NUI Galway
OÉ Gaillimh

Autumn Examinations 2012

Exam Code(s)	3IF1, 3BP1
Exam(s)	Third Year Information Technology Third Year Electronic and Computer Engineering
Module Code(s)	CT326
Module(s)	Programming III
Paper No.	1
External Examiner(s)	Prof. M. O'Boyle
Internal Examiner(s)	Prof. G. Lyons Dr. M. Madden *Dr. D. Chambers
Instructions:	Answer any 4 questions. All questions carry equal marks.
Duration	3 hrs
No. of Pages	5
Department(s)	Information Technology
Requirements	None

1:a: Using a simple example, discuss why casting a superclass reference to a subclass reference is potentially dangerous. 5 MARKS

b: What is the difference between **abstract** classes and interfaces? Should all the methods in an **abstract** superclass be declared **abstract**? 5 MARKS

c: Consider the inheritance hierarchy of **Figure** 1 below. For each class, indicate some common attributes and methods consistent with the hierarchy. Assume that a CurrentAccount provides overdraft facilities and that a StudentAccount is similar to a Current Account but has no transaction charges. Write simple Java implementations for each of the classes shown. The base Account class should be declared as an abstract class. 15 MARKS

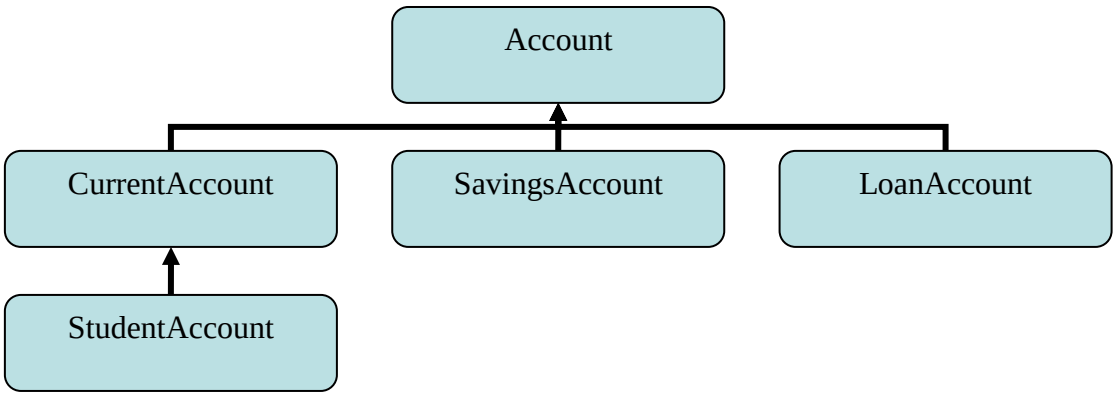

Figure 1 - Inheritance Hierarchy for Bank Accounts

2.a: Describe the general structure and purpose of the IO Streams classes provided in the Java programming environment. What Java Class is used to support random file access? 4 MARKS

b: Write a Java application that inputs a date as a string in the form 17/02/2010 The program should use an object of class *StringTokenizer* to extract the various components of the date string as tokens. The program should then convert the day, month and year to int values and display them. 6 MARKS

c: Write a simple Employee class that includes an id number, a name, and salary details and a suitable constructor method. Then write a Java program that uses an ArrayList to store a collection of Employee 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 Employee objects based on their id number. Finally, use the version of the Collections.sort() method that allows you to pass your own Comparator object to sort the list of Employee objects. 15 MARKS

3.a: Describe briefly the purpose and operation of the following code idiom:

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

5 MARKS

b: Create a standalone Java application that will count and sum up the number of lines in the text file passed as an argument on the command line. The program should create a **FileReader** object and pass this in the constructor of a **LineNumberReader** object to handle the file reading required.

The **LineNumberReader** class has two useful methods (that could be used):

public String readLine() throws IOException; This method reads a line of text. It returns a String containing the contents of the line, not including any line-termination characters, or null if the end of the stream has been reached.

public int getLineNumber(); This method returns the current line number.

10 MARKS

c: 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 vlaue. 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's wrong with this design? Suggest a better design approach based on using the dependency inversion principle. 10 MARKS

4.a: Discuss briefly the differences between a process and a thread. How should executing threads be stopped (assuming they still haven't finished their work)? 5 MARKS

b: Write a JAVA animation applet that uses a thread to continuously scroll a text message across the screen from right to left. The message itself and the rate at which the text scrolls can be passed to the applet as HTML based parameters. 10 MARKS

c: 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 at exactly once and there may be multiple producer and consumer threads executing (and attempting to access the buffer) concurrently. 10 MARKS

5.a: What types of Sockets are supported in the Java networking package? Which type of Socket would you recommend for a VOIP type application and a File Transfer type application? 5 MARKS

b: Write a Java application that uses Stream type sockets to exchange Java Objects using object serialisation. The client side should connect to the server and send it an Integer Object. The server should print out this value and respond to the client with a text based response encapsulated in a String Object. The client should receive the String Object from the server and print out this response. 10 MARKS

c: Write another Java application with the same functionality as outlined above, in part b 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. 10 MARKS

6. Assume that a Sports Club at the University wishes to store details about its members. Design and implement a Java application to support this requirement. The application should be able to print out and manage information about the members of the club. The following guidelines should be used to construct the application:

   a: A Java class, called Member, should be defined to store and manage student details. The class should include methods for updating member details and querying their registration status i.e. are they fully paid up members of the club. Each member of the club should also have a unique membership id number, this number is automatically assigned when the member object is created. 10 MARKS

   b: Define another Java class, called SportsClub, that will be used to manage club membership and access details about individual members. Member objects added to the SportsClub should be stored using a suitable collection object. SportsClub should include methods for adding new members, removing members, getting a list of current members and accessing information about an individual member (based on their name or id number). 10 MARKS

   c: Write a short driver program, in a class called ClubManager, that creates an instance of SportsClub and uses its methods to add, lookup and remove club members. 5 MARKS