SCIT

School of Computing and Information Technology

Spring 2015

CSCI213/MCS9213/CSCI813 — Java Programming and Applications

Assignment 1 (10 marks)

Due Time and Date:

23:59:59 on Sunday, 30 August

Objectives

This assignment requires you to design and implement Java classes to support a Java quiz system. You will apply your knowledge and skills in Java class design and implementation to develop a reusable class library in packages and with the proper API documentation. Java programming is to craft new classes and reuse existing classes. Indeed, Java is not just a programming language; it is a platform on which Java developers can work to achieve true reusability and rapid application development.

Upon completion of this assignment, you should be able to model abstraction, encapsulation and packages; to design and implement class, member, attribute, method, constructor and package; to use interface, inheritance and polymorphism.

Tasks and Requirements

In this assignment, you will design and implement Java classes that support a Java quiz system. The application program, called `JavaQuiz.java`, is provided in the Appendix A. You should not modify the application program. Instead, you should design and implement your classes to support it. The `Question` interface used in the program is provided in the Appendix B. You should not modify the `Question` interface. All your classes and the interface should be arranged in suitable packages, named `au.edu.uow.QuestionLibrary` and `au.edu.uow.UserInterface`.

A sample screen shot of running the Java Quiz Application is shown in the Appendix C.

General Requirements:

- You should observe the common principles in abstraction and encapsulation when you design your classes.
- You should create your classes by making use of features of Java classes and OO programming such as inheritance and polymorphism etc. where they are appropriate.
- You should make documentation and implementation comments in your codes where they are necessary.
- Logical structures and statements are properly used for specific purposes.
- Your code should comply with the major requirements of Java Code Conventions.

Task 1: Implementation of the `Student` Class

Requirements:

You should implement the `Student` class to support the Java quiz application to register the student's name, record scores and give the final score at the end of the quiz. The API (public methods) of the `Student` class is defined as follows:

```
public void setName(String name)
public String getName()
public void recordScore(boolean isCorrect)
public int getScore()
```

You should write code to implement above methods. This class could be in the package `au.edu.uow.UserInterface`.

Suggestions: Before you implement most functionality of your classes, the Java quiz application provided in the Appendix A may not be able to run properly. You can make a small application to test your classes while you develop them. It is a good idea that you always have a working program as you progress in implementing more methods.

Task 2: Design and Implementation of the `UserInterface` Class

Requirements:

You should design and implement a `UserInterface` class to support the Java quiz application to provide a console based user interface to display information and take user input. A sample screen shot that shows how this class interacts with users is shown in the Appendix C. This class should have at least the following methods as required by the Java quiz application:

```
public Student getStudent()
public void startQuiz(List<Question> quiz, Student student)
public void showStudentMarks(Student student)
```

The `getStudent()` method registers the student in a simple login process; the `startQuiz()` method presents the questions and records answers; the `showStudentMarks()` method shows the student marks. This class should be in the package `au.edu.uow.UserInterface`.

Task 3: Design and Implementation of the `QuestionLibrary` Class

Requirements:

You should design and implement a `QuestionLibrary` class to support the Java quiz application to store all questions of the question library and make quizzes from the question library. This class should have at least the following methods as required by the Java quiz application:

```
public static boolean buildLibrary(String qFile)
public static List<Question> makeQuiz(int noOfQuestions)
```

The `buildLibrary()` method loads all questions from a question library file with a format as specified in the Appendix D to build a list of all questions; the `makeQuiz()` method makes a quiz with the specified number of questions from the question library. Each quiz should not have duplicates of questions and quizzes in different runs should contain different sets of randomly selected questions. This class should be in the package `au.edu.uow.QuestionLibrary`.

Task 4: Design and Implementation of Classes to Represent Questions

Requirements:

You should design and implement two classes called `MultipleChoiceQuestion` and `TrueAndFalseQuestion` that implement the `Question` interface to represent the two types of questions. You may create more supporting classes if appropriate. You should not convert the true and false questions to a special case of multiple-choice questions. These classes should be in the package `au.edu.uow.QuestionLibrary`.

Task 5: Creation of the API Documentation

Requirements:

You should use the `javadoc` tool to produce the API documentation of your classes. You should have made necessary documentation comments while you code your program. All the documents should go into a subdirectory called `doc` in the directory where the Java quiz application is. You should produce documentation for both public and private methods and include your name as the author. The Java API documentation would be a good reference for you to determine what and how much information should be documented for your classes, methods and class fields.

You can produce the documentation by issuing the following command in the directory where the Java quiz application is:

```
javadoc -private -tag author:a:"Author:" -d doc au.edu.uow.UserInterface au.edu.uow.QuestionLibrary
```

Submission

- The **`@author`** line in all your source code files should include **your Subject Code**, **your name**, **student ID**, and **Unix login name**.
- IMPORTANT: Any submission that cannot be decompressed may receive zero mark. Any submission that resulted in compiling errors may have 50% marks deducted.
- You should first zip your files including subdirectory paths into a file called **`a1.zip`**. For this assignment you submit all your programs including the Java quiz application, Question interface and the generated API documentation, basically all files in the working directory and subdirectories **excluding** files that may be generated by the IDE you might have used. Failing to do so may result in deduction of marks.
- Submit the file from your University Unix account on Banshee (`banshee.uow.edu.au`) as follows:

  ```
  turnin -c csci213 -a a1 a1.zip
  ```

or, if you submit it after the deadline:

```
turnin -c csci213 -a a1-late a1.zip
```

- Use the following command to check the submitted file:

  ```
  turnout -c csci213 -a a1
  ```

  or, if you submit it after the deadline:

  ```
  turnout -c csci213 -a a1-late
  ```

Marking Scheme

Mark to 0.5 divisions.

For CSCI213/MCS9213/CSCI813 students

General requirements	-0.5 ~ -1.0 mark for each error
Task 1	1
Task 2	3
Task 3	3
Task 4	2
Task 5	1

Appendix A: Java Quiz Application Program

(This program is available for downloading)

You should not modify this program. Instead, you should create your classes to support the program.

```
import java.util.List;

import au.edu.uow.QuestionLibrary.*;
import au.edu.uow.UserInterface.*;

/**
 * This is the application that tests students with Java quiz. <br>
 * This program is provided for the CSCI213 Assignment. <br>
 * Note: You should not modify this program. This program will use
 *       classes you created as instructed in the Assignment paper.
 *
 * @author Lei Ye
 *
 */
public class JavaQuiz {

        /**
         * This constant specifies the number of questions in each quiz.
         */
        final private static int NoOfQuestions = 5;

        /**
         * This is the entry point of the application
         * @param args Command line options.
         */
        public static void main(String[] args){

                if(args.length == 1){

                        /* You should create the QuestionLibrary class */
                        boolean isQuestionLibraryReady = QuestionLibrary.buildLibrary(args[0]);
                        if (isQuestionLibraryReady == true) {
                                System.out.println("===== Java Quiz, v1.0 =====\n");
                                /* You should create the UserInterface class */
                                UserInterface ui = new UserInterface();

                                System.out.println("-- Student login --");
                                /* You should create the Student class */
                                Student student = ui.getStudent();

                                System.out.println("\n-- Quiz begins --");
                                List<Question> quiz = QuestionLibrary.makeQuiz(NoOfQuestions);
                                ui.startQuiz(quiz, student);

                                System.out.println("\n-- Student marks  --");
                                ui.showStudentMarks(student);

                        }else{
                               System.err.println("Error: failed to build a question library. Exiting ...");
                               System.exit(0);
                        }
                }else{
                        System.err.println("Usage: java JavaQuiz questionFileName");
                }
        }
}
```

Appendix B: Question Interface

(This program is available for downloading)

You should not modify this interface declaration. Instead, you should create your classes to implement it.

```
package au.edu.uow.QuestionLibrary;

import java.util.List;

/**
 * This interface specifies the requirements of implementation classes.<br>
 *
 * This program is provided for the CSCI213 Assignment. <br>
 * Note: You should not modify this program but create suitable
 *       classes to implement this interface.
 *
 * @author Lei Ye
 *
 */
public interface Question {

	/**
	 * This method returns the question text.
	 * @return The question text in a list
	 * @see #getChoices()
	 * @see #compareAnswer(int)
	 */
	List<String> getQuestion();

	/**
	 * This method returns the multiple choices.
	 * @return The list of choices
	 * @see #getQuestion()
	 * @see #compareAnswer(int)
	 */
	List<String> getChoices();

	/**
	 * This method compares the student's answer to the standard answer.
	 * @see #getQuestion()
	 * @see #getChoices()
	 * @param ans The student's answer
	 * @return True for the correct answer; false for incorrect answers.
	 */
	boolean compareAnswer(int ans);
}
```

Appendix C: A Sample Screen Shot of Running the Java Quiz Application

```
===== Java Quiz, v1.0 =====

-- Student login --
Your name:Lei

-- Quiz begins --

Question No 1:

All method declarations contain arguments.

Answer Choices:
1: True
2: False

Choose your answer:2

Question No 2:

Which of the following statements is correct?

Answer Choices:
1: A class in Java can extend one or more classes
2: A class in Java can extend one class and implement one or more interfaces
3: A class in Java can extend one interface and implement one or two classes
4: A class in Java can extend one class and implement one interface

Choose your answer:2

Question No 3:

In Java, the concept of keeping data private is known as _______.

Answer Choices:
1: polymorphism
2: information hiding
3: data deception
4: concealing fields

Choose your answer:2

. . .


-- Student marks  --
Result of Lei: 4 out of 5
```

Note: the texts in the green colour are user inputs.

Appendix D: Question Library File Format

All questions and their sections in the question library document are marked with start-tags and matching end-tags, a format conforming to XML 1.0 Specification.

A XML document is essentially a marked up string of characters, which can be read as a text file.

The root tag for the document is `JavaQuestions`. The multiple choice questions, tagged as `MQuestion`, have three sections that are `question`, `choices` and `answer` while the true and false questions, tagged as `TFQuestion`, have two sections that are `question` and `answer`. Sections may appear in any order and each section may have different numbers of lines or choices.

You should not change the format. A sample question file, called `questions.xml`, is provided for your conformance test.

A segment of an example question file is shown below.

```
<?xml version="1.0"?>
<JavaQuestions>
<MQuestion>
<question>
Which of the following statements is correct?
</question>
<choices>
A class in Java can extend one or more classes
A class in Java can extend one class and implement one or more interfaces
A class in Java can extend one interface and implement one or two classes
A class in Java can extend one class and implement one interface
</choices>
<answer>
2
</answer>
</MQuestion>
<TFQuestion>
<question>
Superclasses can contain abstract methods.
</question>
<answer>
true
</answer>
</TFQuestion>

. . .

</JavaQuestions>
```