**CARDIFF UNIVERSITY**
**EXAMINATION PAPER**

| | |
|---|---|
| **Academic Year:** | 2012/2013 |
| **Examination Period:** | Spring |
| **Examination Paper Number:** | CMT205 |
| **Examination Paper Title:** | Object Oriented Development with Java |
| **Duration:** | 2 hours |

**Do not turn this page over until instructed to do so by the Senior Invigilator.**

**Structure of Examination Paper:**

There are 4 pages.
There are 4 questions in total.

The maximum mark for the examination paper is 100 and the mark obtainable for a question or part of a question is shown in brackets alongside the question.

**Students to be provided with:**

The following items of stationery are to be provided:
ONE answer book.

**Instructions to Students:**

Answer ALL the questions.
Students are permitted to introduce to the examination any textbook, any printed / handwritten notes, and other similar materials. Use of annotations, highlighting and bookmarks is permitted.
The use of calculators is permitted in this examination.
The use of translation dictionaries between English or Welsh and a foreign language bearing an appropriate departmental stamp is permitted in this examination.

Q1. (a) What are the values of the variables `result1`, `result2`, `result3`, `result4`, `result5` after the following Java statements have been executed? [5]

```
int inum = 11;
double dnum = 3.7;
double result1, result2;
result1 = (double) inum / 4 + dnum;
decResult2 = (double)((1/3) * inum + (1/2) * dnum);
int result3;
result3 = (inum / 2) << 2;
String result4 = dnum + " " + inum;
StringBuffer buffer = new StringBuffer(result4);
buffer.replace(2, 5, "***");
String result5 = buffer.toString();
```

(b) Complete the following code to check if the command line contains exactly three arguments, with each being a string representing a properly formatted 32-bit integer. Output "Pass" if the command line arguments are correct and "Fail" otherwise. [9]

```
public class InputTest
{
    public static void main (String [] args)
    {
        // TODO: Complete the code here
    }
}
```

(c) Draw a circle in **red** with centre at (120, 150) and a radius of 50, given a Graphics object `g`. [3]

(d) Suppose that you are writing a class `ConsoleOutput` that keeps track of information printed in the console window. There is only one console window available and only one instance of the class should be created. What design pattern will be useful in this scenario? Briefly give one approach to implement this design pattern (up to three sentences). [3]

(e) Assume that you have a class `TestThread` that has the folllowing code structure:

```
public class TestThread extends Thread
{
    public void run()
    {
        // code omitted for simplicity
    }
}
```

i. Rewrite the class `TestThread` such that `TestThread` still contains the code that runs in a separate thread but also needs to be inherited from another class `GeneralTest`.

ii. Write the Java statements to create an instance of the thread for the rewritten class, and start running it.

[5]

Q2. (a) What is the *Model-View-Controller (MVC)* paradigm and why is it important to Graphical User Interface (GUI) Design? Briefly describe the function of *each* of the MVC *units*. [5]

(b) How do the MVC Architectures in *Classic GUI Design* and Java Swing *differ*? What is the reason for the difference? [4]

(c) What are the *main features* of *Java Swing Components*? What is meant by the term *lightweight components* in relation to Swing? [4]

(d) For most applications the *default* Look-and-Feel libraries provided by Swing are adequate. However, there may be some occasions when a customised Look-and-Feel is more appropriate.

How is *customised rendering* of the look and feel of Swing Components facilitated by the MVC architecture?

Give two such situations with reasons why a *customised* Look-and-Feel should be adopted. Describe how you would achieve this with *fragments* of JAVA Swing code [7]

(e) What are *actions* in relation to Java Swing Components? Give an *example application* where actions might typically be used. [5]

Q3. (a) Suppose we wish to store employee records in a structured binary random access file such that a record can be updated (or added/deleted):

i. Suggest an advantage of using a random access file for this purpose. What are the disadvantages of using this type of file? [3]

ii. Assume a record includes the following fields (name and type)
`employeeID: int` (the ID of the employee)
`name: String` (the full name, with up to 50 characters)
`salary: double` (the salary)
What is the space (number of bytes) each field takes on the disk? [3]

(b) Given a text file '`document.txt`', complete the following Java method to count how many times each character of interest appears in the text file. The method takes one argument of type `String` which contains characters of interest (in the order as they appear in the String). If there are $n$ characters of interest, the output to the console should involve $n$ lines, with each line of the form

`<character>: <count>`

where `<character>` and `<count>` are replaced with the actual character/number. Your code should only read the text file once and the text file/user input may contain general Unicode characters. You need to handle exceptions properly.

The following skeleton code is provided and only the code to complete the program needs to be provided. [19]

```
import java.io.*;
public class CharCount
{
    public static void CountPrint(String charList)
    {
        // TODO: Complete the code
    }
}
```

Q4. Write a Java program `TotalServer` which listens to incoming connections at the port 12345. 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 one or more integers (separated by blankspace(s)). The server's job is to calculate two numbers: the first is the sum of all the numbers in the current line, and the second is the sum of all the numbers since connected. The server sends back a line containing these two numbers separated by a blankspace every time a line is received from the client.

You may assume that the content received from the client is always in the correct format but communication exceptions need to be handled properly. [25]

**Example:**

Received:

```
2  5
6 -3 2
```

Send back:

```
7 7
5 12
```

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

```
import java.net.*;
import java.io.*;
import java.util.*;
public class TotalServer
{
    public static void main()
    {
        // Create a Server Socket object
        ServerSocket sSock = null;
        try
        {
            sSock = new ServerSocket( 12345 );
        }
        catch( IOException e )
        {
            System.err.println(e);
            System.exit(1);
        }
        // TODO: Complete the program
    }
}
```