Computing and Information Systems/Creative Computing
University of London International Programmes
CO1109 Introduction to Java and object-oriented programming
Coursework assignment 1 2016–17

Introduction

This is Coursework assignment 1 (of two coursework assignments total) for 2016–17. 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

- *PartB.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 the hand-in requirements supersede the generic University of London 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 examiner intends to compile and run your Java programs. For this reason, students who hand in files containing their Java classes that cannot be compiled (*e.g.* PDFs) will not be given any marks for that part of the coursework assignment.

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

CO1109 Coursework assignment 1

PART A

Consider the *PartA* class. The version that you have been given will not compile, as its main method attempts to call static methods that do not exist in the class. Printed in the appendix (see the final page of this document) is an example output of the completed *PartA* program. It is your job to write the methods that will make the statements in the main method work, the program as a whole compile, and will give similar output as that given in the appendix. Please note that you should not change any of the statements in the main method and, if you do, you will not receive any marks for that part of the question.

Please complete the following tasks:

1. Write the *censor(String)* method. [3 marks]

2. Write the *print(String)* method. [3 marks]

3. Write the *magic8Ball()* method.

   Note that the *magic8Ball()* method is returning a *String* chosen at random from the *answers* array. For this reason, the output of the *for* loop in the main method should be 20 members of the *answers* array, each on its own line. Since each array item is randomly chosen, the output that you get is likely to be different to the output given in the appendix. However, it should be the same in the sense that it should comprise 20 separate lines, each one containing an item randomly chosen from the *answers* array. [3 marks]

4. Write the *drawStars(int, int)* method. [3 marks]

5. Write the *print(String[ ])* method. [3 marks]

6. The *for* loop in the main method has been written to test the *magic8Ball()* method. In your *PartA* class write a comment above or below the *for* loop to explain what it is that the test is looking for – *i.e.* what output would constitute a successful test?

   Note that your comment should be no longer than a paragraph, and that a paragraph has **at most** 8 sentences. [3 marks]

Reading for part A

Volume 1 of the CO1109 subject guide, and in particular:

- Section 2.7.2 (writing a comment over more than one line)
- Section 2.9 *print* ***vs.*** *println*
- Chapter 5, sections 5.1 to 5.5 (calling methods)
- Section 5.6 *Some Simple Methods for Random Numbers*
- Section 5.8 *Method Signatures*

- Section 8.4 *Syntax of for Loops*
- Section 9.4.3 (*String.length()*)
- Chapter 9 *More on Calling Methods* sections 9.1–9.4.1
- Section 9.5 *Type Checking*
- Section 9.6 *Parsing Strings that Represent Integers*
- Section 9.7 *Method Overloading*
- Chapter 10 *One-Dimensional Arrays*
- Chapter 11 *Nested Loops*
- Chapter 12 *Defining Your Own Methods*

Deliverable for part A

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

PART B

In this part of the coursework assignment you are required to answer a question about the *PartB* class.

Consider the program *PartB.java*. Firstly, compile and run it as it is. Secondly, delete the comment marks from the `scanner.nextLine();` statement, re-compile and run the program a second time. Can you explain the different behaviour of the program with and without the `scanner.nextLine();` statement?

Write your answer as a comment in the *PartB* class. You should write no more than a paragraph. Note that a paragraph has *at most* 8 sentences. [5 marks]

Reading for part B

You should read sections 6.1–6.5.1 (user input with the *Scanner* class) of volume 1 of the CO1109 subject guide. Remember you can also read the Java API and search online for help.

Java API for the Scanner class:
https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

Deliverable for part B

- An electronic copy of *PartB.java (with a comment explaining its behaviour).*

PART C

In this part of the coursework assignment you are required to write the class *TaxAdvisor.* The *TaxAdvisor* class asks the user for certain information and outputs the yearly tax breakdown for a given salary.

Class variables

Your *TaxAdvisor* class should use (static) 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 *answers* array in the *PartA* class is an example of a class variable – note that it is declared inside the class, but outside of all of the methods. Class variables can be accessed by static methods without needing to be included in the method's parameter list.

The BigDecimal class

In real world calculations with currency, Java's *BigDecimal* class would be used for its accuracy. You may use the *BigDecimal* class if you wish; however, there are no marks for doing so.

Rules for calculating tax on a worker's salary

The *TaxAdvisor* class should make use of the following simplified tax band and personal allowance rules for calculating the tax to be paid on the salary of British workers.

Band	***Taxable income***	***Tax rate***
Personal allowance	£X (can vary)	0%
Basic rate	£X.01 to £43,000	20%
Higher rate	£43,000.01 to £150,000	40%
Additional rate	over £150,000	45%

Personal allowance

1. Every worker earning under a certain amount gets a personal allowance on which they do not pay tax.
2. The basic personal allowance is £11,000.
3. To the basic personal allowance the following can be added as appropriate:
   - Additional personal allowance for registered blind people: £2,290 per year.
   - Additional personal allowance for laundry of work clothes for manual workers and those who have to wear a uniform for work: £60 per year.
   - Additional personal allowance for those who work exclusively from home: £200 per year.
4. Any worker whose salary is £122,000 a year or more does not receive any personal allowance including additional personal allowances.

Output of the TaxAdvisor class
Below are three examples of output from the *TaxAdvisor* class:

1. The user earns £160,000:

```
Your salary is 160000
Your tax allowance is: 0
You owe 8600 in the first tax band (20% tax) paid on taxable
income up to 43,000
You owe 42800 in the second tax band (40% tax) paid on
taxable income above 43,000 and up to 150,000
You owe 4500 in the third tax band (45% tax) paid on taxable
income over 150,000
Total tax owed on your salary = 55900
```

2. The user earns £45,117 and is a registered blind person:

```
Your salary is 45117
Your tax allowance is: 13290
You owe 5942.0 in the first tax band (20% tax) paid on
taxable income up to 43,000
You owe 846.80 in the second tax band (40% tax) paid on
taxable income above 43,000 and up to 150,000
You owe 0 in the third tax band (45% tax) paid on taxable
income over 150,000
Total tax owed on your salary = 6788.80
```

3. The user earns £32,592 and wears a uniform to work:

```
Your salary is 32592
Your tax allowance is: 11060
You owe 4306.40 in the first tax band (20% tax) paid on
taxable income up to 43,000
You owe 0 in the second tax band (40% tax) paid on taxable
income above 43,000 and up to 150,000
You owe 0 in the third tax band (45% tax) paid on taxable
income over 150,000
Total tax owed on your salary = 4306.40
```

Please complete the following tasks:

---

1. Write the *TaxAdvisor* class to implement the above rules for determining how much tax a worker should pay on their yearly salary, and to tell the worker their personal allowance and tax breakdown. Your class should be divided up into methods. You may write any additional methods that you wish, but your class **must have** the following **six** methods:

    1. A method to ask the user for their yearly salary and ask further questions to determine the user's eligibility for additional personal allowances. (3 marks)
    *You **do not** have to make sure that this method can recover from invalid input by the user, and can achieve full credit with a method that will end the program* (*i.e*. *throw an exception*) *if it receives invalid input.*
    2. A method to work out the user's personal allowance. (3 marks)
    3. A method to calculate the tax the user should pay in the basic tax band. (3 marks)
    4. A method to calculate the tax the user should pay in the higher rate tax band. (3 marks)
    5. A method to calculate the tax the user should pay in the additional rate tax band. (3 marks)
    6. A method to explain the user's tax breakdown to them. (3 marks).
    Please note that to run the TaxAdvisor class, only the method to explain the user's tax breakdown should need to be run. This method should itself call all other methods as necessary. [18 marks]

2. Make sure that your *TaxAdvisor* program does the following:
    - The user is only asked to input information once, *e.g*. once you have asked for the user's salary, you store it so that you do not need to ask for it again (1 mark).
    - The program uses class variables allowing the same information to be shared by different methods. (1 mark)
    - The user only has to call one method in order to run the entire program and receive a tax breakdown (1 mark).
    - The program should not have any unnecessary extra steps or obstacles for the user (1 mark). For example, if you are asking the user yes/no questions, your program should recognise any reasonable answer, *e.g.* 'y' or 'yes' or 'Y' or 'Yes' or 'yeah'. [4 marks]

3. In your program you should only write a block of code to do a particular job once; this reflects the software developer's maxim DRY: *Don't Repeat Yourself*. If you should find that you are writing the same block of code in more than one method, or repeating the same block of code in one particular method, then you are expected to put that code into its own method. There are 2 marks for not [2 marks]

repeating yourself. You can only get these marks if you have handed in an attempt at writing a complete *TaxAdvisor* class (it does not have to work perfectly, but it should compile).

4. Write a main method in your *TaxAdvisor* class. The main method should run the program so that you, and the examiner, can test your class. [1 mark]

Reading for part C

You should read Volume 1 of the CO1109 subject guide, paying particular attention to the topics noted below:

- Chapter 3 *Arithmetic Expressions* (including how to concatenate *Strings*)
- Chapter 6 *Keyboard Input* (User input with the *Scanner* class)
- Chapter 7 *Boolean Expressions and Conditional Statements* (*i.e*. the *if – else* statement)
- Sections 9.3 to 9.3.3 (calling methods, void and typed methods)
- Section 9.4.4 *charAt()*
- Section 9.6 *Parsing Strings that Represent Integers*
- Chapter 12 *Defining Your Own Methods*

Deliverable for part C

- An electronic copy of your program: *TaxAdvisor.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 are another 50 marks available from Coursework assignment 2.

Total marks for part a	[18 marks]
Total marks for part b	[5 marks]
Total marks for part c	[25 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]

Appendix: An example output of the completed PartA class

```
*****
***************
************************

Outlook unclear
Without a doubt
Concentrate and ask again
Yes
Yes
Without a doubt
Very doubtful
Yes
Outlook unclear
Concentrate and ask again
Outlook unclear
Don't count on it
Without a doubt
Yes
Concentrate and ask again
Concentrate and ask again
Outlook unclear
Yes
Don't count on it
Very doubtful

**********
**********
**********

********************
********************
********************
********************
********************
********************
********************
********************
********************
********************

******************************
******************************
******************************
******************************
******************************
******************************
```