



## **Spring Examinations 2012 / 2013**

**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? Do all the methods in an **abstract** superclass have to be declared as **abstract**? 5 MARKS

c: Consider the inheritance hierarchy of **Figure 1** below. For each class, indicate the 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 including constructors and other methods required. The base Account class should be declared as an abstract class and details of all transactions (debits and credits) should be stored in an ArrayList of Transaction objects i.e. define a class called Transaction suitable for this purpose. 15 MARKS

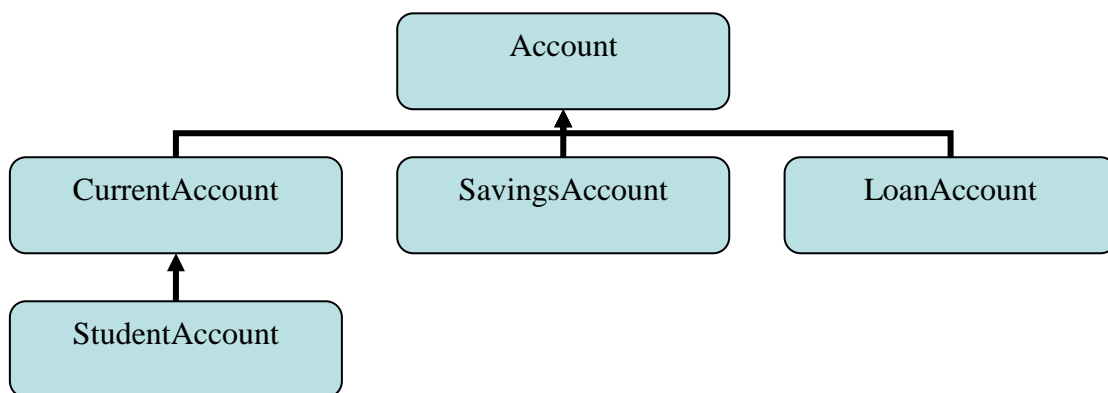


Figure 1 - Inheritance Hierarchy for Bank Accounts

- 2.a: 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

- b: 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: 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 a String object. The server should print out the String and respond to the client with a text based response encapsulated in another 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

4.a: Discuss briefly the differences between a process and a thread. What is the best way to stop 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: Create a class called **Complex** for performing arithmetic with complex numbers. Complex numbers have the form:

**realPart + imaginaryPart \* i** where *i* is the square root of -1.

a: Use floating-point variables to represent the **private** data of the class. Provide constructor method(s) that enable object of this class to be fully initialized.  
5 MARKS

b: Provide a **public** method to add two **Complex** numbers: the real parts are added together and the imaginary parts are added together to create the result. This method should return a new **Complex** object initialized with the result e.g. if **c1** and **c2** are objects of type **Complex**, calling **c3 = c1.add(c2)** would add the value of **c2** to **c1** and then return a new object initialized with the result. The original values of **c1** and **c2** would not change.  
7 MARKS

c: Provide a **public** method for subtraction of two **Complex** numbers: the real part of the right operand is subtracted from the real part of the left operand, the imaginary part of the right operand is subtracted from the imaginary part of the left operand. In the same way as for (b), this method should also return a new **Complex** method initialized with the result e.g. if **c1** and **c2** are objects of type **Complex**, calling **c3 = c1.subtract(c2)** would subtract the value of **c2** from **c1** and then return a new object initialized with the result. The original values of **c1** and **c2** would not change  
7 MARKS

d: Provide a **public** method for printing **Complex** numbers in the form **(a+bi)** where **a** is the real part and **b** is the imaginary part and write a short driver program to test your class.  
6 MARKS

6. 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.