SCIT

School of Computing and Information Technology

Spring 2015
CSCI213/MCS9213/CSCI813 — Java Programming and Applications

Assignment 3 (10 marks)

Due Time and Date:

23:59:59 on Sunday, 4 October

Objectives

This assignment requires you to design and implement a graphic user interface (GUI) for the Java quiz system using Java Swing. You will apply your knowledge and skills in Java GUI design and event-driven programming to develop a GUI application that allows users to register and answer quiz questions.

Upon completion of this assignment, you should be able to make use of reusable classes to develop end user GUI applications; to design and implement program functionality according to design requirements; to design control structure and handle events to support user interaction; to make use of a configuration file to allow users to configure the software startup options.

Tasks and Requirements

In this assignment, you will design and implement a GUI application that supports the Java quiz system using Java Swing according to the user requirements.

Your main application program should be called `JavaQuizGUI.java`. All your GUI classes should be in the package called `UserInterfaceGUI`. The `Student` class is provided in the `UserInterfaceGUI` package with documentation. The classes that represent multiple choice questions and true-and-false questions and the class to load questions from a question file and make the quiz are also provided in the `QuestionLibrary` package with documentation.

General Requirements:

- You should observe the common principles of OO programming when you design your Java classes, which include abstraction, encapsulation, inheritance and polymorphism.
- You should create your classes by making use of features of the Java classes
- You should make proper documentation and implementation comments in your codes where they are necessary.
- Logical structures and statements are properly used for specific purposes.
- Your code should comply with the major requirements of Java Code Conventions.

Structure of the Java packages:

The Java classes created in this assignment will have the following package structure. You may add more supporting classes in appropriate packages if necessary.

```
/WorkingDirectory
      JavaQuizGUI.java
      /au/edu/uow
            /UserInterfaceGUI
                  QuizGUIFrame.java
                  (Other GUI related classes)
                  Student.class
            /QuestionLibrary
                  Question.class
                  MultipleChoiceQuestion.class
               TrueAndFalseQuestion.class
                  QuestionLibrary.class
```

The `Student` class and `QuestionLibrary` package with documentation are provided.

Task 1: GUI Design and Implementation

The user requirements of your Java quiz GUI application are specified by the following program flow in 3 stages.

Stage 1 – Registration Screen:

Fig. 1 is the screen of the GUI in first stage. Your program window should be located at the centre of the screen when it starts. It provides an interface to register the student name. In this screen, there should be one tool bar at the top containing a "**Score**" button, an "**Exit**" button and an area for student name registration. There should be sufficient space between buttons and the registration area. The registration area should contain a label to guide student to input the name in a text field, a text field to input the name and a "**Register**" button. All of them should locate closely to indicate they are in a group to serve one purpose.

A message is display at the centre (vertically and horizontally) of the program window as shown in Fig. 1. The message is displayed in three centre-aligned lines with the first line displaying the program name with a larger font size and in a different colour from the following two lines.

Before the name is typed into the name field, a dialogue box should pop up to remind the registration if the "**Score**" button or "**Register**" button is pressed, or a return key is pressed in the text field, as shown in the Fig 1. The dialogue box should have the same *Look and Feel* as the main frame.

After the student name is typed into the text field and a return key is pressed or the register button is pressed, the program should progress to the next stage.

At any stages, if the **Exit** button is pressed, the program should exit.

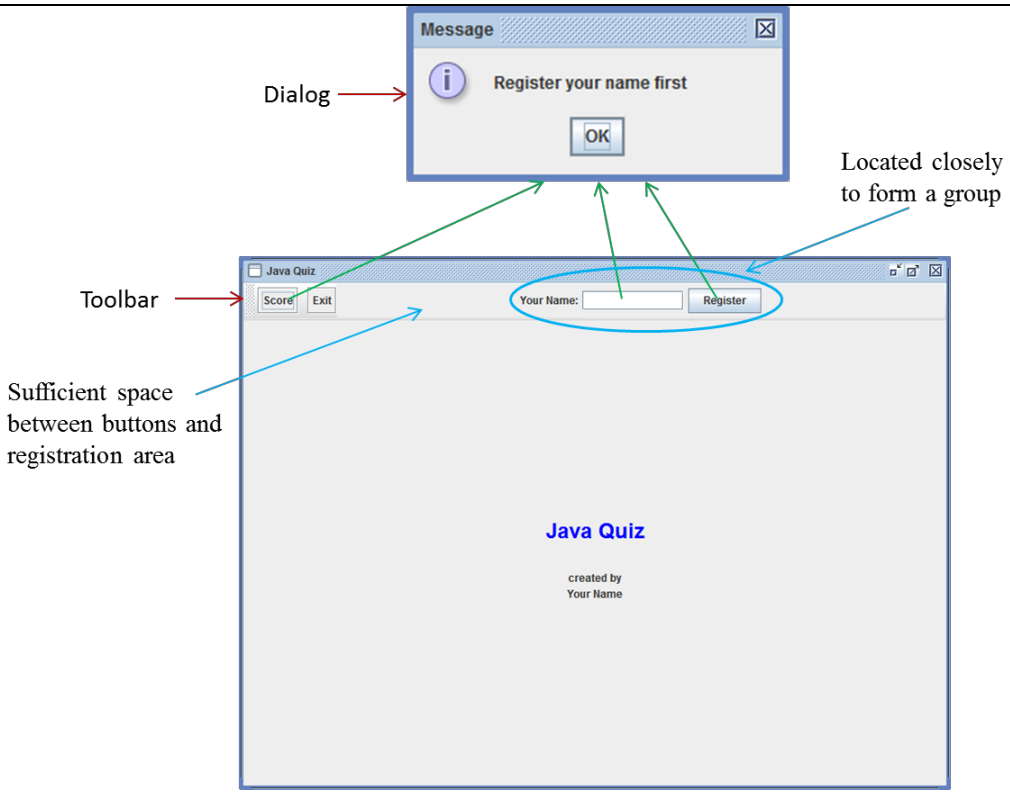

Fig 1: First stage screen shot

Stage 2 – Question Screen:

Fig. 2 is the screen of the GUI in second stage for true and false questions. It presents the questions to users and receives the user answer. In this screen, the registration area in the tool bar should be disabled to prevent any modification to the student name. It should contain an area to present question and an area to receive the answer. Radio buttons should be used for answer selection and the selection should be exclusive. (Only one answer can be selected.) At the bottom of the screen, there should be a "**Next**" button to proceed to the next question. (If no answer is selected before pressing the "**Next**" button, it should be considered as a wrong answer.) In this stage, if the score button is pressed, the current score should be displayed in a dialogue box. The layout requirement is specified in Fig. 2.

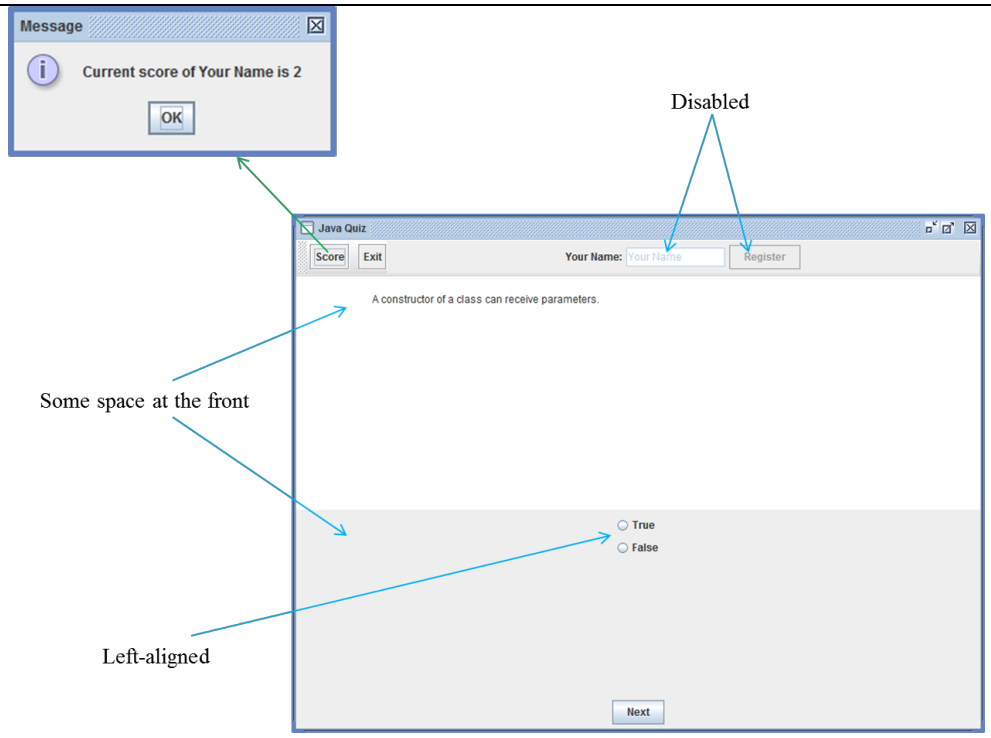

Fig 2: Second stage screen shot of True and False Questions

Fig. 3 shows the screen for multiple choice questions. The number of radio buttons should be the same as the number of answer choices. Other requirements are the same as for the true and false questions screen.

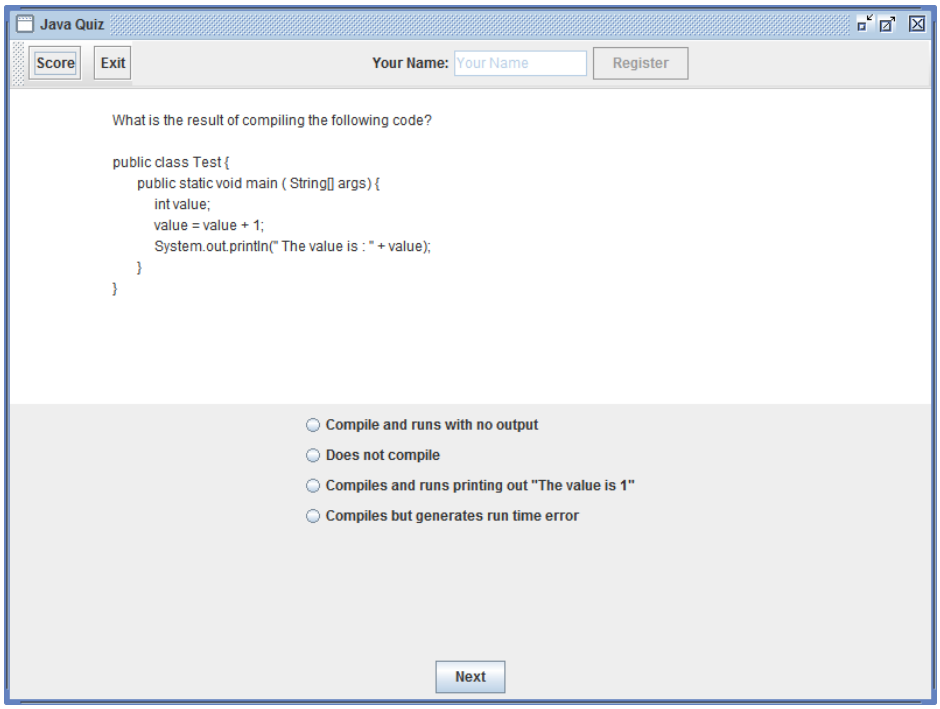

Fig 3: Second stage screen shot of Multiple Choice Questions

Fig. 4 shows the screen for the last question where the "**Next**" button should change to the "**Get Marks**" button. When the "**Get Marks**" button is pressed, the program should progress to the next stage.

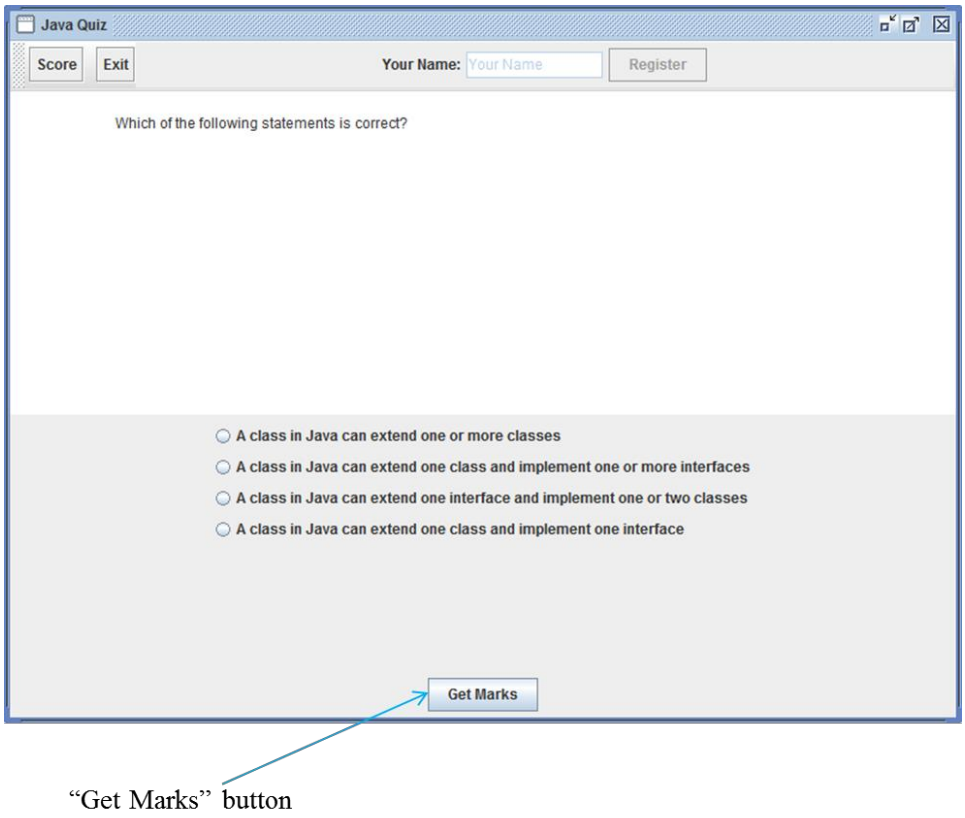

Fig 4: The screen shot of the last question in the second stage

Stage 3 – Marks Presentation:

Fig. 5 is the screen in the third stage for the score presentation. It should present the final marks with the student name and the possible maximal marks. It should contain an "**Exit**" button. It should not contain the tool bar in previous stages.

In this stage, you can design any layout you like as long as you meet the above requirements.

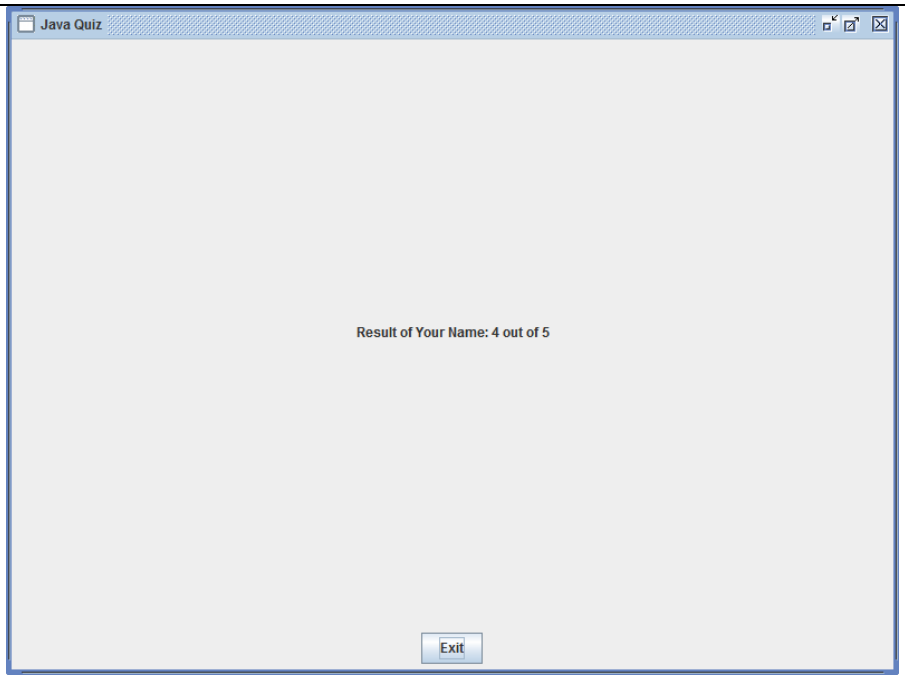

Fig 5: Third stage screen shot for the final marks presentation

Task 2: Program Configuration

Requirements:

Your program should be configurable. Users of your program should be able to configure the *Look and Feel*, the window size, the number of questions in each quiz session and the question file name using a configuration file.

A Java program can set up its own configurable attributes, called *program attributes*. Program attributes allow the user to configure various startup options, preferred window size, and so on for the program. Sometimes the term *preferences* is used instead of program attributes.

You should create a configuration file named "`JavaQuizGUI.conf`" for users to configure the program. Your program should use Java `Properties` class API to load the program attributes into your program. At least, the following configurations should be included in your configuration file.

- The size of your program window;
- The *Look and Feel* of your program;
- The number of questions in each quiz session;
- The Question file name;

The configuration file should accompany your program and in the same directory as your GUI application.

In your "`JavaQuizGUI.conf`" file, there should be a key/attribute pair as bellow to set the Look and Feel of your program to Java Metal Look and Feel:

```
LookAndFeel=javax.swing.plaf.metal.MetalLookAndFeel
```

Extension Task: Innovative Development
(***Optional for CSCI213/MCS9213 students; compulsory for CSCI813 students***)

In this task, you should develop further the Java quiz GUI application with your innovative design, either artistic or technical. For the artistic design, the mark will only be given if it was implemented programmatically.

You can load some image or graphics for information or decoration purposes as the background or graphical buttons etc. If graphical buttons (icons instead of text on the button) are used to replace the text button, tooltips should be used to indicate the function of those buttons. Examples include displaying some messages in blank areas in the window in all stages or creating some graphical decorations. Only the completed work for all stages will be accepted for marking. Trivial design such as simply setting some background colours and so on will not be satisfactory.

You can extend the functionality of the quiz system. All of them should harmonically make the application as a quiz system more user-friendly and professional. One example is display the student answers and correct answers in a tabular format for individual questions along with the final marks presentation in Stage 3.

You should not change the layout requirements in the Stage 1 and Stage 2 in the Task 1 when you attempt this task. You can rearrange the layout of the Stage 3 in the Task 1.

You should list the innovative developments you have made in the header of the `JavaQuizGUI.java` file. Only the listed developments will be checked and marks will be given to satisfactory completion.

Submission

- The `@author` line in all your source code files should include **your Subject Code**, **your name**, **student ID**, and **Unix login name**.
- IMPORTANT: Any submission that cannot be decompressed may receive zero mark. Any submission that resulted in compiling errors may have 50% marks deducted.
- You should first zip your files including subdirectory paths into a file called called **`a3.zip`**. For this assignment, you should submit all your programs including the Java quiz GUI application, `JavaQuizGUI.conf`, the provided question file and the packages, basically all files that required to run your program **excluding** files that may be generated by the IDE you might have used. Failing to do so may result in deduction of marks.
- Submit the file from your University Unix account on Banshee (`banshee.uow.edu.au`) as follows:

  ```
  turnin -c csci213 -a a3 a3.zip
  ```

  or, if you submit it after the deadline:

  ```
  turnin -c csci213 -a a3-late a3.zip
  ```

- Use the following command to check the submitted file:

  ```
  turnout -c csci213 -a a3
  ```

or, if you submit it after the deadline:

```
turnout -c csci213 -a a3-late
```

Marking Scheme

Mark to 0.5 divisions.

For CSCI213/MCS9213

General requirements	-0.5 ~ -1.0 mark for each error
Task 1 Stage 1	2
Task 1 Stage 2	4
Task 1 Stage 3	2
Task 2: Program Configuration	2
Extension Task: Innovative Development	2

Satisfactory completion of the Extension Task will be rewarded extra marks that will be added to the final mark for assignments and labs. The maximal final marks for all assignments and labs remain 50.

For CSCI813 students

General requirements	-0.5 ~ -1.0 mark for each error
Task 1 Stage 1	2
Task 1 Stage 2	3.5
Task 1 Stage 3	1.5
Task 2: Program Configuration	1.5
Extension Task: Innovative Development	1.5

Extension Task is compulsory for CSCI813 students.