

Design Tasks 2&3

Sample Project for Assignment 4
CISC 323, winter 2004

General Notes:

- I have used some notations which are not official UML but should be clear:
 - "Vector of C" means a Vector, all of whose elements are of class C
- I expect that I will be writing some toString methods and main methods for debugging purposes. I'm not documenting these here because they're not really part of the program that will be used in the end.

QuizIO

```
+readQuiz(infile: File, quiz: Quiz)  
+writeQuiz(outfile: File, quiz: Quiz)
```

The QuizIO class contains static methods for reading a quiz from a file and writing a quiz to a file. The readQuiz method discards all questions in the quiz and reads new questions for it. The writeQuiz method writes all the questions from the quiz to a file.

The methods throw IOExceptions if there is an I/O error such as the file not existing. They throw QuizException if a file being read is in a bad format.

Quiz
-questions: Vector of Question
+Quiz() +getNumQuestions(): int +getQuestion(i: int): Question +getAllQuestions(): Vector +addQuestion(q: Question) +deleteQuestion(index: int)

A Quiz object represents an entire quiz, which is basically a collection of questions. When a Quiz object is created, it contains no questions.

The getAllQuestions method returns a *copy* of the Vector of Questions in the Quiz. Adding to or deleting from this Vector doesn't change the Quiz.

<i>Question</i>
-text: String
+getText(): String +setText(text: String) <i>+copy(): Question</i> <i>+equals(Object obj): boolean</i>

A Question object represents a single quiz question. All questions have a text – the String that is displayed to ask the question. The rest depends on the subclass.

The copy method makes a complete copy of the Question. It will be needed for editing questions, so that the program may revert to the old question if the user cancels an editing transaction.

MCQuestion
-choices: Vector of Strings -correctChoice: int
+MCQuestion(text: String, choices: Vector, correct: int) +getNumChoices(): int +getChoice(i: int): String +getCorrectChoice(): int +copy(): Question +equals(Object obj): boolean

An MCQuestion represents a multiple-choice question. It contains a text, a Vector of choices, and the index of the correct choice. The methods `getNumChoices` and `getChoice` provide a way for a user of this class to loop through the choices for the question.

Note: In this diagram and the diagrams for all other subclasses, we don't show attributes and methods which are inherited from `Question` unless this class overrides them with a new version.

SAQuestion
-answer: String
+SAQuestion(text: String, answer: String) +getAnswer(): String +copy(): Question +equals(Object obj): boolean

A SAQuestion represents a short-answer question. It contains a text and a correct answer.

TFQuestion
-answer: boolean
+TFQuestion(text: String, answer: boolean) +getAnswer(): boolean +copy(): Question +equals(Object obj): boolean

A TFQuestion represents a true/false question. It contains a text and a correct answer.

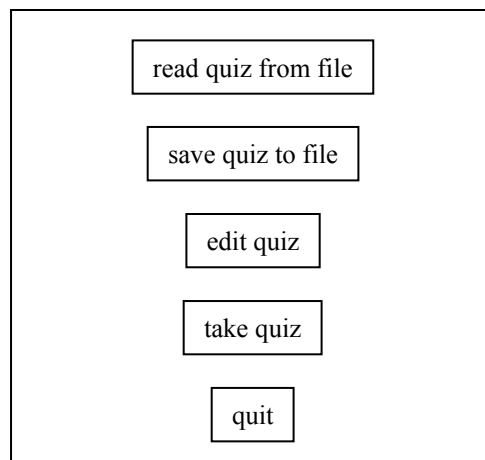
QuizException
+QuizException(msg: String)

QuizException is a subclass of Exception. The access-layer and business-layer class methods in this program throw QuizExceptions when they find certain kinds of errors. The view-layer methods catch and report them. Each QuizException will contain an error message suitable for displaying to the user.

MainQuizFrame
+MainQuizFrame() +actionPerformed(ActionEvent e)

MainQuizFrame is a subclass of JFrame and is the initial frame seen by the user. It offers the user a choice of basic operations, which will lead to different dialogs. It creates an empty quiz when it starts up and uses that Quiz object throughout the run of the program. If the quiz is empty, the save quiz and take quiz buttons are disabled. It's still OK to edit the quiz (for adding questions).

This class contains private methods for administering a quiz, which will involve popping up question dialogs and reporting the score at the end (with a JOptionPane).



<i>QuestionDialog</i>
<pre>+\$CORRECT_STATUS: int = 0 {invariant} +\$INCORRECT_STATUS: int = 1 {invariant} +\$QUIT_STATUS: int = 2 {invariant} -status: int</pre>
<pre>+showDialog(q: Question) +actionPerformed(ActionEvent e) +getStatus(): int</pre>

The program uses a *QuestionDialog* to ask a single question. No picture is shown here because the appearance will be different for each subclass of question. The program will call *showDialog* with the question as a parameter, then call *getStatus* to find out whether the user answered correctly or incorrectly or clicked a quit button to abort the quiz.

TFQuestionDialog

```
+TFQuestionDialog(parent: JFrame)  
+actionPerformed(ActionEvent e)  
+showDialog(q: Question)
```

TFQuestionDialog is a subclass of QuestionDialog and is used to ask the user a true/false question. The parameter to showDialog must always be a TFQuestion.

The capital of Alberta is Calgary

true

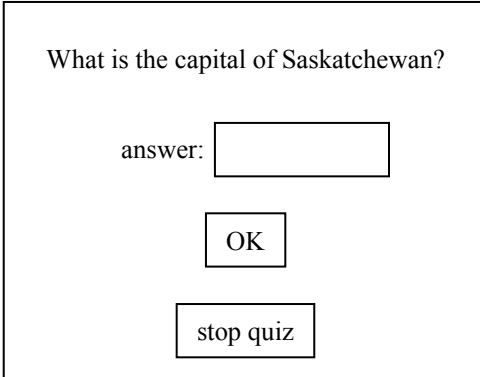
false

stop quiz

SAQuestionDialog

```
+TFQuestionDialog(parent: JFrame)  
+actionPerformed(ActionEvent e)  
+showDialog(q: Question)
```

SAQuestionDialog is a subclass of QuestionDialog and is used to ask the user a short answer question. The parameter to showDialog must always be an SAQuestion.



What is the capital of Saskatchewan?

answer:

OK

stop quiz

MCQuestionDialog

```
+MCQuestionDialog(parent: JFrame)  
+actionPerformed(ActionEvent e)  
+showDialog(q: Question)
```

MCQuestionDialog is a subclass of QuestionDialog and is used to ask the user a multiple choice question. The parameter to showDialog must always be an MCQuestion.

What is the capital of Ontario?

Londaon

Kingston

Toronto

Windsor

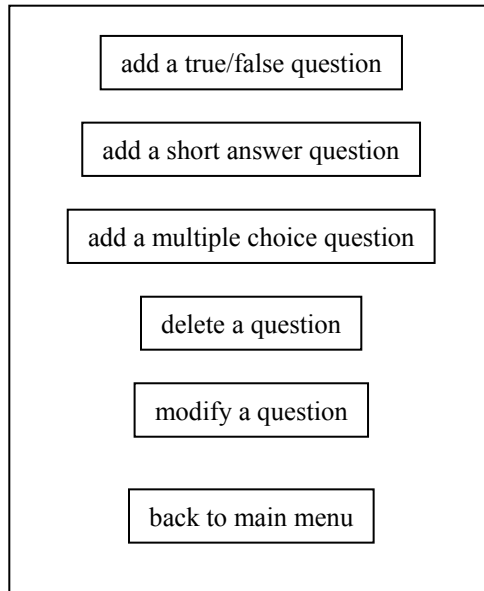
Barrie

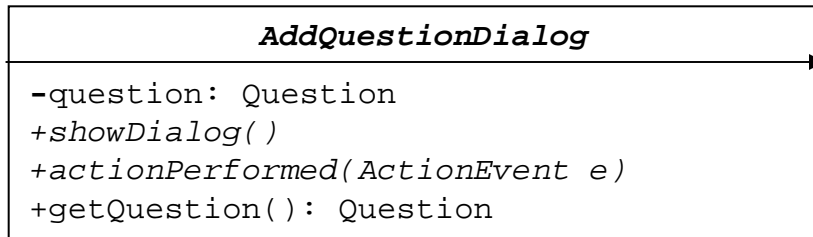
stop quiz

EditDialog

```
+EditDialog(parent: JFrame, quiz: Quiz)  
+actionPerformed(ActionEvent e)  
+showDialog()
```

The EditDialog lets the user choose between the main editing options.





The program uses an *AddQuestionDialog* to create a new question to add to the quiz. It stores the new question in the question attribute. The program will call `showDialog` and then call `getQuestion` to get the question the user created. The question will be null if the user clicked "cancel".

AddSAQuestionDialog

```
+AddSAQuestionDialog(parent: JDialog)  
+showDialog()  
+actionPerformed(ActionEvent e)
```

The program uses this dialog to let the user specify a new short answer question to add to the quiz.

text of question:

correct answer:

OK

cancel

AddTFQuestionDialog

```
+AddTFQuestionDialog()  
+showDialog()  
+actionPerformed(ActionEvent e)
```

The program uses this dialog to let the user specify a new true/false question to add to the quiz.

text of question:

correct answer:

AddMCQuestionDialog

```
+AddMCQuestionDialog()  
+showDialog()  
+actionPerformed(ActionEvent e)
```

The program uses this dialog to let the user specify a new multiple choice question to add to the quiz. It starts out with space for three choices. The user can click the "more choices" or "fewer choices" button to add or delete a choice from the end. The user must click one of the correct answer buttons to specify which choice is the correct one. There's a type of component called a radio button that we haven't discussed in class which will do this very easily. If I didn't know about radio buttons, I could just use regular buttons and remember which button was clicked most recently (allowing the user to change his/her mind). I could change the appearance of that button – for example, changing the colour. If the user clicked "OK" without ever checking a correct answer button, the program could pop up an error message or use a default such as the first choice.

text of question:	<input type="text"/>
answer choices:	check correct answer:
<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	<input type="checkbox"/>
<input type="button" value="more choices"/>	
<input type="button" value="fewer choices"/>	
<input type="button" value="OK"/>	
<input type="button" value="cancel"/>	

ChooseQuestionDialog
-questionIndex: int
+ChooseQuestionDialog(parent: JDialog, quiz: Quiz) +showDialog(delete: boolean) +actionPerformed(ActionEvent e) +getQuestion(): Question +getQuestionIndex(): int

The program uses this dialog to allow the user to choose a question from the quiz – either to modify or to delete it. The parameter to the showDialog method is true if the user is choosing a question to delete and false if the user is choosing a question to modify. This is used for the prompt in the dialog.

The getQuestion method will return the index of the question the user chose, or -1 if the user clicked cancel.

select a question to delete:

- The capital of Alberta is Calgary ▼
- What is the capital of Saskatchewan?
- What is the capital of Ontario?

OK

cancel

<i>EditQuestionDialog</i>

<i>+showDialog(question: Question)</i> <i>+actionPerformed(ActionEvent e)</i>
--

The program uses an `EditQuestionDialog` to modify a question.

EditSAQuestionDialog

```
+EditSAQuestionDialog(parent: JDialog)  
+showDialog(question: Question)  
+actionPerformed(ActionEvent e)
```

The program uses this dialog to let the user modify a short answer question. It displays the text and correct answer in text fields which the user can modify. If the user clicks "OK", it uses the user's input to change the question. If the user clicks "cancel", the question is left unchanged.

text of question:	<input type="text" value="What is the capital of Saskatchewan?"/>
correct answer:	<input type="text" value="Regina"/>
	<input type="button" value="OK"/>
	<input type="button" value="cancel"/>

EditTFQuestionDialog

```
+EditTFQuestionDialog(parent: JDialog)  
+showDialog(question: Question)  
+actionPerformed(ActionEvent e)
```

The program uses this dialog to let the user modify a true/false question. It displays the text and correct answer in a field and combo box which the user can modify. If the user clicks "OK", it uses the user's input to change the question. If the user clicks "cancel", the question is left unchanged.

The dialog window contains the following elements:

- A label "text of question:" followed by a text input field containing "The capital of Alberta is Calgary".
- A label "correct answer:" followed by a dropdown menu with "true" selected and a downward arrow, and "false" as an alternative option.
- An "OK" button.
- A "cancel" button.

EditMCQuestionDialog

```
+EditMCQuestionDialog(parent: JDialog)  
+showDialog(question: Question)  
+actionPerformed(ActionEvent e)
```

The program uses this dialog to let the user modify a multiple-choice question. It displays the text, choices, and correct answer, which the user can modify. If the user clicks "OK", it uses the user's input to change the question. If the user clicks "cancel", the question is left unchanged. There is no separate dialog for adding a new choice; this can be done with a JOptionPane dialog.

text of question:	<input type="text" value="What is the capital of Ontario?"/>	
answer choices:	check correct answer:	delete this choice:
<input type="text" value="Kingston"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="text" value="Toronto"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="text" value="Ottawa"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="button" value="add new choice"/>		
<input type="button" value="OK"/>		
<input type="button" value="cancel"/>		