Computing and Information Systems/Creative Computing
University of London International Programmes
CO1109 Introduction to Java and object-oriented programming
Coursework assignment one
2015–16

Introduction

This is Coursework assignment 1 of two for 2015–16. The coursework assignment asks that you demonstrate an understanding of static methods and return types, `Arrays`, random numbers (part a), and, in part b, static methods, variables (including `ints`, `Strings` and `boolean`), user input, the `if – else` statement and `while` loops.

Electronic files you should have:

Part a

- *Speak.java*

Part b

- *InvoiceCreator.java*
- *inv.txt*

Notes

Please do not hand in your entire directory structure, just hand in the files you are asked for. Also please do not compress your files. It takes time to go through the directory structure and to decompress files, and this time soon adds up when the examiner has to do it more than 200 times.

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.

Coursework assignment 1

PART A

Consider the class, *Speak.java*, below:

```
public class Speak{
  private static String[] words_1 = {"Mr Spock","Captain",
     "Bones","Scotty","Uhura","Sulu","Checkov"};
  private static String[] words_2 = {"your","my","their","the"};
  private static String[] words_3 = {"best friend","warp drive",
     "lunch","tribble","Klingon Ambassador","tricorder"};
  private static String[] words_4 = {"is life but not as we know
     it","is behind you","ate my hamster","has taken over the
     ship","cannae take much more of this","has run amok","is
     looking at me strangely","is decloaking"};
}
```

Make the following changes to the Speak class:

1.	Add a `Static` method of type `String`, that takes a `String` array and returns an element chosen at random from the array. Call your method *getRandomString().* Hint: if *a* is an array then *a.length* is the number of elements in *a.*	[4 marks]
2.	Write a second `Static String` method, *getRandomSentence()*, that uses your *getRandomString()* method to return a random sentence. The sentence should be composed by taking one element at random from the *words_1* array, followed by another randomly chosen element from *words_2,* followed by a third element chosen randomly from *words_3* and the last part of the sentence should be taken at random from *words_4*. This should produce random sentences such as: ◦ Mr Spock, your phaser ate my hamster. ◦ Uhura, my lunch cannae take much more of this. ◦ Captain, my tribble has taken over the ship. ◦ Scotty, their Klingon Ambassador is behind you.	[4 marks]
3.	Write a main method to test your *getRandomString()* method. Please note that the *Speak* class you hand in should have at least one test statement in the main method.	[4 marks]
4.	In your testing of the *getRandomString()* method, explain what you tested for, and how you fixed any errors that you found. You may write this as a comment in your *Speak* class. Please do not write more than two paragraphs (a paragraph is at most eight sentences) and please do not describe finding and fixing syntactical errors. See Volume 1 of the subject guide, Chapter 2, page 11, section 2.7.2, on how to write a comment over more than one line.	[4 marks]

Reading for part a.
Volume 1 of the subject guide, and in particular:

- Chapter 2, page 11, section 2.7.2.
- Chapter 5, sections 5.1 to 5.6 pages 39–40, and section 5.8 pages 42–43.
- Chapter 9.
- Chapter 10.
- Chapter 12.

Deliverable for part a.

- An electronic copy of your program: *Speak.java.* This should include an explanation of what you were looking for in your testing, what you found and how you fixed it (if necessary) written as a comment.

PART B

Consider the *InvoiceCreator* class. You will note that many of the statements in the *InvoiceCreator* class have been written as comments, so that they do not compile. This is because as statements they would cause compilation errors as they try to use methods that do not exist.

Compile and run the program. You should see the following output:

```
                Quality Employment Agency


INVOICE

24/10/15

Quality Employment Agency
10 Sixth Road
Lodgate
LO15 9EE

TO



JOB TITLE           HOURLY RATE    HOURS WORKED   AMOUNT

Total Hours                                       0
Total Before VAT                                  Â£0
VAT @ 20%                                         Â£0

=================================================================
TOTAL PAYABLE                                     Â£0
=================================================================


DETAILS
See attached time sheets.


PAYMENT
Make all cheques payable to Quality Employees Inc.
Thank you for your custom.


          Quality Employment Agency: where quality comes first

Tool completed successfully
```

NOTE: If the output you see has any strange characters such as “Â” ignore them as this would be an issue with character encoding which this coursework assignment is not covering.

Compare the above output with the file *inv.txt*. The task you have is to write methods such that the output of the *InvoiceCreator* program, given the right input, looks similar to the output in the *inv.txt* file. There are six methods for you to write. Once you have written your methods remove the comment marks “//” from statements, and the comment delimiters “/*” and “*/” around the `while` loop in the *create()* method. Compile and run the *InvoiceCreator* class again and test that the output is as it should be. You may wish to use the input data used to create the *inv.txt* file, in your testing. This data is:

name : LutherCorp

address : 16 Lodgate Hill Road
Lodgate
LO16 1TH

jobTitle	Receptionist	Janitor	Handyman
hourlyRate	17	15	20
hours	90	100	50

1.	Write the *getFromUser()* method. This method is used wherever you see the “******TASK ONE******” comment. You will see that it is used three times to get input from the user via the keyboard. Note that this method should work correctly in each of the three statements.	[5 marks]
2.	Write the *getAddress()* method. See the “******TASK TWO******” comment in the *create()* method.	[5 marks]
3.	Write the *getIntFromUser()* method. See the “******TASK THREE******” comments in the *create()* method. In this method make sure that the number entered by the user is greater than zero. If the number is zero or negative then tell the user they have made a mistake and ask them to re-enter the number. Keep doing this until the number entered is greater than zero.	[5 marks]
4.	Write the *getVat()* method. See the “******TASK FOUR*******” comment in the *printInvoice()* method.	[5 marks]
5.	Write the *addVat()* method. See the “******TASK FIVE*******” comment in the *printInvoice()* method.	[5 marks]
6.	Write the *getDateIn30Days()* method. See the “******TASK SIX*******” comment in the *printInvoice()* method.	[5 marks]
7.	Would `StringBuilder` have been a better choice than `String` for the variable *s* in the *printInvoice()* method? Explain why or why not in a paragraph (a paragraph has no more than eight sentences) included as	[2 marks]

a comment in your *InvoiceCreator* class. You can find out more about `StringBuilder` in the Java API: http://docs.oracle.com/javase/7/docs/api/java/lang/StringBuilder.html	

NOTE: You are only asked to write methods and add them to the *InvoiceCreator* class. Do not change any other part of the class, except for removing comment marks and delimiters.

Reading for part b.

You should read the following chapters of Volume 1 of the subject guide, paying particular attention to the topics noted below.

- Chapter 2 – defining `String` constants, difference between `print` and `println`.
- Chapter 3 – arithmetic expressions and how to concatenate `Strings`.
- Chapter 4 – variables.
- Chapter 5 – methods.
- Chapter 6 – keyboard input, the `Scanner` class, how to input `ints` from the keyboard.
- Chapter 7 – boolean expressions and conditionals (ie the `if-else` statement).
- Chapter 8 – `while` loops.
- Chapter 9 – calling methods, `static` methods, non-void (ie typed) methods

Remember you can also search online for help as well as reading the Java API.

Deliverable for part b.

- Electronic file of your amended program: *InvoiceCreator.java,* including a comment answering Question 7.

MARKS FOR 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 buried inside a directory structure. This makes 50 marks altogether. There are another 50 marks available from the Coursework assignment 2.

Total marks for part a.	[16 marks]
Total marks for part b.	[32 marks]
Mark for giving in uncompressed files	[1 mark]
Mark for giving in stand-alone files, ie files ***not*** enclosed in a directory	[1 mark]
Total marks for Coursework assignment 1	**[50 marks]**

[END OF COURSEWORK ASSIGNMENT 1]