Unit 7 Programming Assignment

In this assignment, you will again modify your Quiz program from the previous assignment. You will replace the text based dialog boxes with button-based dialog boxes specialized for multiple-choice and true-false questions.

This assignment uses many classes and methods described in Chapter 6 of Eck (2019), and the instructions below include references to the relevant sections of Chapter 6.

First, modify your main program for easier testing.

- Open your "CS1102" project in Eclipse.
- Bring up "Quiz.java" in the editor pane.
- Use "/*" and "*/" to comment out all your questions except one true/false question.
- Run your program to make sure it asks just one true/false question.

Create a new class for your specialized dialog boxes.

- Select "New" -> "Class" from the File menu.
- Enter the name "QuestionDialog" and click "Finish".

This class will extend the Java component "JDialog" to create a dialog window. ("JDialog" is similar to "JFrame".) Your class will also implement the interface "ActionListener" to respond to button clicks. (See Section 6.1.3.)

- Add import statements for "JDialog" and "ActionListener". Use "*" to simplify the list.
```
import java.awt.event.*;
import javax.swing.*;
```
- Exend "JDialog" and promise to implement "ActionListener".
```
public class QuestionDialog extends JDialog implements
ActionListener {
```

The dialog box will store the user's answer based on a button click.

- Add a String instance variable called "answer".
```
String answer;
```
- Add an implementation of the "actionPerformed" method to support the "ActionListener" interface.
```
public void actionPerformed(ActionEvent e) {
```
- Inside the method, set "answer" to the label of the button that generated the event "e". As described in Section 6.5.1, this label is returned by "getActionCommand".
```
answer = e.getActionCommand();
```

When a button is clicked in a "QuestionDialog", "answer" will get set to its label. This is all the dialog box needs to do, so it can then return control to the main program.

- Inside the "actionPerformed" method, after setting "answer", return control by closing the dialog. This is done with the "dispose()" method that "QuestionDialog" inherits from a parent class.
  ```
  dispose();
  ```

The editor pane for the class should show no error warnings. (The editor may show an informational warning about a missing ID for serialization, but you may ignore this since you will not be using the serialization features. Serialization is the ability to read and write object data from and to files.)

Now modify the "Question" class to use the new "QuestionDialog".

- Change the "question" instance variable from a String to a QuestionDialog.
  ```
  QuestionDialog question;
  ```

You will make more changes to the "Question" class later.

Next modify "TrueFalseQuestion" to use "QuestionDialog". Start with the "ask" method, which becomes much simpler. It only needs to do two things: make the question dialog visible and then get the answer from it.

- Delete the current statements from the method "ask" in "TrueFalseQuestion".
- Add a statement to make the "question" dialog box visible. Like for "JFrame", "JDialog" uses the "setVisible" method.
  ```
  question.setVisible(true);
  ```
  This will "turn on" the "question" dialog box, which runs until it calls "dispose".
- Once the "question" dialog box returns from "setVisible", it has set its answer. (This is because the dialog box is "modal", as explained below.) Have the "ask" method return the answer.
  ```
  return question.answer;
  ```

Next comes the more-complicated task of building up the "QuestionDialog" box for a true/false question. You do this in the constructor for "TrueFalseQuestion".

- Change the import statement in "TrueFalseQuestion.java" to include all Swing classes, and add an import for the AWT classes.
  ```
  import java.awt.*;
  import javax.swing.*;
  ```
- Replace the String initialization of "this.question" in the "TrueFalseQuestion" constructor with a new "QuestionDialog".
  ```
  this.question = new QuestionDialog();
  ```
- The dialog box will have a single question with buttons below. Give it a grid layout with one column and any number of rows. (See Section 6.1.1.)
  ```
  this.question.setLayout(new GridLayout(0,1));
  ```

- Add the "question" String to the window as a "JLabel". Center it using "JLabel.CENTER". Add some spaces to each end of the String to keep it from getting too close to the edges of the dialog window. (See Section 6.5.2.)
```
this.question.add(new JLabel("      "+question+"
     ",JLabel.CENTER));
```
- Create a new "JPanel" to organize the true/false buttons. You will add this panel to the dialog box later.
```
JPanel buttons = new JPanel();
```

The next step is to add buttons to the "buttons" panel. You use a separate panel for the buttons so they appear in a single row (using the default layout, "FlowLayout", as described in Section 6.6.1).

Because you will add more than one button, encapsulate the steps in an "addButton" method.

- Add an "addButton" method to the "TrueFalseQuestion" class. The method takes the buttons panel and a button label as parameters.
```
void addButton(JPanel buttons, String label) {
```
- In the "addButton" method, create a new button with the provided label.
```
JButton button = new JButton(label);
```
- The "question" object is programmed to respond to button clicks (by implementing "actionPerformed" from the "ActionListener" interface). Wire it up to the new button by adding it as a listener. (See Section 6.1.3.)
```
button.addActionListener(question);
```
- Finally, add the button to the "buttons" panel. (See Section 6.1.2.)
```
buttons.add(button);
```

Use this "addButton" method to add true/false buttons to the dialog box.

- In the "TrueFalseQuestion" constructor, add a "TRUE" button to the "buttons" panel. This statement must come after the definition of the "buttons" panel.
```
addButton(buttons,"TRUE");
```
- Add another button for "FALSE".
- Add the "buttons" panel to the dialog box itself. The multi-column panel will be added to the single-column grid layout.
```
this.question.add(buttons);
```

The construction of the dialog box requires a few additional steps in the "TrueFalseQuestion" constructor for proper behavior, size, and location.

- Make the dialog box "modal", as described in Section 6.7.2. This will keep the dialog box from returning control to the program until it is disposed. (Recall that this disposal happens after a button is clicked, thanks to your implementation of "actionPerformed" in "QuestionDialog".)
```
this.question.setModal(true);
```
- Use the "pack" method to resize the dialog box based on its contents (Section 6.7.3).

```
this.question.pack();
```
- Center the dialog box on the screen using the "setLocationRelativeTo" method with a "null" argument. This method is not described in Eck (2014), but it appears in standard Java documentation.
```
this.question.setLocationRelativeTo(null);
```

That completes the construction of the dialog box "question" in the "TrueFalseQuestion" constructor. Note that the constructor should still include the initialization of the "answer" String from the previous assignment.

The editor pane for "TrueFalseQuestion.java" should have no errors. If you open "MultipleChoiceQuestion.java", however, you should see errors from the initialization of "question", which is now a "QuestionDialog" instead of a String.

- Delete the "question" initialization statements from the "MultipleChoiceQuestion" constructor.

If you properly commented out all but one true/false question in your main program, you should now be able to run. The program should display a dialog box with your true/false question, along with "TRUE" and "FALSE" buttons. Clicking one of the buttons should close the window and replace it with the appropriate correct/incorrect dialog.

Next comes building the dialog box for "MultipleChoiceQuestion". The first step is to re-use what you can from the "TrueFalseQuestion" definition. You can do this by moving parts of the definition to the parent class, "Question".

For example, the "ask" method in "TrueFalseQuestion" is now also appropriate for "MultipleChoiceQuestion" also.

- In "Question.java", replace the existing import statement for "JOptionPane" with the two imports in "TrueFalseQuestion.java".
```
import java.awt.*;
import javax.swing.*;
```
- Replace the abstract "ask" declaration in "Question" with the concrete "ask" definition from "TrueFalseQuestion".
- Delete the "ask" definition from "TrueFalseQuestion".

Some of the statements in the "TrueFalseQuestion" constructor can be shared with "MultipleChoiceQuestion", but the order of the statements is important. Statements at the beginning of the constructor can be moved to a "Question" constructor for sharing.

- Create a "Question" constructor that takes a String "question" parameter.
```
Question(String question) {
```
- Copy the first three lines of the "TrueFalseQuestion" constructor and paste them into the "Question" constructor. These lines initialize the "question" instance variable, set its layout manager, and add the question text to its content.

```
this.question = new QuestionDialog();
this.question.setLayout(new GridLayout(0,1));
his.question.add(new JLabel(" "+question+"
",JLabel.CENTER));
```
Note that you still need "this" to distinguish the "QuestionDialog" instance variable "question" from the String parameter "question".

- In the "TrueFalseQuestion" constructor, replace these lines with a single call to the "Question" constructor.
  ```
  super(question);
  ```
  Recall that "super" must be the first statement in the "TrueFalseQuestion" constructor.
- Add the equivalent "super" statement to the "MultipleChoiceQuestion" constructor to eliminate error warnings. Note that the actual argument in this case may be "query" instead of "question".
  ```
  super(query);
  ```
- You can now delete the "java.awt.*" import statement from "TrueFalseQuestion" because that dependence has been moved to "Question".

You should now have no error warnings, and your program should work as before. Give it a try.

The "TrueFalseQuestion" constructor has more statements to share, but they come after some statements that are specific to "TrueFalseQuestion". Encapsulate the sharable statements in a "Question" method.

- Create a method in "Question" to finish the initialization of the "question" instance variable. It needs no parameters, and it returns nothing.
  ```
  void initQuestionDialog() {
  ```
- Copy the statements from the "TrueFalseQuestion" constructor that make "question" modal, set its size, and set its position.
  ```
  this.question.setModal(true);
  this.question.pack();
  this.question.setLocationRelativeTo(null);
  ```
- Paste them into "initQuestionDialog". Note that "this" is no longer needed for these statements, but it does not hurt.
- Replace the three statements in the "TrueFalseQuestion" constructor with a call to "initQuestionDialog", which is now inherited from "Question".
  ```
  initQuestionDialog();
  ```

Again, you should have no error warnings, and your program should work as before. Test once more.

Now you are ready to create the dialog box for "MultipleChoiceQuestion".

- In "MultipleChoiceQuestion.java", replace the import for "JOptionPane" with the imports for all AWT and Swing classes.
  ```
  import java.awt.*;
  import javax.swing.*;
  ```

- Delete the "ask" method. Instead it will use the new one inherited from "Question".

Just as you gave "TrueFalseQuestion" the method "addButton", you will give "MultipleChoiceQuestion" a method to create its answer choices. Each choice will need both a button and a text label, however, and the choices should be arranged in a column instead of a row.

- In "MultipleChoiceQuestion", create a method called "addChoice" that takes two String arguments, one for the button name and one for the answer label.
  ```
  void addChoice(String name, String label) {
  ```
- In this method, create a panel to arrange the button and the label. Use a "BorderLayout", as described in Section 6.6.1.
  ```
  JPanel choice = new JPanel(new BorderLayout());
  ```
- Create a button with the given name.
  ```
  JButton button = new JButton(name);
  ```
- Wire this button into the "actionPerformed" method of the "question" object, so it will record the answer if the button is clicked.
  ```
  button.addActionListener(question);
  ```
- Add this button to the "choice" panel, aligned to the left, or "WEST", border of the panel.
  ```
  choice.add(button,BorderLayout.WEST);
  ```
- Add the answer label to the center section of the panel, with some extra spaces on the end to keep the label from appearing too close to the edge of the dialog box. Align the text to the left so it starts near the button.
  ```
  choice.add(new JLabel(label+"
       ",JLabel.LEFT),BorderLayout.CENTER);
  ```
- Finally add the panel holding the button and answer label to the dialog box.
  ```
  question.add(choice);
  ```
  Recall that the dialog box has a grid layout with a single column, so this will make a column of answer choices.

The final step is to update the "MultipleChoiceQuestion" constructor to build the dialog box.

- Use the "addChoice" method in the "MultipleChoiceQuestion" constructor to add choice "A" using formalparameter "a" as the answer text.
  ```
  addChoice("A",a);
  ```
  Note that this statement must come after the call to "super" in the constructor.
- Add choices for the other formal parameters: "b", "c", "d", and "e".
- Perform the rest of the initialization of the dialog box by calling "initQuestionDialog".
  ```
  initQuestionDialog();
  ```
- Make sure that the constructor still includes the initialization of "correctAnswer" from the previous assignment.

You should now be able to use "MultipleChoiceQuestion".

- Un-comment all the questions in your main program in "Quiz". The file "Quiz.java" should now be unchanged from the previous assignment.

You should not have any error warnings, and you should be able to run your program to ask all your previous questions, but with your new dialog boxes!

Your assignment will be graded by your peers using the following criteria. Each item is worth 10 points, for a total of 90 points.

- Does the submission include a file "QuestionDialog.java" with a class that extends "JDialog" and implements "ActionListener"?
- Does the class "QuestionDialog" have a method "actionPerformed" that uses its "ActionEvent" parameter to set the instance variable "answer" and then calls the inherited "dispose" method?
- Does the submission include a file "Question.java" with a constructor that initializes instance variable "question", gives it a single-column grid layout, and adds a text label using the constructor's String parameter?
- Does class "Question" have a method "ask" that makes the instance variable "question" visible and returns the value "question.answer"?
- Does class "Question" have a method "initQuestionDialog" that makes instance variable "question" modal, sets its size with "pack", and positions it in the center of the screen?
- Does the class "TrueFalseQuestion" have a method "addButton" that constructs a button using its String parameter, adds the instance variable "question" as a listener for that button, and adds the button to its "JPanel" parameter?
- Does the class "TrueFalseQuestion" have a constructor that calls its superclass constructor with its first String parameter, calls "addButton" to add "TRUE" and "FALSE" buttons to a panel, adds that panel to the instance variable "question", calls "initQuestionDialog", and initializes the instance variable "correctAnswer" with its second String parameter?
- Does class "MultipleChoiceQuestion" have a method "addChoice" that creates a panel with a border layout, creates a button using its first String parameter, adds the instance variable "question" as a listener for that button, adds the button to the left side of the panel, adds a label to the center of the panel using its second String parameter, and adds the panel to the instance variable "question"?
- Does the class "MultpleChoiceQuestion" have a constructor that calls its superclass constructor with its first parameter, calls "addChoice" using its next five parameters, calls "initQuestionDialog", and initializes the instance variable "correctAnswer" with its last parameter?