Quiztime™

Java 'O' Level Project
Written and Compiled by Maximus C. Rossi

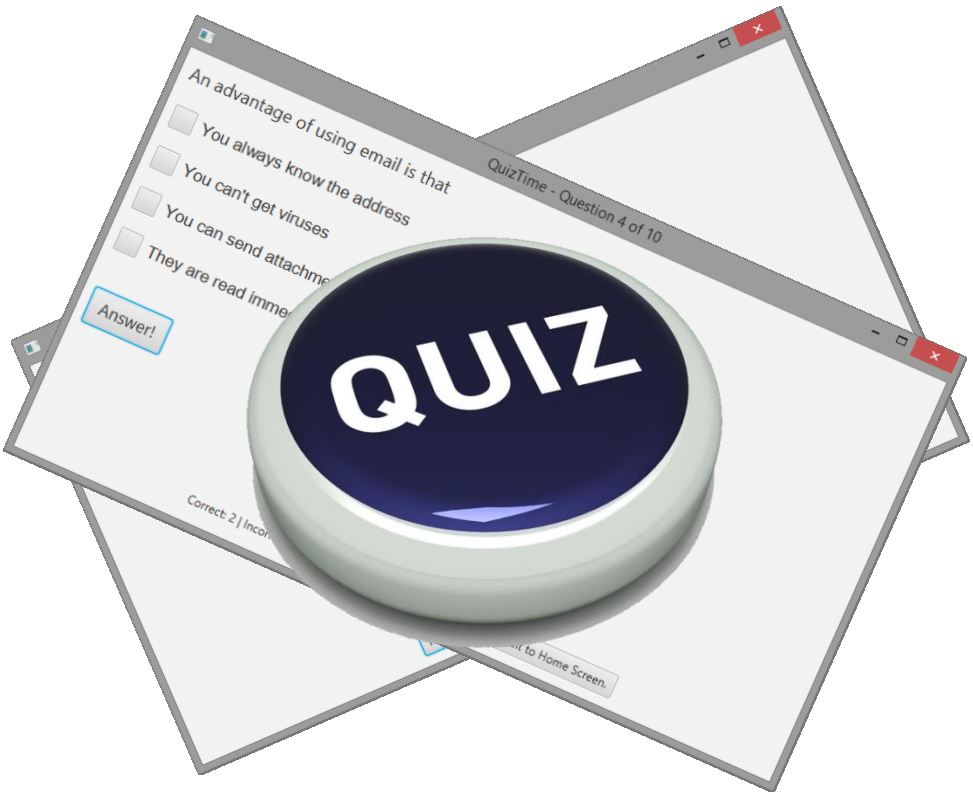


Table of contents

SECTION 1 - Definition of the problem

1.1 - Problem to be tackled

Milton High School found in Cambridge, United Kingdom is reaching its 100th year since its foundation. The school currently caters for about three thousand pupils ranging from ten to seventeen year olds. It employs over one hundred fifty staff and teachers. Milton High still operates in an old fashioned style which is not ideal for the twenty first century.

Each year a large amount of students fail their mid yearly and annual exam's which is not acceptable for a school as prestigious as Milton High. This hasn't been a problem in the past because Milton High was one of the only schools in the country, however due to the rise of schools across the country, there is quite a bit of competition between the schools. It is a common belief that parents would want to send their children to the best schools, not a school with a low pass rate / high fail rate. In the past years the school administration team has noticed a decrease in pupils applying for admission to the school. Fortunately, the administrator team has decided to take action. They have come up with a plan to convert their teaching programs to fit twenty first century learning.

The use of twenty first century learning aims to exercise certain aspects that aren't practised on during traditional learning. There are a countless number of advantages, some of which are; creativity and innovation, critical thinking, problem solving, communication and collaboration.

The administrator team requested that the Computing students of level eleven are to design quizzes for each and every subject taught at the school. The quiz will be uploaded to the school's website so that any student is able to download the quiz, and play it at home. This provides a more enjoyable experience to learning rather than the traditional studying off books and notes. As to reiterate the new system of twenty first century learning, multiple quizzes can be made in order to conduct tests on the school's computers.

In fact, students at Milton High are just about to return from their two week half term break and the school administrator team has instructed each and every computing student to custom build a quiz for each subject. We have been instructed to make a quiz for the IT department. This aims to test all students' knowledge, also testing if any of them revised over the course of the two week break. The quiz is intended to be fun for all students. It will consist of the following topics; Input and Output, Hardware and Software, Networking and some common knowledge questions.

The quiz will be held in the first week that the students return from their half term break. It will also contain a database which is able to store every student's marks and prizes will be given out to the highest scoring twenty students.

The quiz is not the only activity the school will be organising in order to transfer to 21st century learning. Several lessons have been re-thought in order to achieve this.

1.2 - Statement of the results to be required

All results will be saved at the end of a quiz and stored in a .txt file named. All the data found in the file can either be viewed by opening it, or by clicking a menu button which is found on the home screen of the program.

Below are listed all the different values that are created or saved once the quiz has ended.

Field Name	Data Type	Obtained by
Name	String	Entered by user at the end of the quiz
Total Score	Double	Score + Bonus
Bonus	Double	A bonus is given if the quiz is completed quickly
Score	Double	Correct amount x10
Correct	int	Number of correct answers
Incorrect	int	Number of incorrect answers
Time	int	Time taken to complete quiz
Date	String	Date that the quiz was played

1.3: Details of any input/output information required

Input information

Screen	What is Inputted?
Home	Menu navigation
Question	Answer of choice
End	Username

Output information

Screen	What is displayed?
Home	Menu, Features, Help
Scores	User score information
Help	Help information
Question	Question, Answers, Correct / Incorrect info, Hangman
End	Correct / Incorrect score, Time taken, Save

SECTION 2 – Solution to the problem

2.1 – Details of any special design features.

In the following section, a few methods have been selected and will be explained with the use of pseudo-code.

2.1.1 - Pseudo-code of the generateQuestions() algorithm

The main use of this method is to obtain a shuffled list of numbers at the beginning of the program so that every quiz has a different set of questions.

1. In the first for loop, randomNumbers[] is filled up with the numbers 1 to 50. This is done by adding the increment value "i" to the array element.

2. In the second loop, the numbers in randomNumbers[] are shuffled using a separate class called "Collections".

```
public void generateQuestions() {
    for (int i = 0; i < randomNumbers.length; i++) {
        randomNumbers[i] = i;
    }
    Collections.shuffle(Arrays.asList(randomNumbers));
    for (int i = 0; i < randomNumbers.length; i++) {
        shuffledNumbers[i] = randomNumbers[i];
    }
}
```

Figure 2.1: Screenshot of the generateQuestions() method.

2.1.2 - Pseudo-code of the handleOptions() algorithm

The use of this method is to check whether the user is entering the correct information when playing the quiz. At the end, the scores are summed and other methods are run. Each time the user submits an answer, the handleOptions() method is called.

1. In the first 6 'if' statements, the program checks whether two or more answers are selected.

2. If two or more answers are selected, the error status is given a specific value, then all the checkboxes are cleared by calling the clearCheckBoxes method and a command is sent to open the error window, warning the user that more than one answer was selected. This cancels the event and the question will not be answered.

3. The program then checks if at least one answer is selected and sends a command to the correctClass class which checks whether the question is correct or incorrect.

4. If no answers were selected, the error status is given a value and a command is sent to the errorWindow class which opens up a warning window explaining to the user that no answers were selected. The event is also cancelled and the user is forced to select one answer.

5. Then the correctScore value is obtained from the correctClass using a method called getGood(). The incorrectScore is also obtained from the correctClass using the getBad() method.

6. The status is then reset from the previous question and it is re-set to the current question which is then displayed on the GUI. It is used to show whether the last question answered was correct or incorrect.

7. The program then checks if 7 questions were answered incorrectly and if the statement is true, the quiz is terminated, if not, the quiz continues.

8. The program then checks whether it was the last question, if the statement is true, the quiz is terminated, the timers are stopped and the endQuiz() method is called which properly ends the quiz, if not, the quiz continues to the next question.

```
public void handleOptions(CheckBox answerOne, CheckBox answerTwo, CheckBo

    Correct correctClass = new Correct();

    if (answerOne.isSelected() && answerTwo.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
    } else if (answerOne.isSelected() && answerThree.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);

        errorWindow.generalErrors(errorStatus);
        return;
    } else if (answerOne.isSelected() && answerFour.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
    } else if (answerTwo.isSelected() && answerThree.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
    } else if (answerTwo.isSelected() && answerFour.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
    } else if (answerThree.isSelected() && answerFour.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
```

Figure 2.2: Screenshot of the handleOptions() method.

```
} else if (answerOne.isSelected()) {
    correctClass.correctAnswer(shuffledNumbers[i], 1);
} else if (answerTwo.isSelected()) {
    correctClass.correctAnswer(shuffledNumbers[i], 2);
} else if (answerThree.isSelected()) {
    correctClass.correctAnswer(shuffledNumbers[i], 3);
} else if (answerFour.isSelected()) {
    correctClass.correctAnswer(shuffledNumbers[i], 4);
} else {
    errorStatus = "nothingSelectedError";
    errorWindow.generalErrors(errorStatus);
    return;
correctScore = correctScore + correctClass.getGood();
incorrectScore = incorrectScore + correctClass.getBad();
status = "";
status = status + correctClass.getStatus();

try {
    if (incorrectScore == 7) {
        window.close();
        lostHangMan(primaryStage);
    } else if (i < 9) {
        beginQuiz(primaryStage);
    } else {
        endTimer();
        sumTimer();
        sumScore();
        endQuiz(primaryStage);
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

Figure 2.2: Screenshot of the handleOptions() method.

2.1.3 - Pseudo-code of the sumScore() algorithm

This method adds an extra feature to Quiztime. This feature gives the player more points, the faster the quiz is completed.

1. The sumScore[] method is called from the handleOptions[] method.

2. Ten times the correct score amount is then added to the total score using a mathematical operation.

3. For the next 'if' statements, the program decides how much of a bonus is to be added with the total score of the user. As seen in the method, the quicker the user completes the quiz, the bigger of a bonus the user gets which is then added to the total score.

4. The program then calculates bonusScore and noBonusScore, which are then scored in "quiztime_data.txt" in another method.

```
public void sumScore() {
    DecimalFormat roundFormat = new DecimalFormat("0.00");
    int time = Integer.parseInt(userTime);
    totalScore = correctScore * 10;
    if (time <= 10) {
        totalScore = totalScore * 3;
    } else if (time >= 11 && time <= 15) {
        totalScore = totalScore * 2.8;
    } else if (time >= 16 && time <= 20) {
        totalScore = totalScore * 2.6;
    } else if (time >= 21 && time <= 25) {
        totalScore = totalScore * 2.4;
    } else if (time >= 26 && time <= 30) {
        totalScore = totalScore * 2.2;
    } else if (time >= 31 && time <= 35) {
        totalScore = totalScore * 2;
    } else if (time >= 36 && time <= 40) {
        totalScore = totalScore * 1.8;
    } else if (time >= 41 && time <= 45) {
        totalScore = totalScore * 1.6;
    } else if (time >= 46 && time <= 50) {
        totalScore = totalScore * 1.4;
    } else if (time >= 51 && time <= 55) {
        totalScore = totalScore * 1.2;
    } else {
        totalScore = correctScore * 10;
    }
    String score = roundFormat.format(totalScore);
    try {
        totalScore = Double.parseDouble(score);
        bonusScore = totalScore - (correctScore * 10);
        noBonusScore = totalScore - bonusScore;
    } catch (Exception e) {
        errorStatus = "sumScoreError";
        errorWindow errorWindow = new errorWindow();
        errorWindow.generalErrors(errorStatus);
```

Figure 2.3: Screenshot of the sumScore() method.

2.1.4 – Pseudo-code of the correctAnswers() algorithm

The use of this method found in the ‘Correct’ class checks whether the answer selected by the user is correct or incorrect, and returns the value.

1. The integer ‘num1’ contains the question number generated in the ‘Main’ class. The other integer ‘num2’ contains the value of the selected answer (1, 2, 3 or 4).

2. The first integer is checked for in each ‘if statement’ and if the value matches, the ‘if statement’ goes onto the next step.

3. The second ‘if statement’ checks whether the value of the selected answer chosen by the user matches the value inside the ‘if statement’.

4. If the value of ‘num2’ matches the value in the ‘if statement’, ‘good’ is incremented by 1, the getGood() method is called and ‘status’ is set to ‘Correct’.

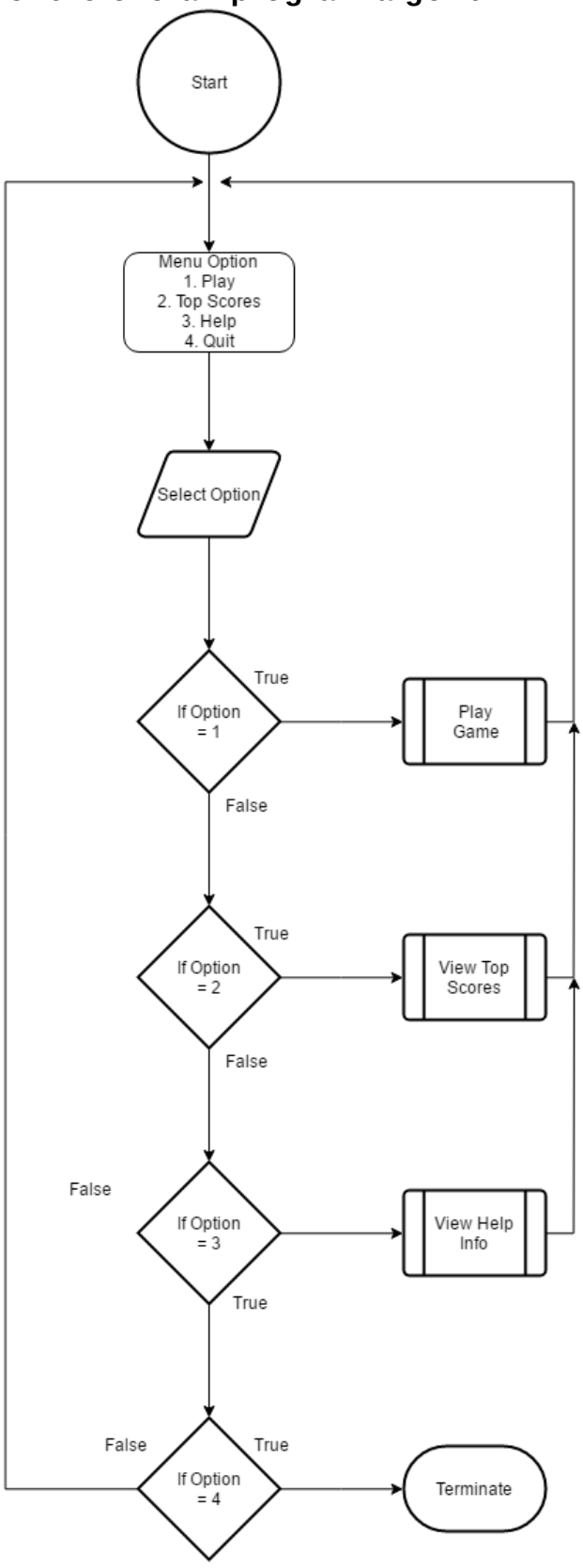
5. If the value of ‘num2’ does not match the value in the ‘if statement’, it is ignored and the ‘else’ part of the code is executed. Therefore ‘bad’ is incremented by 1, the getBad() method is called and ‘status’ is set to ‘Incorrect’.

6. The methods getGood(), getBad() and getStatus() return the values to the main class ‘Main’.

```
public void correctAnswer(int num1, int num2) {

    if (num1 == 0) {
        if (num2 == 3) {
            good++;
            getGood();
            status = "Correct";
        } else {
            bad++;
            getBad();
            status = "Incorrect";
        }

    }

    if (num1 == 1) {
        if (num2 == 1) {
            good++;
            getGood();
            status = "Correct";
        } else {
            bad++;
            getBad();
            status = "Incorrect";
        }

    }

 public int getGood() {
     return good;
 }

 public int getBad() {
     return bad;
 }

 public String getStatus() {
     return status;
 }
```

Figure 2.4: Screenshot of the correctAnswer() method.

2.1.4 - Flowchart of the overall program algorithm



2.2 - Computer listing of program

2.2.1 - Main Class

```
package me.max.main;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Collections;
import java.util.Date;
import java.util.Scanner;

import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
```

```
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;
import me.max.main.calculations.Correct;
import me.max.main.calculations.SetupTable;
import me.max.main.data.Answers;
import me.max.main.data.Questions;
import me.max.main.windows.errorWindow;
import me.max.main.windows.helpWindow;

public class Main extends Application {

        errorWindow errorWindow = new errorWindow();
        DateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");

        Stage window;
        TableView<SetupTable> table;
        Integer[] randomNumbers = new Integer[50];
        String[] file_username = new String[1000];
        Double[] file_usertotalscore = new Double[1000];
        Double[] file_userbonus = new Double[1000];
        Double[] file_userscore = new Double[1000];
        int[] file_usercorrect = new int[1000];     //initializing and assigning variables
        int[] file_userincorrect = new int[1000];
        int[] file_usertime = new int[1000];
        String[] file_userdate = new String[1000];
        int[] shuffledNumbers = new int[50];
        String status = "No answer";
        String userDate = "";
        int i = 0;
        double totalScore;
        double bonusScore;
        double noBonusScore;
        int correctScore = 0;
        int incorrectScore = 0;
```

```
int scoreCount = -1;
int tableCount = 0;
boolean error = false;
public String errorStatus = "";
String startTime = "";
String endTime = "";
String userTime = "";

public static void main(String[] args) {
	launch(args);
}

@Override
public void start(Stage primaryStage) throws Exception {
	generateQuestions();
	correctScore = 0;          //menu method
	incorrectScore = 0;
	bonusScore = 0;
	i = 0;

	window = primaryStage;
	window.setTitle("QuizTime - Home");

	HBox topMenu = new HBox();
	topMenu.setPadding(new Insets(10, 10, 10, 10));

	VBox midMenu = new VBox();
	midMenu.setPadding(new Insets(10, 10, 10, 10));   //building scene
	midMenu.setSpacing(20);

	HBox midMenu1 = new HBox();
	midMenu1.setPadding(new Insets(10, 10, 10, 10));
	midMenu1.setSpacing(10);

	VBox midMenu2 = new VBox();
	midMenu1.setPadding(new Insets(10, 10, 10, 10));
```

```
midMenu1.setSpacing(20);

HBox botMenu = new HBox();
botMenu.setPadding(new Insets(10, 10, 10, 10));
botMenu.setSpacing(10);

Label updateLabel = new Label("Last Updated: 07/12/2016");
Button helpButton = new Button("Help");

// Welcome Label
Label welcomeLabel = new Label("Welcome to QuizTime!");
welcomeLabel.setFont(Font.font("Verdana Pro Semibold", 22));

// Play Button
Button playButton = new Button("Play");
playButton.setFont(Font.font("Verdana Pro Semibold", 22));

// Top Score Button
Button topScoreButton = new Button("Top Scores");
topScoreButton.setFont(Font.font("Verdana Pro Semibold", 22));

// Quit Button
Button quitButton = new Button("Quit");
quitButton.setFont(Font.font("Verdana Pro Semibold", 22));

/*
 * Button debugButton = new Button("Debug"); debugButton.setOnAction(e
 * -> { endQuiz(primaryStage); });
 */

Label featureLabel1 = new Label("Features:");
featureLabel1.setFont(Font.font("Verdana Pro Semibold", FontWeight.BOLD, 22));

Label featureLabel2 = new Label("\n[*] High scores table.\n" + "[*] Easy and simple
to use.\n" + "[*] Includes hang-man.\n"
+ "[*] Keep track of time taken to complete.\n" + "[*] Keep track of correct and
incorrect score.\n"
```

```
+ "[*] Save stats with custom username.\n" + "[*] Quicker to finish = more points +
more comming soon.\n");


featureLabel2.setFont(Font.font("Verdana Pro Semibold", 12));


topMenu.getChildren().addAll(welcomeLabel);
midMenu1.getChildren().addAll(playButton, topScoreButton, quitButton);
midMenu2.getChildren().addAll(featureLabel1, featureLabel2);
midMenu.getChildren().addAll(midMenu1, midMenu2);
botMenu.getChildren().addAll(helpButton, updateLabel);


topMenu.setAlignment(Pos.CENTER);
midMenu.setAlignment(Pos.CENTER);
midMenu1.setAlignment(Pos.CENTER);
midMenu2.setAlignment(Pos.CENTER);
botMenu.setAlignment(Pos.CENTER);


BorderPane borderPane = new BorderPane();


borderPane.setTop(topMenu);
borderPane.setCenter(midMenu);
borderPane.setBottom(botMenu);


Scene homeScene = new Scene(borderPane, 815, 430);
window.setScene(homeScene);
window.show();


playButton.setOnAction(e -> {
       beginQuiz(primaryStage);
       startTimer();
});


topScoreButton.setOnAction(e -> {
       scoreDisplay(homeScene);
});


quitButton.setOnAction(e -> window.close());
```

```
        helpButton.setOnAction(e -> {
                helpWindow helpWindow = new helpWindow();
                helpWindow.window();
        });

}

public void beginQuiz(Stage primaryStage) {

        Questions questionClass = new Questions();
        Answers answerClass = new Answers();
        window.setTitle("QuizTime - Question " + (i + 1) + " of 10");

        questionClass.listQuestions();
        answerClass.listAnswers();              //method sets up question and answers

        Button answerButton = new Button("Answer!");
        answerButton.setFont(Font.font("Verdana Pro Semibold", 17));
        Button quitButton = new Button("Quit to Home Screen.");
        quitButton.setFont(Font.font("Verdana Pro Semibold", 12));

        GridPane layout = new GridPane();
        layout.setPadding(new Insets(10, 10, 10, 10));
        layout.setVgap(20);
        layout.setHgap(10);

        Label questionLabel = new Label(questionClass.printQuestion(shuffledNumbers[i]));
        questionLabel.setFont(Font.font("Verdana Pro Semibold", 18));
        GridPane.setConstraints(questionLabel, 0, 0);

        CheckBox answerOne = new CheckBox(answerClass.printAnswer(shuffledNumbers[i], 0));
        answerOne.setFont(Font.font("Arial Unicode MS", 17));
        GridPane.setConstraints(answerOne, 0, 1);

        CheckBox answerTwo = new CheckBox(answerClass.printAnswer(shuffledNumbers[i], 1));
```

```
answerTwo.setFont(Font.font("Arial Unicode MS", 17));

GridPane.setConstraints(answerTwo, 0, 2);


CheckBox answerThree = new CheckBox(answerClass.printAnswer(shuffledNumbers[i],
2));

answerThree.setFont(Font.font("Arial Unicode MS", 17));

GridPane.setConstraints(answerThree, 0, 3);


CheckBox answerFour = new CheckBox(answerClass.printAnswer(shuffledNumbers[i], 3));

answerFour.setFont(Font.font("Arial Unicode MS", 17));

GridPane.setConstraints(answerFour, 0, 4);


GridPane.setConstraints(answerButton, 0, 6);


Pane spacePane = new Pane();

GridPane.setConstraints(spacePane, 0, 5);


layout.getChildren().addAll(spacePane, questionLabel, answerButton, answerOne,
answerTwo, answerThree, answerFour);


BorderPane borderPane = new BorderPane();

borderPane.setTop(layout);


Scene questionsScene = new Scene(borderPane, 815, 430);

window.setScene(questionsScene);

window.show();


Label lastAnswerLabel = new Label();

lastAnswerLabel.setText("Correct: " + correctScore + " | Incorrect: " +
incorrectScore + " | Last Question: " + status + " | Chances: "

+ (6 - incorrectScore));


lastAnswerLabel.setFont(Font.font("Verdana Pro Semibold", 12));


// Bottom Area

HBox bottomMenu = new HBox();

bottomMenu.setPadding(new Insets(10, 10, 10, 10));

bottomMenu.setSpacing(10);
```

```
bottomMenu.setAlignment(Pos.CENTER);

bottomMenu.getChildren().addAll(lastAnswerLabel, quitButton);


borderPane.setBottom(bottomMenu);


int question = i;

answerButton.setOnAction(e -> {

i++;

handleOptions(answerOne, answerTwo, answerThree, answerFour, question,
primaryStage);

});


quitButton.setOnAction(e -> {

	try {

		start(primaryStage);

	} catch (Exception e1) {

		e1.printStackTrace();

	}

});


Label hangLabel = new Label(); //setting up hangman

hangLabel.setFont(Font.font("Verdana Pro Semibold", 12));


if (incorrectScore == 1) {

	hangLabel.setText("|\n|\n|\n|\n|\n|");

}


if (incorrectScore == 2) {

	hangLabel.setText("_________\n|\n|\n|\n|\n|\n|");

}


if (incorrectScore == 3) {

hangLabel.setText("_________\n" + "|             |\n" + "|\n" + "|\n" + "|\n" +
"|\n" + "|");

}


if (incorrectScore == 4) {
```

```
            hangLabel.setText("_________\n" + "|             |\n" + "|             0\n" + "|\n"
            + "|\n" + "|\n" + "|");

            }


            if (incorrectScore == 5) {

            hangLabel.setText("_________\n" + "|             |\n" + "|             0\n" + "|
            /|\\\n" + "|\n" + "|\n" + "|");

            }


            if (incorrectScore == 6) {

            hangLabel.setText("_________\n" + "|             |\n" + "|             0\n" + "|
            /|\\\n" + "|            / \\\n" + "|\n"

                                 + "|  Last Chance!");

            }


            borderPane.setCenter(hangLabel);


      }


      public void handleOptions(CheckBox answerOne, CheckBox answerTwo, CheckBox answerThree,
             CheckBox answerFour, int i, Stage primaryStage) {


             Correct correctClass = new Correct();


             if (answerOne.isSelected() && answerTwo.isSelected()) { //verifying answer

                    errorStatus = "twoOrMoreSelectedError";

                    clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);

                    errorWindow.generalErrors(errorStatus);

                    return;

             } else if (answerOne.isSelected() && answerThree.isSelected()) {

                    errorStatus = "twoOrMoreSelectedError";

                    clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);


                    errorWindow.generalErrors(errorStatus);

                    return;

             } else if (answerOne.isSelected() && answerFour.isSelected()) {

                    errorStatus = "twoOrMoreSelectedError";

                    clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
```

```
        errorWindow.generalErrors(errorStatus);
        return;
} else if (answerTwo.isSelected() && answerThree.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
} else if (answerTwo.isSelected() && answerFour.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
} else if (answerThree.isSelected() && answerFour.isSelected()) {
        errorStatus = "twoOrMoreSelectedError";
        clearCheckBoxes(answerOne, answerTwo, answerThree, answerFour);
        errorWindow.generalErrors(errorStatus);
        return;
} else if (answerOne.isSelected()) { //verifying answer
        correctClass.correctAnswer(shuffledNumbers[i], 1);
} else if (answerTwo.isSelected()) {
        correctClass.correctAnswer(shuffledNumbers[i], 2);
} else if (answerThree.isSelected()) {
        correctClass.correctAnswer(shuffledNumbers[i], 3);
} else if (answerFour.isSelected()) {
        correctClass.correctAnswer(shuffledNumbers[i], 4);
} else {
        errorStatus = "nothingSelectedError";
        errorWindow.generalErrors(errorStatus);
        return;
}

correctScore = correctScore + correctClass.getGood(); //retrieving info from
incorrectScore = incorrectScore + correctClass.getBad(); //respective methods
status = "";
status = status + correctClass.getStatus();
```

```
		try {
			if (incorrectScore == 7) {
				window.close();
				lostHangMan(primaryStage);
			} else if (i < 9) {
				beginQuiz(primaryStage);
			} else {
				endTimer();
				sumTimer();
				sumScore();
				endQuiz(primaryStage);
			}
		} catch (Exception e) {
			e.printStackTrace();
		}

	}

	public void endQuiz(Stage primaryStage) { //executed when quiz is over
		window = primaryStage;
		window.setTitle("QuizTime - End");

		// layout one

		VBox topMenu = new VBox();
		topMenu.setPadding(new Insets(10, 10, 10, 10));
		topMenu.setAlignment(Pos.CENTER);

		HBox midMenu = new HBox();
		midMenu.setPadding(new Insets(10, 10, 10, 10));     //building scene
		midMenu.setSpacing(20);
		midMenu.setAlignment(Pos.CENTER);

		HBox botMenu = new HBox();
		botMenu.setPadding(new Insets(10, 10, 10, 10));
		botMenu.setAlignment(Pos.CENTER);
```

```
Label label1 = new Label("Quiz finished!");
label1.setFont(Font.font("Verdana Pro Semibold", 22));

Label label2 = new Label("Enter name to save score(Not more than 21 chars).");
label2.setFont(Font.font("Verdana Pro Semibold", 22));

Label label3 = new Label("Correct: " + correctScore + " | Incorrect: " +
incorrectScore + " | Time Taken: " + userTime + " seconds.");
label3.setFont(Font.font("Verdana Pro Semibold", 12));

TextField usernameField = new TextField();
usernameField.setFont(Font.font(18));
usernameField.setPromptText("username");

Button saveButton = new Button("Save");
saveButton.setFont(Font.font("Verdana Pro Semibold", 18));

topMenu.getChildren().addAll(label1, label2);
midMenu.getChildren().addAll(usernameField, saveButton);
botMenu.getChildren().addAll(label3);

BorderPane borderPane = new BorderPane();
borderPane.setTop(topMenu);
borderPane.setCenter(midMenu);
borderPane.setBottom(botMenu);

Scene scene1 = new Scene(borderPane, 815, 430);
window.setScene(scene1);
window.show();

saveButton.setOnAction(e -> {
       if (usernameField.getText().equalsIgnoreCase("")) {
              errorStatus = "nonameError";
              errorWindow.generalErrors(errorStatus);
              return;
       } else if (usernameField.getLength() > 21) {
```

```
                        usernameField.clear();
                        errorStatus = "usernameLengthError";
                        errorWindow.generalErrors(errorStatus);
                } else {
                        loadDate();
                        writeScore(usernameField, totalScore, userTime, userDate);
                        try {
                                start(primaryStage);
                        } catch (Exception e1) {
                                e1.printStackTrace();
                        }
                }
        });

}

public void lostHangMan(Stage primaryStage) { //executed when all user chances
        window = primaryStage;                // are used up
        window.setTitle("Lost");

        HBox layout = new HBox();
        layout.setAlignment(Pos.CENTER);

        Label label = new Label("You've lost the hangman!  ");
        Button button = new Button("Continue");

        layout.getChildren().addAll(label, button);

        Scene scene = new Scene(layout, 300, 200);
        window.setScene(scene);
        window.show();

        button.setOnAction(e -> {
                window.close();
                try {
                        start(primaryStage);
```

```
                } catch (Exception e1) {
                        e1.printStackTrace();
                }
        });
    }

    public void generateQuestions() {                        //generates numbers from 0-49
        for (int i = 0; i < randomNumbers.length; i++) { //stored in variable
                randomNumbers[i] = i;
        }
        Collections.shuffle(Arrays.asList(randomNumbers));
        for (int i = 0; i < randomNumbers.length; i++) { //shuffles the numbers to be used
                shuffledNumbers[i] = randomNumbers[i];   //to randomize quiz questions
        }
    }

public void writeScore(TextField name, Double score, String userTime, String userDate) {
        try (FileWriter fw = new FileWriter("quiztime_data.txt", true);
                        BufferedWriter bw = new BufferedWriter(fw);
                        PrintWriter out = new PrintWriter(bw)) {
                        out.println(name.getText() + "\n" + Double.toString(score) + "\n" +
                        bonusScore + "\n" + noBonusScore + "\n" + correctScore + "\n" +
                        incorrectScore + "\n" + userTime + "\n" + userDate);
        } catch (Exception e) {
                e.printStackTrace();
        }
    }

    @SuppressWarnings("unchecked")
    public void scoreDisplay(Scene homeScene) { //Top Score table method
        loadScores();
        if (error == true) {
                return;
        }
        window.setTitle("QuizTime - Scores");
        // Name column
        TableColumn<SetupTable, String> usernameColumn = new TableColumn<>("Name");
```

```
usernameColumn.setMinWidth(150);
usernameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

// Score column
TableColumn<SetupTable, Double> usertotalscoreColumn = new TableColumn<>("Total
Score");
usertotalscoreColumn.setMinWidth(75);
usertotalscoreColumn.setCellValueFactory(new PropertyValueFactory<>("totalscore"));

// Bonus column
TableColumn<SetupTable, Double> userbonusColumn = new  TableColumn<>("Bonus");
userbonusColumn.setMinWidth(75);
userbonusColumn.setCellValueFactory(new PropertyValueFactory<>("bonus"));

// Score column
TableColumn<SetupTable, Double> userscoreColumn = new TableColumn<>("Score");
userscoreColumn.setMinWidth(75);
userscoreColumn.setCellValueFactory(new PropertyValueFactory<>("score"));

// Correct column
TableColumn<SetupTable, Integer> usercorrectColumn = new TableColumn<>("Correct");
usercorrectColumn.setMinWidth(75);
usercorrectColumn.setCellValueFactory(new PropertyValueFactory<>("correct"));

// Incorrect column
TableColumn<SetupTable, Integer> userIncorrectColumn = new
TableColumn<>("Incorrect");
userIncorrectColumn.setMinWidth(75);
userIncorrectColumn.setCellValueFactory(new PropertyValueFactory<>("incorrect"));

// Time taken Column
TableColumn<SetupTable, Integer> usertimeColumn = new TableColumn<>("Time (s)");
usertimeColumn.setMinWidth(75);
usertimeColumn.setCellValueFactory(new PropertyValueFactory<>("time"));

// Date Column
TableColumn<SetupTable, String> userdateColumn = new TableColumn<>("Date");
```

```
        userdateColumn.setMinWidth(75);
        userdateColumn.setCellValueFactory(new PropertyValueFactory<>("date"));


        Button backButton = new Button("Back");


        table = new TableView<>();
        table.setItems(getUsername());
        table.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
        table.getColumns().addAll(usernameColumn, usertotalscoreColumn, userbonusColumn,
        userscoreColumn, usercorrectColumn, userIncorrectColumn, usertimeColumn,
        userdateColumn);


        VBox vBox = new VBox();
        vBox.getChildren().addAll(table);


        HBox hBox = new HBox();
        hBox.setAlignment(Pos.CENTER);
        hBox.getChildren().addAll(backButton);


        BorderPane borderPane = new BorderPane();
        borderPane.setTop(vBox);
        borderPane.setBottom(hBox);


        Scene scoreDisplayScene = new Scene(borderPane, 815, 430);
        window.setScene(scoreDisplayScene);


        backButton.setOnAction(e -> {
                window.setTitle("QuizTime - Home");
                window.setScene(homeScene);
        });


    }


    public ObservableList<SetupTable> getUsername() { //fills up table with data
        ObservableList<SetupTable> username = FXCollections.observableArrayList();
        for (tableCount = 0; tableCount <= scoreCount; tableCount++) {
        username.add(new SetupTable(file_username[tableCount],
        file_usertotalscore[tableCount], file_userbonus[tableCount],
```

```
                file_userscore[tableCount], file_usercorrect[tableCount],
                file_userincorrect[tableCount], file_usertime[tableCount],
                file_userdate[tableCount]));

                }

                return username;

        }


        public void loadScores() { //fills up table variables from file

                File file = new File("quiztime_data.txt");

                scoreCount = -1;

                try {

                        Scanner sc = new Scanner(file);

                        while (sc.hasNextLine()) {

                                scoreCount++;

                                String uname = sc.nextLine();

                                file_username[scoreCount] = uname;

                                String utotalscore = sc.nextLine();

                                file_usertotalscore[scoreCount] = Double.parseDouble(utotalscore);

                                String ubonus = sc.nextLine();

                                file_userbonus[scoreCount] = Double.parseDouble(ubonus);

                                String uscore = sc.nextLine();

                                file_userscore[scoreCount] = Double.parseDouble(uscore);

                                String ucorr = sc.nextLine();

                                file_usercorrect[scoreCount] = Integer.parseInt(ucorr);

                                String uincorr = sc.nextLine();

                                file_userincorrect[scoreCount] = Integer.parseInt(uincorr);

                                String utime = sc.nextLine();

                                file_usertime[scoreCount] = Integer.parseInt(utime);

                                String udate = sc.nextLine();

                                file_userdate[scoreCount] = udate;

                        }

                        sc.close();

                } catch (Exception e) {

                        errorStatus = "fileError";

                        errorWindow.generalErrors(errorStatus);

                        error = true;

                }
```

```
	}

	public void loadDate() { //creating date to be used in user final score
		DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
		Date date = new Date();
		userDate = dateFormat.format(date);
	}

	public void startTimer() {
		Date time = new Date();
		startTime = timeFormat.format(time);
	}

	public void endTimer() {
		Date time = new Date();
		endTime = timeFormat.format(time);
	}

	public void sumTimer() {		//summing total time taken to complete
		try {
			Date timeStarted = timeFormat.parse(startTime);
			Date timeEnded = timeFormat.parse(endTime);
				long difference = (timeEnded.getTime() -
			timeStarted.getTime()) / 1000;
			userTime = Long.toString(difference);
		} catch (ParseException e) {
			e.printStackTrace();
		}
	}

public void clearCheckBoxes(CheckBox answerOne, CheckBox answerTwo, CheckBox answerThree, CheckBox
		answerFour) {
		answerOne.setSelected(false);		//clear checkboxes on error
		answerTwo.setSelected(false);
		answerThree.setSelected(false);
		answerFour.setSelected(false);
```

```
	}

	public void sumScore() {
		DecimalFormat roundFormat = new DecimalFormat("0.00");
		int time = Integer.parseInt(userTime);
		totalScore = correctScore * 10;		//sum score at the end of the quiz
		if (time <= 10) {
			totalScore = totalScore * 3;
		} else if (time >= 11 && time <= 15) {
			totalScore = totalScore * 2.8;
		} else if (time >= 16 && time <= 20) {
			totalScore = totalScore * 2.6;
		} else if (time >= 21 && time <= 25) {
			totalScore = totalScore * 2.4;
		} else if (time >= 26 && time <= 30) {
			totalScore = totalScore * 2.2;
		} else if (time >= 31 && time <= 35) {
			totalScore = totalScore * 2;
		} else if (time >= 36 && time <= 40) {
			totalScore = totalScore * 1.8;
		} else if (time >= 41 && time <= 45) {
			totalScore = totalScore * 1.6;
		} else if (time >= 46 && time <= 50) {
			totalScore = totalScore * 1.4;
		} else if (time >= 51 && time <= 55) {
			totalScore = totalScore * 1.2;
		} else {
			totalScore = correctScore * 10;
		}
		String score = roundFormat.format(totalScore);
		try {
			totalScore = Double.parseDouble(score);
			bonusScore = totalScore - (correctScore * 10);
			noBonusScore = totalScore - bonusScore;
		} catch (Exception e) {
			errorStatus = "sumScoreError";
```

```
                    errorWindow errorWindow = new errorWindow();
                    errorWindow.generalErrors(errorStatus);
                }
        }

}
```

2.2.2 - Correct Class

```
package me.max.main.calculations;

public class Correct {
	String[][] correct = new String[30][4];
	public int good = 0;
	public int bad = 0;
	public String status;

	public void correctAnswer(int num1, int num2) {  //method to verify user input

		if (num1 == 0) {
			if (num2 == 3) {
				good++;
				getGood();
				status = "Correct";
			} else {
				bad++;
				getBad();
				status = "Incorrect";
			}

		}

		if (num1 == 1) {
			if (num2 == 1) {
				good++;
				getGood();
				status = "Correct";
			} else {
				bad++;
				getBad();
				status = "Incorrect";
			}

		}

		if (num1 == 2) {
			if (num2 == 2) {
				good++;
				getGood();
				status = "Correct";
			} else {
				bad++;
				getBad();
				status = "Incorrect";
			}

		}

		if (num1 == 3) {
			if (num2 == 4) {
				good++;
				getGood();
				status = "Correct";
			} else {
				bad++;
				getBad();
				status = "Incorrect";
			}

		}

		if (num1 == 4) {
			if (num2 == 4) {
```

```
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 5) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 6) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 7) {
        if (num2 == 4) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 8) {
        if (num2 == 3) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 9) {
        if (num2 == 1) {
                good++;
```

```
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 10) {
        if (num2 == 3) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 11) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 12) {
        if (num2 == 1) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 13) {
        if (num2 == 4) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 14) {
        if (num2 == 2) {
                status = "Correct";
                good++;
```

```
                getGood();
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 15) {
        if (num2 == 1) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 16) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 17) {
        if (num2 == 1) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 18) {
        if (num2 == 3) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 19) {
        if (num2 == 4) {
                good++;
                getGood();
                status = "Correct";
```

```
		} else {
			bad++;
			getBad();
			status = "Incorrect";
		}

	}

	if (num1 == 20) {
		if (num2 == 3) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";

		}

	}

	if (num1 == 21) {
		if (num2 == 2) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";

		}

	}

	if (num1 == 22) {
		if (num2 == 1) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";

		}

	}

	if (num1 == 23) {
		if (num2 == 3) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";
		}

	}

	if (num1 == 24) {
		if (num2 == 4) {
			good++;
```

```
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 25) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 26) {
        if (num2 == 3) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 27) {
        if (num2 == 1) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 28) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";
        }

}

if (num1 == 29) {
        if (num2 == 1) {
                good++;
                getGood();
```

```
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 30) {
        if (num2 == 4) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 31) {
        if (num2 == 3) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 32) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 33) {
        if (num2 == 4) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}
```

```
if (num1 == 34) {
	if (num2 == 2) {
		good++;
		getGood();
		status = "Correct";
	} else {
		bad++;
		getBad();
		status = "Incorrect";

	}

}

if (num1 == 35) {
	if (num2 == 4) {
		good++;
		getGood();
		status = "Correct";
	} else {
		bad++;
		getBad();
		status = "Incorrect";

	}

}

if (num1 == 36) {
	if (num2 == 1) {
		good++;
		getGood();
		status = "Correct";
	} else {
		bad++;
		getBad();
		status = "Incorrect";

	}

}

if (num1 == 37) {
	if (num2 == 3) {
		good++;
		getGood();
		status = "Correct";
	} else {
		bad++;
		getBad();
		status = "Incorrect";

	}

}

if (num1 == 38) {
	if (num2 == 1) {
		good++;
		getGood();
		status = "Correct";
	} else {
		bad++;
		getBad();
		status = "Incorrect";
```

```
		}

	}

	if (num1 == 39) {
		if (num2 == 4) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";

		}

	}

	if (num1 == 40) {
		if (num2 == 2) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";

		}

	}

	if (num1 == 41) {
		if (num2 == 3) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";

		}

	}

	if (num1 == 42) {
		if (num2 == 1) {
			good++;
			getGood();
			status = "Correct";
		} else {
			bad++;
			getBad();
			status = "Incorrect";

		}

	}

	if (num1 == 43) {
		if (num2 == 2) {
			good++;
			getGood();
			status = "Correct";
		} else {
```

```
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 44) {
        if (num2 == 2) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 45) {
        if (num2 == 3) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 46) {
        if (num2 == 4) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 47) {
        if (num2 == 3) {
                good++;
                getGood();
                status = "Correct";
        } else {
                bad++;
                getBad();
                status = "Incorrect";

        }

}

if (num1 == 48) {
        if (num2 == 2) {
```

```
                        good++;
                        getGood();
                        status = "Correct";
                } else {
                        bad++;
                        getBad();
                        status = "Incorrect";

                }

        }

        if (num1 == 49) {
                if (num2 == 3) {
                        good++;
                        getGood();
                        status = "Correct";
                } else {
                        bad++;
                        getBad();
                        status = "Incorrect";

                }

        }

    }

    public int getGood() {
            return good;
    }

    public int getBad() {
            return bad;
    }

    public String getStatus() {
            return status; //returning whether question is correct or not
    }
}
```

2.2.3 - SetupTable Class

```
package me.max.main.calculations;

public class SetupTable {

	private String name;
	private Double totalscore;
	private Double bonus;
	private Double score;
	private int correct;
	private int incorrect;
	private int time;
	private String date;

	public SetupTable() {
		this.name = "";
		this.totalscore = 0.0;
		this.bonus = 0.0;
		this.score = 0.0;
		this.correct = 0;
		this.incorrect = 0;
		this.time = 0;
		this.date = "";
	}

public SetupTable(String name, Double totalscore, Double bonus, Double score, int  correct, int
		incorrect, int time, String date) {
		this.name = name;
		this.totalscore = totalscore;
		this.bonus = bonus;
		this.score = score;
		this.correct = correct;
		this.incorrect = incorrect;
		this.time = time;
		this.date = date;
	}

	public String getName() {
		return name;
	}

	public void getName(String name) {
		this.name = name;
	}

	public Double getTotalscore() {
		return totalscore;
	}

	public void setTotalscore(Double totalscore) {
		this.totalscore = totalscore;
	}

	public Double getBonus() {
		return bonus;
	}

	public void setBonus(Double bonus) {
		this.score = bonus;
	}

	public Double getScore() {
		return score;
	}
```

```
	public void setScore(Double score) {
		this.score = score;
	}

	public int getCorrect() {
		return correct;
	}

	public void setCorrect(int correct) {
		this.correct = correct;
	}

	public int getIncorrect() {
		return incorrect;
	}

	public void setIncorrect(int incorrect) {
		this.incorrect = incorrect;
	}

	public int getTime() {
		return time;
	}

	public void setTime(int time) {
		this.time = time;
	}

	public String getDate() {
		return date;
	}

	public void setDate(String date) {
		this.date = date;
	}

}
```

2.2.4 - Answers Class

```
package me.max.main.data;

public class Answers {

	public String[][] answers = new String[50][4];

	public void listAnswers(){ //storing the answers in multi-dimensional arrays

		answers[0][0] = "Personal computer";
		answers[0][1] = "Mainframe";
		answers[0][2] = "PDA";
		answers[0][3] = "SuperComputer";

		answers[1][0] = "Monitor";
		answers[1][1] = "Keyboard";
		answers[1][2] = "Mouse";
		answers[1][3] = "Scanner";

		answers[2][0] = "Expensive";
		answers[2][1] = "Something you can touch";
		answers[2][2] = "Difficult to purchase";
		answers[2][3] = "Code writren by programmers";

		answers[3][0] = "Word Processor";
		answers[3][1] = "Spreadsheet";
		answers[3][2] = "Web browser";
		answers[3][3] = "Operating System";

		answers[4][0] = "Personal Computer";
		answers[4][1] = "Stand alone";
		answers[4][2] = "Work alone";
		answers[4][3] = "Work Station";

		answers[5][0] = "Internet";
		answers[5][1] = "Extranet";
		answers[5][2] = "Intranet";
		answers[5][3] = "Network";

		answers[6][0] = "A piece of software";
		answers[6][1] = "A connected hardware device";
		answers[6][2] = "Computer instructions";
		answers[6][3] = "A diskdrive";

		answers[7][0] = "A firewall";
		answers[7][1] = "Defragmentation Software";
		answers[7][2] = "An operating system";
		answers[7][3] = "Anti-virus software";

		answers[8][0] = "Read access memory";
		answers[8][1] = "Ready access memory";
		answers[8][2] = "Random access memory";
		answers[8][3] = "ROM access memory";

		answers[9][0] = "Shareware";
		answers[9][1] = "Beta Software";
		answers[9][2] = "Freeware";
		answers[9][3] = "Netware";

		answers[10][0] = "Single user licence";
		answers[10][1] = "Multiple user licence";
		answers[10][2] = "Site licence";
		answers[10][3] = "Copyright licence";

		answers[11][0] = "OMR";
```

```
answers[11][1] = "MICR";
answers[11][2] = "OCR";
answers[11][3] = "PCR";

answers[12][0] = "Web browser";
answers[12][1] = "Search engine";
answers[12][2] = "Email";
answers[12][3] = "Web sites";

answers[13][0] = "Laptop";
answers[13][1] = "PDA";
answers[13][2] = "Personal Computer";
answers[13][3] = "Supercomputer";

answers[14][0] = "output, processing, input";
answers[14][1] = "input, processing, output";
answers[14][2] = "processing, output, input";
answers[14][3] = "output, input, processing";

answers[15][0] = "WAN";
answers[15][1] = "LAN";
answers[15][2] = "Intranet";
answers[15][3] = "Extranet";

answers[16][0] = "ANSI";
answers[16][1] = "ASCII";
answers[16][2] = "HTML";
answers[16][3] = "IANA";

answers[17][0] = "Graphical User interface";
answers[17][1] = "Graphical User Internet";
answers[17][2] = "Good User Internet";
answers[17][3] = "Good User Interface";

answers[18][0] = "Teleconferencing";
answers[18][1] = "Slacking";
answers[18][2] = "Teleworking";
answers[18][3] = "Working";

answers[19][0] = "Computer Misuse Act";
answers[19][1] = "Copyright Designs & Patents Act";
answers[19][2] = "Computer Protection Act";
answers[19][3] = "Data Protection Act";

answers[20][0] = "Latop";
answers[20][1] = "Supercomputer";
answers[20][2] = "Mainframe";
answers[20][3] = "Personal computer";

answers[21][0] = "Decryption";
answers[21][1] = "Encryption";
answers[21][2] = "Protection";
answers[21][3] = "Detection";

answers[22][0] = "OMR";
answers[22][1] = "OCR";
answers[22][2] = "MICR";
answers[22][3] = "PDA";

answers[23][0] = "Spreadsheet";
answers[23][1] = "Word processor";
answers[23][2] = "Database";
answers[23][3] = "Web browser";

answers[24][0] = "Purchased";
answers[24][1] = "Copied";
```

```
answers[24][2] = "Replaced";
answers[24][3] = "Updated";

answers[25][0] = "ROM";
answers[25][1] = "RAM";
answers[25][2] = "Hard disk";
answers[25][3] = "Floppy Disk";

answers[26][0] = "Bus";
answers[26][1] = "Ring";
answers[26][2] = "Star";
answers[26][3] = "Mesh";

answers[27][0] = "Online banking";
answers[27][1] = "Automatic banking";
answers[27][2] = "E-banking";
answers[27][3] = "Flexible banking";

answers[28][0] = "Alpha testing";
answers[28][1] = "Beta testing";
answers[28][2] = "Delta testing";
answers[28][3] = "Free testing";

answers[29][0] = "Process data and instructions";
answers[29][1] = "Create files and folders";
answers[29][2] = "Allocate out processing time";
answers[29][3] = "Defragment the disk";

answers[30][0] = "Keyboard";
answers[30][1] = "Printer";
answers[30][2] = "Monitor";
answers[30][3] = "Touch screen";

answers[31][0] = "Locks on doors";
answers[31][1] = "Video cameras";
answers[31][2] = "Firewalls";
answers[31][3] = "Security Guards";

answers[32][0] = "Super computer";
answers[32][1] = "Latop";
answers[32][2] = "Personal Computer";
answers[32][3] = "Palmtop";

answers[33][0] = "Computer Aided Disaster";
answers[33][1] = "Computer Aided Drawing";
answers[33][2] = "Computer Aided Desktop";
answers[33][3] = "Computer Aided Design";

answers[34][0] = "Telephone";
answers[34][1] = "Fax";
answers[34][2] = "Video Conference";
answers[34][3] = "Instant Messaging";

answers[35][0] = "Expensive";
answers[35][1] = "Something you can pick up";
answers[35][2] = "Difficult to purchase";
answers[35][3] = "Code or instructions";

answers[36][0] = "Stand alone";
answers[36][1] = "Work alone";
answers[36][2] = "Work station";
answers[36][3] = "Personal station";

answers[37][0] = "Analysis";
answers[37][1] = "Design";
answers[37][2] = "Evaluation";
```

```
        answers[37][3] = "Programming";

        answers[38][0] = "DVD";
        answers[38][1] = "CD";
        answers[38][2] = "USB stick";
        answers[38][3] = "Magnetic tape";

        answers[39][0] = "Palmtop";
        answers[39][1] = "Supercomputer";
        answers[39][2] = "Mainframe";
        answers[39][3] = "Personal computer";

        answers[40][0] = "Extranet";
        answers[40][1] = "Intranet";
        answers[40][2] = "Network";
        answers[40][3] = "Internet";

        answers[41][0] = "Bytes";
        answers[41][1] = "Kilobytes";
        answers[41][2] = "Gigahertz";
        answers[41][3] = "Seconds";

        answers[42][0] = "Utility package";
        answers[42][1] = "Spreadsheet";
        answers[42][2] = "Web browser";
        answers[42][3] = "Desktop publisher";

        answers[43][0] = "Monitor";
        answers[43][1] = "Keyboard";
        answers[43][2] = "Printer";
        answers[43][3] = "Speaker";

        answers[44][0] = "How long you can log on for";
        answers[44][1] = "What you can access on the system";
        answers[44][2] = "Which workstation you must use";
        answers[44][3] = "Your passwords";

        answers[45][0] = "Bytes";
        answers[45][1] = "Kilobytes";
        answers[45][2] = "Megabytes";
        answers[45][3] = "Gigabytes";

        answers[46][0] = "Stop viruses spreading";
        answers[46][1] = "Make people work harder";
        answers[46][2] = "Allocate access rights";
        answers[46][3] = "Keep the network secure";

        answers[47][0] = "You always know the address";
        answers[47][1] = "You can't get viruses";
        answers[47][2] = "You can send attachments";
        answers[47][3] = "They are read immediately";

        answers[48][0] = "Overwrite the last backup";
        answers[48][1] = "Store the backup off site";
        answers[48][2] = "Make a backup of the backup";
        answers[48][3] = "Store the backup in the office";

        answers[49][0] = "CR-R";
        answers[49][1] = "CD-ROM";
        answers[49][2] = "CD-RW";
        answers[49][3] = "CD-WORM";


    }
```

```
	public String printAnswer(int num1, int num2){
				return answers[num1][num2];
	} //returning answers to question scene to be displayed to the user
}
```

2.2.5 - Questions Class

```
package me.max.main.data;

public class Questions {
	public String[] questions = new String[50];

	public void listQuestions(){ //storing the questions
			questions[0] = ("A suitable computer to keep track of appointments. It is
		small and portable.");
		questions[1] = ("Which of these is NOT an input device?");
		questions[2] = ("Hardware is");
		questions[3] = ("An example of system software is ");
		questions[4] = ("A computer which is attached to a network is called a");
		questions[5] = ("Part of a companies' internal website which is available to some
		\ncustomers/suppliers");
		questions[6] = ("A peripheral is");
		questions[7] = ("To detect and deal with a virus which software would be
		required?");
		questions[8] = ("RAM stands for");
		questions[9] = ("Software which can be used for a period of time before payment
		needs to be\n made is called");
		questions[10] = ("If you want to install a piece of software on the whole  network
		you would\n need to purchase a");
		questions[11] = ("Which of these is used to read the magnetic ink on a cheque?");
		questions[12] = ("The following software enables you to view web pages");
		questions[13] = ("The most suitable computer for processing trillions of
		calculations a second\n would be a");
		questions[14] = ("Which of the following is the correct order for computer
		processing?");
		questions[15] = ("A company wants to connect their offices globally, Which should
		they use?");
		questions[16] = ("Which of the following is used to represent alphanumeric
		characters\n in Computers?");
		questions[17] = ("GUI stands for");
		questions[18] = ("Working from home is otherwise known as");
		questions[19] = ("Any personal data which is stored on a computer is covered by
		which of the \nfollowing Acts?");
		questions[20] = ("A utility company such as a gas or water company would use which
		type of \ncomputer for calculating bills?");
		questions[21] = ("What is encoding or scrambling data for transmission across a
		network called?");
		questions[22] = ("Which of these input devices would be most suitable for entering
		answers on \nfrom a multiple choice test?");
		questions[23] = ("Which application would be used to store medical records about
		hospital patients?");
		questions[24] = ("It is important that anti-virus software is regularly ....");
		questions[25] = ("Which of these storage media is also known as 'volatile
		memory'?");
		questions[26] = ("Which of the following network topologies connects every
		workstation with its\n own cable?");
		questions[27] = ("Which of the following is the term used to describe internet
		banking?");
		questions[28] = ("Customers are sometimes asked to test software during the final
		phase of \ndevelopment. This is called");
		questions[29] = ("The job of the CPU is to ...");
		questions[30] = ("Which of these is both an input and an output device?");
		questions[31] = ("Which of these is NOT an example of physical security?");
		questions[32] = ("What would a protable, briefcase sized computer be called?");
		questions[33] = ("What does CAD stand for?");
		questions[34] = ("Which is the best method of communication if you want to send a
		map and \ndirections?");
		questions[35] = ("Software is");
		questions[36] = ("A computer not attached to a network is called a");
		questions[37] = ("The final, review stage of the Systems Life Cycle is called");
```

```
                questions[38] = ("Which of these storage devices would be most suitable for storing
                a movie?");
                questions[39] = ("The most suitable computer for someone working at home would be a
                ");
                questions[40] = ("A private network owned by a company and available to employees
                is called ");
                questions[41] = ("CPU speed is measured in");
                questions[42] = ("Which of these is an example of systems software?");
                questions[43] = ("Which of these is NOT an output device?");
                questions[44] = ("Access rights govern");
                questions[45] = ("RAM is usually measured in");
                questions[46] = ("A firewall is used to");
                questions[47] = ("An advantage of using email is that");
                questions[48] = ("When you make a back up of your system you should");
                questions[49] = ("Which of the following could you regularly save data to?");

        }

        public String printQuestion(int num){
                return questions[num];
        } //returning question to question scene to be answered by user
}
```

2.2.6 - errorWindow Class

```
package me.max.main.windows;

import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class errorWindow {                                  //error method, executed if user
	public void generalErrors(String errorStatus) {    //choose more than one answer or
		Stage errorStage = new Stage();             //no answers at all.
		errorStage.setTitle("QuizTime - Error");
		errorStage.initModality(Modality.APPLICATION_MODAL);

		Label errorLabel = new Label("");
		if (errorStatus.equals("nonameError")) {
			errorLabel.setText("Please enter a valid username.");
		} else if (errorStatus.equals("fileError")) {
		errorLabel.setText("Program encountered an file error.\nAt least one score has to
		be saved at"
		+ " the end of the quiz\nthen restart the program.");
		} else if (errorStatus.equals("nothingSelectedError")) {
			errorLabel.setText("Please select an answer.");
		} else if (errorStatus.equals("twoOrMoreSelectedError")) {
			errorLabel.setText("Only one answer may be selected.");
		} else if (errorStatus.equals("usernameLengthError")) {
			errorLabel.setText("Username is exceeds character limit.");
		} else if (errorStatus.equals("sumScoreError")){
			errorLabel.setText("Error when summing score.");
		}

		Button backButton = new Button("Back");
		backButton.setOnAction(e -> errorStage.close());

		VBox layout = new VBox();
		layout.getChildren().addAll(errorLabel, backButton);
		layout.setAlignment(Pos.CENTER);
		layout.setSpacing(20);

		Scene errorScene = new Scene(layout, 350, 150);
		errorStage.setScene(errorScene);
		errorStage.show();
	}

}
```

2.2.7 - helpWindow Class

```
package me.max.main.windows;

import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class helpWindow {
	public void window(){ //executed when user selects help button on the menu
		Stage helpStage = new Stage();
		helpStage.setTitle("QuizTime - Help");
		helpStage.initModality(Modality.APPLICATION_MODAL);

		Label help1Label = new Label("1) If 'Top Scores' button doesn't open, please
		restart the program. This only happens on the first time.\n"
		+ "2) If 'Top Scores' opens up an error window, make sure at least one game has
		been played.");

		Button closeButton = new Button("Back");
		closeButton.setOnAction(e -> helpStage.close());

		VBox layout = new VBox();
		layout.setSpacing(10);
		layout.setPadding(new Insets(10, 10, 10, 10));
		layout.setAlignment(Pos.CENTER);
		layout.getChildren().addAll(help1Label, closeButton);

		Scene helpScene = new Scene(layout, 560, 250);
		helpStage.setScene(helpScene);
		helpStage.show();
	}
}
```

Section 3 – Running the program

3.1 – Evidence that the solution works.

When the 'Play' button is pressed, the user should be presented with the first question as shown in screenshot 3.1. This screenshot is testing the beginQuiz() method in the 'Main' class.

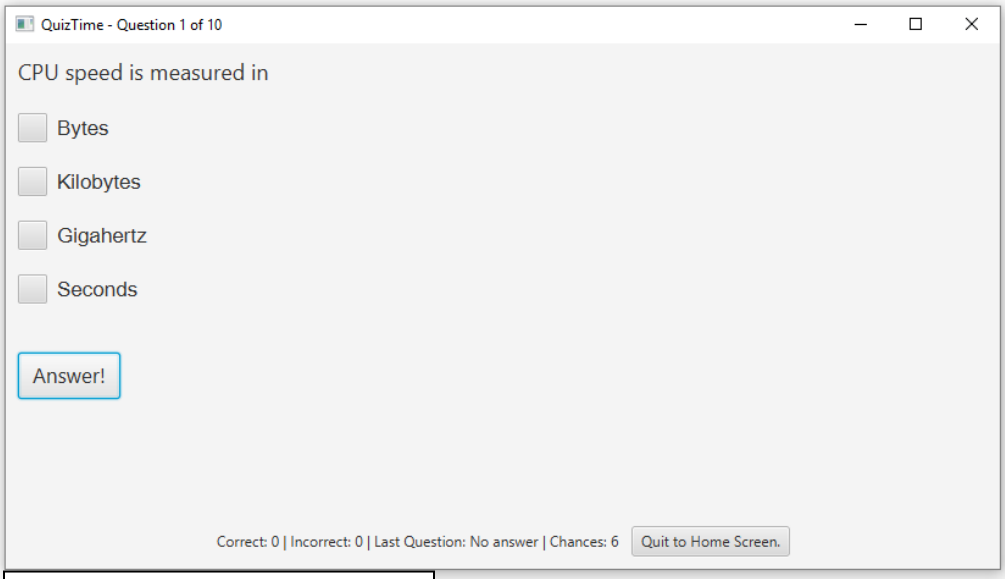


Figure 3.1: Showing first question.

The user selects an answer and then presses the 'Answer!' button as shown in screenshot 3.2. The methods beginQuiz() and handleOptions() are being tested in the following screenshot.

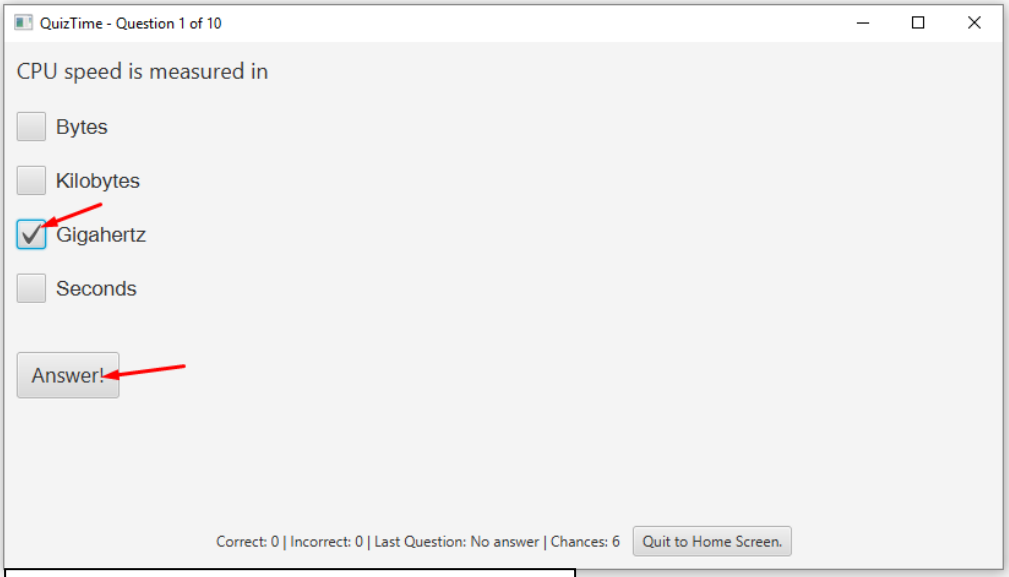


Figure 3.2: Selecting and answering a question.

The answer was correct, this is shown using screenshot 3.3 at the bottom of the screen 'Last Question: Correct' with the use of the beginQuiz(), correctAnswer(), getGood(), getBad() and getStatus() methods.

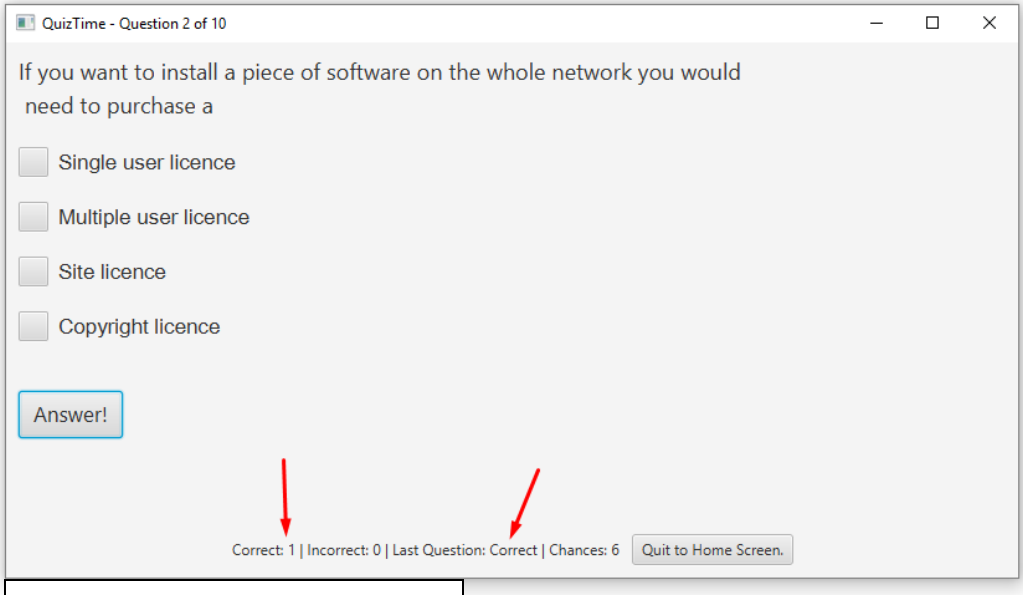


Figure 3.3: Showing change in labels.

The user selects another answer as shown in screenshot 3.4.

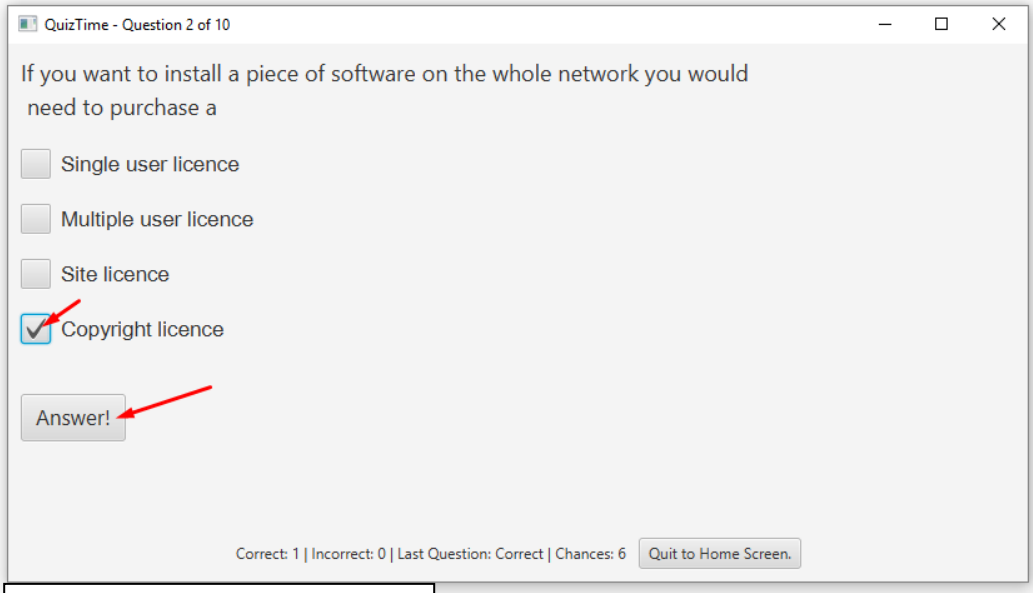


Figure 3.4: Choosing a question.

This time the answer is Incorrect and the value of 'Last Question' has been set to 'Incorrect'. You can also notice the chance decreased by 1 value and the hangman began as shown in screenshot 3.5. The methods beginQuiz(), correctAnswer(), getGood(), getBad() and getStatus() are being tested.

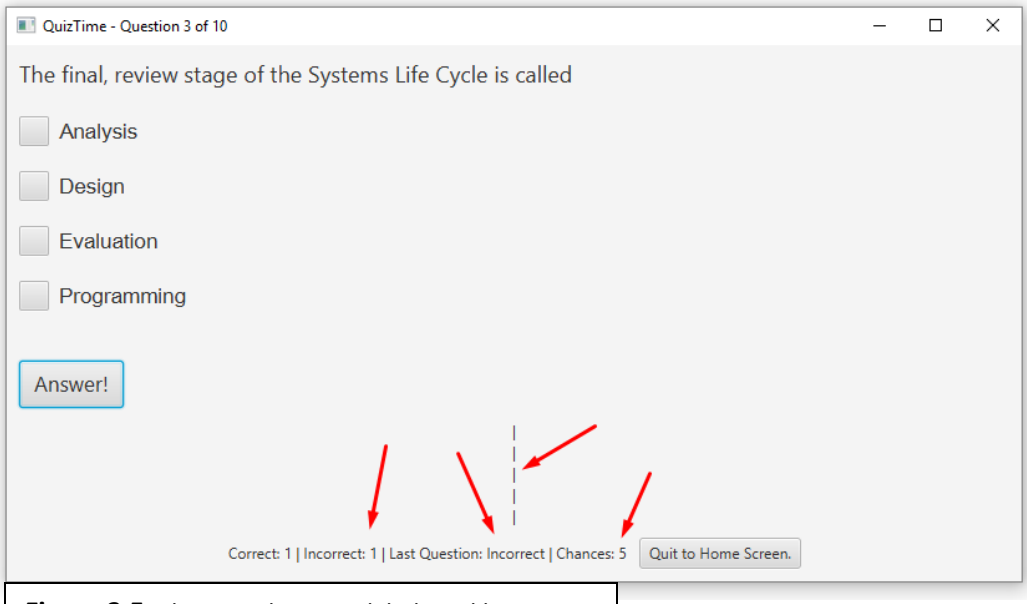


Figure 3.5: Showing change in labels and hangman.

The user selects more than one answer and presses the 'Answer!' button. This is shown in screenshot 3.6.

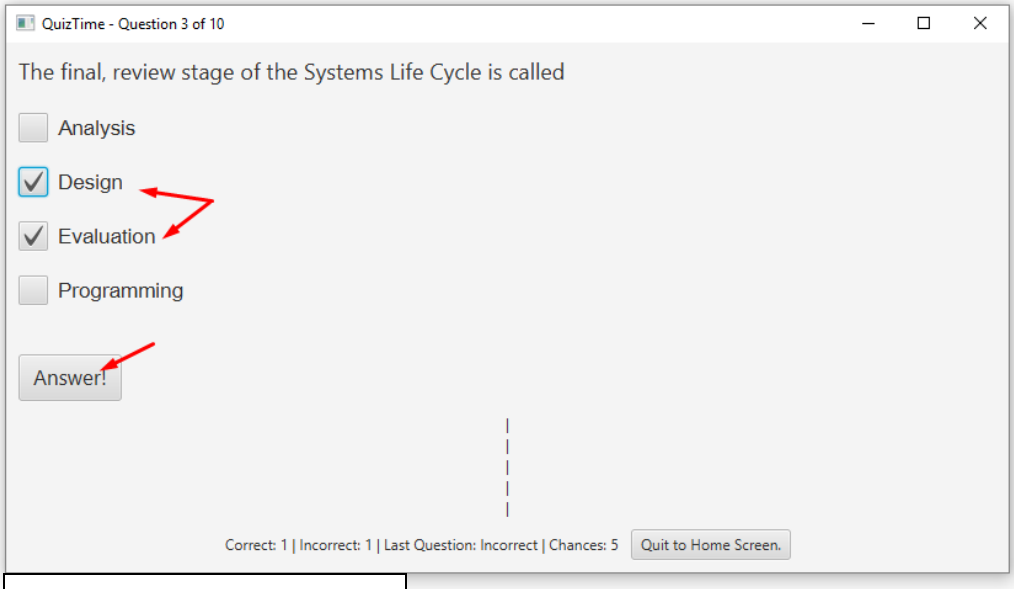


Figure 3.6: Choosing a question.

Only one answer can be chosen so the user is presented with an error screen and the event is cancelled, shown in screenshot 3.7. The method errorWindow() is being used to display the error message.

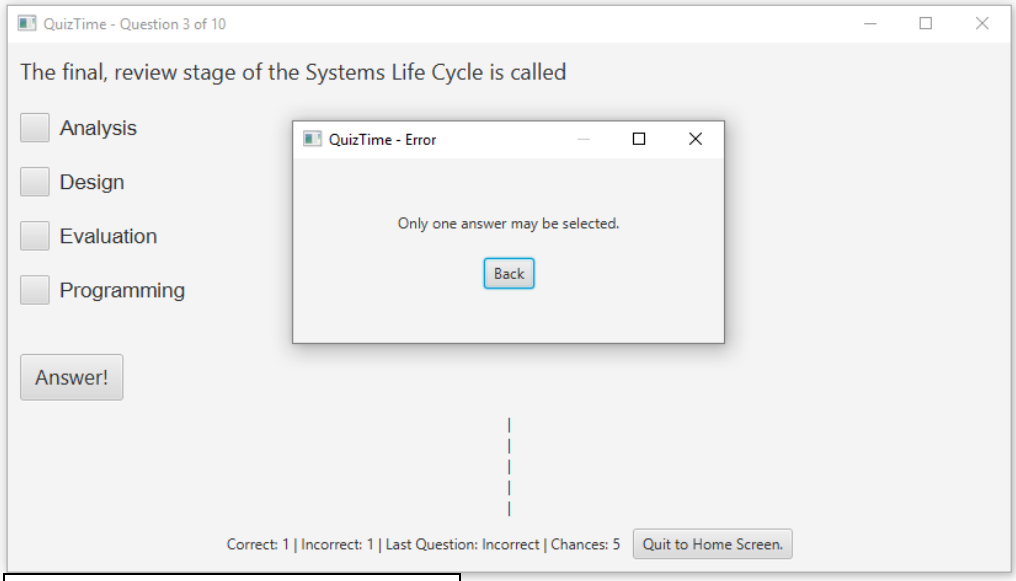


Figure 3.7: Showing error scene.

The user selects no answers and presses the 'Answer!' button shown in screenshot 3.8.

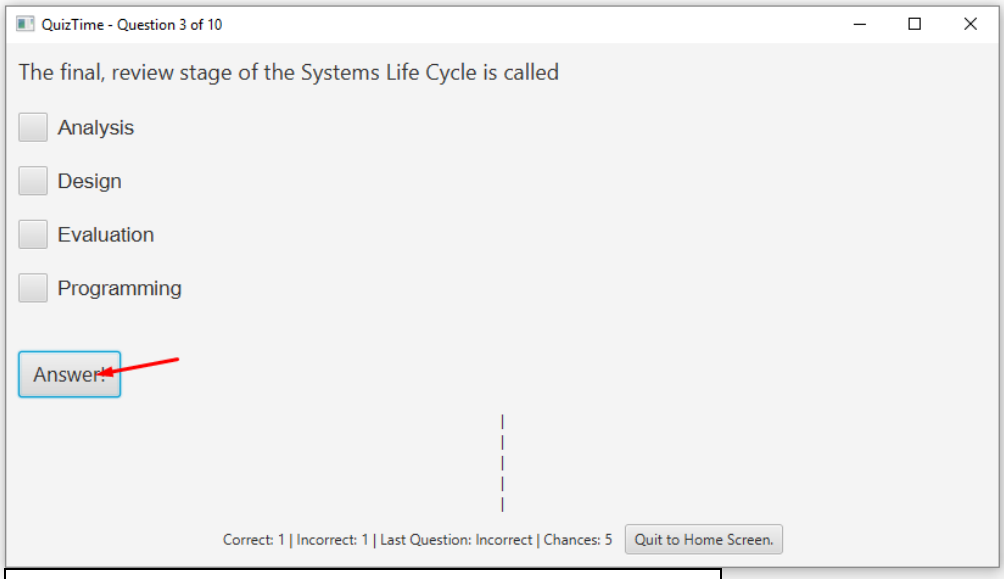


Figure 3.8: Choosing a question without selecting an answer.

At least one answer must be chosen so the user is presented with an error screen and the event is cancelled shown in screenshot 3.9. The errorWindow() message is being tested.

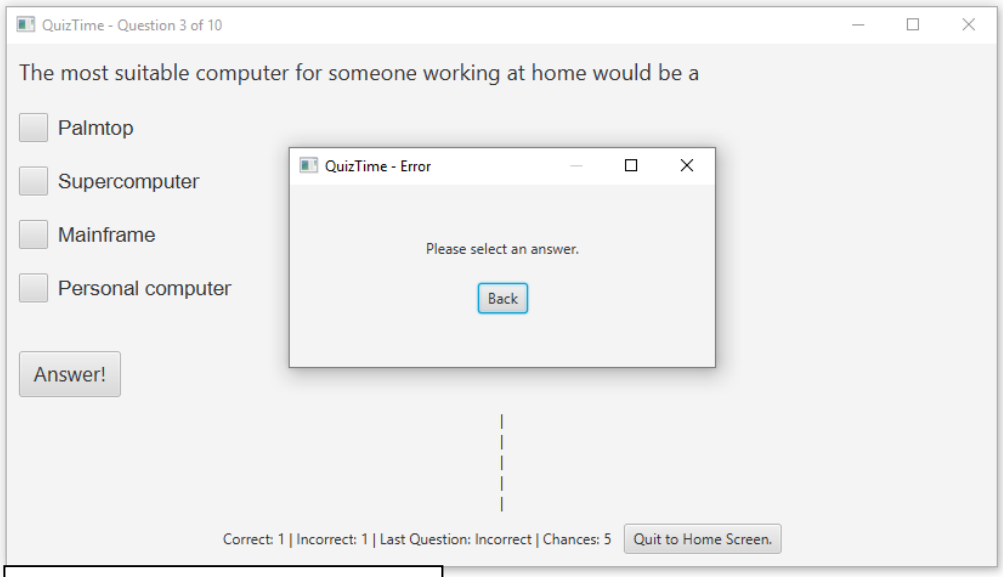


Figure 3.9: Showing error scene.

Once 10 questions have been answered, the quiz terminates using the endQuiz() method and the user is presented with a screen asking for a unique username to determine their score in the .txt file. This is shown in screenshot 3.10.

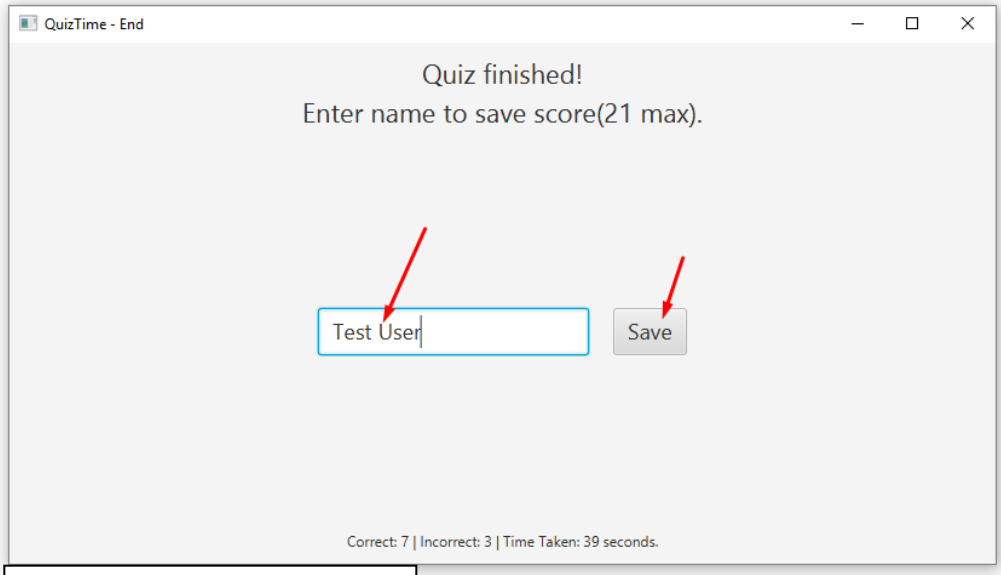


Figure 3.10: Showing end scene.

The user enters their username and selects the ‘Save’ button which writes the score to the .txt file using the method writeScore(), shown in screenshot 3.11. This scene is then closed and the user is directed to the home screen with the use of the start() method shown in the next screenshot.

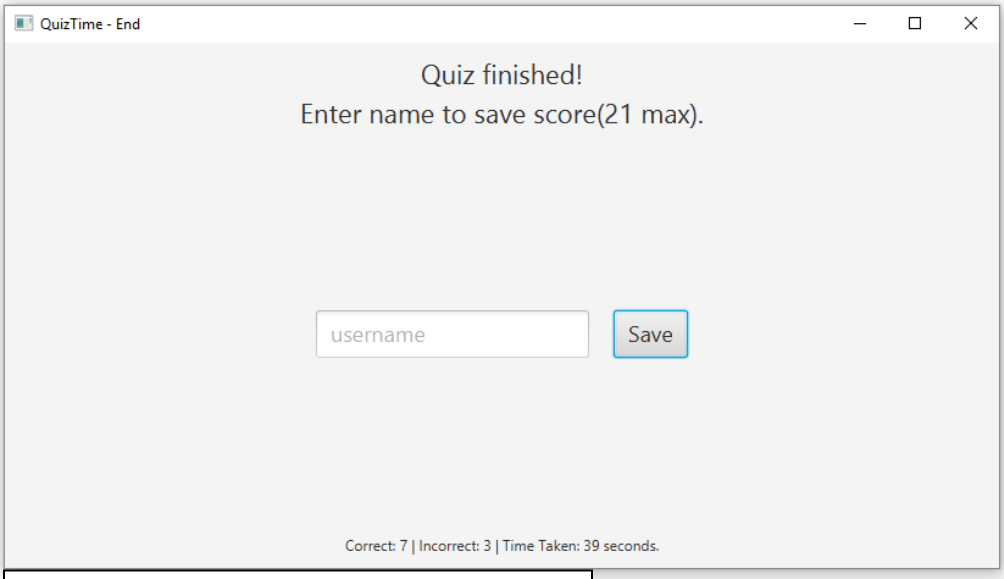


Figure 3.11: Entering username and saving score.

Back on the home screen, the ‘Top Scores’ button is pressed. Shown in screenshot 3.12.

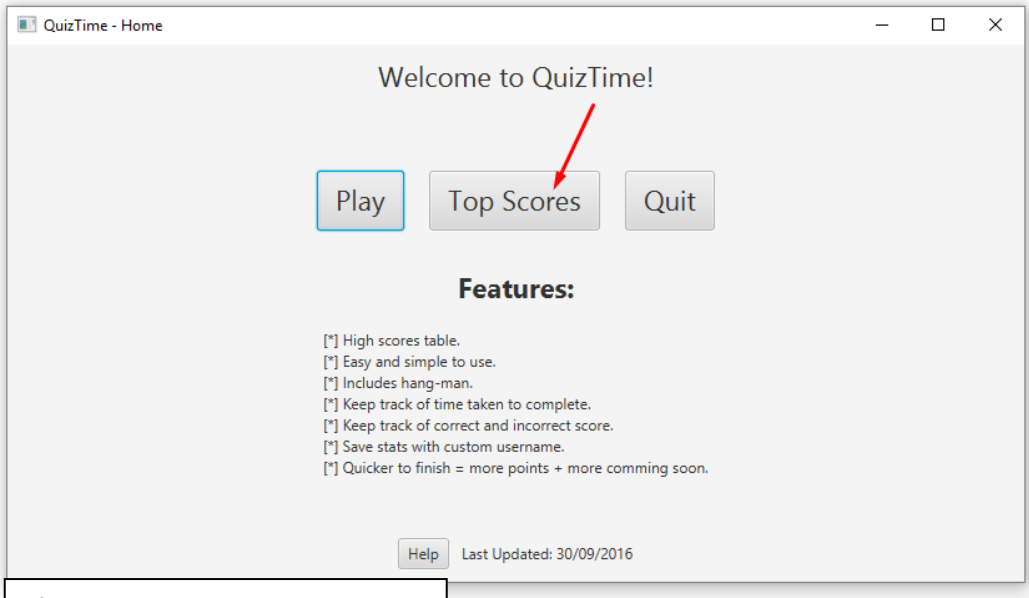


Figure 3.12: Showing home screen.

The users' score can be seen in the scores table with the username entered previously. The table is gathering the information from the quiztime_data.txt file in the programs' directory using the methods scoreDisplay() and loadScores(). Shown in screenshot 3.13.

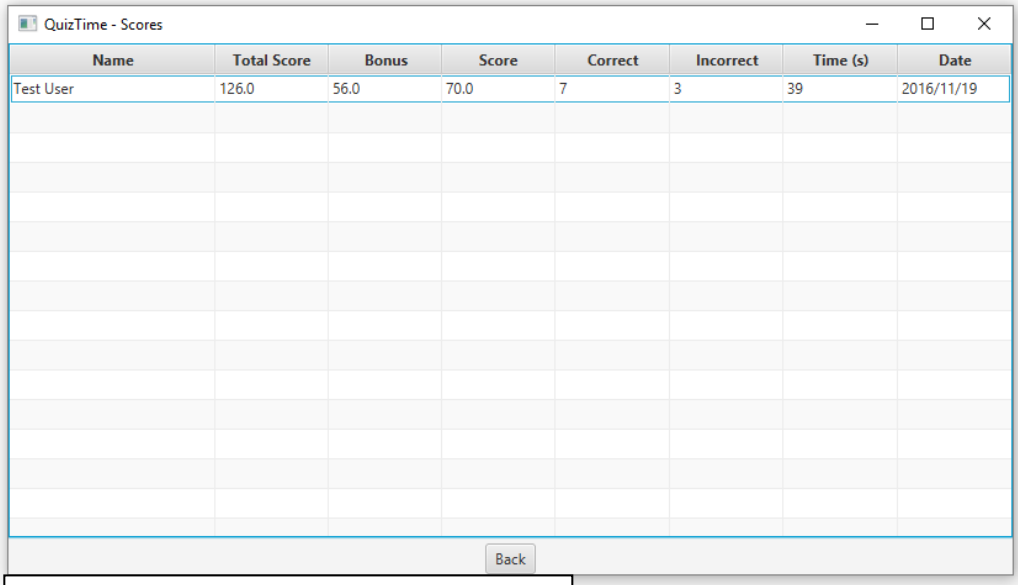


Figure 3.13: Showing users' score in table view.

3.2 – Details of test data.

This section shows the values and data used to test program along with their output.

Input	Input - Location	Input - Type	Output
One answer selected	Questions scene	Radio Check Box	Correct/incorrect + next question
More than one answer selected	Questions scene	Radio Check Box	Error window, event cancelled
No answers selected	Questions scene	Radio Check Box	Error window, event cancelled
Username	End scene	String	Saves to file, viewed in top scores
Top Scores	Scores scene	Button	Top scores scene

Section 4 – User Instructions

Table of Contents

The Aim

The aim of this quiz is to get as many questions answered correctly in order to achieve the highest score possible. One will find out that the faster the quiz is completed, the more bonus points are earned. Avoid accumulating incorrect points, if too many are obtained the quiz will terminate and no score will be saved.

Getting Started

Set up your Desktop or Laptop by installing the newest version of the **Java Runtime Environment**. This program can be run as a **Runnable JAR File** or with a **Java Editor** such as Eclipse or IntelliJ which would also require the **JDK 8**.

When the quiz starts up for the first time, no data will be saved in the save file, therefore the top scores will not function. You are required to have at least one saved record. In some cases the top scores will not work and you will be required to restart the program.

Basic Controls

This program requires a keyboard and mouse in order to be operated. The keyboard is required to type in your username allowing you to save your unique record and the mouse is required to navigate through the program using the left mouse button only.

Starting the Program

The first screen you will access contains the main menu, it is called the Home screen. Use the mouse to hover over a menu item and then press the left mouse button to select it.

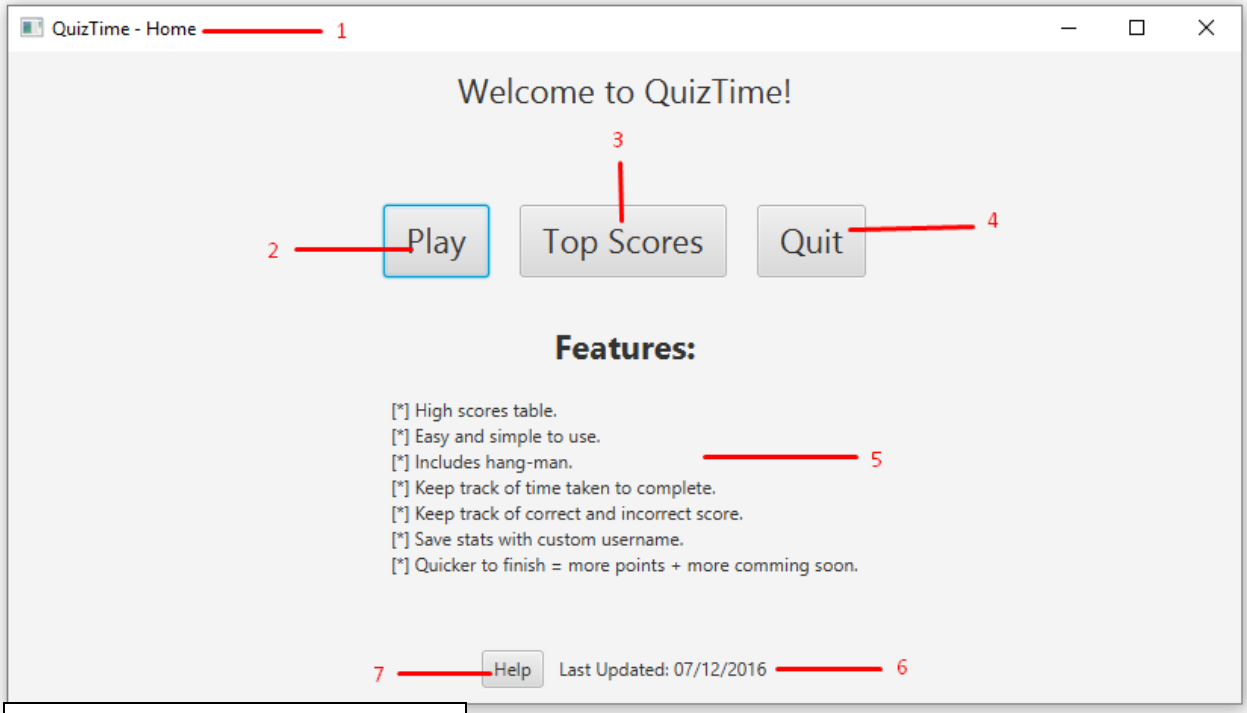


Figure 4.1: Displaying Home Screen.

- **1: Screen Name** – Title of the current screen that is being viewed.
- **2: Play Button** – This button starts the quiz on a separate scene.
- **3: Top Scores Button** – This button opens up a table with all the information of users who have played the game previously, this contains the; name, total score, bonus, score, correct, incorrect, time and date fields.
- **4: Quit Button** – Quit the program.
- **5: Features List** – A list of a few features the program has to offer.
- **6: Last Update** – Date the program was last updated.
- **7: Help Button** – This button provides some troubleshooting information.

Starting the Quiz

Figure 4.2: Displaying question scenario.

- **1: Question Number** – Current question being viewed.
- **2: Question Label** – This is the question that the user is required to answer.
- **3: Multiple Choice Answers** – Select one of these answers by checking the empty box. Only one box can be selected and at least one box may be selected to proceed to the next question.
- **4: Answer Button** – Select this button when you want to proceed to the next question.
- **5: Hangman** – Hangman progresses as questions are answered incorrectly.
- **6: Quit to Home Screen Button** – Exit to Home screen. Data will not be saved.
- **7: Correct Label** – Contains the amount of correct answers.
- **7: Incorrect Label** – Contains the amount of incorrect answers.
- **7: Last Question Label** – Displays whether the last question answered was correct or incorrect.
- **7: Chances Label** – Chances left until quiz terminates.

Saving the Score

Once the quiz is completed, the user will be asked to enter their desired username as to associate it to their score. Once the user has entered their username they must select the Save Button and the score will be stored safely in a file.

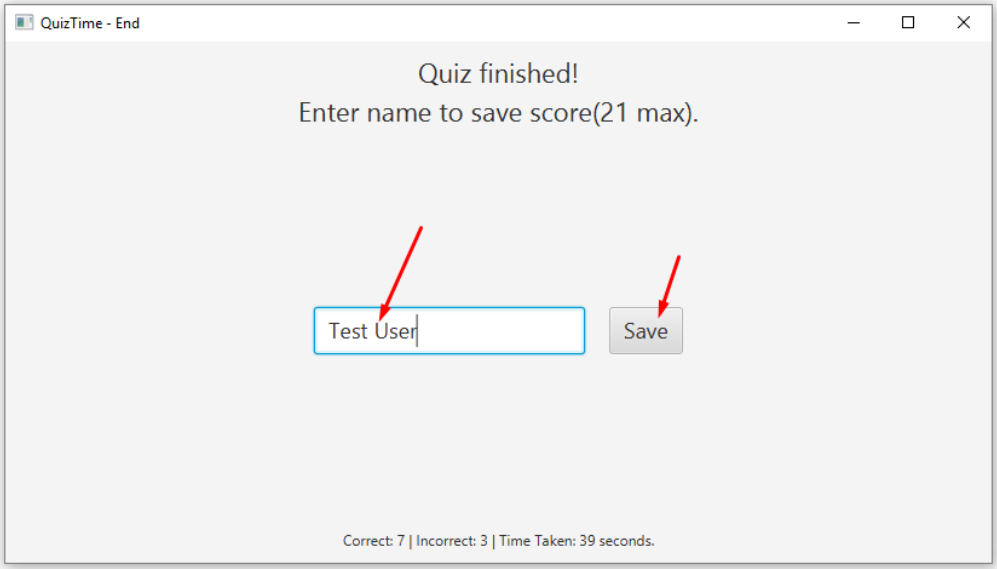


Figure 4.3: Displaying saving score scenario.

Viewing the Scores

To view the saved scores, navigate through the main menu and located at the Home screen and select the Top Scores button. There you will find all the scores saved in the programs' data file.

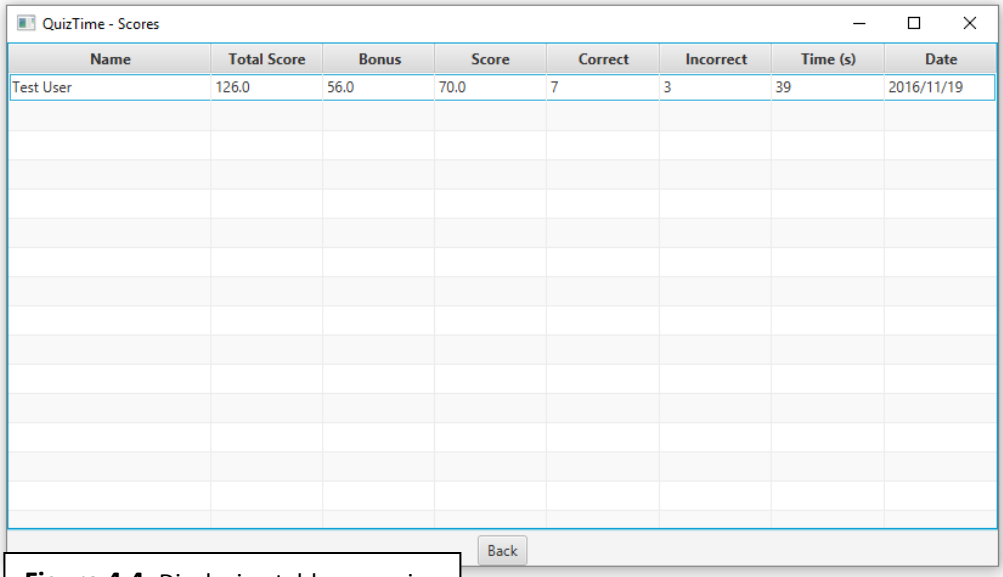


Figure 4.4: Displaying table scenario.

Section 5 – Improvements and conclusion

Improvements

1. On the whole, the program is easy and straightforward to use. It is immediately clear that the program is simply a quiz and it is very easy to get started as all one has to do is select the play button. There aren't too many icons and messages on the display which minimizes usability issues and worries for the user. It is also clear that the display isn't very striking at all. This is one of the main setbacks of the program which could easily be improved on in the future.

2. The program wasn't designed to be used over long periods of time but rather a few tries on the quiz and that's all. Not having an aim besides trying to obtain a good score and testing ones knowledge, doesn't render any motive to continue playing the quiz. Implementing a progress system which includes levels containing more challenging questions and possibly even different subjects, could be added, creating a better reason to play the quiz for longer periods of time and possibly attracting a larger audience.

3. The high scores data is currently limited to local files. Connecting the program to a remote database can easily add some extra fun to the program, allowing users from around the world to compete between each other.

Conclusion

I have had a great experience working on this project as it has also helped on improving my programming knowledge. Quiztime has surely worked out properly and has helped Milton High School achieve higher grades in comparison to previous years. It definitely matches the experience I was aiming towards to provide at this stage; an easy, user friendly quiz aimed at the younger generation.
