Spring Examinations 2013 / 2014

Exam Code(s)	3BCT, 3BP1
Exam(s)	Third Year Computer Science & Information Technology Third Year Electronic and Computer Engineering
Module Code(s)	CT326
Module(s)	Programming III
Paper No.	1
External Examiner(s)	Dr. J. Power
Internal Examiner(s)	Prof. G. Lyons Dr. M. Madden *Dr. D. Chambers
Instructions:	Answer any 4 questions. All questions carry equal marks.
Duration	2 hrs
No. of Pages	5
Department(s)	Information Technology
Requirements	None

1.a: Write an **abstract** Java class called **Account** to hold information about bank accounts. The **Account** class should contain suitable **private** instance variables to represent the account number, the balance of the account, the name of the account holder and the address of the holder. It should also contain a **List** of transaction objects for that account. Provide a constructor method that enables an object of this class to be fully initialised when it is declared. Also provide other **public** methods to facilitate simple transactions on the account object and to retrieve the various account attributes (e.g. the balance, account holder details, transactions). Provide the implementation for a suitable class that contains details about past transactions. The Transaction class will need suitable **private** instance variables to represent the type of tansaction, the amount, the date / time, and the balance after the transaction was applied. You can use the **java.util.Date** class to represent the date / time within the Transaction class. The list of transactions in the Account will be stored in the order the transactions were made. 10 MARKS

b: Write another Java class called **SavingsAccount** that inherits the **Account** class. This derived class should contain an additional **private** instance variable called **annualInterestRate** and an additional method called **addMonthlyInterest** that calculates the monthly interest and adds it to the account balance. The interest should be based on the closing balance each day for the past month. The closing balance can be obtained by iterating through the list of Transaction objects for the past month and then multiplying the closing account balance, as stored in the final Transaction object for each day, by **annualInterestRate** and then divide this amount by 365; this daily interest should then be added to the account balance. Provide a suitable constructor method that enables an object of this class to be fully initialised when it is declared. You might find the following code sample useful in using and comparing date objects :

```
Calendar cal = Calendar.getInstance();
cal.setTime(someDate); // someDate is a Date object
int day = cal.get(Calendar.DAY_OF_MONTH);
```

10 MARKS

c: Finally, write a driver program to test class **SavingsAccount**. Instantiate two **SavingsAccount** objects called **saver1** and **saver2**, with initial account balances of €2000 and €3000, respectively. This program should set **annualInterestRate** to 4%, then calculate the monthly interest and print the new balances for each of the savers. 5 MARKS

2.a Write a Java program that uses an ArrayList to store a collection of Vehicle objects. Also, write the code for a Comparator class i.e. a class that implements the Comparator interface, that can be used to compare two Vehicle objects based on their engine size. The Vehicle class should implement an accessor method called getEngineSize() that returns an int representing the engine size for that vehicle e.g. 1895. Finally, use the version of the Collections.sort() method that allows you to pass your own Comparator object to sort the list of Vehicle objects.
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 cricumstances 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. Write a Java class called **Rational** for performing arithmetic with fractions.

Use integer variables to represent the **private** instance variables of the class - the **numerator** and the **denominator**. Provide a constructor method that enables an object of this class to be initialised when it is declared e.g. the fraction 2/3 would be stored in the object as 2 in the **numerator** and 3 in the **denominator**.
4 MARKS

Provide **public** methods for each of the following:

(a) Addition of two **Rational** numbers. The result should be stored in the target object e.g. if **r1** and **r2** are objects of type **Rational**, calling **r1.add(r2)** would add the value of **r2** to **r1** and then store the new value in **r1**. (Hint: adding 2/3 and 3/4 gives the result 17/12). 7 MARKS

(b) Multiplication of two **Rational** numbers. In the same way as for (a), the result should be stored in the target object. (Hint: multiplying 2/3 and 3/4 gives the result 6/12). 7 MARKS

(c) Printing **Rational** numbers in the form **a/b**, where **a** is the **numerator** and **b** is the **denominator**. 3 MARKS

Finally, write a suitable driver program that could be used to test your class.
4 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 should include full source code for the server 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. 13 MARKS

5. Using Java Remote Method Invocation, write the Java code for a remote compute server that could be used to remotely execute arbitrary Task objects. The server allows clients to submit Task objects, that is objects that implement the Task interface, for remote execution on the server and are then returned the result as a Java object. The following Java interfaces / classes should be provided:

- *Compute* - this remote interface should provide a method to upload Task objects to the server and to then run the task and return the result back to the client when execution is complete. 4 MARKS

- *Task* - this interface should define an arbitrary task object that may be passed as a parameter to the compute server. 4 MARKS

- *MathTask* – this class provides an implementation of the Task interface and is used to perform some calculation that returns an Integer object. The calculation itself can be just some simple arithmetic e.g. add two numbers. 6 MARKS

- *ComputeServer* - this class should provide an implementation of the Compute interface as well as the code required to initialise the server and make the remote object locatable for clients in the RMI registry. The server runtime should be protected so that objects uploaded to the server can not cause any harm. 6 MARKS

- *ComputeClient* – this should provide a simple client program that creates a MathTask object and submits it to the server for remote execution and then displays the result. 5 MARKS

The design of the system should make it possible for new Task classes to be easily added to the system in the future, making the system very flexible. The design should use Java RMI and Object Serialisation to submit Task objects and to return the result back to the client.

6.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