```java
package fi.paatti.mobile.views.eventlistview;


import com.vaadin.data.Item;
import fi.paatti.mobile.applicationinfo.CssStyleNames;
import fi.paatti.mobile.views.listview.ListItem;
import java.io.Serializable;


/**
 * A single item in an event list.
 * <p/>
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi @date 24.4.2012
 * @since 0.0.5
 */
public class EventListItem extends ListItem implements Serializable {

  private static final long serialVersionUID = 1L;
  private int itemStatus = -1;

  /**
   * The value that is used to show that the event item is available (it can be started).
   */
  public static final int STATUS_AVAILABLE = 0;

  /**
   * The value that is used to show that the event item is not available (it cannot be started yet).
   */
  public static final int STATUS_UNAVAILABLE = 1;

  /**
   * The value that is used to show that the event item is available and is unfinished (it is over it´s due time).
   */
  public static final int STATUS_UNFINISHED = 2;

  /**
   * The value that is used to show that the event item is a voluntary event.
   */
  public static final int STATUS_VOLUNTARY = 3;

  /**
   * Constructor.
   * @param item The item that has the event values.
   * @param idColumn The id property in the event item.
   * @param textRows The properties that will be used in list items text.
   */
```

```
  public EventListItem(Item item, String idColumn, String... textRows) {
    super(item, idColumn, textRows);
    this.addStyleName(CssStyleNames.EVENTLISTITEM);
    this.setOpenButtonText("Aloita");
  }

  /**
   * Set the status of the event item.
   * <ol>
   * <li>AVAILABLE (the event can be started)</li>
   * <li>UNAVAILABLE (the event is not yet ready)</li>
   * <li>UNFINISHED (the event shoud have been done allready)</li>
   * <li>VOLUNTARY (the event has no time limit and can be done multiple times)</li>
   * </ol>
   * @param status The new status for the event.
   */
  public void setStatus(int status) {
    this.itemStatus = status;

    switch (itemStatus) {
    case STATUS_AVAILABLE:
      this.removeStyleName(CssStyleNames.EVENTLISTITEM_UNAVAILABLE + " " + CssStyleNames.EVENTLISTITEM_UNFINISHED);
      this.addStyleName(CssStyleNames.EVENTLISTITEM_AVAILABLE);
      break;

    case STATUS_UNAVAILABLE:
      this.removeStyleName(CssStyleNames.EVENTLISTITEM_AVAILABLE + " " + CssStyleNames.EVENTLISTITEM_UNFINISHED);
      this.addStyleName(CssStyleNames.EVENTLISTITEM_UNAVAILABLE);
      break;

    case STATUS_UNFINISHED:
      this.removeStyleName(CssStyleNames.EVENTLISTITEM_UNAVAILABLE + " " + CssStyleNames.EVENTLISTITEM_AVAILABLE);
      this.addStyleName(CssStyleNames.EVENTLISTITEM_UNFINISHED);
      break;

    case STATUS_VOLUNTARY:
      this.addStyleName(CssStyleNames.EVENTLISTITEM_VOLUNTARY);
      break;
    }
  }

  /**
   * Get the status of the event item.
   * <ol>
   * <li>AVAILABLE (the event can be started)</li>
   * <li>UNAVAILABLE (the event is not yet ready)</li>
```

```
   * <li>UNFINISHED (the event shoud have been done allready)</li>
   * <li>VOLUNTARY (the event has no time limit and can be done multiple times)</li>
   * </ol>
   * @return The status of the event.
   */
  public int getStatus() {
    return itemStatus;
  }

}
```