```java
package fi.paatti.research.paattiapplication.views.eventview.tools;


import fi.paatti.research.paattiapplication.views.eventview.EventTool.NODE_MODES;
import fi.paatti.research.paattiapplication.views.eventview.tools.NodeFactory.NODE_TYPE;
import fi.paatti.research.paattiapplication.views.eventview.tools.NodeFactory.TreeNode;
import java.awt.Color;
import java.awt.Graphics;
import java.io.Serializable;
import java.util.LinkedList;


/**
 * ContextMenu is a simple clickable "container" that holds other clickables
 * knows as MenuItems. ContextMenu acts merely as a larger click detector around
 * the defined sub-clickables, returning the values of the selected menu items
 * (if any).
 *
 * @author Tapio Keränen, t.tapio.keranen@student.jyu.fi
 */
public class ContextMenu extends Clickable implements Serializable {

  // Default sizes for the menu items and the context menu (width).
  private static final int MENU_ITEM_WIDTH = 150, MENU_ITEM_HEIGHT = 20;
  // ContextMenu's visibility.
  private boolean visible;
  // Clickables to display when the item clicked in the tree view was a node.
  private LinkedList<MenuItem> nodeMenuItems;
  // Clickables to display when the item clicked in the tree view was null.
  private LinkedList<MenuItem> nullMenuItems;
  // Current displayed list of clickables.
  private LinkedList<MenuItem> currentItemList;

  /**
   * ContextMenu constructor.
   */
  public ContextMenu() {
    width = MENU_ITEM_WIDTH;

    nodeMenuItems = new LinkedList<MenuItem>();
    nodeMenuItems.add(new MenuItem("Siirrä solmua", NODE_MODES.MOVE_NODE, null));
    nodeMenuItems.add(new MenuItem("Lisää lapsi", NODE_MODES.ADD_CHILD, null));
    nodeMenuItems.add(new MenuItem("Poista solmu", NODE_MODES.REMOVE_NODE, null));

    nullMenuItems = new LinkedList<MenuItem>();
    nullMenuItems.add(new MenuItem("Lisää tekstisolmu", NODE_MODES.ADD_NODE, NodeFactory.NODE_TYPE.DEFAULT));
```

```
    nullMenuItems.add(new MenuItem("Lisää radiosolmu", NODE_MODES.ADD_NODE, NodeFactory.NODE_TYPE.RADIO));
  }

  /**
   * Moves the context menu to the given coordinates and sets the menu visible,
   * displaying a list of clickable items based on the type of the node given
   * as a parameter.
   *
   * @param x the x coordinate.
   * @param y the y coordinate.
   * @param node the node that was clicked (if any).
   */
  public void show(int x, int y, TreeNode node) {
    this.x = x;
    this.y = y;

    currentItemList = (node == null) ? nullMenuItems : nodeMenuItems;

    for (int i = 0; i < currentItemList.size(); i++) {
      currentItemList.get(i).setPos(x, y + i * MENU_ITEM_HEIGHT);
    }

    height = currentItemList.size() * MENU_ITEM_HEIGHT;

    visible = true;
  }

  /**
   * Returns the visibilty state of the context menu.
   *
   * @return true if visible, otherwise false.
   */
  public boolean isVisible() {
    return visible;
  }

  /**
   * Returns the menu item found at (x,y) and hides the context menu. If no
   * menu item was found, a null value is returned instead.
   *
   * @param x the x coordinate.
   * @param y the y coordinate.
   * @return the context menu item found, if any.
   */
  public MenuItem getMenu(int x, int y) {
    MenuItem selectedMenuItem = null;
```

```
    if (contains(x, y)) {
      for (MenuItem menuItem : currentItemList) {
        if (menuItem.contains(x, y)) {
          selectedMenuItem = menuItem;
          break;
        }
      }
    }

    visible = false;

    return selectedMenuItem;
  }

  /**
   * Paints the items added to the context menu.
   *
   * @param g graphics context.
   */
  public void paint(Graphics g) {
    if (!visible) {
      return;
    }

    for (MenuItem menuItem : currentItemList) {
      menuItem.paint(g);
    }
  }

  /**
   * MenuItem is a clickable that contains event tree tool enums used in
   * controlling the tool states and the types of nodes to be added to the
   * tree (if any).
   */
  public class MenuItem extends Clickable implements Serializable {

    private String caption;
    private NODE_MODES nodeMode;
    private NODE_TYPE nodeType;

    private MenuItem(String caption, NODE_MODES nodeMode, NODE_TYPE nodeType) {
      this.caption = caption;
      this.nodeMode = nodeMode;
      this.nodeType = nodeType;
      super.width = MENU_ITEM_WIDTH;
```

```
      super.height = MENU_ITEM_HEIGHT;
    }

    /**
     * Positions menu item to the coordinates given as parameters.
     *
     * @param x the x coordinate
     * @param y the y coordinate
     */
    private void setPos(int x, int y) {
      this.x = x;
      this.y = y;
    }

    /**
     * Returns the node mode set to the menu item.
     *
     * @return node mode (tree tool state)
     */
    public NODE_MODES getNodeMode() {
      return nodeMode;
    }

    /**
     * Returns the node type set to the menu item.
     *
     * @return node type
     */
    public NODE_TYPE getNodeType() {
      return nodeType;
    }

    /**
     * Paints the menu item to the current graphics context.
     *
     * @param g graphics context
     */
    public void paint(Graphics g) {
      g.setColor(Color.white);
      g.fillRect(x, y, MENU_ITEM_WIDTH, MENU_ITEM_HEIGHT);

      g.setColor(Color.black);
      g.drawRect(x, y, MENU_ITEM_WIDTH, MENU_ITEM_HEIGHT);

      g.drawString(caption, x, y + MENU_ITEM_HEIGHT);
    }
```

```
  }
}
```