Project 9: Quiz Creator

Due: Sunday, November 6, 2011 by 11:59 PM

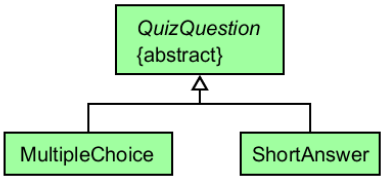
Deliverables:

The following project files must be submitted by the due date and time specified above (see the Lab Guidelines for information on submitting project files).A portion of the lab is due on Wednesday in lab. If the Wednesday portion is not complete, there will be a 5% deduction on your project grade. **Projects sent via e-mail past the deadline at 11:59 PM will not be accepted without a university-approved excuse.**

Files to submit to Web-CAT:

- QuizQuestion.java
- ShortAnswer.java and ShortAnswerTest.java
- MultipleChoice.java and MultipleChoiceTest.java

Overview: This week you will create classes to represent questions on a quiz. Your class hierarchy should look as follows:



Specifications:

Requirements: You will be creating classes to represent a multiple choice question and short answer question on a quiz. Each has an index and question text:

For a multiple choice question, possible answers are listed with the question. In the question key, only the correct answer is printed:

Question Display (displayQuestion method)	Question Key (displayKey method)
Question 1A: What is the function of the Java compiler? A. Run java programs B. Compile source code C. All of the above	Question 1A: What is the function of the Java compiler? B. Compile source code

There can be up to 52 choices on a multiple choice question. Answers should be marked A through Z and then starting at lowercase a through z.

For a short answer question, possible answers represent correct answers that could be provided by the user. In the question key, all correct answers are printed:

Question Display (displayQuestion method)	Question Key (displayKey method)
Question 1A: What is the name of the Java compiler?	Question 1A: What is the name of the Java compiler? javac javac.exe

Possible answers

QuizQuestion.java

- **Design**: Though a QuizQuestion object will never be instantiated, it contains methods and attributes that are common to MultipleChoice and ShortAnswer. Before you start coding QuizQuestion, make sure that you read the descriptions for MultipleChoice and ShortAnswer so that you'll know what is common to the two classes and can be placed in QuizQuestion.

  The following methods must be available in QuizQuestion. They can be either defined in QuizQuestion or be declared as abstract (whichever you think is most appropriate):
  - `setQuestionText` and `getQuestionText` are getter and setter methods for the question text (a String). The set method does not have to return a value.
  - `setQuestionIndex` and `getQuestionIndex` are getter and setter methods for the question index (a String such as "4A"). The set method should set the index and return true only if the index has a length of 0 to 10 characters (leading and trailing whitespace should not be counted).
  - `addAnswer`: Takes a String parameter representing a possible answer and returns true and adds the answer if or only if the answer is between 1 and 40 characters (leading and trailing whitespace should not be counted).
  - `displayQuestion`: Returns the question display as a String (see diagrams above and description of other 2 classes).
  - `equals`: takes a `QuizQuestion` object as a parameter. Two quiz questions are equal if their question text and index are the same.

  The following method must be abstract:
  - `displayKey`: Returns the question key as a String (see diagrams above and description of other 2 classes).
    ```
    public abstract String displayKey()
    ```

MultipleChoice.java

- **Design**: A MultipleChoice question can be instantiated with no parameters or can be instantiated with an index:
  ```
  MultipleChoice mc1 = new MultipleChoice();
  MultipleChoice mc2 = new MultipleChoice("4a");
  ```

  The following methods are inherited from QuizQuestion (if a method is abstract then you will have to define it):
  - `setQuestionText`, `getQuestionText`, `setQuestionIndex` and `getQuestionIndex`: same function as that described in QuizQuestion

- `addAnswer`: Includes all of the functionality described in the QuizQuestion. Additionally, the question will NOT be added if there are more than 52 existing choices.
- `displayQuestion`: If there is no index, then just display the Question text and possible answers (starting with A. B. C. etc). If there is an index, then Question ___: must be displayed on the line before the text (where ___ is the index). When printing the question text, at most 50 characters can be displayed in each line. New line escape sequences ("\r\n") must be inserted (without splitting words) so that no line exceeds the 50 character limit. So, a question with the following answer text:

```
What must one do when the question has more than 50 characters
in a line? How do you think that this would be accomplished?
```

  Must be printed as follows:

```
What must one do when the question has more than
50 characters in a line? How do you think that
this would be accomplished?
```

- `displayKey`: Prints the question exactly as in displayQuestion, but only prints the correct option.

The following method must also be added:

- `setCorrectOption`: takes a char parameter that represents the correct answer. Sets the correct answer and returns true if an only if the answer is a valid choice for that question.
- `getCorrectOption`: returns the character representing the correct answer

For example, the question on page 1 would be created as follows:

```
MultipleChoice mc2 = new MultipleChoice("4a");
mc2.setQuestionText("What is the function of the Java compiler?");
mc2.addAnswer("Run Java Programs");
mc2.addAnswer("Compile Source Code");
mc2.addAnswer("All of the above");
mc2.setCorrectOption('B'); // should return true
```

ShortAnswer.java

- **Design**: A ShortAnswer question can be instantiated with no parameters or can be instantiated with an index:

```
ShortAnswer sa1 = new ShortAnswer();
ShortAnswer sa2 = new ShortAnswer("4a");
```

  The following methods are inherited from QuizQuestion (if a method is abstract then you will have to define it):

  - `setQuestionText`, `getQuestionText`, `setQuestionIndex` and `getQuestionIndex`: same function as that described in QuizQuestion
  - `addAnswer`: Has all of the functionality described in the QuizQuestion section.

- `displayQuestion`: Same as MultipleChoice, but do not print the answers that have been added.
- `displayKey`: Prints the question exactly as in displayQuestion, but prints all possibly correct answers as well (one per line).

For example, the question on page 2 would be created as follows:

```
ShortAnswer sa1 = new ShortAnswer();
sa1.setQuestionText("What is the name of the Java compiler?");
sa1.addAnswer("javac");
sa1.addAnswer("javac.exe");
```

Code & Test: Download the two test files on the class website and add them to a project with your three classes. Make sure to mark the two test files as tests (right click -> "Mark as Test" in the Open Projects pane) before attempting to compile your files. Fill in the tests, following the instructions in the test comments.

- **Statement-level coverage**: Your tests must cover every single line of code in your program as they cover your program functionality. You can make sure that this is performed by running the test in debug mode and making sure that you have traversed every line of code in each method during the course of testing. **You may have to add additional test methods to achieve statement-level coverage.**
- **Design note**: Anytime that you see a method that performs a similar function in both ShortAnswer and QuizQuestion, you can include that code in QuizQuestion. If the method then requires more functionality in a subclass, you can override the method with new method in the subclass. The new method can include a call to super.methodName() along with statements for the additional functionality that you need for the subclass.
- **Repeated code**: You will lose points for repeating code. Suggestions:
  - Keep as much common functionality as you can in QuizQuestion.
  - Never override a method unless necessary. For example, MultipleChoice only really needs 2 constructors and 5 public methods in its class body; the rest can be inherited "as is" from QuizQuestion.
  - Including private or protected support methods if necessary.
- **Misc**: You will lose points for having instance variables that should be local variables (a Scanner, for example, should probably be local). You will also lose points for using literal values like 52 in your code rather than well-named constants.

HINTS (optional for ambitious and / or reading-averse students):

- You can use a Scanner object on the Question code string and add individual words to each line of the output in displayQuestion (use the next method) until a length limit is reached (at which point you will need to add a new line).
- The above code can be placed in the displayQuestion method of QuizQuestion; that way when you override **both displayQuestion and displayKey** you can invoke super.displayQuestion() and then add anything else that you might need to the output.
- Start early. Don't panic.