Presented by: C Griffin

t00175569

OOP2-Project 2014

Networking quiz

December 8, 2014

# OOP2-Project 2014

## Project Summary

The function of this project is to allow a user to take a multiple choice quiz with a graphical user interface.

The program will allow a user via a JMenu to take different size quizzes and also has administration options of adding or deleting questions from the quiz.

When a user invokes the quiz function, the program will check the users answers against an answer key and will keep track of their overall score in the quiz and display that result on the JFrame when the quiz is finished.

The GUI consists of a main JFrame with several JPanels with JLabels and JTextAreas. JOptionPanes are also employed to display questions and intake user answers.

This Java project consists of 3 new classes. Along with a GUI driver, there are two instantiable classes, Question class and Answer class.

Question class uses an instance of type Answer. Specifically Question class has an attribute of an ArrayList of type Answer.

The GuiQuizDriver class inherits from both the Java JFrame class and Java ActionListener class.

Persistent memory is used to store and retrieve the quiz questions.

## Program code

### question class

[Describe the process that your team will follow to document and approve changes to the project. If your team uses a change control document, identify how and when team members should fill it out.]

//Question.java

//This is the Question class that will be used by the driver

**import** java.io.\*;

**import** java.util.\*;

**public** **class** Question **implements** Serializable

{

//attributes

**private** String question;

**private** ArrayList<Answer> answers;

**private** String answerKey="";

//constructor methods

**public** Question()

{

setQuestion(question);

setAns(answers);

setKey(answerKey);

}

**public** Question(String question,ArrayList<Answer> answers)

{

setQuestion(question);

setAns(answers);

setKey(answerKey);

}

//mutator methods

**public** **void** setQuestion(String question)

{

**this**.question=question;

}

**public** **void** setAns(ArrayList answers)

{

**this**.answers=answers;

}

**public** **void** setKey(String answerKey)

{

**this**.answerKey=answerKey;

}

//accessor methods

**public** String getQuestion()

{

**return** question;

}

**public** ArrayList getAns()

{

**return** answers;

}

**public** String getKey()

{

**char** letter='@';

**for**(Answer a: answers)

{

letter++;

**if**(a.getIsCorrect()==**true**)

answerKey+=letter;

}

**return** answerKey;

}

//toString Method- no t/f

**public** String toString()

{

**char** letter='@';

//adds question to output

String output = "Question: " + getQuestion()+"?\n\n";

//loop to add answers to string output

**for**(Answer a: answers)

{

letter++;

output+= (**char**)(letter)+ ". " + a.toString() + "\n";

}//end loop

**return** output;

}//end toString

//toStringFull method with t/f

**public** String toStringFull()

{

**char** letter='@';

//adds question to output

String output = "Question: " + getQuestion()+"?\n\n";

//loop to add answers to string output

**for**(Answer a: answers)

{

letter++;

output+= (**char**)(letter)+ ". " + a.toStringTF() + "\n";

}//end loop

**return** output;

}//end toStringFull

}

### Answer Class

**import** java.io.\*;

**public** **class** Answer **implements** Serializable

{

//attributes

**private** String answer;

**private** **boolean** isCorrect;

//constructor

**public** Answer()

{

**this**("Blank",**false**);

}

**public** Answer(String answer, Boolean isCorrect)

{

setAnswer(answer);

setIsCorrect(isCorrect);

}

//mutators

**public** **void** setAnswer(String answer)

{

**this**.answer=answer;

}

**public** **void** setIsCorrect(**boolean** isCorrect)

{

**this**.isCorrect=isCorrect;

}

//accessors

**public** String getAnswer()

{

**return** answer;

}

**public** **boolean** getIsCorrect()

{

**return** isCorrect;

}

//to String with t/f

**public** String toStringTF()

{

String output = getAnswer()+"\t\t"+getIsCorrect();

**return** output;

}

//to String with no t/f

**public** String toString()

{

String output = getAnswer();

**return** output;

}

}

### GuiQuizDriver Class

//this is the gui driver for the quiz

**import** java.awt.BorderLayout;

**import** java.awt.CardLayout;

**import** java.awt.Color;

**import** java.awt.Container;

**import** java.awt.Font;

**import** java.awt.GridLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.util.ArrayList;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuBar;

**import** javax.swing.JMenuItem;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JTextField;

**public** **class** GuiQuizDriver **extends** JFrame **implements** ActionListener {

// global attributes needed for methods and driver

JPanel kingPanel, blank, holdAllPanel, questionPanel, optionsPanel, aPanel,

bPanel, cPanel, dPanel, ePanel, fPanel;

JMenu playMenu, adminMenu;

CardLayout c1 = **new** CardLayout();

GridLayout optionsLayout = **new** GridLayout(6, 3);

Container cPane;

**int** i = 0;

JLabel questionLabel, aLabel, bLabel, cLabel, dLabel, eLabel, fLabel,

blankLabel;

JTextField questionField, aField, bField, cField, dField, eField, fField,

aTf, bTf, cTf, dTf, eTf, fTf;

**boolean** isCorrect, addMore = **true**;

String answer = "start", question, isCorrectString, addMoreString, choice,

answerKey, resultString = "Welcome to the QUIZ";

ArrayList<Answer> answers;

Answer ans;

Question q1;

ArrayList<Question> questions = **new** ArrayList<Question>();

// main start

**public** **static** **void** main(String[] args) {

GuiQuizDriver runBox1 = **new** GuiQuizDriver();

runBox1.setVisible(**true**);

runBox1.pack();

}// end main

// start of driver constructor

**public** GuiQuizDriver() {

setTitle("Networking Quiz");

setLocation(100, 200);

setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);

cPane = getContentPane();

// methods to add menu items

/\*

\* Title:JMenuFrame.javaAuthor:John Walshdate:06/11/2014 15:37

\* Availability: X-Drive\John W\Sample Programs\Units13\_15Comment: I

\* used this code create my Menu items

\*/

createPlayMenu();

createAdminMenu();

JMenuBar menuBar = **new** JMenuBar();

setJMenuBar(menuBar);

menuBar.add(playMenu);

menuBar.add(adminMenu);

// my panels

kingPanel = **new** JPanel();

kingPanel.setLayout(c1);

questionPanel = **new** JPanel();

questionLabel = **new** JLabel("Please enter the question");

questionField = **new** JTextField(**null**, 50);

questionPanel.add(questionLabel);

questionPanel.add(questionField);

aLabel = **new** JLabel("Option A");

aField = **new** JTextField(**null**, 20);

aTf = **new** JTextField("True/False", 10);

bLabel = **new** JLabel("Option B");

bField = **new** JTextField(**null**, 20);

bTf = **new** JTextField("True/False", 10);

cLabel = **new** JLabel("Option C");

cField = **new** JTextField(**null**, 20);

cTf = **new** JTextField("True/False", 10);

dLabel = **new** JLabel("Option D");

dField = **new** JTextField(**null**, 20);

dTf = **new** JTextField("True/False", 10);

eLabel = **new** JLabel("Option E");

eField = **new** JTextField(**null**, 20);

eTf = **new** JTextField("True/False", 10);

fLabel = **new** JLabel("Option F");

fField = **new** JTextField(**null**, 20);

fTf = **new** JTextField("True/False", 10);

optionsPanel = **new** JPanel();

optionsPanel.setLayout(optionsLayout);

optionsPanel.add(aLabel);

optionsPanel.add(aField);

optionsPanel.add(aTf);

optionsPanel.add(bLabel);

optionsPanel.add(bField);

optionsPanel.add(bTf);

optionsPanel.add(cLabel);

optionsPanel.add(cField);

optionsPanel.add(cTf);

optionsPanel.add(dLabel);

optionsPanel.add(dField);

optionsPanel.add(dTf);

optionsPanel.add(eLabel);

optionsPanel.add(eField);

optionsPanel.add(eTf);

optionsPanel.add(fLabel);

optionsPanel.add(fField);

optionsPanel.add(fTf);

holdAllPanel = **new** JPanel();

JButton submit = **new** JButton("Submit Question");

submit.addActionListener(**this**);

holdAllPanel.add(questionPanel);

holdAllPanel.add(optionsPanel);

holdAllPanel.add(submit);

holdAllPanel.setBackground(Color.***BLUE***);

blank = **new** JPanel();

blank.setBackground(Color.***GREEN***);

blankLabel = **new** JLabel(resultString);

blankLabel.setFont(**new** Font("Serif", Font.***PLAIN***, 54));

blank.add(blankLabel);

/\*

\* Title:How to change the size of the font of a JLabel to take the

\* maximum sizeAuthor:Asaf DavidDate:Apr 26'10 at 16:33

\* Availability:http:

\* //stackoverflow.com/questions/2715118/how-to-change-

\* the-size-of-the-font-of-a-jlabel-to-take-the-maximum-sizeComment: I

\* used this bit of code in order to resize my JLabel on the landing

\* panel(blankPanel)

\*/

kingPanel.add(blank, "2");

kingPanel.add(holdAllPanel, "1");

/\*

\* Title: CardLayoutEX.javaAuthor: John WalshDate:02/12/2014 09:25

\* Availability: X-Drive\John W\Sample Programs\Multiple PanelsComment:

\* I used this to deal with how to display panels at different times

\* during runtime

\*/

cPane.add(kingPanel, BorderLayout.***CENTER***);

**this**.add(kingPanel);

/\*

\* Title: BicyleFrame4.javaAuthor:John Walshdate:05/12/2014 12:41

\* Availability: X-Drive\John W\Sample Programs\Units16\_18

Comment: I used this code to model my own input/output streams and to save dat files

\*/

**try** {

System.***out***.println("made into addNewQuestion() try1");

ObjectInputStream is;

is = **new** ObjectInputStream(**new** FileInputStream("allQuestion.dat"));

questions = (ArrayList<Question>) is.readObject();

is.close();

} **catch** (Exception e) {

System.***out***.println("made into addNewQuestion() catch1");

// this is left empty to prevent exceptions being displayed in

// output

}

}// end constructor

// my event handler method for menu items

/\*

\* Title:JMenuFrame.javaAuthor:John Walshdate:06/11/2014 15:37Availability:

\* X-Drive\John W\Sample Programs\Units13\_15Comment: I used this code to

\* model my own event handling

\*/

**public** **void** actionPerformed(ActionEvent event) {

String menuName;

menuName = event.getActionCommand();

**if** (menuName.equals("10 Questions")) {

**try** {

take10Quiz();

} **catch** (IOException e) {

e.printStackTrace();

}

} **else** **if** (menuName.equals("25 Questions")) {

**try** {

take25Quiz();

} **catch** (IOException e) {

e.printStackTrace();

}

} **else** **if** (menuName.equals("Every Question")) {

**try** {

takeQuiz();

} **catch** (IOException e) {

e.printStackTrace();

}

} **else** **if** (menuName.equals("Add Question")) {

resultString = "Welcome to the QUIZ";

blankLabel.setText(resultString);

newQuestionForm();

} **else** **if** (menuName.equals("Edit Question")) {

**try** {

deleteQuestion();

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

save();

} **catch** (IOException e) {

e.printStackTrace();

}

resultString = "Welcome to the QUIZ";

blankLabel.setText(resultString);

} **else** **if** (menuName.equals("Submit Question")) {

**try** {

addNewQuestion();

} // end try

**catch** (IOException f) {

System.***out***.print("Not able to save the file:\n"

+ "Check the console printout for clues to why ");

f.printStackTrace();

}// end catch

} **else** {

System.*exit*(0);

}// end of if else statements

}// end action handler

// method to create PlayMenu items

**private** **void** createPlayMenu() {

JMenuItem item;

playMenu = **new** JMenu("Play");

item = **new** JMenuItem("10 Questions");

item.addActionListener(**this**);

playMenu.add(item);

item = **new** JMenuItem("25 Questions");

item.addActionListener(**this**);

playMenu.add(item);

item = **new** JMenuItem("Every Question");

item.addActionListener(**this**);

playMenu.add(item);

playMenu.addSeparator();

item = **new** JMenuItem("Quit");

item.addActionListener(**this**);

playMenu.add(item);

} // end createPlayMenu method

// method to create AdminMenu items

**private** **void** createAdminMenu() {

JMenuItem item;

adminMenu = **new** JMenu("Admin");

item = **new** JMenuItem("Add Question");

item.addActionListener(**this**);

adminMenu.add(item);

item = **new** JMenuItem("Edit Question");

item.addActionListener(**this**);

adminMenu.add(item);

}// end createAdminMenu method

**private** **void** newQuestionForm() {

c1.show(kingPanel, "1");

questionField.grabFocus();

}// END OF TESTING METHOD

// addNewQuestion is method to allow user to add questions to the quiz and

// set the options to T/F

**private** **void** addNewQuestion() **throws** IOException {

question = questionField.getText();

questionField.setText("");

intakeOptions();

q1 = **new** Question(question, answers);

questions.add(q1);

**try** {

save();// method to save questions ArrayList to dat file

} // try

**catch** (IOException f) {

f.printStackTrace();

}// catch

resultString = "Welcome to the QUIZ";

c1.show(kingPanel, "2");

}// end of addNewQuestions method

// method to write questions ArryList to dat file

**public** **void** save() **throws** IOException {

ObjectOutputStream os;

os = **new** ObjectOutputStream(**new** FileOutputStream("allQuestion.dat"));

os.writeObject(questions);

os.close();

}

// this method will ask the user every question

**private** **void** takeQuiz() **throws** IOException {

// staring test quiz

**int** n = 0;

**double** totalRight = 0, totalQuestion = 0;

**for** (Question q1 : questions) {

q1 = questions.get(n);

answerKey = q1.getKey();

choice = JOptionPane.*showInputDialog*(q1.toString()

+ "\n\nYour Answer");

totalRight += *scoreAnswers*(answerKey, choice);

totalQuestion++;

n++;

}

resultString = "You scored "

+ String.*format*("%.0f", ((totalRight / totalQuestion) \* 100))

+ "%";

blankLabel.setText(resultString);

}// end takeQuiz() method

// 10 question quiz method

**private** **void** take10Quiz() **throws** IOException {

// to keep track of used questions

ArrayList<Integer> used = **new** ArrayList<Integer>();

**double** totalRight = 0;

**int** randomNum;

// String myNums="";

**if** (questions.size() >= 10) {

randomNum = (**int**) (Math.*random*() \* questions.size());

used.add(randomNum);

**for** (**int** h = 0; h < 10; h++) {

**for** (**int** y = 0; y < h; y++) {

**while** (used.contains(randomNum)) {

randomNum = (**int**) (Math.*random*() \* questions.size());

System.***out***.println(randomNum);

}

}

used.add(randomNum);

q1 = questions.get(randomNum);

choice = JOptionPane.*showInputDialog*(q1.toString()

+ "\n\nYour Answer");

answerKey = q1.getKey();

totalRight += *scoreAnswers*(answerKey, choice);

// myNums+=randomNum; to test that numbers are not being

// repeated

}

// System.out.print(myNums);

resultString = "You scored "

+ String.*format*("%.0f", ((totalRight / 10) \* 100)) + "%";

blankLabel.setText(resultString);

} **else**

JOptionPane.*showMessageDialog*(**null**,

"There are not enough questions stored to take this Quiz",

"", JOptionPane.***WARNING\_MESSAGE***);

}// end take10Quiz() method

// 25 question quiz method

**private** **void** take25Quiz() **throws** IOException {

// to keep track of used questions

ArrayList<Integer> used = **new** ArrayList<Integer>();

**double** totalRight = 0;

**int** randomNum;

**if** (questions.size() >= 25) {

randomNum = (**int**) (Math.*random*() \* questions.size());

used.add(randomNum);

**for** (**int** h = 0; h < 25; h++) {

**for** (**int** y = 0; y < h; y++) {

**while** (used.contains(randomNum)) {

randomNum = (**int**) (Math.*random*() \* questions.size());

System.***out***.println(randomNum);

}

}

used.add(randomNum);

q1 = questions.get(randomNum);

choice = JOptionPane.*showInputDialog*(q1.toString()

+ "\n\nYour Answer");

answerKey = q1.getKey();

totalRight += *scoreAnswers*(answerKey, choice);

}

resultString = "You scored "

+ String.*format*("%.0f", ((totalRight / 25) \* 100)) + "%";

blankLabel.setText(resultString);

} **else**

JOptionPane.*showMessageDialog*(**null**,

"There are not enough questions stored to take this Quiz",

"", JOptionPane.***WARNING\_MESSAGE***);

}// end take25Quiz() method

// this method will tabulate the score for takeQuiz method

**public** **static** **double** scoreAnswers(String key, String choices) {

**int** outOf = key.length();

**double** questionScore = 0, score = 0;

**char** pick;

**if** (choices.length() > key.length()) {

**return** questionScore;

} **else** {

**for** (**int** i = 0; i < choices.length(); i++) {

choices = choices.toUpperCase();

pick = choices.charAt(i);

**for** (**int** n = 0; n < key.length(); n++) {

**if** (pick == key.charAt(n))

score++;

}

}

questionScore = (score / outOf);

}// end if/else

**return** questionScore;

}// end scoreAnswers() method

**private** **void** intakeOptions() {

String answerA, answerB, answerC, answerD, answerE, answerF;

answers = **new** ArrayList<Answer>();

answerA = aField.getText();

aField.setText("");

**if** (!answerA.equals("")) {

isCorrectString = aTf.getText();

aTf.setText("True/False");

**if** (isCorrectString.charAt(0) == 'f'

|| isCorrectString.charAt(0) == 'F')

isCorrect = **false**;

**else**

isCorrect = **true**;

ans = **new** Answer(answerA, isCorrect);

answers.add(ans);

}

answerB = bField.getText();

bField.setText("");

**if** (!answerB.equals("")) {

isCorrectString = bTf.getText();

bTf.setText("True/False");

**if** (isCorrectString.charAt(0) == 'f'

|| isCorrectString.charAt(0) == 'F')

isCorrect = **false**;

**else**

isCorrect = **true**;

ans = **new** Answer(answerB, isCorrect);

answers.add(ans);

}

answerC = cField.getText();

cField.setText("");

**if** (!answerC.equals("")) {

isCorrectString = cTf.getText();

cTf.setText("True/False");

**if** (isCorrectString.charAt(0) == 'f'

|| isCorrectString.charAt(0) == 'F')

isCorrect = **false**;

**else**

isCorrect = **true**;

ans = **new** Answer(answerC, isCorrect);

answers.add(ans);

}

answerD = dField.getText();

dField.setText("");

**if** (!answerD.equals("")) {

isCorrectString = dTf.getText();

dTf.setText("True/False");

**if** (isCorrectString.charAt(0) == 'f'

|| isCorrectString.charAt(0) == 'F')

isCorrect = **false**;

**else**

isCorrect = **true**;

ans = **new** Answer(answerD, isCorrect);

answers.add(ans);

}

answerE = eField.getText();

eField.setText("");

**if** (!answerE.equals("")) {

isCorrectString = eTf.getText();

eTf.setText("True/False");

**if** (isCorrectString.charAt(0) == 'f'

|| isCorrectString.charAt(0) == 'F')

isCorrect = **false**;

**else**

isCorrect = **true**;

ans = **new** Answer(answerE, isCorrect);

answers.add(ans);

}

answerF = fField.getText();

fField.setText("");

**if** (!answerF.equals("")) {

isCorrectString = fTf.getText();

fTf.setText("True/False");

**if** (isCorrectString.charAt(0) == 'f'

|| isCorrectString.charAt(0) == 'F')

isCorrect = **false**;

**else**

isCorrect = **true**;

ans = **new** Answer(answerF, isCorrect);

answers.add(ans);

}

}// end of intake method

**public** **void** deleteQuestion() **throws** IOException {

String[] buttons = { "Delete Question", "Keep Question" };

**for** (**int** g = 0; g < questions.size(); g++) {

q1 = questions.get(g);

**int** returnValue = JOptionPane.*showOptionDialog*(**null**,

q1.toStringFull(), "Delete Quiz Question",

JOptionPane.***WARNING\_MESSAGE***, 0, **null**, buttons, buttons[i]);

/\*From the use of a string array for button names, until the line above, I found something on the internet to show me how

\* to do that but I am unable to find that code again to reference it correctly. \*/

**if** (returnValue == 0) {

System.***out***.println("into delete if");

questions.remove(g);

}

}

}// end deleteQuestion method

}// end class

### Question class JavaDoc





 