Victoria University of Wellington
School of Engineering and Computer Science

SWEN225: Software Design, Assignment 2

Due: Monday, 9 August, before midnight

1 Outline

In this assignment, you will add a Graphical User Interface to your previous implementation of the "Murder Madness" game. We suggest that you continue to work with your previous team partners, but you may choose different partners for this assignment. Either way, i.e., even if you keep the current team composition, you need to register your team for assignment 2, using the team signup system.

2 Requirements

In this assignment, you are required to use at least the following Swing components, but you may use more:

- `JMenuBar` and `JMenuItem`. A menu bar for your Murder Madness game must be provided.
- `JButton`. Buttons should be used in your GUI. For example, you might provide a button to start the next turn and/or roll the dice.
- `Canvas` / `JPanel`. Your GUI should use a `Canvas` (or `JPanel`) to draw the current state of the game, including the Murder Madness board itself and the position of all players on that board. The location of all weapons should also be visible, as should the cards held by the current player.
- `JTextField`. A typical use for a text field is to allow each player to enter their name.
- `JRadioButton`. Radio buttons should be used when entering the details of a player. In this case, the available characters (e.g. Lucilla, Bert, etc) should be displayed using radio boxes, thus allowing the user to choose exactly one. Characters which have already been chosen should not be selectable (and ideally, should be grayed out).
- `JDialog`/`JOptionPane`. Dialogs should be used in a number of places. For example, when the user attempts to close the main window, a dialog should seek confirmation from the player.
- `MouseListener`. A mouse listener should be used to process mouse events. For example, a player might be able to move to a square by clicking on it (provided the intended move is legal).

N.B., if you want to use third-party assets, it is not necessary to ensure that you are using non-copyrighted work only since you are not allowed to publish your work as per the submission rules anyhow.

In this assignment, you will also have to use Javadoc to create documentation for your code, and use JUnit to test some of the functionality of your code (see the submission instructions for further details).

3 Overall Design

Use the Model View Controller (MVC) design, as discussed in lectures, to implement the game with a GUI. There is no need to establish dedicated controllers, but game data and its presentation should be separated according to the MVC design.

4 Extensions

There are a number of ways in which you can make your GUI more interesting, and extra marks will be awarded for this. Opportunities to improve the GUI include:

- **Mouse Hover Behaviour**. Hovering the mouse over various items on the board should produce little pop-up messages.
- **Shortcut Keys**. GUIs often provide shortcut keys enabling experienced users to operate the program more efficiently via keyboard shortcuts. For example, a shortcut key could be used to start the next turn, and/or to access menu items. A `KeyListener` is needed to implement this.
- **Animations**. Providing an animation of certain events in the game will make it more engaging. For example, when a player's character token is being moved to another square, you might animate the token rather than moving it there immediately. Similarly, you could have the tokens themselves be animated to perform different actions when different events happen to them.
- **Resizeable Display**. An interesting challenge is to make the main window resizeable, along with all of the items being displayed (e.g. the board). In this case, the size of items on the board will depend upon the size of the main window.

5 Tasks

If you have not used Swing before, you may find the official Java Swing documentation or the selection of tutorials on our reading list useful.

5.1 Team Work

You need to work with a team on assignment 2 and must register team via the team signup system, even if you are continuing with your assignment 1 team. Ensure to be registered at least three days before the submission deadline as team formation will be closed at that point in time.

6 Submission

The following artefacts need to be *submitted electronically*:

1. **Program code, including source code, and Javadoc files.** (filename = `code.jar`)
2. **UML State Diagram**, specifying the global states your software can be in, along with appropriate external and internal event transitions. (filename = `state-diagram.*`)
3. **Reflection**, written in your own words (1000 word limit) giving an overview of the design for your graphical user interface. This should include brief discussions on how the GUI is organised, making reference to Swing components used in the code. Also explain what you had to change in the Murder Madness implementation you started from in order to add the GUI and whether a different original design could have made it easier to add a GUI, and, if so how so. (filename = `reflection.pdf`, to be submitted to its dedicated submission item)

Your submitted code must be packaged into a `code.jar` file, including the source code and Javadoc documentation (see for instance this "Exporting to Jar" guide) and must run on an ECS machine (JDK build 11.0.10+9). You must not use any third-party libraries; the code must compile in a standard, plain Java environment. For the Javadoc documentation to make sense, your source code must include appropriate Javadoc comments. The code should also provide a suite of five to ten JUnit test cases.

The other submission artefacts must adhere to the naming scheme detailed above and must be provided as PDF or PNG files; other formats will not be accepted.

NOTE: The written reflection must be your own individual work; you cannot collaborate with any of your team partners, i.e., you cannot submit a reflection which shares content with that of any of your team partners.

Furthermore, for all submitted work, we cannot acknowledge material copied from other sources, such as the internet or textbooks. If you include any material that you have not authored yourself then you need to state this clearly with a disclaimer and the material will then be discounted for marking purposes. This may or may not impact on your mark, depending on the nature of the material used. If you fail to include such a disclaimer for copied material, unintentionally or not, you will be subject to a plagiarism investigation which may result in the loss of all marks and an entry into the academic misconduct register. Note that a person sharing material will be implicated in a plagiarism investigation as well and that it is not allowed to upload VUW assignment descriptions or solutions to VUW assignments to a public repository.

We will be using automated systems such as Turnitin and MOSS to check for plagiarism and with your submission you consent to us using these systems.

7 Marking Guide

7.1 Individual Reflection [82 marks]

For the individual reflection, marks will be awarded on an *individual basis* as follows:

- **Grammar and Spelling (5 marks)**. Reflections should be free from typographical and other presentational errors.
- **GUI Discussion (25 marks)**. Marks will be awarded for how well the individual reflection conveys aspects of the GUI design.
- **Design Discussion (27 marks)**. Marks will be awarded for how well the individual reflection explains the challenges in modifying the previous code to feature a GUI, rather than textual input/output. In particular the reflection should elaborate on how a good design supports changes like adding a GUI, e.g., what properties of the original code made it hard or easy to add the GUI.
- **State Diagram Discussion (25 marks)**. Marks will be awarded for how well the individual reflection describes the states that the software may be in.

7.2 Group Design [6 marks]

For the design document, marks will be awarded on a *group basis* as follows:

- **State Diagram (6 marks)**. Marks will be awarded for correct use of the following: *states*, *transitions*, *transition actions*, *transition guards*, *state activities*, and *advanced state chart notions* such as *super-states*.

7.3 Group Implementation [12 marks]

For the main implementation, marks will be awarded on a *group basis* as follows

- **Swing Components (4 marks)**. Marks will be awarded for programs which make appropriate use of the required Swing components, as set out in section 2.
- **Model-View-Controller (3 marks)**. Marks will be awarded for programs which make appropriate use of the *Model-View-Controller* approach, as discussed in lectures.
- **Extensions (2 marks)**. A mark will be awarded for the incorporation of an additional feature which improves the overall quality of the GUI.
- **Coding Style (1 mark)**. A mark will be awarded for overall coding style which includes: appropriate naming of classes, fields, methods, and variables; good division of work into methods, in order to avoid long and complex chunks of code; consistent formatting (e.g., indentation, etc).
- **Documentation (1 mark)**. A mark will be awarded for good use of Javadoc comments on all public classes, interfaces and methods.
- **Testing (1 mark)**. A mark will be awarded for the inclusion of sensible JUnit tests. These should cover the basic functionality of the program. Between five and ten tests should be defined.