`writer` - `BufferedWriter` to use for writing.

(continued from last page)

Throws:

`IOExceptionis` - thrown if something goes wrong when accessing the file.

createDotFile

```
private java.io.File createDotFile(DotPanel.DotJob job)
```

Writes the analyze model to an external file in dot syntax.

Parameters:

`job` - `DotJob` that contains the `AnalyzeModel` we need to write.

Returns:

`File` containing the analyze model in dot language. `Null` is returned if something went wrong.

mapCurrentModel

```
private void mapCurrentModel(DotPanel.DotJob job)
```

Method for mapping the current analyze model. This is required to figure out differences with the submodel that needs to be highlighted.

Parameters:

`job` - `DotJob` where mapping will be done.

mapHighlightRequest

```
private void mapHighlightRequest(DotPanel.DotJob job,
         boolean drawNewElements)
```

Method for mapping the submodel for highlight request.

Parameters:

`job` - `DotJob` to be handled.

`drawNewElements` - Indicates wether or not the new elements are supposed to be drawn in the highlighted model. New elements are those that exist in the highlight request but do not exist in the current analyze model.

highlight

```
public void highlight(AnalyzeModel highlight,
         boolean drawNewElements)
```

Method for highlighting submodels from the analyze model.

Parameters:

`highlight` - The submodel to be highlighted. If this argument is `null`, then all applied highlights are removed.

`drawNewElements` - Indicates wether or not those elements which do not exist in the current analyze model should be also drawn and highlighted.

(continued from last page)

highlight

```
public void highlight(AnalyzeModel highlight)
```

Method for highlighting submodels from the analyze model. As default we assume that new entities don't need to be drawn.

Parameters:
`highlight` - Submodel to be highlighted.

updateModel

```
private void updateModel(DotPanel.DotJob job)
```

Method for updating the model. This method pushes the given dot job to the top of the dot job stack and notifies the running thread about it. If no updation thread is running yet, one is created.

Parameters:
`job` - `DotJob` to be run next.

updateModel

```
public void updateModel(AnalyzeModel analyzeModel)
```

Method for updating the analyze model.

Parameters:
`analyzeModel` - New `AnalyzeModel`.

finalize

```
protected void finalize()
  throws java.lang.Throwable
```

This finalize method makes sure the thread running for panel update shuts down.

paintComponent

```
public void paintComponent(java.awt.Graphics g)
```

This method draws the graph generated by dot to the panel's canvas.

executeDot

```
private void executeDot()
```

(continued from last page)

This method should be executed in its own thread. This thread keeps running until `threadRunning` class variable is set to false.

It takes the latest job from the dot job stack and disposes all the other jobs in the stack at the same time. The newest job is then modeled and drawn to the canvas. After that the thread sleeps until waken again by notification [`executing.notify()`].

setTimerComponent

`public void` **`setTimerComponent`**`(javax.swing.JLabel timerComponent)`

`JLabel` where the updation time of this `DotPanel` is drawn.

Parameters:

`timerComponent` - `JLabel` on which you need to get the updation time to.

ucot.ui.gui.dot
Class DotPanel.DotJob

```
java.lang.Object
   |
   +-ucot.ui.gui.dot.DotPanel.DotJob
```

private class **DotPanel.DotJob**
extends java.lang.Object

This class is a container for a dot job which includes all required information for rendering both highlighted and regular dot graphs. A container like this is required when dot execution is done in threads.

Fields

analyzeModel

`private ucot.model.AnalyzeModel` **`analyzeModel`**

highlightModel

`private ucot.model.AnalyzeModel` **`highlightModel`**

entitiesToBeHighlighted

`private java.util.Set` **`entitiesToBeHighlighted`**

entitiesInCurrentModel

`private java.util.Set` **`entitiesInCurrentModel`**

entitiesRequestedForHighlighting

`private java.util.Set` **`entitiesRequestedForHighlighting`**

parentsToBeHighlighted

`private java.util.Set` **`parentsToBeHighlighted`**

(continued from last page)

parentsInCurrentModel

`private java.util.Set` **`parentsInCurrentModel`**

---

parentsRequestedForHighlighting

`private java.util.Set` **`parentsRequestedForHighlighting`**

---

methodsToBeHighlighted

`private java.util.Set` **`methodsToBeHighlighted`**

---

methodsInCurrentModel

`private java.util.Set` **`methodsInCurrentModel`**

---

methodsRequestedForHighlighting

`private java.util.Set` **`methodsRequestedForHighlighting`**

---

attributesToBeHighlighted

`private java.util.Set` **`attributesToBeHighlighted`**

---

attributesInCurrentModel

`private java.util.Set` **`attributesInCurrentModel`**

---

attributesRequestedForHighlighting

`private java.util.Set` **`attributesRequestedForHighlighting`**

Constructors

DotPanel.DotJob

`private` **`DotPanel.DotJob`**`()`

ucot.ui.gui.dot Class DotPanel.ExportJob

```
java.lang.Object
   |
   +-ucot.ui.gui.dot.DotPanel.ExportJob
```

private class **DotPanel.ExportJob**
extends java.lang.Object

Class for giving needed info for `executeExport()` method.

Needs to stored in own class cause `ExportJob`s are stored in stack before `executeExport()` accesses them.

Author:
tujupien

Fields

graph

`private java.awt.image.BufferedImage **graph**`

imageType

`private ucot.ui.gui.dot.DotPanel.ExportImageType **imageType**`

targetFile

`private java.io.File **targetFile**`

Constructors

DotPanel.ExportJob

`private **DotPanel.ExportJob**()`

ucot.ui.gui.dot
Class DotPanel.ExportImageType

```
java.lang.Object
   |
   +-java.lang.Enum
      |
      +-ucot.ui.gui.dot.DotPanel.ExportImageType
```

All Implemented Interfaces:
    java.io.Serializable, java.lang.Comparable

public static final class **DotPanel.ExportImageType**
extends java.lang.Enum

Imageformats that are usable when exporting graph as image.
Author:
    ilanliuk

Fields

PNG

`public static final ucot.ui.gui.dot.DotPanel.ExportImageType` **`PNG`**

JPG

`public static final ucot.ui.gui.dot.DotPanel.ExportImageType` **`JPG`**

EPS

`public static final ucot.ui.gui.dot.DotPanel.ExportImageType` **`EPS`**

PDF

`public static final ucot.ui.gui.dot.DotPanel.ExportImageType` **`PDF`**

Constructors

DotPanel.ExportImageType

`private` **`DotPanel.ExportImageType`**`()`

Methods

(continued from last page)

values

```
public final static DotPanel.ExportImageType[] values()
```

valueOf

```
public static DotPanel.ExportImageType valueOf(java.lang.String name)
```

Package ucot.ui.gui.tree.analyzemodeltree

Classes related to a tree that displays analyzemodel in a tree.

ucot.ui.gui.tree.analyzemodeltree
Class AbstractEntityTreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.analyzemodeltree.TreeItem
      |
      +-ucot.ui.gui.tree.analyzemodeltree.AbstractEntityTreeItem
```

All Implemented Interfaces:
java.lang.Comparable

Direct Known Subclasses:
AttributeTreeItem, ChildTreeItem, EntityTreeItem

---

public abstract class **AbstractEntityTreeItem**
extends TreeItem

This class represents all TreeItems in the AnalyzeModelTrre. It gathers common functionality shared among TreeItemss that represents entities in different levels in the tree.
Author:
UCOT

Fields

expand

`protected boolean **expand**`

Should this tree node be expandape (should it show its childs).

Constructors

AbstractEntityTreeItem

```
public AbstractEntityTreeItem(java.lang.String name,
                              AnalyzeTreeModel model,
                              TreeItem parent)
```

Constructor for the class.

Parameters:
`name` - The name of the entity.
`model` - The model where the entity resides.
`parent` - The parent node of this `TreeItem`.

Methods

getChildren

`public java.util.List **getChildren**()`

Returns the child items of this item.

getMethodTreeItems

`protected java.util.List **getMethodTreeItems**()`

Returns the `TreeItem`s for the entity's methods.

Returns:
List of `MethodTreeItem`s representing entity's method.

getAttributeTreeItems

`protected java.util.List **getAttributeTreeItems**()`

Returns the `TreeItem`s for the entity's attributes.

Returns:
List of `AttributeTreeItem`s representing entity's attributes.

getChildTreeItems

`protected java.util.List **getChildTreeItems**()`

Returns the `TreeItem`s for the entity's childs.

Returns:
List of `ChildTreeItem`s representing entity's childs.

getIcon

`public abstract javax.swing.Icon **getIcon**()`

Returns `Icon` for this `TreeItem`.

ucot.ui.gui.tree.analyzemodeltree
Class AnalyzeModelTree

```
java.lang.Object
  |
  +-java.awt.Component
      |
      +-java.awt.Container
          |
          +-javax.swing.JComponent
              |
              +-javax.swing.JTree
                  |
                  +-ucot.ui.gui.tree.analyzemodeltree.AnalyzeModelTree
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.Scrollable

public class **AnalyzeModelTree**
extends javax.swing.JTree

Customized `JTree` for showing `AnalyzeModel`s entity structure.

The data model of this tree is kept in `AnalyzeTreeModel`.

Author:
UCOT

See Also:
`javax.swing.JTree`

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`262462830033456547`**

panel

`private ucot.ui.gui.dot.DotPanel` **`panel`**

atm

`private ucot.ui.gui.tree.analyzemodeltree.AnalyzeTreeModel` **`atm`**

entitiesIcon

`protected static javax.swing.Icon` **`entitiesIcon`**

(continued from last page)

entityIcon

`protected static javax.swing.Icon **entityIcon**`

methodIcon

`protected static javax.swing.Icon **methodIcon**`

attributeIcon

`protected static javax.swing.Icon **attributeIcon**`

childIcon

`protected static javax.swing.Icon **childIcon**`

Constructors

AnalyzeModelTree

```
public AnalyzeModelTree(GraphicalUI owner)
```

Default constructor for `AnalyzeModelTree`.

Parameters:
`owner` - `GraphicalUI` where this `AnalyzeModelTree` is located.

Methods

createMouseListener

```
public void createMouseListener(ControlInterface core,
        EntityPropertiesDialog entityPropertiesDialog)
```

Creates `MouseListener` for `AnalyzeModelTree` to handle mouse clicking events over the tree.

Parameters:
`core` - `ControlInterface`
`entityPropertiesDialog` - `EntityPropertiesDialog` for editing entity properties.

getTreeSelectionListener

```
private javax.swing.event.TreeSelectionListener getTreeSelectionListener()
```

(continued from last page)

Creates and returns new `TreeSelectionListener` for `AnalyzeModelTree`.

This `TreeSelectionListener` tells `DotPanel` to highlight selected entities.

Returns:
`TreeSelectionListener`

setStructurePanel

`public void` **`setStructurePanel`**`(DotPanel dotPanel)`

Sets dotpanel

Parameters:
`dotPanel` - dotpanel to use

ucot.ui.gui.tree.analyzemodeltree
Class AnalyzeModelTree.AnalyzemodelTreeCellRenderer

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JLabel
               |
               +-javax.swing.tree.DefaultTreeCellRenderer
                  |
                  +-
ucot.ui.gui.tree.analyzemodeltree.AnalyzeModelTree.AnalyzemodelTreeCellRenderer
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.SwingConstants, javax.swing.tree.TreeCellRenderer

---

private class **AnalyzeModelTree.AnalyzemodelTreeCellRenderer**
extends javax.swing.tree.DefaultTreeCellRenderer

Custom `TreeCellRenderer` for rendering icons for treeitems.
Author:
ilanliuk

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`-1904905029126681422`**

Constructors

AnalyzeModelTree.AnalyzemodelTreeCellRenderer

`private` **`AnalyzeModelTree.AnalyzemodelTreeCellRenderer`**`()`

Methods

(continued from last page)

getTreeCellRendererComponent

```
public java.awt.Component getTreeCellRendererComponent(javax.swing.JTree tree,
         java.lang.Object value,
         boolean sel,
         boolean expanded,
         boolean leaf,
         int row,
         boolean hasFocus)
```

ucot.ui.gui.tree.analyzemodeltree
Class AnalyzeTreeModel

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.analyzemodeltree.AnalyzeTreeModel
```

All Implemented Interfaces:
java.util.Observer, javax.swing.tree.TreeModel

public class **AnalyzeTreeModel**
extends java.lang.Object
implements javax.swing.tree.TreeModel, java.util.Observer

This is the `TreeModel` for `JTree` representing the `AnalyzeModel`. It is able to listen the `Udation` messages coming from the `AnalyzeModel` and its ask the tree to update itself when the model is updated.

This tree and the `TreeItem` classes it uses are actually a adapter for the analyze model because the `TreeItems` are created dynamically based on the status of the actual model.

Known problems

Because of the dynamic nature of this tree there are problems informing the actual tree about the modifications and currently the tree is invaliated completely after every minor change. Modify `Update(Observer,Object)`-method to to fix this.
See Also:
`AnalyzeModel`, `javax.swing.JTree`, `javax.swing.tree.TreeModel`, `TreeItem`
Author:
pajumasu

Fields

owner

`private ucot.ui.gui.GraphicalUI` **`owner`**

model

`ucot.model.AnalyzeModel` **`model`**

listeners

`java.util.Set` **`listeners`**

(continued from last page)

root

```
ucot.ui.gui.tree.analyzemodeltree.TreeItem root
```

Constructors

AnalyzeTreeModel

```
public AnalyzeTreeModel(GraphicalUI owner)
```

Creates the object.

Parameters:
`owner` - The owner of this component.

Methods

update

```
public void update(java.util.Observable o,
         java.lang.Object arg)
```

getAnalyzeModel

```
public AnalyzeModel getAnalyzeModel()
```

Returns the analyze model which this item models.

Returns:
the analyze model which this item models.

updateAnalyzeModel

```
public void updateAnalyzeModel()
```

Updates the `AnalyzeModel` from the `Core` component. This should be called if the `AnalyzeModel in the` Core is changed to other object.

getRoot

```
public java.lang.Object getRoot()
```

getChild

```
public java.lang.Object getChild(java.lang.Object parent,
         int index)
```

getChildCount

```
public int getChildCount(java.lang.Object parent)
```

isLeaf

```
public boolean isLeaf(java.lang.Object node)
```

valueForPathChanged

```
public void valueForPathChanged(javax.swing.tree.TreePath path,
         java.lang.Object newValue)
```

getIndexOfChild

```
public int getIndexOfChild(java.lang.Object parent,
         java.lang.Object child)
```

addTreeModelListener

```
public void addTreeModelListener(javax.swing.event.TreeModelListener l)
```

removeTreeModelListener

```
public void removeTreeModelListener(javax.swing.event.TreeModelListener l)
```

sendTreeNodesChanged

```
protected void sendTreeNodesChanged(javax.swing.event.TreeModelEvent event)
```

Sends `treeNodesChanged` event to all `TreeModelListeners` listening this object.

Parameters:
`event` - The actual event to be sended.

See Also:
`TreeModelListener.treeNodesChanged(javax.swing.event.TreeModelEvent)`

sendTreeNodesInserted

```
protected void sendTreeNodesInserted(javax.swing.event.TreeModelEvent event)
```

(continued from last page)

Sends `treeNodesInserted` event to all `TreeModelListeners` listening this object.

Parameters:
`event` - The actual event to be sended.

See Also:
`TreeModelListener.treeNodesInserted(javax.swing.event.TreeModelEvent)`

---

sendTreeNodesRemoved

```
protected void sendTreeNodesRemoved(javax.swing.event.TreeModelEvent event)
```

Sends `treeNodesRemoved` event to all `TreeModelListeners` listening this object.

Parameters:
`event` - The actual event to be sended.

See Also:
`TreeModelListener.treeNodesRemoved(javax.swing.event.TreeModelEvent)`

---

sendTreeStructureChanged

```
protected void sendTreeStructureChanged(javax.swing.event.TreeModelEvent event)
```

Sends `treeStructureChanged` event to all `TreeModelListeners` listening this object.

Parameters:
`event` - The actual event to be sended.

See Also:
`TreeModelListener.treeStructureChanged(javax.swing.event.TreeModelEvent)`

ucot.ui.gui.tree.analyzemodeltree
Class AttributeTreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.analyzemodeltree.TreeItem
      |
      +-ucot.ui.gui.tree.analyzemodeltree.AbstractEntityTreeItem
          |
          +-ucot.ui.gui.tree.analyzemodeltree.AttributeTreeItem
```

All Implemented Interfaces:
java.lang.Comparable

---

public class **AttributeTreeItem**
extends AbstractEntityTreeItem

This class represents entity object that resides as an attribute in the analyze model tree.
Author:
pajumasu

Constructors

AttributeTreeItem

```
public AttributeTreeItem(java.lang.String name,
                         AnalyzeTreeModel model,
                         TreeItem parent)
```

Constructs the object.

Parameters:
`name` - The name of the attribute entity.
`model` - The `TreeModel` this item belongs to.
`parent` - The parent node.

Methods

getIcon

```
public javax.swing.Icon getIcon()
```

Returns `Icon` for this `TreeItem`.

ucot.ui.gui.tree.analyzemodeltree
Class ChildTreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.analyzemodeltree.TreeItem
      |
      +-ucot.ui.gui.tree.analyzemodeltree.AbstractEntityTreeItem
          |
          +-ucot.ui.gui.tree.analyzemodeltree.ChildTreeItem
```

All Implemented Interfaces:
java.lang.Comparable

---

public class **ChildTreeItem**
extends AbstractEntityTreeItem

Represents child entity in analyze model tree.
Author:
pajumasu

Constructors

ChildTreeItem

```
public ChildTreeItem(java.lang.String name,
                     AnalyzeTreeModel model,
                     TreeItem parent)
```

Constucts the object.

Parameters:
`name` - The name of the child entity.
`model` - The `TreeModel` this item belongs to.
`parent` - The parent node.

Methods

getIcon

```
public javax.swing.Icon getIcon()
```

Returns `Icon` for this `TreeItem`.

ucot.ui.gui.tree.analyzemodeltree
Class EntitiesTreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.analyzemodeltree.TreeItem
      |
      +-ucot.ui.gui.tree.analyzemodeltree.EntitiesTreeItem
```

All Implemented Interfaces:
java.lang.Comparable

---

public class **EntitiesTreeItem**
extends TreeItem

This represents the entities `TreeItem` that contains all the entities of the analyze model.
Author:
pajumasu

Constructors

EntitiesTreeItem

```
public EntitiesTreeItem(AnalyzeTreeModel model,
                        TreeItem parent)
```

Constructs the object.

Parameters:
`model` - The `TreeModel` this item belongs to.
`parent` - The parent node.

Methods

getIcon

```
public javax.swing.Icon getIcon()
```

Returns `Icon` for this `TreeItem`.

---

getChildren

```
public java.util.List getChildren()
```

Returns the child items of this item.

---

isEntityInfluencingEntity

```
private boolean isEntityInfluencingEntity(java.lang.String sourceEntity,
        java.lang.String targetEntity)
```

Retuns true is targetEntity is referenced by sourceEntity. TargetEntity os referenced if it is sourceEntitys attribute, child or is target of sourceEntitys methods.

(continued from last page)

Parameters:

`sourceEntity` - name of the entity that is influencing.
`targetEntity` - name of the entity that is influenced.

Returns:

true if sourceEntity is influencing targetEntity.

ucot.ui.gui.tree.analyzemodeltree
Class EntityTreeItem

```
java.lang.Object
   |
   +-ucot.ui.gui.tree.analyzemodeltree.TreeItem
      |
      +-ucot.ui.gui.tree.analyzemodeltree.AbstractEntityTreeItem
         |
         +-ucot.ui.gui.tree.analyzemodeltree.EntityTreeItem
```

All Implemented Interfaces:
java.lang.Comparable

public class **EntityTreeItem**
extends AbstractEntityTreeItem

This class represents entity tree item.
Author:
pajumasu

Constructors

EntityTreeItem

```
public EntityTreeItem(java.lang.String name,
                      AnalyzeTreeModel model,
                      TreeItem parent)
```

Constructs the object.

Parameters:
`name` - The name of the entity.
`model` - The `TreeModel` this item belongs to.
`parent` - The paren of this node.

EntityTreeItem

```
public EntityTreeItem(java.lang.String name,
                      AnalyzeTreeModel model,
                      TreeItem parent,
                      boolean expand)
```

Constructs the object.

Parameters:
`name` - The name of the entity.
`model` - The `TreeModel` this item belongs to.
`parent` - The paren of this node.
`expand` - Should the node be expandable (Show childs or not).

Methods

(continued from last page)

getIcon

`public javax.swing.Icon **getIcon**()`

Returns `Icon` for this `TreeItem`.

ucot.ui.gui.tree.analyzemodeltree
Class MethodTreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.analyzemodeltree.TreeItem
      |
      +-ucot.ui.gui.tree.analyzemodeltree.MethodTreeItem
```

All Implemented Interfaces:
java.lang.Comparable

---

public class **MethodTreeItem**
extends TreeItem

This class represents method `TreeItem`.

Fields

entity

`java.lang.String` **`entity`**

The name of the entity that this method belongs to.

Constructors

MethodTreeItem

```
public MethodTreeItem(java.lang.String entity,
                      java.lang.String method,
                      AnalyzeTreeModel model,
                      TreeItem parent)
```

Costructs the object.

Parameters:
`entity` - The name of the entity that the method belongs to.
`method` - The name of the menthod.
`model` - The `TreeModel` this item belongs to.
`parent` - The parent node.

Methods

getChildren

`public java.util.List` **`getChildren`**`()`

Returns the child items of this item.

(continued from last page)

getIcon

```
public javax.swing.Icon getIcon()
```

Returns `Icon` for this `TreeItem`.

ucot.ui.gui.tree.analyzemodeltree
Class TreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.analyzemodeltree.TreeItem
```

All Implemented Interfaces:
java.lang.Comparable

Direct Known Subclasses:
AbstractEntityTreeItem, EntitiesTreeItem, MethodTreeItem

---

public abstract class **TreeItem**
extends java.lang.Object
implements java.lang.Comparable

`TreeItem` is basic building block of `AnalyzeTreeModel`. Its subclasses are used to create dynmic internal model of the analyzemodel which can be used like a tree. By using these classes `AnalyzeTreeModel` can serve the actual tree representing the analyze model.
See Also:
`javax.swing.JTree`, `AnalyzeModel`

Fields

name

```
private java.lang.String name
```

The name of the item. This should be shown by the tree

---

model

```
private ucot.ui.gui.tree.analyzemodeltree.AnalyzeTreeModel model
```

The model this `TreeItem` fetches its data (mostly childs).

---

parent

```
private ucot.ui.gui.tree.analyzemodeltree.TreeItem parent
```

The parent `TreeItem` of this node.

Constructors

TreeItem

```
public TreeItem(java.lang.String name,
                AnalyzeTreeModel model,
                TreeItem parent)
```

Constructs the object.

(continued from last page)

Parameters:
`name` - The name of the item.
`model` - The `TreeModel` this item belongs to.
`parent` - The parent node for this item.

Methods

getTreeModel

`public AnalyzeTreeModel **getTreeModel**()`

---

getAnalyzeModel

`public AnalyzeModel **getAnalyzeModel**()`

Returns the analyze model used by this item.

Returns:
the analyze model used by this item.

See Also:
`AnalyzeModel`

---

getName

`public java.lang.String **getName**()`

Returns the name of this item.

Returns:
the name of this item.

---

toString

`public java.lang.String **toString**()`

Returns the name of this item.

See Also:
`getName()`

---

getChildren

`public java.util.List **getChildren**()`

Returns the child items of this item.

Returns:
The childs.

---

getPath

`public java.util.List **getPath**()`

(continued from last page)

Returns the path to this item inside the tree. Path is list of tree items where the most higest parent is first and the node ask to return the path is last in the list.

Returns:
List of `TreeItem` which represents the path to this item.

getParent

```
public TreeItem getParent()
```

Returns the parent item of this item.

Returns:
the parent item of this item.

getIcon

```
public abstract javax.swing.Icon getIcon()
```

Returns `Icon` for this `TreeItem`.

Returns:
`Icon` for this `TreeItem`.

compareTo

```
public int compareTo(java.lang.Object other)
```

Compares this treeitem to another

Parameters:
`other` - treeitem to compare to

Returns:
0 if the treeitems are equal

Package ucot.ui.gui.tree.usecasetree

Classes related to tree that displays loaded usecases.

ucot.ui.gui.tree.usecasetree
Class FilesTreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.usecasetree.TreeItem
     |
     +-ucot.ui.gui.tree.usecasetree.FilesTreeItem
```

public class **FilesTreeItem**
extends TreeItem

Root node for `UseCaseTree`.
Author:
ilanliuk

Fields

directoryIcon

`private static javax.swing.Icon` **`directoryIcon`**

Constructors

FilesTreeItem

```
public FilesTreeItem(UseCaseTreeModel treemodel,
                     TreeItem parent)
```

Default constructor for FilesTreeItem.

Parameters:
`treemodel` - `UseCaseTreeModel` that uses this `FilesTreeItem`.
`parent` - parent node of this node.

Methods

getIcon

`public javax.swing.Icon` **`getIcon`**`()`

Returns `Icon` for this `TreeItem`.

getName

`public java.lang.String` **`getName`**`()`

(continued from last page)

Returns name of this item.

getChildren

```
public java.util.List getChildren()
```

Return children of this `TreeItem` in a `List`.

getDirectoryIcon

```
private static javax.swing.Icon getDirectoryIcon()
```

Returns file systems default directory icon.

Ask file systems default directory `Icon` from `FileSystemView`.

Returns:
File systems `Icon` for directory.

See Also:
`javax.swing.filechooser.FileSystemView`

ucot.ui.gui.tree.usecasetree Class FileTreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.usecasetree.TreeItem
      |
      +-ucot.ui.gui.tree.usecasetree.FileTreeItem
```

public class **FileTreeItem**
extends TreeItem

`TreeItem` to hold data of source containing usecases.
Author:
ilanliuk

Fields

iconsForFileExtensions

`private static java.util.Map iconsForFileExtensions`

url

`private java.net.URL url`

Constructors

FileTreeItem

```
public FileTreeItem(UseCaseTreeModel treemodel,
                    TreeItem parent,
                    java.net.URL url)
```

Default constructor for `FileTreeItem`.

Parameters:
`treemodel` - `UseCaseTreeModel` wich uses this `FileTreeItem`.
`parent` - `FilesTreeItem` wich is parent of this.
`url` - `URL` to source this `FileTreeItem` contains.

Methods

getName

`public java.lang.String getName()`

(continued from last page)

Returns name of this item.

getChildren

`public java.util.List **getChildren**()`

Return children of this `TreeItem` in a `List`.

getIcon

`public javax.swing.Icon **getIcon**()`

Returns `Icon` for this `TreeItem`.

getFilesystemIcon

`private static javax.swing.Icon **getFilesystemIcon**(java.lang.String s)`

Returns file systems default icon for file described in string.

Ask file systems default `Icon` for given file from `FileSystemView`.

Parameters:
`s` - name of the file we want to get `Icon` for.

Returns:
`Icon` for given file.

See Also:
`javax.swing.filechooser.FileSystemView`

getUrl

`public java.net.URL **getUrl**()`

Returns `URL` of this `FileTreeItem`.

Returns:
`URL` of this `FileTreeItem`.

ucot.ui.gui.tree.usecasetree
Class TreeItem

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.usecasetree.TreeItem
```

Direct Known Subclasses:
FilesTreeItem, FileTreeItem, UseCaseTreeItem

public abstract class **TreeItem**
extends java.lang.Object

`Abstract` superclass for all `UseCaseTree`s tree nodes.
Author:
ilanliuk

Fields

data

```
java.lang.Object data
```

name

```
java.lang.String name
```

treemodel

```
ucot.ui.gui.tree.usecasetree.UseCaseTreeModel treemodel
```

parent

```
ucot.ui.gui.tree.usecasetree.TreeItem parent
```

Constructors

TreeItem

```
public TreeItem(UseCaseTreeModel treemodel,
                TreeItem parent)
```

Constructor for TreeItem.

(continued from last page)

Parameters:
`treemodel` - `UseCaseTreeModel`
`parent` - `TreeItem` wich is parent of this `TreeItem`.

Methods

getTreeModel

`public UseCaseTreeModel **getTreeModel**()`

Returns treemodel.

Returns:
`UseCaseTreeModel`

---

getUseCaseCollection

`public UseCaseCollection **getUseCaseCollection**()`

Returns `UseCaseCollection` of the `UseCaseTreeModel`.

Returns:
`UseCaseCollection`

---

getName

`public abstract java.lang.String **getName**()`

Returns name of this item.

Returns:
`String` name.

---

getData

`public java.lang.Object **getData**()`

Returns data contained by this TreeItem.

Returns:
Data of this `TreeItem` as `Object`.

---

toString

`public java.lang.String **toString**()`

(continued from last page)

getChildren

`public java.util.List **getChildren**()`

Return children of this `TreeItem` in a `List`.

Returns:
`List` of children.

---

getPath

`public java.util.List **getPath**()`

Returns path to this item as a `List` of `TreeItem`s.

Returns:
`List` of items in the path.

---

isLeaf

`public boolean **isLeaf**()`

Returns is this `TreeItem` leaf-node or not.

Returns:
is this `TreeItem` leaf-node or not

---

getIcon

`public abstract javax.swing.Icon **getIcon**()`

Returns `Icon` for this `TreeItem`.

Returns:
`Icon` for this `TreeItem`.

ucot.ui.gui.tree.usecasetree
Class UseCaseTree

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JTree
               |
               +-ucot.ui.gui.tree.usecasetree.UseCaseTree
```

All Implemented Interfaces:

java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.Scrollable

---

public class **UseCaseTree**
extends javax.swing.JTree

`JTree` for displaying usecases that are loaded into program.

The data model of this tree is kept in `UseCaseTreeModel`.

Root node of this tree is `FilesTreeItem`. Root has all `UseCase` sources loaded into program as children as `FileTreeItem`s. `FileTreeItem` holds `UseCase`s from that source as it's child nodes as `UseCaseTreeItem`s.

Author:
ilanliuk

See Also:
`javax.swing.JTree`

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

Constant value: **`-6655430305570481199`**

---

popupMenuActionListener

`private java.awt.event.ActionListener` **`popupMenuActionListener`**

---

useCaseTreeModel

`private ucot.ui.gui.tree.usecasetree.UseCaseTreeModel` **`useCaseTreeModel`**

(continued from last page)

dotPanel

`private ucot.ui.gui.dot.DotPanel` **`dotPanel`**

core

`private ucot.core.ControlInterface` **`core`**

useCasePanel

`private ucot.ui.UseCasePanelInterface` **`useCasePanel`**

usecaseIcon

`protected static javax.swing.Icon` **`usecaseIcon`**

usecaseLeafIcon

`protected static javax.swing.Icon` **`usecaseLeafIcon`**

usecaseIconNotInModel

`protected static javax.swing.Icon` **`usecaseIconNotInModel`**

usecaseLeafIconNotInModel

`protected static javax.swing.Icon` **`usecaseLeafIconNotInModel`**

ADD_SINGLE_POPUP_ACTION

`private static final java.lang.String` **`ADD_SINGLE_POPUP_ACTION`**

Constant value: **`ADD_SINGLE`**

ADD_SINGLE_WITH_POPUP_ACTION

`private static final java.lang.String` **`ADD_SINGLE_WITH_POPUP_ACTION`**

Constant value: **`ADD_SINGLE_WITH`**

(continued from last page)

ADD_SUBTREE_POPUP_ACTION

`private static final java.lang.String` **`ADD_SUBTREE_POPUP_ACTION`**

Constant value: **`ADD_SUBTREE`**

ADD_SUBTREE_WITH_POPUP_ACTION

`private static final java.lang.String` **`ADD_SUBTREE_WITH_POPUP_ACTION`**

Constant value: **`ADD_SUBTREE_WITH`**

Constructors

UseCaseTree

`public` **`UseCaseTree`**`(ControlInterface core)`

Default constructor for `UseCaseTree`.

Parameters:
`core` - `ControlInterface` which holds the `UseCaseCollection` which contains the `UseCases` we want to show.

Methods

getSubtreeInVector

`private java.util.Vector` **`getSubtreeInVector`**`(UseCaseTreeItem item)`

Method to add `UseCase` pointed at `UseCaseTreeItem` and it's sub use cases into `Vector`.

Parameters:
`item` - `UseCaseTreeItem` to add into `Vector`.

Returns:
`Vector` containing `UseCase`s.

getSubtreeInVector

`private java.util.Vector` **`getSubtreeInVector`**`(FileTreeItem item)`

Method to add all `UseCases` from file pointed out at `FileTreeItem` into `Vector`.

Parameters:
`item` - `FileTreeItem` which contains the source from which we want to get the use cases.

Returns:
`Vector` containing `UseCases` from given source.

(continued from last page)

getSubtreeInVector

`private java.util.Vector getSubtreeInVector(TreeItem item)`

Method to add `UseCase`s from subtree of this `TreeItem` into `Vector`.

Parameters:
`item` - `TreeItem` which sub use cases we want to get.

Returns:
`Vector` containing `UseCase`s.

getSubtreeInVector

`private java.util.Vector getSubtreeInVector(FilesTreeItem item)`

Method to add all use case sources that are loaded into program into `Vector`.

Parameters:
`item` - `FilesTreeItem`

Returns:
`Vector` containing `UseCase`s.

addToAnalyzeModel

```
private void addToAnalyzeModel(UseCase usecase,
         ParserInterface parser,
         HeuristicInterface heuristic)
```

Adds single `UseCase` into analyzemodel with given `ParserInterface` and `HeuristicInterface`.

Parameters:
`usecase` - `UseCase` to add into model.
`parser` - `ParserInterface` to use for parsing `UseCase`.
`heuristic` - `HeuristicInterface` to use on the use case.

addToAnalyzeModel

`private void addToAnalyzeModel(UseCase usecase)`

Adds single `UseCase` into analyzemodel with default `ParserInterface` and `HeuristicInterface`.

Parameters:
`usecase` - `UseCase` to add into model.

(continued from last page)

addToAnalyzeModel

```
private void addToAnalyzeModel(java.util.Vector usecases,
         ParserInterface parser,
         HeuristicInterface heuristic)
```

Add `Vector` of use cases into analyzemodel with given `ParserInterface` and `HeuristicInterface`.

Parameters:
`usecases` - `Vector` of `UseCase`s to add.
`parser` - `ParserInterface` to use.
`heuristic` - `HeuristicInterface` to use.

---

addToAnalyzeModel

```
private void addToAnalyzeModel(java.util.Vector usecases)
```

Add `Vector` of use cases into analyzemodel with default `ParserInterface` and `HeuristicInterface`.

Parameters:
`usecases` - `Vector` of `UseCase`s to add.

---

addToAnalyzeModelWith

```
private void addToAnalyzeModelWith(UseCase usecase)
```

Method to ask from user which `ParserInterface` and `HeuristicInterface` should be used to add use case into model. Uses `AddToModelWithDialog` to ask which the wanted `ParserInterface` and `HeuristicInterface` are.

Parameters:
`usecase` - `UseCase` to add into model.

---

addToAnalyzeModelWith

```
private void addToAnalyzeModelWith(java.util.Vector usecases)
```

Method to ask from user which `ParserInterface` and `HeuristicInterface` should be used to add usecases into model. Uses `AddToModelWithDialog` to ask which the wanted `ParserInterface` and `HeuristicInterface` are.

Parameters:
`usecases` - `Vector` of `UseCase`s to add.

---

getTreeSelectionListener

```
private javax.swing.event.TreeSelectionListener getTreeSelectionListener()
```

Creates and returns `TreeSelectionListener` for `UseCaseTree`.

(continued from last page)

Returns:
`TreeSelectionListener`

getMouseListener

`private java.awt.event.MouseListener **getMouseListener**()`

Creates and returns `MouseListener` for `UseCaseTree`.

Returns:
`MouseListener`

getPopupMenuFor

`private javax.swing.JPopupMenu **getPopupMenuFor**(TreeItem treeitem)`

Creates and returns `JPopupMenu` for given `TreeItem`.

MenuItems in popup menu are depending of the type of `TreeItem` we need the menu for.

Parameters:
`treeitem` - `TreeItem` we need a popup menu for.

Returns:
`JPopupMenu`

getRefreshFileActionListener

`private java.awt.event.ActionListener **getRefreshFileActionListener**()`

Returns `ActionListener` for refreshing files action from `UseCaseTree` popup menu.

Returns:
`ActionListener`

getActionListenerForPopupMenu

`private java.awt.event.ActionListener **getActionListenerForPopupMenu**()`

Return `ActionListener` for `UseCaseTrees` popup menu

Returns:
`ActionListener` for popup menu

setDotPanel

`public void **setDotPanel**(DotPanel panel)`

Method to set the `DotPanel` which is used to display the model.

Parameters:
`panel` - `DotPanel`

setUseCasePanel

```
public void setUseCasePanel(UseCasePanelInterface useCasePanel)
```

Method to set the `UseCasePanelInterface` which is used to display steps of the selected use case.

Parameters:
`useCasePanel` - `UseCasePanelInterface`.

clear

```
public void clear()
```

Clears contest of this tree.

ucot.ui.gui.tree.usecasetree
Class UseCaseTree.UseCaseTreeCellRenderer

```
java.lang.Object
   |
   +-java.awt.Component
      |
      +-java.awt.Container
         |
         +-javax.swing.JComponent
            |
            +-javax.swing.JLabel
               |
               +-javax.swing.tree.DefaultTreeCellRenderer
                  |
                  +-ucot.ui.gui.tree.usecasetree.UseCaseTree.UseCaseTreeCellRenderer
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable, javax.accessibility.Accessible, javax.swing.SwingConstants, javax.swing.tree.TreeCellRenderer

private class **UseCaseTree.UseCaseTreeCellRenderer**
extends javax.swing.tree.DefaultTreeCellRenderer

Custom `TreeCellRenderer` for highlighting tree nodes and adding custom icons for them.

Fields

serialVersionUID

`private static final long serialVersionUID`

Constant value: **-3044905713499088318**

defaultFont

`private java.awt.Font defaultFont`

highlightFont

`private java.awt.Font highlightFont`

Constructors

UseCaseTree.UseCaseTreeCellRenderer

`private UseCaseTree.UseCaseTreeCellRenderer()`

(continued from last page)

Methods

getTreeCellRendererComponent

```
public java.awt.Component getTreeCellRendererComponent(javax.swing.JTree tree,
         java.lang.Object value,
         boolean sel,
         boolean expanded,
         boolean leaf,
         int row,
         boolean hasFocus)
```

ucot.ui.gui.tree.usecasetree
Class UseCaseTreeItem

```
java.lang.Object
   |
   +-ucot.ui.gui.tree.usecasetree.TreeItem
      |
      +-ucot.ui.gui.tree.usecasetree.UseCaseTreeItem
```

public class **UseCaseTreeItem**
extends TreeItem

`TreeItem` to contain single `UseCase`.
Author:
ilanliuk

Fields

usecase

`private ucot.input.UseCase` **`usecase`**

Constructors

UseCaseTreeItem

```
public UseCaseTreeItem(UseCaseTreeModel treemodel,
                       TreeItem parent,
                       UseCase usecase)
```

Default constructor for `UseCaseTreeItem`.

Parameters:
`treemodel` - `UseCaseTreeModel` that uses this `TreeItem`.
`parent` - Parent `TreeItem` of this `TreeItem`.
`usecase` - `UseCase` this `UseCaseTreeItem` should contain.

Methods

getName

`public java.lang.String` **`getName`**`()`

Returns name of this item.

isUsecaseInAnalyzemodel

`public boolean` **`isUsecaseInAnalyzemodel`**`()`

(continued from last page)

Returns is `UseCase` this `UseCaseTreeItem` contains in analyzemodel.

Returns:
is `UseCase` this `UseCaseTreeItem` contains in analyzemodel.

getUseCase

`public UseCase getUseCase()`

Return `UseCase` this `UseCaseTreeItem` contains.

Returns:
`UseCase` this `UseCaseTreeItem` contains.

getChildren

`public java.util.List getChildren()`

Return children of this `TreeItem` in a `List`.

getIcon

`public javax.swing.Icon getIcon()`

Returns `Icon` for this `TreeItem`.

ucot.ui.gui.tree.usecasetree
Class UseCaseTreeModel

```
java.lang.Object
  |
  +-ucot.ui.gui.tree.usecasetree.UseCaseTreeModel
```

All Implemented Interfaces:
java.util.Observer, javax.swing.tree.TreeModel

public class **UseCaseTreeModel**
extends java.lang.Object
implements javax.swing.tree.TreeModel, java.util.Observer

Implementation of `TreeModel` for holding data model of `UseCaseTree`.
Author:
ilanliuk

Fields

listeners

`protected java.util.Set **listeners**`

collection

`private ucot.input.UseCaseCollection **collection**`

root

`private ucot.ui.gui.tree.usecasetree.TreeItem **root**`

Constructors

UseCaseTreeModel

`public **UseCaseTreeModel**(UseCaseCollection u)`

Default constructor for `UseCaseTreeModel`.

Adds given `UseCaseCollection` observed to get notified when data of the `UseCaseCollection` changes.

Parameters:
`u` - `UseCaseCollection` from wich this `TreeModel` is constructed.

(continued from last page)

Methods

update

```
public void update(java.util.Observable o,
        java.lang.Object arg)
```

getRoot

```
public java.lang.Object getRoot()
```

getChild

```
public java.lang.Object getChild(java.lang.Object parent,
        int index)
```

getChildCount

```
public int getChildCount(java.lang.Object parent)
```

isLeaf

```
public boolean isLeaf(java.lang.Object node)
```

valueForPathChanged

```
public void valueForPathChanged(javax.swing.tree.TreePath path,
        java.lang.Object newValue)
```

getIndexOfChild

```
public int getIndexOfChild(java.lang.Object parent,
        java.lang.Object child)
```

addTreeModelListener

```
public void addTreeModelListener(javax.swing.event.TreeModelListener l)
```

removeTreeModelListener

```
public void removeTreeModelListener(javax.swing.event.TreeModelListener l)
```

(continued from last page)

getUseCaseCollection

public UseCaseCollection **getUseCaseCollection**()

Returns `UseCaseCollection` from wich this `TreeModel` is contructed.

Returns:
`UseCaseCollection` from wich this `TreeModel` is contructed.

clear

public void **clear**()

Method for clearing this `TreeModel`.

When called clears contest of this data model and also clears contest of `UseCaseCollection` from wich this `TreeModel` is constructed.

Package
ucot.utils

Miscellaneous tools used by other classes.

ucot.utils
Class CustomFileFilter

```
java.lang.Object
   |
   +-javax.swing.filechooser.FileFilter
      |
      +-ucot.utils.CustomFileFilter
```

public class **CustomFileFilter**
extends javax.swing.filechooser.FileFilter

CustomFileFilter for JFileChooser that accepts extensions that are given in constructor.

How to use:

```
JFileChooser f = new JFileChooser(".");
  f.addChoosableFileFilter(
      new CustomFileFilter(".jpg , .jpeg", new String[] {"jpg", "jpeg"}) );
```

Author:
ilanliuk

Fields

description

`private java.lang.String` **`description`**

extensions

`private java.util.Vector` **`extensions`**

Constructors

CustomFileFilter

```
public CustomFileFilter(java.lang.String description,
                        java.lang.String[] acceptableExtensions)
```

Creates new CustomFileFilter for filtering files in JFileChooser.

Parameters:
`description` - short description of what filter accepts (for example "image files" etc.)
`acceptableExtensions` - array of extensions that are accepted by filter

(continued from last page)

Methods

parseExtensions

`private void **parseExtensions**(java.lang.String[] extensions)`

Creates vector from given String-array

Parameters:

`extensions` - aray of String

accept

`public boolean **accept**(java.io.File f)`

getDescription

`public java.lang.String **getDescription**()`

getExtensions

`public java.util.Vector **getExtensions**()`

Returns extensions this FileFilter accepts in a vector.

Returns:

Vector of acceptable extensions

ucot.utils
Class FileTools

```
java.lang.Object
  |
  +-ucot.utils.FileTools
```

public class **FileTools**
extends java.lang.Object

This class contains some operations for file handling.

Fields

FILE_NOT_FOUND

`public static final java.lang.String` **`FILE_NOT_FOUND`**

Message for file not found -situation.

FILE_NOT_READABLE

`public static final java.lang.String` **`FILE_NOT_READABLE`**

Message for file not readable -situation.

FILE_NOT_WRITABLE

`public static final java.lang.String` **`FILE_NOT_WRITABLE`**

Message for file not writable -situation.

OPERATION_FORBIDDEN

`public static final java.lang.String` **`OPERATION_FORBIDDEN`**

Message for operation not permited -situation.

Constructors

FileTools

`public` **`FileTools`**`()`

Methods

copyFile

```
public static void copyFile(java.io.File from,
          java.io.File to)
   throws java.io.IOException
```

(continued from last page)

Method copies file. It copies from source to target file. If the target file exists its is overwriten. If the target file is direcoty then the source file is copied with its original name under the target directory.

Parameters:

`from` - The source file to copy from. It must exist.

`to` - The target file or directory.

Throws:

`IOException`

copyFile

```
public static void copyFile(java.net.URL from,
        java.net.URL to)
  throws java.io.IOException
```

Copies file represented by url to the the file represented by anaother url. Uses `copyFile(File,File)`-method to perform the actual copy.

Parameters:

`from` - The URL of source file.

`to` - The URL of target file or directory.

Throws:

`IOException`

See Also:

`copyFile(File, File)`

copyFile

```
public static void copyFile(java.lang.String from,
        java.lang.String to)
  throws java.io.IOException
```

Copies file represented by path to target path. Uses `copyFile(File,File)`-method to perform the actual copy.

Parameters:

`from` - The path of the source file.

`to` - The path of the target file or directory.

Throws:

`IOException`

See Also:

`copyFile(File, File)`

changeExtension

```
public static java.io.File changeExtension(java.io.File original,
        java.lang.String extension)
```

Changes given file's extension

Parameters:

`original` - original file

`extension` - new extension for the file

(continued from last page)

Returns:
file with changed extension

ucot.utils
Class PropertiesTools

```
java.lang.Object
  |
  +-ucot.utils.PropertiesTools
```

public class **PropertiesTools**
extends java.lang.Object

Tools for properties management. Saving and loading from file and such.
Author:
UCOT

Fields

propertiesFile

```
private static final java.io.File propertiesFile
```

propertiesURL

```
public static final java.net.URL propertiesURL
```

Constructors

PropertiesTools

```
public PropertiesTools()
```

Methods

getPropertiesURL

```
private static java.net.URL getPropertiesURL()
```

Method for creating the URL from the properties file.

Returns:
URL to the properties file.

merge

```
public static java.util.Properties merge(java.util.Properties overriding,
         java.util.Properties virtual)
```

Method for merging two sets of properties in such a way where overriding properties' values will override any values in virtual properties set that have the same key as in overriding set.

(continued from last page)

Parameters:

`overriding` - Overriding properties set.
`virtual` - Virtual properties set.

Returns:

Merged properties set.

---

saveProperties

```
public static void saveProperties(java.util.Properties properties,
         java.net.URL propertiesURL)
  throws java.io.IOException
```

Method for saving current settings to the properties XML file. Notice that only the values for keys given in properties will be changed and all other possible keys and values in propertiesFile will remain untouched.

Parameters:

`properties` - Properties to be saved.
`propertiesURL` - Target file.

---

saveProperties

```
public static void saveProperties(java.util.Properties properties)
  throws java.io.IOException
```

Method for (re)saving properties to the current properties XML file.

Parameters:

`properties` - Properties to be saved.

---

loadProperties

```
public static java.util.Properties loadProperties(java.util.Enumeration
propertiesKeys,
         java.net.URL propertiesURL)
  throws java.io.IOException
```

Method for loading settings from the properties XML file.

Parameters:

`propertiesKeys` - Properties' keys that are supposed to be loaded. If this is null, then all properties in the properties file are returned.
`propertiesURL` - Target file.

Returns:

Loaded properties.

---

loadProperties

```
public static java.util.Properties loadProperties(java.util.Enumeration
propertiesKeys)
  throws java.io.IOException
```

Method for (re)loading settings from the current properties XML file.

Parameters:

`propertiesKeys` - Properties' keys that are supposed to be loaded.

Returns:

Loaded properties.

ucot.utils
Class Sets

```
java.lang.Object
   |
   +-ucot.utils.Sets
```

public class **Sets**
extends java.lang.Object

Operations to use sets.
Author:
pajumasu

Constructors

Sets

```
public Sets()
```

Methods

intersection

```
public static java.util.Set intersection(java.util.Set set1,
         java.util.Set set2)
```

Item must be in both sets to be in the resulting set.

Parameters:
`set1` - The first set.
`set2` - The second set.

Returns:
The intersection of both sets.

missing

```
public static java.util.Set missing(java.util.Set set1,
         java.util.Set set2)
```

Returns set items that are missing in second set.

Parameters:
`set1`
`set2`

Returns:
The set of items that ar in set2 but not in set1.

ucot.utils
Class StringTools

```
java.lang.Object
  |
  +-ucot.utils.StringTools
```

public class **StringTools**
extends java.lang.Object

This class contians tools for string handling.

Constructors

StringTools

`public StringTools()`

Methods

capitalize

`public static java.lang.String capitalize(java.lang.String str)`

Returns string which starts with uppercase letter.

Parameters:
`str` - The string to be capitalized.

Returns:
The capitalized string.

decapitalize

`public static java.lang.String decapitalize(java.lang.String str)`

Returns string which starts with lowercase letter.

Parameters:
`str` - The string to be decapitalized.

Returns:
@return The decapitalized string.

exceptionStackTrace

`public static java.lang.String exceptionStackTrace(java.lang.Throwable t)`

Returns stack trace of throwable object as a string.

Parameters:
`t` - The throwable object.

(continued from last page)

Returns:
String representation of the stack trace.

removeWhitespaces

```
public static java.lang.String removeWhitespaces(java.lang.String str)
```

Removes all white space characters (spaces, tabs and line breaks).

Parameters:
`str` - The string to be shortened.

Returns:
string with the whitespaces removed

Index