University of London International Programmes

Computing and Information Systems/Creative Computing

CO1109 Introduction to Java and object-oriented programming

Coursework assignment 1 2017–18

Introduction

This is Coursework assignment 1 (of two coursework assignments) for 2017–18. The assignment asks that you demonstrate an understanding of random numbers, variable types (inluding *ints*, *Strings, boolean* and *Arrays*), static class variables, user input with the *Scanner* class, the *if – else* statement, loops and static methods with their return types.

Electronic files you should have:

Part a

- *PartA.java*

Part b

- *CustomerVerifier.java*

What you should hand in: very important

There is one mark allocated for handing in uncompressed files – that is, students who hand in zipped or .tar files or any other form of compressed files can only score 49/50 marks.

There is one mark allocated for handing in just the .java files asked for, without putting them in a directory; students who upload their files in directories can only achieve 49/50 marks.

At the end of each section there is a list of files to be handed in – **please note these hand-in requirements supersede the generic coursework instructions**. Please make sure that you give in **electronic versions** of your .java files since you cannot gain any marks without handing in the .**java** files asked for. Class files are **not** needed, and any student giving in only a class file will not receive any marks for that part of the coursework assignment, **so please be careful about what you upload as you could fail if you submit incorrectly**.

Programs that do not compile will not receive any marks.

The examiners will compile and run your Java programs; students who hand in files containing Java classes that cannot be compiled (*e.g*. PDFs) will not be given any marks for that part of the assignment.

Please put your name and student number as a comment at the top of each .java file that you hand in.

Part A

Consider the *PartA* class. Compile and run the program. You may find it helpful to know that CTRL + 'c' will terminate a program before it has completed running. Each of the methods in the *PartA* class has a comment explaining what it is that the method is intended to do. None of the methods work completely as intended.

Please complete the following task:

Directly under the comment at the top of each method, write three more comments:

- The first comment should start with “PROBLEM:” and it should detail what the method is doing that it should not be doing, or conversely, what the method is not doing that it should be doing.
- The second comment should start “REASON:” and should explain why the method has the problem described in the first comment.
- The third comment should start “SOLUTION:” and should explain how the method can be corrected so that it works as intended.

You will find that for some methods you will only need to write a sentence or two for each comment, others may take more work to document. [18 marks]

You may wish to correct the methods in the PartA class in order to better understand the problem. While this is a sensible strategy, ***no marks*** *will be given by the examiners for corrections to Java code. Only your comments will be marked.*

Example: given the following method:

```
//loop 4 times
private static void loopX() {
      for(int i = 0; i != 10; i += 3) {
            System.out.println("In loop "+i);
      }
      System.out.println("Out of loop");
}
```

A good answer might look like the following:

```
//loop 4 times

//PROBLEM: Infinite loop.

/*REASON: The loop starts when i is zero, and 3 is added to i
at each iteration, meaning that i can only take a value that is
divided by 3 (i = 0, 3, 6, 9, 12, 15,...). The loop will
continue while i does not equal 10. Since 10 is not divided by
3 i can never equal 10, hence the stopping condition (i != 10)
cannot be met. */
```

```
/*SOLUTION: make the stopping condition (1) a value that i can
take and (2) a value that will mean 4 iterations only. i!=12;
will satisfy these two conditions.*/
```

Reading for part A (all from Volume one of the subject guide)

- Section 2.7.2 (writing a comment over more than one line)
- Section 5.6 (random numbers)
- Chapter 8 (simple loops)
- Chapter 11 (nested loops).

Deliverables for part A

Electronic file of your documented program: *PartA.java*

Part B

You have been given the *CustomerVerifier* class, which contains four class variables, and the following static methods:

- `boolean askUserToContinue()`
- `String getCustomerFromUser() {`
- `int getPinFromUser()}`
- `String getUserInput(String)`
- `boolean isValidPin(String, int)`
- `boolean isValidCustomer(String)`
- `int[] getDiscreteRandomInts(int, int)`
- `String charsAt(String, int[])`
- `String getMemorableWordCharsFromUser(int[]) {`
- `String getMemorableWord(String)`
- `void verifiedCustomer(String, int, String)`
- `void incorrectPin(String, int)`
- `void invalidMemorableWord(String)`
- `void invalidCustomer(String)`

In this part of the coursework assignment you are required write two additional methods for the *CustomerVerifier* class, plus a main method.

Class variables
The *CustomerVerifier* class uses class variables. Class variables are variables that (among other things) can be accessed and updated by any static method in the class. They are declared outside of a method. The arrays in the *CustomerVerifier* class are examples of class variables – note that they are declared and initialised inside the class, but outside of all of the methods. Class variables can be accessed by static methods within the class without needing to be included in the method's parameter list.

The *CustomerVerifier* class
The *CustomerVerifier* class has methods that are intended to be used in a two-stage verification process. Firstly, the user inputs the customer name, and this is checked against the customer pin. If this is correct, then the class will go on to ask for two characters from the user's memorable word.

Note for maximum marks you are asked to write a `boolean` method, *userKnowsRandomCharsFromMemorableWord(String) a*nd a `void` method *verify()*. Once you have answered questions 1 and 2, you should find that you have used every method in the *CustomerVerifier* class as given to you, in either the *verify()* or the *userKnowsRandomCharsFromMemorableWord (String)* method.

1. a. Write a `boolean` method, *userKnowsRandomCharsFromMemorableWord(String).* This method should ask the user for 2 different random characters from the current customer's memorable word. The customer's memorable word is the word stored in the *memorableWords* array, at the same index as the customer's name in the *customer* array. The method should return *true* if both characters given by the user match the memorable word characters asked for, and *false* otherwise. [5 marks]

   b. It is possible to write the above method with three statements incorporating calls to existing methods in the *CustomerVerifier* class as given to you, plus one or two further statement(s) to return a value. A single return statement is possible with the right choice of `String` method. You do not have to write your method in this way, but if you do not then the most you can get for answering question 1, is 5 marks out of 10. [5 marks]

2. a. Write a *Verify()* method to implement the following algorithm:
    1. Ask for the customer name.
    2. Check to see if the customer name is valid, *i.e*. is in the array of customers.
        a. If the customer name is ***not*** in the array, tell the user that the customer is not valid, and ask if they wish to validate another customer.
            i. If yes, back to step 1.
            ii. If no, end the loop.
        b. If the customer name is in the array move to step 3
    3. Ask for the customer's PIN.
    4. Check if the PIN is valid, *i.e.* is it in the *pins* array, at the same index as the name is in the *customers* array.
        a. If the pin is ***not*** valid then tell the user the customer is not valid, and ask if they want to verify another customer.
            i. If yes, back to step 1.
            ii. If no, end the loop.
        b. If the pin is valid move to step 5.
    5. Ask for two random characters from the customer's memorable word.
    6. Check to see if the characters from the memorable word are valid. Note that the customer's memorable word is in the same indexed location in the *memorableWords* array, as the customer name is in the *customers* array.
        a. If the memorable word is ***not*** valid then tell the user the customer is not valid, and ask if they want to verify another customer.
            i. If yes, back to step 1.
            ii. If no, end the loop.
        b. If the memorable word is valid move to step 7.
    7. Tell the user that the customer has been verified, and ask the user if they want to verify another customer.
            i. If yes, back to step 1.
            ii. If no, end the loop.

[10 marks]

b. The *verify()* method should make use of the methods that exist in the *CustomerVerifier* class as given to you, plus the *userKnowsRandomCharsFromMemorableWord(String)* method that you have written. You should not need to write any other methods, or to write any statements that do not include method calls to methods within the *CustomerVerifier* class with one single exception: you may wish to write a goodbye message to the user when leaving the loop.

The above means that every statement in the *verify()* method should incorporate either a call to a method already in the *CustomerVerifier* class as given to you, or a call to your *userKnowsRandomCharsFromMemorableWord(String)* method, except for the optional goodbye message.

[8 marks]

If you do not write your *verify()* method as described above, you can achieve at most 10 marks for this question.

3. Write a main method in your *CustomerVerifier* class. The main method should run the program so that you, and the examiner, can test your class.

   *Note that this assignment **does not ask** for any verification of data entered by the user, hence, if the PIN entered by the user cannot be parsed to an `int`, a run-time exception will be thrown. There are no marks in this assignment for verifying user input.* [2 marks]

Reading for part b

- Look at the Java String API, for a method to use in the return statement of question 1: https://docs.oracle.com/javase/7/docs/api/java/lang/String.html
- Volume one of the subject guide, paying particular attention to the chapters and sections noted below:
  - Chapter 5, *Calling methods,* in particular:
    - Sections 5.1 to 5.5 (*calling methods*)
    - Section 5.6 *Some Simple Methods for Random Numbers*
    - Section 5.8 *Method Signatures*
  - Chapter 6 user input with the *Scanner* class
  - Chapter 7 *Boolean Expressions and Conditional Statements* (*i.e.* the *if – else* statement)
  - Chapter 9, *More on calling methods,* in particular:
    - Sections 9.1 - 9.3.3 (*calling methods, void and typed methods*)
    - Section 9.4.3 *String.length()*
    - Section 9.4.4 *charAt()*
    - Section 9.5 *type checking*
  - Chapter 12 *Defining Your Own Methods,* sections 12.1 – 12.13 and ***in particular*** sections 12.12 and 12.13 (about void and typed methods)

Deliverable for part b

- An electronic copy of your revised program: *CustomerVerifier.java.*

Marks for CO1109 Coursework assignment 1

The marks for each section of Coursework assignment 1 are clearly displayed against each question and add up to 48. There are another two marks available for giving in uncompressed .Java files and for giving in files that are not contained in a directory. This amounts to 50 marks altogether. There will be a further 50 marks available for Coursework assignment 2.

Total marks for part A	[18 marks]
Total marks for part B	[30 marks]
Mark for giving in uncompressed files	[1 mark]
Mark for giving in standalone files; namely, files **not** enclosed in a directory	[1 mark]
Total marks for Coursework assignment 1	**[50 marks]**

[END OF COURSEWORK ASSIGNMENT 1]