Lab Assignment 4:
eQuiz System

Due Date (a one-week LA)
03/02/16 11:59pm – For Wednesday Lab Sections **03/03/16 11:59pm – For Thursday Lab Sections**

Objectives

- Abstract Classes
- Polymorphism

Problem Specification

For this lab assignment, you are going to assume that you are a professor designing your own quiz program. Your quiz can have multiple question types, all children of the Question class. Your program will consist of two parts, the Question classes that extend the Question class, the quiz, which knows nothing about the implementation of the Question but can have many questions, and the main class which handles all that.

Input: The input is handled by the createQuiz() method. You don't have to worry about file io for this assignment. Assume, however, that the questions in the method might change.

Sample Output:
Program Menu:
1. Take Quiz.
2. Exit
Please choose an option: 1
Question 1: The cat _____ the rat.
Fill in the blank.
Enter your answer: ate //note: Case does not matter.
Your answer is correct.
Question 2: 1 is less than 2
Enter true or false.
Enter your answer: false //note: Case does not matter. T and f are also valid answers.
Your answer is incorrect.
Question 3: What color is snow?
Type in a one word answer.
Enter your answer: White //note: Case does not matter.
Your answer is correct.

You have finished the exam.
Out of 3 questions, you answered 2 correctly. //note: these numbers should not be hardcoded. The quiz can have many questions, the user might make no mistakes, etc…

For the menu, any option other than 0 or 1 should prompt the user to enter a correct option.
The prompts after the question are not part of the question. The text for question three is “What color is snow?” The phrase “Type in a one word answer.” is appended by the getQuestion method. Read the comments in the classes below for more details.

IQuestion Interface:

```
public interface IQuestion {


	/**
	 * @return returns the question text plus any other extra prompts or details.
	 */
	public String getQuestion();

	/**
	 *
	 * @return the answer as a string
	 */
	public String getAnswer();

	/**
	 *
	 * @param answer the answer entered by the user
	 * @return true if the answer is correct, false otherwise.
	 */
	public boolean checkAnswer(String answer);
}
```

Details:

- You are to implement this interface in a class called Question.

- Add the following method to your implementation.

```
/**
 *
 * @return the wording of the question as a string
 */
final String getText(){
	…..
}
```

- getQuestion, getAnswer, and checkAnswer should be abstract.

IQuiz Interface

```
public interface IQuiz {
	/**
	 * @param questions list of questions in the quiz
	 */
	public void setQuestions(List<IQuestion> questions);
	/**
	 * @return List of questions in the quiz
	 */
	public List<IQuestion> getQuestions();
	/**
	 * @return number of questions in the quiz
	 */
	public int getNumberOfQuestions();
	/**
	 * @return True if a quiz has been started. False otherwise
	 */
	public boolean isStarted();
	/**
	 * @return True if the last question has been answer, false otherwise.
	 */
	public boolean isCompleted();
	/**
	 * @return true if there is a question after the current one, false otherwise.
	 */
	public boolean hasNextQuestion();
	/**
	 * Sets isCompleted to false if there are no more questions after returning;
	 * @return the next question that should be answered.
	 */
	public IQuestion getNextQuestion();
	/**
	 * sets the isStarted variable to true if it is false and returns the current question
	 * @return the current question waiting for an answer.
	 */
	public IQuestion getCurrentQuestion();
	/**
	 * @return the answer to the current question
	 */
	public String getCurrentAnswer();
	/**
	 * checks if the user input matches the question's answer.
	 * @param answer the user input as a string
	 * @return true if the answer matches, false otherwise.
	 */
	public boolean checkAnswer(String answer);
	/**
	 * Resets the number of incorrect answers and the current question to 0,
	 * isStarted and isCompleted to false;
	 */
	public void reset();

	/**
	 * @return the number of the current question
	 */
	public int getQuestionNumber();
}
```

Quiz UI Interface

```
public interface IQuizUI {

	/**
	 * Creates a quiz out of a set of questions
	 * @param questions a list of questions
	 * @return the quiz object
	 */
	public IQuiz createQuiz(List<IQuestion> questions);

	/**
	 * takeQuiz will go through all of the questions,
	 * handle user input, and check the answers
	 * @param  the quiz to be taken
	 */
	public void takeQuiz(IQuiz quiz);
}
```

Main Class: DO NOT MODIFY

```
public class Main {

	public static int getUserChoice(Scanner in){

		int choice =0;
		do{
			System.out.println("Program Menu");
			System.out.println("0. Take Quiz");
			System.out.println("1. Exit");
			System.out.print("Please choose an option: ");
			try{
				choice=Integer.parseInt(in.nextLine());
			}catch(NumberFormatException e){
				System.out.println("Please enter only a number.");
				choice = -1;
			}
		}while(choice != 0 && choice != 1);
		return choice;
	}
	public static void main(String[] args){
		List<IQuestion> questions = new ArrayList<IQuestion>();
		questions.add(new BlankFill("The cat _____ the rat.", "ate"));
		questions.add(new TrueFalse("1 is less than 2", true));
		questions.add(new ShortAns("What color is snow?", "white"));
		QuizUI qUI = new QuizUI();
		IQuiz myQuiz = qUI.createQuiz(questions);
		if(myQuiz==null)
			return;
		Scanner scanner = new Scanner(System.in);
		while(getUserChoice(scanner)!=1){
			myQuiz.reset();
			qUI.takeQuiz(myQuiz);
		}
	}
}
```

Requirements:

1. Create a project and name it edu.wmich.cs1120.spring16.quiz.YOUR_ID
    a. YOUR_ID is the id you used to log in to gowmu.
2. Create four packages:
    a. edu.wmich.cs1120.spring16.quiz.YOUR_ID
    b. edu.wmich.cs1120.spring16.quiz.YOUR_ID.Driver
    c. edu.wmich.cs1120.spring16.quiz.YOUR_ID.Questions
    d. edu.wmich.cs1120.spring16.quiz.YOUR_ID.Quiz
3. Add the Main class to the Driver package.
4. Add the IQuestion Interface to the Question package and implement it in Question. Don't forget to read the details below the IQuestion interface.
    a. Add the BlankFill Class to the question package and extend Question
        i. Constructor: `public BlankFill(String questionText, String answer)`
        ii. The constructor will have to call Question's constructor to set the questionText variable.
        iii. Note that getText is final and package private.
        iv. Override the methods in Question. DO NOT CHANGE the signature or the return type of any of the methods.
        v. The getQuestion method should add "\nFill in the blank." to the question text.
    b. Add the ShortAns Class to the question package and extend Question
        i. Constructor: `public ShortAns(String questionText, String answer)`
        ii. Requirements 4.a.ii – 4.a.iv apply.
        iii. The getQuestion method should add "\Type in a one word answer" to the question text.
    c. Add the TrueFalse class to the question package and extend Question
        i. Constructor: `public TrueFalse(String questionText, boolean answer)`
        ii. Requirements 4.a.ii – 4.a.iv apply.
        iii. Note that it takes a Boolean as the answer. Do not change the method signatures or return types. (hint: look at Boolean.toString(…) and Boolean.parseBoolean(..)).
        iv. The getQuestion method should add "\nEnter true or false." To the question text.
        v. The getAnswer method should return a string representation of the answer (true for true and false for false).
        vi. The checkAnswer should accept a string with true, t, false, f in lower case, upper case, or a mix.
    d. Add the IQuiz interfance to the quiz package and implement it in the Quiz class. Also add IQuizUI and implement it in QuizUI.
        i. Note that the quiz class has no idea what the type of the question is or what it contains. All it has access to is the Question class.
    e. Do not assume that the questions will remain as those given to you.

Additional Requirements

Coding Standards

You must adhere to all conventions in the CS 1120 Java coding standard. This includes the use of white spaces for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming classes, variables, method parameters and methods. You must use the package instructions included above.

Assignment Submission Lab Portion

- A word or pdf document containing your UML diagrams plus pseudo code
- A zip file containing your Javadocs.

Assignment Submission Home Portion

- Generate a .zip file that contains all your files, including:
  - Program Files
    - including any input or output files
  - Java docs

NOTE: Standards will be strictly enforced.