CARDIFF UNIVERSITY

EXAMINATION PAPER

Academic Year:	**2019-2020**
Examination Period:	**Spring**
Examination Paper Number:	**CMT205**
Examination Paper Title:	**Object-Oriented Development with Java**
Duration:	**Three hours**

Structure of Examination Paper:

There are FIVE pages.
There are FOUR questions in total.
There are no appendices.
The maximum mark for the examination paper is 60 and the mark obtainable for a question or part of a question is shown in brackets alongside the question.

Instructions to Students:

Answer THREE questions.

Important note: if you answer more than the number of questions instructed, then answers will be marked in the order they appear only until the above instruction is met. Extra answers will be ignored. Clearly cancel any answers not intended for marking.

Question 1

(a) What are the *values* of the variables `r1`, `r2`, `r3` and `r4` after the following sequence of Java statements has been executed? [4 marks]

```
double s = 2.7;
double t = 3.2;
int i = 2;
double r1 = (int)s/i+t;
double r2 = s+i*t;
String r3 = s + "+" + (i*t);
StringBuffer buf = new StringBuffer(r3);
buf.insert(2, "$");
buf.replace(5, 7, "#");
String r4 = buf.toString();
```

(b) Write Java statements for the following tasks:

i. Given a `double` array `nums`, print the average of all the **positive** values in the array. You can assume that the array contains at least one positive number. [4 marks]

ii. Take an input from the keyboard as a `String`. If the content of the String corresponds to a **positive** integer that can be represented as an `int`, assign the value to an `int` variable `val`. You can assume the variable is already defined. Otherwise, repeat the process to take another input from the keyboard, until a correct value is assigned. [5 marks]

(c) Given a `String` array `accounts` containing all the student account names, and a `String` variable `query` holding the student account name to be checked, assign whether the account to be queried exists or not, to a `boolean` variable `found`.

i. Write Java statements to achieve the task, without using extra storage. [3 marks]

ii. To make queries efficient, it is possible to utilise a Java collection. Write Java statements for generating the collection from the given array, and using it to achieve the task. [4 marks]

[Question 1 Total: 20 marks]

Question 2

(a) What type of files (text files or binary files) is more suitable for storing numerical data (i.e. numbers) in the following scenarios. Justify your answer.

i. You have a large amount of numerical data to store and the priority is efficiency. [2 marks]
ii. You would like to make sure the file is easily readable and can be exchanged with other software. [2 marks]

(b) Complete the following program `Sum.java` that reads in an input text file, with the file name specified as the first command line argument. The text file has one or more lines, each line containing one or more `double` numbers, separated by a comma between every two numbers in the same line.

The program generates an output text file, with the file name specified as the second command line argument. The output text file will have the same number of lines as the input text file, with each line containing a single `double` number, which is the sum of all the numbers in the corresponding line.

Your program should check if the number of input arguments is correct (and print an appropriate message if incorrect), and handle exceptions related to file I/O. You can assume the input text file satisfies the required format. You can also assume relevant packages have been imported.

Example `input.txt`:

```
2, 3.5, 4.2
-1, -5, 3.1
```

Execute: `java Sum input.txt output.txt`

The generated `output.txt` contains:

```
9.7
-2.9
```

The following code is given and only the missing code needs to be provided.

```
import java.io.*;
public class Sum {
     public static void main(String[] args) {
          // TO DO: Complete the code here
     }
}
```

[16 marks]

[Question 2 Total: 20 marks]

Question 3

Write a Java program using multi-threading to simulate 100 meters sprint with 8 sprinters. You will implement a thread class SprinterThread that simulates a sprinter (with an ID of 0, 1, until 7), and at every time step (a random integer between 14ms and 18ms), it moves the sprinter forward by a random distance (for simplicity, an integer between 180cm and 250cm inclusive). A sprinter completes the game by running a distance of 10000cm. When a sprinter finishes, print the sprinter ID and the amount of time involved.

The main program SprinterSim simulates the competition between 8 sprinters. After all the sprinters finish, print the ID of the winner, i.e. the sprinter taking the minimum amount of time. You may pass on the sprinter ID to the sprinter thread object as a constructor parameter.

(a) The following code is provided for SprinterThread and only the missing code needs to be completed: [8 marks]

```
public class SprinterThread extends Thread
{
	private int ID;			// sprinter ID
	private int time = 0;		// sprinter running time

	public int getTime()
	{
		return time;
	}

	// TODO: Complete the program
}
```

(b) In addition, complete the missing code in the following simulation class SprinterSim: [12 marks]

```
public class SprinterSim
{
	public static void main(String[] args)
	{
		final int SPRINTER_NUM = 8;

		// TODO: Complete the program
	}
}
```

[Question 3 Total: 20 marks]

Question 4

(a) Assume you are developing a Java server program and you are writing a class NetworkMonitor that monitors all the network communication in your application. As only one instance of the class NetworkMonitor should be created, what design pattern will be useful in this scenario? Briefly give one approach to implement this design pattern (no more than three sentences).

[3 marks]

(b) Write a Java program TransServer that receives information of transactions, each containing the unit price and the number of units purchased, and sends back the running total (the total amount based on the transactions received). Your program should listen to incoming **TCP** connections at the port 8000. For simplicity only one connection needs to be dealt with at a time. Every time a new connection is made, the running total is set to 0. While the connection keeps alive, the client sends a line at a time, containing two double numbers, separated by blankspace(s). The first number gives the unit price and the second number gives the number of units purchased. The server should send back a line containing a single number, corresponding to the running total based on the transactions received so far.
You may assume that the content received from the client is always in the correct format but communication exceptions need to be handled properly. You can assume all the necessary classes from the Java standard library are imported.

Example:

Received:

1.5 3
4.8 2.5

Send back:

4.5
16.5

Here, 4.5 = 1.5 * 3, and 16.5 = 1.5 * 3 + 4.8 * 2.5.

The following skeleton program is provided and only the code to complete the program needs to be provided.

[17 marks]

```
import java.net.*;
import java.io.*;
import java.util.*;
public class TransServer
{
   public static void main()
   {
      // Create a Server Socket object
      ServerSocket sSock = null;
      try
      {
         sSock = new ServerSocket( 8000 );
      }
      catch( IOException e )
      {
         System.err.println(e);
         System.exit(1);
      }

      // TODO: Complete the program
   }
}
```

[Question 4 Total: 20 Marks]