F27SB Software Development 2

GUI Coursework 3: Multiple Choice GUI

Write a program to create a multiple choice quiz

A GUI-based program is required that will enable people to take a test from a bank of multiple choice questions. Each question consists of:

- the question text;
- 4 possible answers;
- the number of the correct answer (1-4, **not** 0-3);

Questions are held sequentially in a text file with each question held on 6 consecutive lines. An example entry in the questions text file could look like this:

```
What is your favourite colour?
Red
Green
Blue
Pink
4
How many Harry Potter books are there?
5
6
7
8
3
```

The programme should have the following features:

- A menu including `Open` and `Exit` where `Open` starts a `JFileChooser` to select the file with the questions inside and `Exit` ends the programme.
- Once a file is loaded, the GUI should display one question and its answers at a time.
- The user should be able to select an answer and they should be informed if they were correct or not.
- The user should be made aware of the number of correctly answered and the total number of questions answered.
- The user should only be able to proceed to the next question once they answered the current one.
- Once all questions have been answered, the user should be informed of their overall score and that the game has finished. The Open menu item should now be enabled to start a new quiz. Optionally, you can add a restart menu item to redo the current quiz.

Concrete sub-tasks:

a) define a class called `Question` to hold a single question, i.e. the text, the possible answers, and the correct answer index;

(0.25P)

b) write a method to select a file via a JFileChooser and to read all the questions from that file into an array/list of `Question` objects (assume that file has the structure mentioned above);

(0.25P)

c) design and implement a GUI with the components mentioned above: A menu, ability to display the question and answers, ability to select an answer, show the outcome and score, and proceed to the next question.

(Appropriate layout: 1P,
Class extends JFrame: 0.25P,
Class follows OOP principles: 0.25P,
Global set-up in main method: 0.25P)[1]

d) write a method to display a question on the GUI you designed;

(0.25P)

e) implement an `actionPerformed` method to respond to user interactions with the GUI. Make sure to enable and disable interactive components as required, e.g. the user should not be able to skip to the next question without selecting an answer first and they should not be able to load a new quiz before finishing the current one;

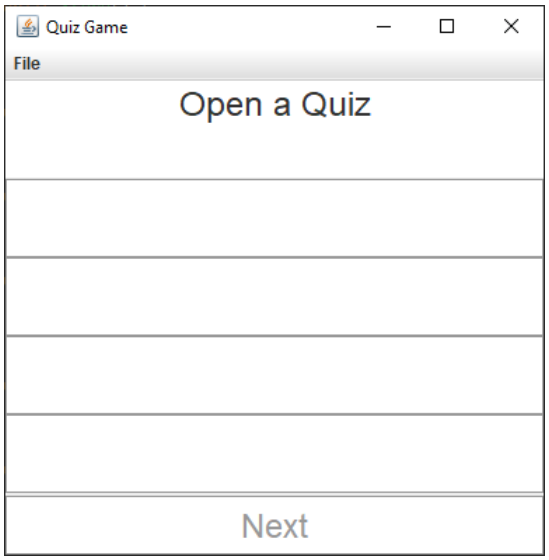
(Class implements ActionListener: 0.25P,
enabling/disabling of components: 0.25P,
Works as required: 1P)

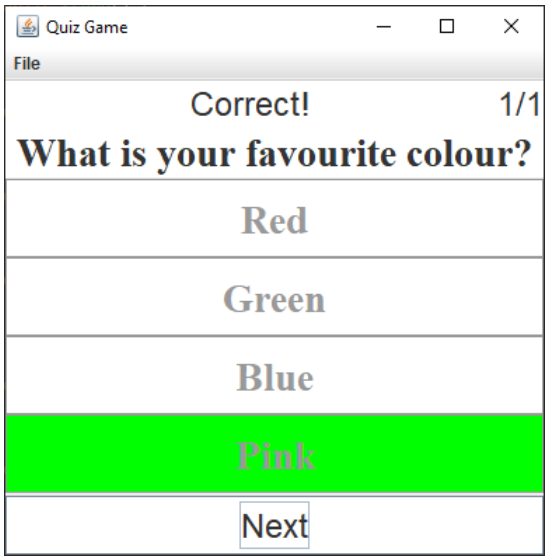
Optional Challenges: Randomise the order in which the questions and the answers are displayed. Problem: How can you know which answer is correct? Once an answer has been chosen, if the user selected the correct answer, highlight it in green. If they selected the wrong answer, highlight their answer in red and the correct one in green. When next is clicked, reset the colour.
Add a dialgoue box confirming if the user really wants to exit the programme when they press Exit in the menu.

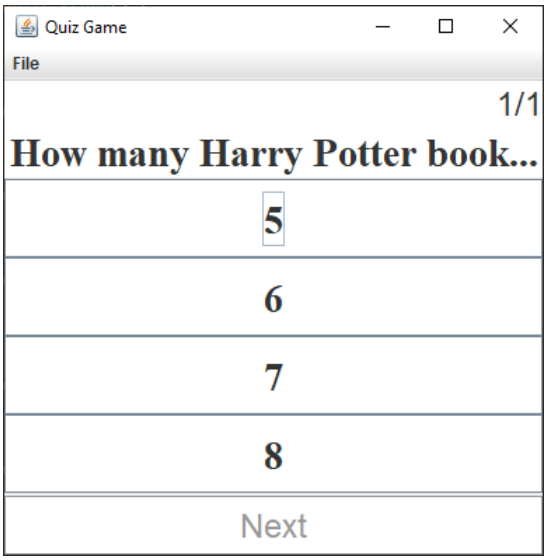
[1] Detailed explanation of points at the bottom of the document.

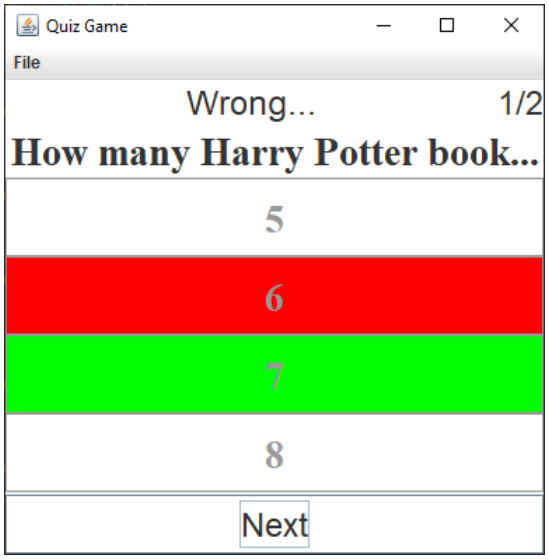
Possible final design

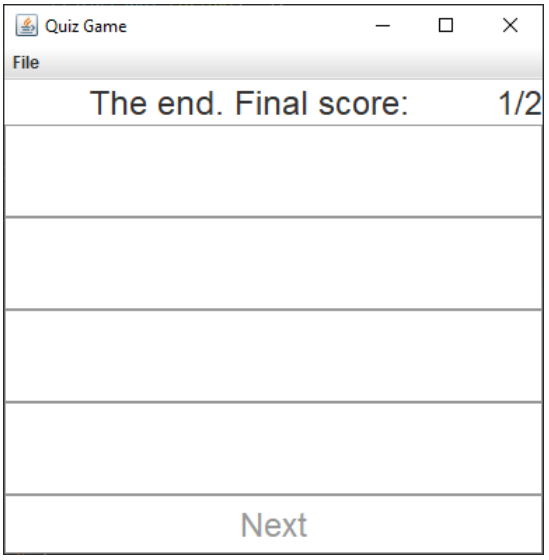
From top left to bottom right, started the programme, loaded a question file, selected the right answer (Pink), clicked next, selected the wrong answer (6), clicked next, the game ended.











Marking

In each coursework, you can achieve a total of 4 points. Each question awards different amounts of points for different parts of the question. Partial completion of a task will award the partial points listed underneath. Most descriptors of points are self-explanatory in context of the task. Where further clarification is needed, you can find that below.

Your work will be checked by a lab helper during your assigned lab slot. Once all tasks are checked, the points will be used to calculate your marks. Please, understand that the lab helpers are not marking your work but are checking the completion of subtasks. As part of this check, you will need to explain how you solved the given task. **Only successfully completed sub-tasks will award points.**

The marks will be released on Canvas after the marking deadline has passed. This is not an automatic process so please, be patient. Once the marks are released, you will be notified via Canvas. Please, **make sure to check your marks as soon as possible**. If there are any issues, please contact your teaching team immediately.

Collusion and Plagiarism

As mentioned above, you will need to explain your work during the demo session in the lab. If you are not able to explain how you arrived at the solution to the task, we need to assume that you did not do the work yourself. We do, however, know that you might be anxious or nervous during the session. Please, rest assured that this is not an interrogation and you can take all the time you need to explain your solution.

If there is reasonable doubt that you solved the given problems yourself, you will not get any points for this task. If there are concrete indications that you copied your answer or colluded with other students, we might also start an official investigation.

Please, make sure to fill and sign the Declaration of Student Authorship form for each coursework and upload it to Canvas. If you do not upload the form, we will not be able to give you any marks for this coursework.

If you feel unjustly accused of plagiarism or collusion, please contact your teaching team.

Coursework submission

Unless stated otherwise, all code parts of your work need to be committed and pushed to your GitLab fork. You need to upload the Declaration of Student Authorship to Canvas, and you need to present your solution to a lab helper. If you fail to do any one of these steps, you will not be awarded any marks.

The deadline for submission can be seen on Canvas. You will need to **present your work to a lab helper any time before the deadline during your allocated lab slot**. If you do not manage to present your work before the deadline, you can do so at the first lab after the deadline but will incur a 30% late penalty. If this late submission was caused by issues out with your control, you can apply for mitigating circumstances to have this late penalty removed.

If you also fail to present your work at the lab following the deadline, we will not be able to give you any marks for your work. Similarly, if this was caused by circumstances out with your control, you should apply for Mitigating Circumstances.
Please, note that we are not allowed to give individual extensions. If you cannot submit your work on time, you will need to apply for Mitigating Circumstances.

Explanation of Points

Appropriate layout: In this coursework, you have complete freedom of how to design the layout for your GUI. It should still use the things we covered in the lectures but you can arrange the components as you see fit. All the functionality needs to be supported but how you do this is up to you. While and example is given, your solution does not have to look like it.
Class extends JFrame: The class that declares and initialises all the GUI components should be a subclass of JFrame and correctly use the methods and variables provided by its super class to achieve the desired layout.

Wrong	Wrong	Right

`public static void main() {`
`  JFrame f = new JFrame();`
`  f.add(new JLabel());`
`  …`
	`public class A {`
	`  public A() {`
	`  JFrame f = new JFrame();`
	`  f.add(new JLabel());`
	`  …`
		`public class A extends JFrame {`
		`  public A() {`
		`  add(new JLabel());`
		`  …`

Class follows OOP principles: You should endeavour to reduce coupling and increase cohesion in your code. That means you should check that every class is responsible for its own components and data and that you do not provide unnecessary public variables or methods. Always think about how someone else could use your class in a different project. Refer to all the guidance given in the first half of the course and make sure to highlight this to the lab helper when explaining.
Global set-up in main method: When you think about someone using your code in their larger GUI, do you think they are happy that the name, size, position, and visibility is defined inside the class where it cannot easily be changed? Make sure to only define these things in the main method where you create a new instance of your class as we do in the lectures.
Class implements ActionListener: To increase cohesion, the class that declares and initialises the components that can trigger action events should be the one that is the action listener. Make sure that this class implements the ActionListener interface and overrides the actionPerformed method.
Works as required: If not all the functionality is implemented, the lab helper may award partial points for this category.