Design Patterms

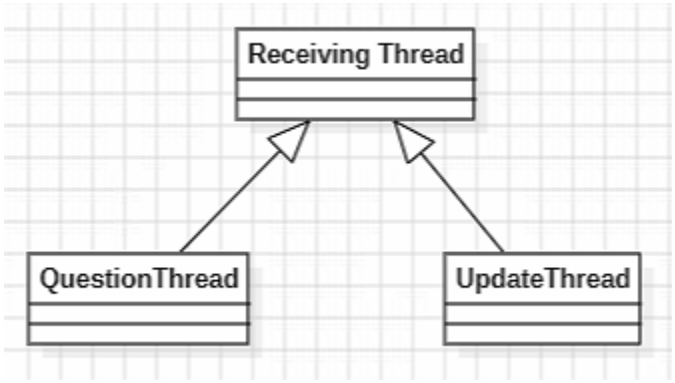
Rafael Tavares 60608

1 – Template Method Pattern

Snippet

```
37  /**T
38   * The thread that checks for incoming messages.
39   */
40  final class ReceivingThread extends Thread {
41
42      private static final Logger logger = Logger.getLogger(ReceivingThread.class.getName());
43
44      /** A class to handle questions. */
45  >   private static class QuestionThread extends Thread {…
98
99  >   private static class UpdateThread extends Thread {…
145
```

Class Diagram



Location

`src/net/sf/freecol/common/networking/ReceivingThread.java`

Rationale

In the class ReceivingThread we can see two other classes being define inside of it, which change very little in terms of implementation, slightly differing, which indicates a sign of the Template Method Pattern. Not only here, but, there are other classes extending the Thread class which are also very similar.

2 – Composite Pattern

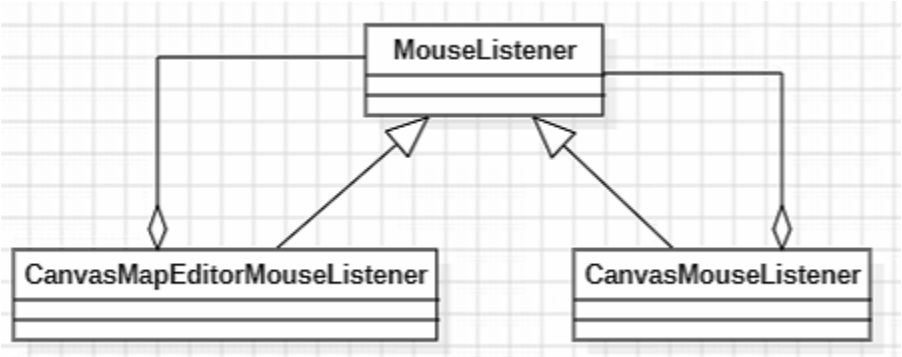
Snippets

```java
/**
 * Listens to mouse buttons being pressed at the level of the Canvas.
 */
public final class CanvasMouseListener extends FreeColClientHolder
    implements MouseListener {
```

and

```java
/**
 * Listens to the mouse being moved at the level of the Canvas.
 */
public final class CanvasMapEditorMouseListener extends FreeColClientHolder
    implements MouseListener, MouseMotionListener {
```

Class Diagram



Location

`src/net/sf/freecol/client/gui/CanvasMapEditorMouseListener.java`
and
`src/net/sf/freecol/client/gui/CanvasMouseListener.java`

Rationale

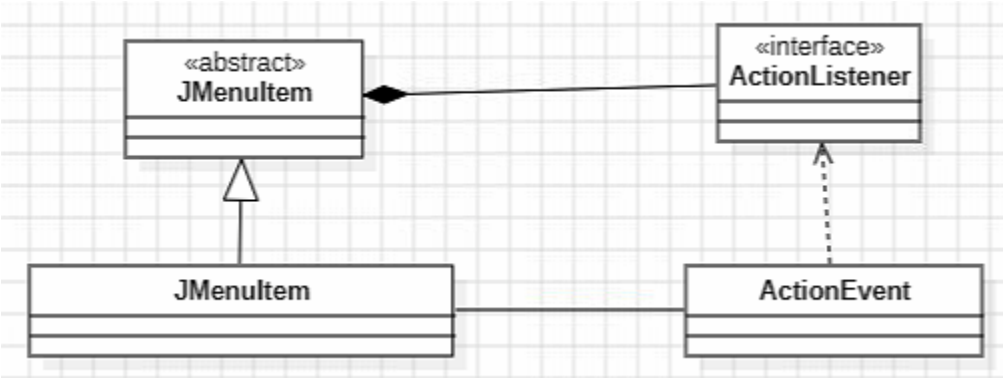
By examining the relations between MouseListener with CanvasMapEditorMouseListener and CanvasMouseListener, we can see their dependency through the implementation of the same interface, which can indicate the Composite Pattern. Moreover, we can conclude our rationale by understanding that both CanvasMapEditorMouseListener and CanvasMouseListener are MouseListeners themselves.

3 – Observer Pattern

Snippet

```
        ji.addActionListener((ActionEvent ae) → {
                igc.changeState(activeUnit, Unit.UnitState.ACTIVE);
            });
        add(ji);
```

Class Diagram



Location

`src/net/sf/freecol/client/gui/TilePopup.java`

Rationale

The method addActionListener was a very strong indication of an Observer pattern, which was later confirmed by examining the code. We can see here, an observer being added to an object, which when triggered will produce a state change.