SITACS

School of Information Technology & Computer Science

Spring 2011
CSCI213 — Java Programming and Applications

Assignment 2 (10 marks)

Due Time and Date:

23:59 on Sunday, 4 September

Aims

This assignment requires you to design and implement a class library 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 a package 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 framework in 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 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 in a package, named au.edu.uow.csci213quiz.

An example 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 implement your classes by making use of features of the 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: Preparation of Quiz Question Library

Requirements:

You should prepare the quiz question library according to the format given in the Appendix D. You should not change the formats of the files in which the questions are provided. The quiz question library should contain two types of questions, multiple-choice questions and true and false questions, which are provided in two separate text files. Each of them should have at least 5 questions. You may use any questions in any Java books or on the Internet. The questions should cover reasonable range of Java programming topics.

Task 2: 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. You need analyse the requirements of this class by reading the Java quiz application and the example screen shot of running the application provided in Appendices A and C.

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 write a small application program to test this class separately.

Task 3: Design and Implementation of the QuestionLibrary Class and Other Supporting Classes

Requirements:

You should design and implement a QuestionLibrary Class to support the Java quiz application to provide questions and answer choices, and compare the student's answer with the standard answer in the quiz question library. The QuestionLibrary class should provide both two types of questions. The QuestionLibrary class should provide the Java quiz application with randomly selected questions of mixed types from the quiz library for each quiz session without repeating the same questions. It should also provide the quiz questions randomly for every session. All the questions should be loaded from the two question library files.

You must design and implement two classes to represent the two types of questions separately. You should not convert the true and false questions to a special case of multiple-choice questions. You may create more supporting classes if appropriate.

Some of your classes need to implement the Question interface that is given in the Appendix B.

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 4: Packaging of Your Classes

Requirements:

You should group your classes, the Question interface and the Student class into a package named au.edu.uow.csci213quiz.

Suggestion: You can do this while you develop your classes or you can do this after you have finished the development.

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.csci213quiz
```

Submission

- The **`@author`** line in all your source code files should include **your name**, **student ID**, and **Unix login name**.
- IMPORTANT: Any submission that can not 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 called **`a2.zip`**. For this assignment you submit all your programs including the Java quiz application,

Question interface, question library files and 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, say on Wumpus (wumpus.uow.edu.au) or Banshee (banshee.uow.edu.au) as follows:

```
turnin -c csci213 -a a2 a2.zip
```

- Use the following command to check the submitted file:

```
turnout -c csci213 -a a2
```

- NOTE: You should make sure a working version of your zipped file can be obtained by the sequence of the following Unix commands:

```
unzip a2.zip
javac JavaQuiz.java
java JavaQuiz
```

Marking Scheme

This information is primarily for the tutors who will be marking the assignment; but some of you might be interested.

Mark to 0.5 divisions.

You will receive an email from your tutor providing you with the marking result.

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

Appendix A: Java Quiz Application Program

(This program is available on eLearning for downloading)

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

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import au.edu.uow.csci213quiz.*;

/**
 * This is the application that tests students with Java quiz.
 * This program is provided for the CSCI213 Assignment 2.
 * Note: You should not modify this program. This program will use
 *       classes you created in the package au.edu.uow.csci213quiz
 *       as instructed in the Assignment 2 paper.
 *
 * @author Lei Ye
 *
 */
public class JavaQuiz {

    /**
     * This constant specifies the number of questions in each quiz.
     * It should not be larger than the total number of questions in
     * the question library.
     */
    final private static int NoOfQuestions = 5;
    /**
     * This is the buffered reader required to receive user answers
     */
    private static BufferedReader breader;

    /**
     * This is the entry point of the application
     * @param args Command line options. Not used in this assignment
     * @throws IOException The exception for errors from reading user answers
     */
    public static void main(String[] args) throws IOException{

        //Preparing to receive user answers
        breader = new BufferedReader(new InputStreamReader(System.in));

        //Getting student details
        /* You should build the Student class */
        Student student = new Student();
        System.out.println("What is your name?");
        String name = breader.readLine();;
        student.setName(name);

        //Creating the question library
        /* You should build the QuestionLibrary class */
        Question question = new QuestionLibrary();

        // Variable to hold the student answer
        int ans=0;
        //The quiz starts here
        for(int i=0; i< NoOfQuestions; i++){

        //Printing the question and choices
        System.out.println("\nQuestion No "+ (i+1) +":");
        question.printQuestion();
        System.out.println("\nAnswer Choices:");
        question.printChoices();
        //Getting the answer
        System.out.println("\nChoose your answer number:");
        ans = Integer.parseInt(breader.readLine());
        //Recording the student answer
        student.recordScore(question.compareAnswer(ans));

        //Resetting the answer variable.
        /* It's a good practice to do so. */
        ans = 0;
    }

    //Printing the quiz result
    System.out.println("\n\nResult of "
                       + student.getName()
                       +": " + student.getScore()
                       +" out of "
                       + NoOfQuestions);
    }
}
```

Appendix B: Question Interface

(This program is available on eLearning for downloading)

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

```
package au.edu.uow.csci213quiz;

/**
 * This interface specifies the requirements of implementation classes.<br>
 * The following methods should be used in sequence.<br>
 * <UL>
 *     <LI>void printQuestion()</LI><br>
 *     <LI>void printChoices()</LI><br>
 *     <LI>boolean compareAnswer(int)</LI>
 * </UL>
 * @author Lei Ye
 *
 */
public interface Question {

    /**
     * This method print a question.
     * @see printChoices()
     * @see compareAnswer(int)
     */
    void printQuestion();

    /**
     * This method prints the choices of the answer.
     * @see printQuestion()
     * @see compareAnswer(int)
     */
    void printChoices();

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

Appendix C: An Example Screen Shot of Running the Java Quiz Application

```
What is your name?
John Citizen

Question No 1:
Superclasses contains more features than their subclasses.

Answer Choices:
1: True
2: False

Choose your answer number:
2

Question No 2:
Analyze the following code:
public class Test {
    private int t;

    public static void main(String[] args) {
        int x;
        System.out.println(t);
    }
}

Answer Choices:
1: The variable t is not initialized and therefore causes errors
2: The variable t is private and therefore cannot be accessed in the main method
3: t is non-static and it cannot be referenced in a static context in the main method.
4: The variable x is not initialized and therefore causes errors.
5: The program compiles and runs fine.

Choose your answer number:
3

…

Result of John Citizen: 4 out of 5
```

Appendix D: Question Library File Formats

For multiple questions, each question consists of three sections. The question section starts with the tag <question>, the choices section with the tag <choices> and the standard answer section with the tag <answer>. For true and false questions, each question consists of two sections. The question section starts with the tag <question> and the standard answer section with the tag <answer>.

Questions in the library should have various numbers of lines and choices.

You should not change the formats of the question sections and files.

An example of formatting the question in the file is shown as follows.

Question section

```
Analyze the following code:
public class Test {
    private int t;

    public static void main(String[] args) {
        int x;
        System.out.println(t);
    }
}
```

Answer Choices section

```
1: The variable t is not initialized and therefore causes errors
2: The variable t is private and therefore cannot be accessed in the main method
3: t is non-static and it cannot be referenced in a static context in the main method.
4: The variable x is not initialized and therefore causes errors.
5: The program compiles and runs fine.
```

```
<question>
Analyze the following code:
public class Test {
    private int t;

    public static void main(String[] args) {
        int x;
        System.out.println(t);
    }
}
<choice>
The variable t is not initialized and therefore causes errors
The variable t is private and therefore cannot be accessed in the main method
t is non-static and it cannot be referenced in a static context in the main method.
The variable x is not initialized and therefore causes errors.
The program compiles and runs fine.
<answer>
3
```

This is the standard answer

The following shows how the questions are arranged in the file. They are simply put together one by one.

Multiple Choice Question Format

```
<question>
MQuestion 1 l1
MQuestion 1 l2
MQuestion 1 l3
MQuestion 1 l4
<choice>
MQuestion 1 choice 1
MQuestion 1 choice 2
MQuestion 1 choice 3
MQuestion 1 choice 4
<answer>
3
<question>
MQuestion 2 l1
MQuestion 2 l2
<choice>
MQuestion 2 choice 1
MQuestion 2 choice 2
MQuestion 2 choice 3
MQuestion 2 choice 4
<answer>
2
<question>
MQuestion 3 l1
MQuestion 3 l2
MQuestion 3 l3
<choice>
MQuestion 3 choice 1
MQuestion 3 choice 2
MQuestion 3 choice 3
<answer>
3

…
```

True and False Question Format

```
<question>
FQuestion 1 l1
FQuestion 1 l2
FQuestion 1 l3
<answer>
true
<question>
FQuestion 2 l1
FQuestion 2 l2
FQuestion 2 l3
<answer>
false
<question>
TQuestion 3 l1
TQuestion 3 l2
TQuestion 3 l3
TQuestion 3 l4
<answer>
true
<question>
FQuestion 4 l1
FQuestion 4 l2
<answer>
false
<question>
TQuestion 5 l1
TQuestion 5 l2
TQuestion 5 l3
TQuestion 5 l4
<answer>
true

…
```