Victoria University of Wellington
School of Engineering and Computer Science

SWEN222: Software Design

Assignment 2

Originally Due on Monday 26th August @ Midday

Now Due on Monday 2th September @ Midnight

1 Introduction

In this assignment, you will build a Graphical User Interface for the cluedo game. As with the previous assignment, you may work in pairs or on your own. **Unless otherwise agreed with the course co-ordinator, you are assumed to be working in the same pair (or on your own) as for assignment 1.**

In this assignment, you will also be **required** to use subversion. Since we want to have control of the way you use subversion, you have to register to `https://ecs.victoria.ac.nz/cgi-bin/teamsignup`, even if your team is composed by only you; and a team repository will be made available.
Team registration closes on 2013-08-15. You must register your team on time. Even if you are already registered for assignament 1, you have to register your group for assignament 2. If you fail to register your group, you assignament could not be marked.
NOTE: Marks will be available for demonstrating good usage of subversion. In particular, checking-in all of the code on the last day does not constitute good usage! We will be looking for good commit messages, that commits have been made on multiple days, and other evidence that the repository was used appropriately.
Useful example commands for using SVN with your team space:

```
% svnadmin create /vol/projects/swen222_2013T2/s222tX/svn

% chgrp -R s222tX ~/svn
% chmod -R g=u ~/svn

%svn+ssh://login@barretts.ecs.vuw.ac.nz/vol/projects/swen222_2013T2/s222tX/svn
```

Here, login should be replaced with your ECS login. Using this location, you will even be able to access your repository remotely from home

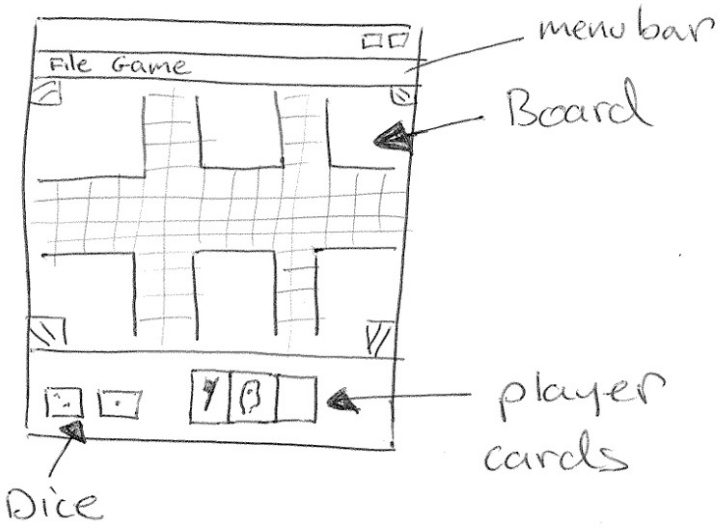
2 Requirements

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

- `JMenuBar` and `JMenuItem`. A menu bar for your cluedo 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`. Your GUI should use a user defined `Canvas` class to draw the current state of the game, including the cluedo board itself and the position of all players on that board. The location of all weapons should also be visible, as should those cards held by the current player.
- `JTextField`. These should be used 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. Miss Scarlet, Professor Plum, etc) should be displayed using radio boxes, thus allowing the user to choose the one they wish. Characters which have already been chosen should not be selectable (and ideally, should be grayed out).
- `JDialog`. Dialogs should be used in a number of places. For example, when the user attempts to close the main window, a dialog should check that they wanted to do this. Similarly, when the game starts, a dialog should prompt the user to input the name of each player, and their token.
- `MouseListener`. A mouse listener should be used to received events from the mouse. For example, the user should be able to move to a square by clicking on it (provided he/she has rolled a sufficiently high number).

An example layout illustrating the main GUI window is given below:



Your GUI window does not have to be exactly the same as this, although it should be similar.

3 Fancy Stuff

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

- **Hovers**. In this case, hovering the mouse over various items on the board should produce little pop-up messages. For example, hovering over a players token will show the players name.

- **Short-Cut Keys**. Modern GUIs provide short-cut keys enabling experienced users to operate the program more efficient. For example, a short-cut key could be used to start the next turn, and 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 fun. For example, when a player's token is being moved to another square, you might animate the motion 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 (e.g. when going through a secret passage).
- **Resizeable Display**. A more complicated problem 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. The code provided in lecture 7 (as well as `ChessView` program used in SWEN221) provides a good example of how to do this.

4 What to Do

If you have not used Swing before, you may find the following tutorial helpful:

`http://download.oracle.com/javase/tutorial/ui/index.html`

For this assignment, you will be using the same shared space of assignament one where you can store your subversion repository,

NOTE: you are expected to use your subversion repository, and marks will award for this. Furthermore, you are expected to use subversion in a "normal manner" — that is, to make regular commits during the assignment and not, for example, simply make all commits right before submission.

5 Submission

Your program code should be submitted electronically via the *online submission system*, linked from the course homepage. Your submitted code should be packaged into a jar file, including the source code (see the export-to-jar tutorial linked from the course homepage). Your program should include appropriate Javadoc comments, and provide JUnit tests where appropriate. In particular, (with the opportune adaptations) all existing JUnit tests from Assignment 1 should still pass. That is, the resulting program of assinament two should also offer an option to keep the text based used interface instead of the graphical one.

In addition to the source code, various pieces of design documentation are required. These should be submitted as either PDF or PNG files; other formats will not be accepted. The required documents are:

1. A Sequence diagram illustrating the most important/intresting scenario about what happens when the user clicks on the board.
2. An object diagram illustrating the structure of the main GUI window. Since the GUI could dynamically change during the game, show the object diagram in the richest case.
3. A two-page report written in your own words giving an overview of the design for your graphical user interface. This should include brief discussions on how the GUI is organised and, in particular, how the main display (e.g. the board and other graphical components) is drawn. Finally, you should also describe the sequence of events that occurs when a user clicks on the board area.

NOTE: Points 1 and 2 are submitted as a team, as for the program itself. Instead, the written report (point 3) must be your own work. That means you cannot submit a report which is identical to that of your other team member(s). Furthermore, it is not acceptable to simply copy material from other sources, such as the internet or text books.

6 Assessment

This assignment will be marked as a letter grade (A+ ... E), based primarily on the following criteria:

- **Design**. This overall quality of the design will be assessed. A good design will provide an appropriate decomposition of real-world objects, and make effective use of inheritance, polymorphism and design patterns (where appropriate).
- **Documentation**. Marks will be awarded on an individual basis for clear communication. In particular, diagrams should focus on correctly identifying the most important issues in the design and communicating them effectively. Marks will also be awarded for good justification of the design, and for showing evidence that alternative approaches were considered.
- **Implementation**. Marks will be awarded for various aspects related to the program itself including, but not limited to, correctness and style. Submitted projects are expected to follow the appropriate style guide, and include comments suitable for documentation purposes (e.g. Javadoc), as well as general understanding. Submitted projects are also expected to provide a range of JUnit tests to give confidence that the program is correct.

NOTE: Marks will be available for demonstrating good usage of subversion. In particular, checking-in all of the code on the last day does not constitute good usage! We will be looking for good commit messages, that commits have been made on multiple days, and other evidence that the repository was used appropriately.