

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**IX610001 Programming 3**

**Lab Assignment 2 – Application**

Study Block 1: 13th Jan to 6th March 2020

Due date to be handed in: Week 8 - (2nd March)

Submission: Upload soft copy to Moodle

Demo: To Lecturer (Compulsory)

Contact Lecturer: Tariq Khan

Total Marks: 40

Weighting/Contribution: 15% to final marks

Learning outcomes covered 1-3

Name: Marlon Steve Parra Ruiz

ID: 1000087091

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# GitHub:

<https://github.com/marlonry/Assigment-2>

# Screens Drawings:

Figure : Design Screen



Figure : Game Over - Final Score

## Description:

Due to time concerns, I have decided to create a simple java quiz that follows the minimum viable requirements, even though my quiz is really simple I have implemented an entry application point, event handlers and a question class where all the questions are held.

As seen in figure 1 we can appreciate a simple design where we a score board, a question, and multiple choices are shown in a quiz format, where a user can pick a answer and it will be immediately told whether the answer is correct or not.

## Parts of my design:

Please reference figure 1:

1. **Score:** This is the score accumulator which lets the user know how many points they have accumulated over the course of the game.
2. **Question:** This is the question text view which fetches a question from the Question Class in a random format and displays it on the text view, each question is fetched which is proper choices.
3. **Choices:** These are the choices, on this part the choices are fetched dynamically from the Question class attributes choices, and whenever an user clicks on the choices the event handler will run a method that checks if the question was correct and if it was it will display another question as well as adding the score, if the question was incorrect a screen will pop letting the user know that the answer was incorrect and also this will conclude the game as soon as they failed one question of the quiz. The screen will also let the user decide if they want to keep going or if they want to exit the game.

**Question:** All the questions are based on New Zealand general information and common knowledge about New Zealand.

Please reference figure 2:

**Pop Up Window:** A pop up window will appear on thescreen letting the user know the current situation. This lets the user know whether they want to play again or exit the game.

this screen also shows the user the current score.

**Code Explanation:**

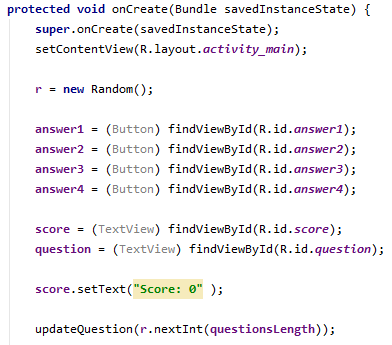
****

Figure : random and initialization

Figure 3: This part of the code shows the initialization of some variables a the initialization of the random variable as well as the usage of the updateQuestion() method which sets the first question on the view.

****

Figure : choices

Figure 4: This is the choices attribute where all the choices are stored, this stores the choices in a 2 dimensional array.

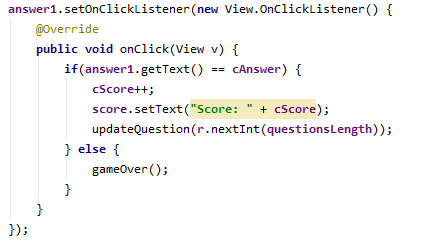
****

Figure : event listener

Figure 5: This is the event listener that add the current score and set the new question on the text view that are store from the question array.

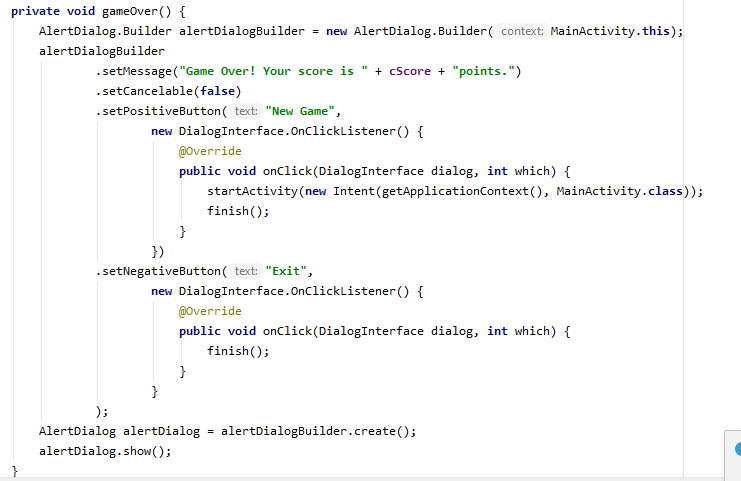
****

Figure : game over method

Figure 6: This is the game over function that creates a new pop up message that sets the message and the user’s score as well as setting up the 2 buttons to let the user decide whether they want to have a new game or exit the game.

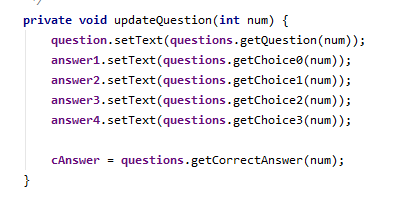
****

Figure : update question method

Figure 7: This is the method that updates the current question and the choices and correct answer for that specific question, it fetches the correct question based on the index for the array that was fetched.

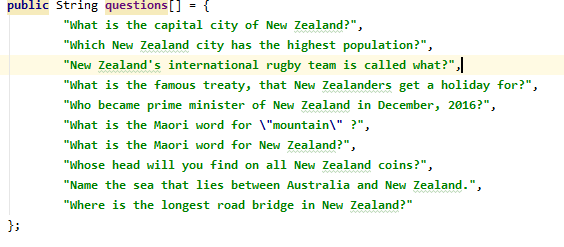
****

Figure : questions

Figure 8: These are the questions that are stored in the array to be fetched later based on the index.