NUI Galway
OÉ Gaillimh

Semester 1 Examinations 2016 / 2017

Exam Code(s)	3BCT, 3BP, 3BLE
Exam(s)	Third Year Computer Science & Information Technology Third Year Electronic and Computer Engineering Third Year Electrical and Electronic Engineering
Module Code(s)	CT326
Module(s)	Programming III
Paper No.	1
External Examiner(s)	Dr. J. Power
Internal Examiner(s)	Dr. J. Duggan *Dr. D. Chambers
Instructions:	Answer any 4 questions. All questions carry equal marks.
Duration	2 hrs
No. of Pages	4
Department(s)	Information Technology
Requirements	None

1. Develop a simple Java based payroll system that can calculate the weekly pay due for different categories of employees. The system should be implemented using the following design guidelines:

   a: Implement an *abstract* base class called Employee that is used to hold basic information about an employee e.g. name, address, etc. This class should also define an *abstract* method called earnings() that returns the weekly pay for each type of employee. The class should include a suitable constructor and accessor methods to retrieve information about the employee. Include the private instance variable joinDate in class Employee to represent when they joined the company. Use the java.util.Date class for this variable. 7 MARKS

   b: Implement a class called Manager, derived from Employee. A manager is paid a fixed weekly salary. The class should include a suitable constructor and should also implement the earnings() method. 5 MARKS

   c: Implement a class called HourlyWorker, derived from Employee. An hourly worker is paid a fixed wage per hour, so in any given week they will be paid for the number of hours worked in the past week. The class should include a constructor and implement the earnings() method. 5 MARKS

   d: Write a short driver program that creates an array of Employee variables to store references to the various employee objects. In a loop, calculate the payroll for each Employee, and add a €100.00 bonus to the person's payroll if they joined the company over 5 years ago. 8 MARKS

2.a: Implement an Enum in Java which enumerates the months of the year. Include in the enum the relevant position of the month in the calender i.e. Jan = 1, Feb = 2, etc and the number of days in the month i.e. Jan = 31, Feb = 28, etc. You do not need to provide support for leap years in the implementation. Provide a suitable toString() method to print information about the enumerated types. 12 MARKS

   b: The JDK contains two general-purpose List implementations i.e. *ArrayList* and *LinkedList*. Why is *ArrayList* generally the best performing implementation? Describe the circumstances under which *LinkedList* might offer better performance. Describe the polymorphic algorithms provided in the JAVA Collections framework. In relation to these algorithms, explain fully the purpose and operaton of the following code idiom:

```
int pos = Collections.binarySearch(l, key);
if (pos < 0)
      l.add(-pos-1, key);
```

13 MARKS

3.a: Write a Java class called Rational for performing arithmetic with positive rational numbers. A positive rational no. is a number that can be expressed in the form of p/q, where p,q are positive integers with q !=0 and p and q have no common divisor. Make the class implement the comparable interface so that each object should be able to compare itself to another object of the same type based on ascending order. 5 MARKS

b: Write a Java program that uses an ArrayList to store a collection of 10 of the Rational objects defined in part a. Sort the ArrayList of Rational objects according to the Natural Order defined in the Comparable interface of the Rational class itself using the sort method provided in the collections framework and then finally print out the list. 8 MARKS

c: Sort the ArrayList of Rational objects again using a separate Comparator class (a class that implements java.util.Comparator) that sorts them in descending order i.e. the opposite of the Natural Order defined in the Rational class itself. 7 MARKS

d: Modify the program to use the built-in Collections.binarySearch() method to check if a particular fraction, inputted through the console, is contained within the list. Print out your results to the console. 5 MARKS

4.a: Write a network Server program in Java where the Server waits for incoming client connections using stream type sockets. Once a Client connects it sends a String object to the server with a simple query – the server then responds with a text based response. The connection is then terminated. The server should use a separate thread of execution for each new client connection and all interaction between the Server and the Client should be done within this thread. The answer only needs to include source code for the server side application. 12 MARKS

b: Write another Java application with the same functionality as outlined above, in part a of this question, but this time using Datagram type sockets. Hint: you can use ByteArrayOutputStream and ByteArrayInputStream to populate and read the array associated with the DatagramPacket object. This application does not need to implement a reliable data transfer protocol. The answer only needs to include source code for the server side application. 13 MARKS

5.a: Evaluate the following code sample (in terms of good design practice). The code is for a simple home heating system. The system is turned on if the current temperature falls below some minimum value. It's then turned off again when it goes above the maximum value. The class is instantiated and started i.e. it runs in its own thread of execution.

```
class Thermostat extends Thread
{
        private final int THERMOMETER = 0x10;
        private final int HEATER = 0xf7;
        private final int HEATER_ON = 0x1;
        private final int HEATER_OFF = 0;
        public void run()
        {
            while (true)
            {
                while (read(THERMOMETER) > min)
                        sleep(100);
                write(HEATER, HEATER_ON);
                while (read(THERMOMETER) < max)
                        sleep(100);
                write(HEATER, HEATER_OFF);
            }
        }
}
```

What is wrong with this code? Suggest a better design approach based on using the dependency inversion principle. 10 MARKS

b: Outline the design and code implementation of the Java class for an object that will be used as a buffer to hold an integer value. The value may be updated randomly by one or more Producer threads, provided that it has already been consumed by one of a number of Consumer threads. Each value produced must be consumed exactly once and there may be multiple producer and consumer threads executing (and attempting to access the buffer) concurrently.
15 MARKS