Programming Coursework Report

Quinn Stevens

June 2017

Contents

Part I

Quiz Game

Chapter 1

Testing Plan

Environment	**Action**	**Expected Result**	**Actual Result**	**Proof**
Program Closed	Run Program	Main Menu form opens	Main Menu form opened	see screenshot 2.1
Main Menu	Click 'Quit' Button	Program closes	Program closed	see screenshot 2.2
Main Menu	Click 'Easy' Button	Asks first easy question	Asked first easy question	see screenshot 2.3
Quiz	Click radio button	Radio button selected	Radio button selected	see screenshot 2.4
Quiz	Click Answer while radio button selected	Ask next question	Asked next question	see screenshot 2.5
Quiz	Click Answer while radio button not selected	Ask next question	Asked next question	see screenshot 2.6
Quiz	Click Skip	Ask Next Question	Asked next question	see screenshot 2.7
Quiz	Finish Quiz	Ask if you want to repeat skipped questions	Asked if you want to repeat skipped questions	see screenshot 2.8
Skipped Questions Dialogue	Click 'Yes'	Re-ask skipped questions	Re-asked skipped questions	see screenshot 2.9

Quiz	Finish Quiz with wrong answers (No skipped questions or click 'No' on skipped questions dialogue or finished retrying skipped questions)	Show end of quiz report with corrections	Showed end of quiz report with corrections	see screenshot 2.10
Quiz	Finish Quiz with no wrong answers	Show end of quiz report without corrections	Showed end of quiz report without corrections	see screenshot 2.11
Main Menu	Click 'Hard' Button	Asks first hard question	Asked first hard question	see screenshot 2.12

Chapter 2

Testing Screenshots

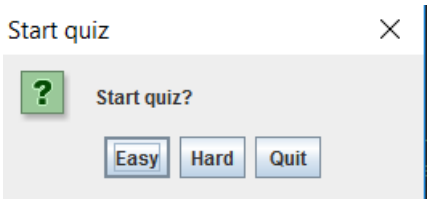

Figure 2.1: Screenshot 01

```
Process finished with exit code 0
```

Figure 2.2: Screenshot 02

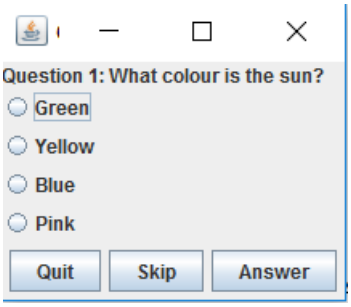

Figure 2.3: Screenshot 03

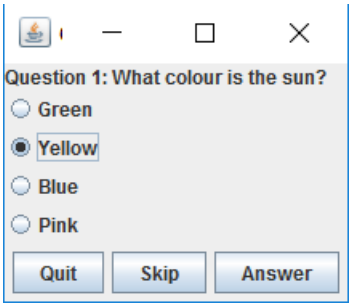

Figure 2.4: Screenshot 04

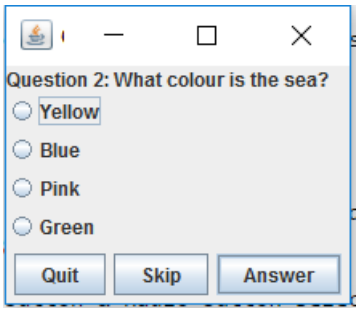

Figure 2.5: Screenshot 05

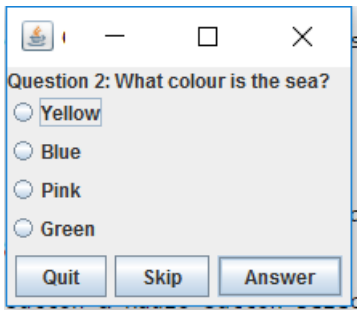

Figure 2.6: Screenshot 06

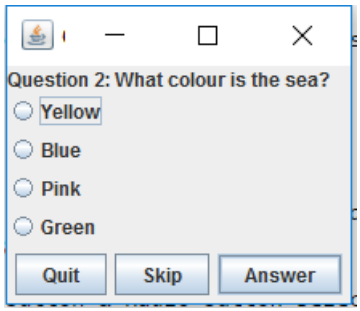

Figure 2.7: Screenshot 07

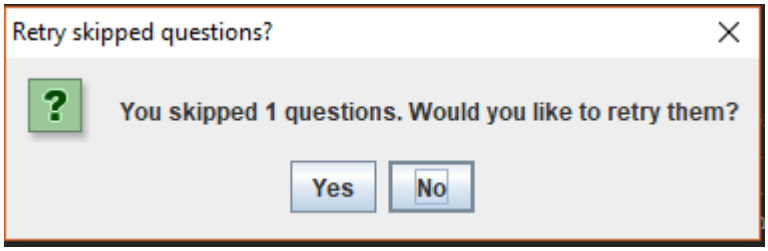

Figure 2.8: Screenshot 08

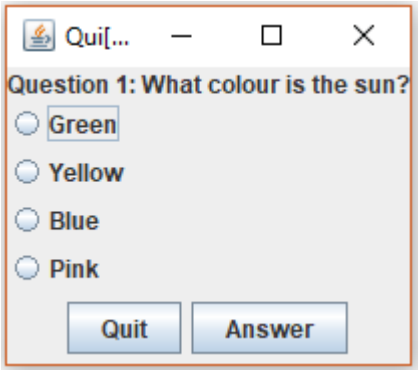

Figure 2.9: Screenshot 09

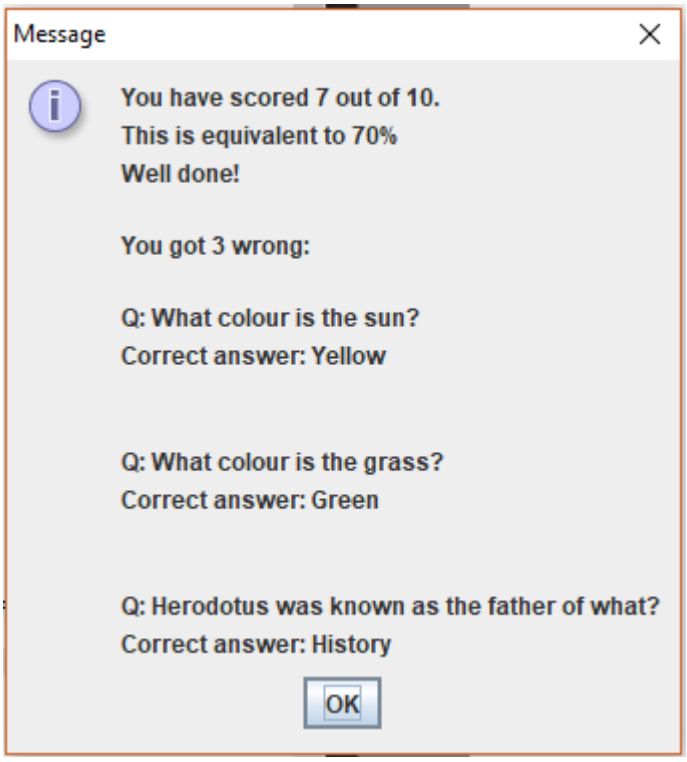

Figure 2.10: Screenshot 10

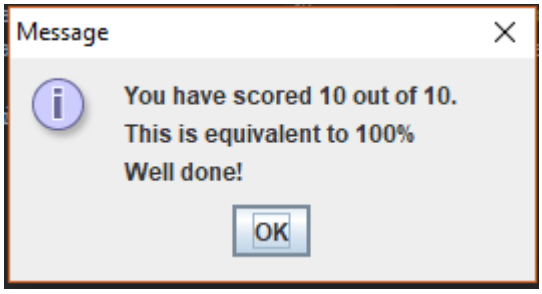

Figure 2.11: Screenshot 11

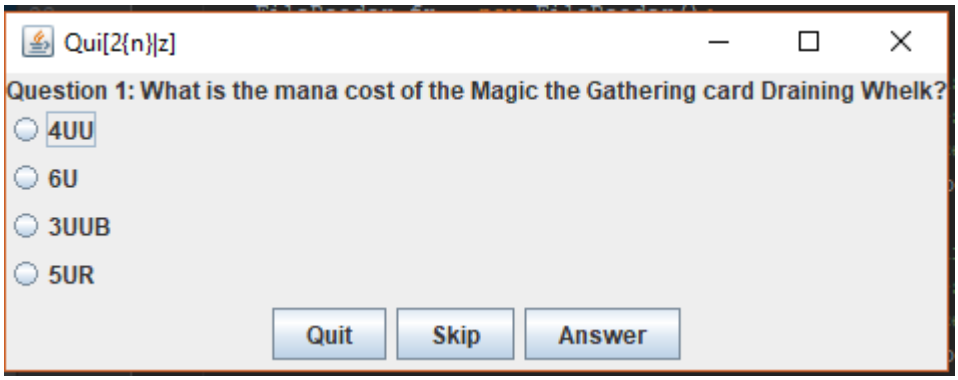

Figure 2.12: Screenshot 12

Chapter 3

Critical Analysis

3.1 Strengths

- Allows users to choose from two different difficulties
- Gives feedback on how the user did
- Allows users to save questions for later

3.2 Weaknesses

- Scoring is binary - it doesn't give the user any second chances
- Doesn't allow users to skip questions multiple times
- Doesn't stop the user if they don't answer a question

3.3 Suggested Enhancements

- Allow user to skip a question multiple times
- Stop the user from continuing with the quiz if they don't select an answer
- Allow the user to retry questions they got wrong, but with a reduced points reward

3.4 Adherence to Specification

Requirement	**Fulfilled?**
The system should be dialogue-based	Yes

Good interface design and programming principles should be applied throughout	Yes
Upon start of the program, the first question with its corresponding answers should be displayed	Yes
The question number should be displayed, and the options labelled appropriately	Yes
When the user clicks the OK button, the second question with its corresponding answers should be displayed	Yes
The above step should be repeated until all 10 questions have been displayed	Yes
When all questions are answered the program should display the user's score along with an appropriate message	Yes
Questions and answers should NOT be hard-coded and must be imported from external files	Yes
When taking the test, if the user tries to view the next question without providing an answer to the current question, a message should be displayed to alert them	No
In returning to a skipped question, an answer must be provided (A question cannot be skipped twice)	Yes (although I do not agree that this improves the test program)
The system should display the questions that have been incorrectly answered. Repeating the failed questions should not change the user's final score.	Yes (again, I believe that changing this would improve the program)
The user should have the choice of the difficulty level of the test: e.g. setting up two or more files each containing a set of questions at different levels of difficulty	Yes

Chapter 4

Source Code

```
import javax.swing.*;

/**
 * Created by Quinn Stevens on 31/05/2017.
 */
public class Quiz {
    static int score = 0;

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

    public static void runQuiz() {
        String[] questions;
        String[][] options;
        String[] correctAnswers;

        Object[] difficulties = {"Easy", "Hard", "Quit"};

        int difficulty =
         ↪ JOptionPane.showOptionDialog(null, "Start
         ↪ quiz?", "Start quiz",
         ↪ JOptionPane.YES_NO_OPTION,
         ↪ JOptionPane.QUESTION_MESSAGE, null,
         ↪ difficulties, difficulties[0]);

        FileReader fr = new FileReader();
        if (difficulty == 0) {
            questions =
             ↪ fr.getArray("resources/easy/questions.txt");
```

```
            options =
             ↪ fr.getAnswerArray("resources/easy/options.txt");
            correctAnswers =
             ↪ fr.getArray("resources/easy/answers.txt");
            QuestionMaster qm = new
             ↪ QuestionMaster(questions, options,
             ↪ correctAnswers);
        } else if (difficulty == 1){
            questions =
             ↪ fr.getArray("resources/hard/questions.txt");
            options =
             ↪ fr.getAnswerArray("resources/hard/options.txt");
            correctAnswers =
             ↪ fr.getArray("resources/hard/answers.txt");
            QuestionMaster qm = new
             ↪ QuestionMaster(questions, options,
             ↪ correctAnswers);
        } else {
            System.exit(0);
        }
    }
}
```

Listing 1: Quiz.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Objects;
import java.util.Stack;

/**
 * Created by Quinn Stevens on 31/05/2017.
 */
public class QuestionMaster implements ActionListener {
    JFrame frame;
    int questionNum= 0;
    int score = 0;
    String currentAnswer;

    String[] questions;
    String[][] options;
    String[] answers;

    boolean skipAllowed;
```

```
Stack skipped = new Stack();
int incorrectAnswers = 0;

String report = "";

public QuestionMaster (String[] questions, String[][]
 ↪  options, String[] answers) {
    this.questions = questions;
    this.options = options;
    this.answers = answers;
    this.questionNum = 0;
    skipAllowed = true;
    askQuestion(questions[questionNum],
     ↪  options[questionNum], answers[questionNum]);
}

private void askQuestion(String question, String[]
 ↪  options, String answer) {
    frame = new JFrame("Qui[2{n}|z]");

     ↪  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocation(600, 300);
    frame.setLayout(new BorderLayout());

    // Panel for question
    JLabel qLabel = new JLabel("Question " +
     ↪  (questionNum+1) + ": " + question);

    // Panel for answer options
    JPanel buttonPanel = new JPanel(new GridLayout(0,
     ↪  1));
    ButtonGroup buttonGroup = new ButtonGroup();
    JRadioButton[] answerButtons = new
     ↪  JRadioButton[options.length];
    for (int i = 0; i < answerButtons.length; i++) {
        answerButtons[i] = new
         ↪  JRadioButton(options[i]);
        answerButtons[i].addActionListener(this);
        buttonGroup.add(answerButtons[i]);
        buttonPanel.add(answerButtons[i]);
    }

    // Panel for controls
    JPanel controlPanel = new JPanel(new
     ↪  FlowLayout());
    // Create quit button
```

```
        JButton quitButton = new JButton("Quit");
        quitButton.setActionCommand("quit");
        quitButton.addActionListener(this);
        controlPanel.add(quitButton);

        if (skipAllowed) {
            // Create skip button
            JButton skipButton = new JButton("Skip");
            skipButton.setActionCommand("skip");
            skipButton.addActionListener(this);
            controlPanel.add(skipButton);
        }

        // Create submit button
        JButton answerButton = new JButton("Answer");
        answerButton.setActionCommand("answer");
        answerButton.addActionListener(this);
        controlPanel.add(answerButton);

        frame.add(qLabel, BorderLayout.NORTH);
        frame.add(buttonPanel, BorderLayout.CENTER);
        frame.add(controlPanel, BorderLayout.SOUTH);
        frame.pack();
        frame.setVisible(true);
    }

    private void nextQuestion() {
        frame.dispose();
        questionNum++;
        if (questionNum < questions.length) {
            askQuestion(questions[questionNum],
             ↪  options[questionNum],
             ↪  answers[questionNum]);
        } else if (!skipped.empty()) {
            Object[] skipOptions = {"Yes", "No"};
            int retry = JOptionPane.showOptionDialog(
                    null,
                    "You skipped " + skipped.size() + "
                     ↪  questions. Would you like to
                     ↪  retry them?",
                    "Retry skipped questions?",
                    JOptionPane.YES_NO_OPTION,
                    JOptionPane.QUESTION_MESSAGE,
                    null,
                    skipOptions,
                    skipOptions[1]);
```

```
            if (retry == 0) {
                skipAllowed = false;
                retrySkipped();
            } else {
                endQuiz();
            }
        } else {
            endQuiz();
        }
    }

    private void updateReport() {
        incorrectAnswers += 1;
        report += "\n\nQ: " + questions[questionNum] +
         ↪ "\nCorrect answer: " + answers[questionNum] +
         ↪ "\n";
    }

    private void endQuiz() {
        if (incorrectAnswers != 0) {
            report = "\n\nYou got " + incorrectAnswers +
             ↪ " wrong:" + report;
        }

        JOptionPane.showMessageDialog(null, "You have
         ↪ scored " + score + " out of 10.\n" +
                "This is equivalent to " +
                 ↪ Math.round(score/10.0*100) + "%\nWell
                 ↪ done!" + report);
    }

    private void checkAnswer() {
        if (Objects.equals(currentAnswer,
         ↪ answers[questionNum])) {
            score ++;
        } else {
            updateReport();
        }
    }

    private void resetQuiz() {
        frame.dispose();

        Quiz.runQuiz();
    }
```

```
    private void retrySkipped() {
        int stackSize = skipped.size();
        String[] skippedQs = new String[stackSize];
        String[][] skippedOpts = new
         ↪ String[stackSize][4];
        String[] skippedAnsw = new String[stackSize];

        for (int i = stackSize-1; i >= 0; i--) {
            int qNum = (int) skipped.pop();
            skippedQs[i] = questions[qNum];
            skippedOpts[i] = options[qNum];
            skippedAnsw[i] = answers[qNum];
        }

        questions = skippedQs;
        options = skippedOpts;
        answers = skippedAnsw;
        questionNum = -1;
        nextQuestion();
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        String command = event.getActionCommand();

        if (command == "answer" && currentAnswer != null)
         ↪ {
            checkAnswer();
            nextQuestion();
        } else if (command == "answer" && currentAnswer
         ↪ == null) {
            JOptionPane.showMessageDialog(null, "You
             ↪ haven't selected an answer!");
        } else if (command == "quit") {
            resetQuiz();
        } else if (command == "skip") {
            skipped.push(questionNum);
            nextQuestion();
        } else {
            currentAnswer = command;
        }
    }
}
```

Listing 2: QuestionMaster.java

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import java.util.Stack;

/**
 * Created by Quinn Stevens on 31/05/2017.
 */
public class FileReader {
    public String[][] getAnswerArray(String filename) {

        String[] inputArray = getArray(filename);
        String[][] outputArray = new
         ↪ String[inputArray.length][4];
        String[] splitText;
        for (int i = 0; i < inputArray.length; i++) {
            splitText = inputArray[i].split(", ");

            for (int j = 0; j < splitText.length; j++) {
                outputArray[i][j] = splitText[j];
            }
        }

        return outputArray;
    }

    public String[] getArray(String filename) {
        String[] text = null;
        try {
            text = readFile(filename);
        } catch (IOException e) {
            e.printStackTrace();
        }

        return text;
    }

    public static String[] readFile(String filename)
     ↪ throws IOException {
        String fileLine;
        Stack textStack = new Stack();

        File questionFile = new File(filename);
        Scanner fileScan = new Scanner(questionFile);

        while (fileScan.hasNext()) {
```

```
            fileLine = fileScan.nextLine();
            textStack.push(fileLine);
        }

        String[] textArray = new
         ↪ String[textStack.size()];

        for (int i = textArray.length -1; i >=0; i--) {
            textArray[i] = (String) textStack.pop();
        }

        return textArray;
    }
}
```

Listing 3: FileReader.java

Appendix I: Banking Program

4.1 Source Code

4.2 Output