Ollscoil na hÉireann, Gaillimh GX_____
National University of Ireland, Galway

Autumn Examinations 2008

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. J. Keane
Internal Examiner(s)	Dr. D. Chambers Prof. G. Lyons

Instructions: Answer any 4 questions.
All questions will be marked equally.

Duration	3hrs
No. of Answer Books	1
No. of Pages	5
Department(s)	Information Technology

1. 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**. Also provide a no-argument constructor with default values in case no initialisers are provided. Provide **public** methods for each of the following:

   i. 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).

   ii. Subtraction of two **Rational** numbers. In the same way as for (a), the result should be stored in the target object. (Hint: subtracting 2/3 from 3/4 gives the result 1/12).

   iii. 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).

   iv. Division of two **Rational** numbers. In the same way as for (a), the result should be stored in the target object. (Hint: dividing 2/3 by 3/4 gives the result 8/9).

   v. Printing **Rational** numbers in the form **a/b**, where **a** is the **numerator** and **b** is the **denominator**.

   Finally, write a suitable driver program that could be used to test your class.
   25 MARKS

2.a: Describe the general structure and purpose of the IO Streams classes provided in the Java programming environment. Is there any mechanism to support random file access in Java? 5 MARKS

b: What information is normally written out during object serialisation in the Java programming environment? Using a suitable example, describe how you can provide custom serialisation for your own classes. 10 MARKS

c: Write a Java applicaion that prompts the user to input their Name, Address, Date of Birth and Student ID number using either the standard input *System.in* or a GUI based input dialog - this information should then be saved to a file named *studentData*. The program should use the *FileWriter* class and an appropriate processing stream to handle the data output. 10 MARKS

3.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: Show (using simple code examples) how threads may be created (and started) using the following mechanisms:
(i) Application class extends the Thread class.
(ii) Application class implements the Runnable interface.
Assume you have a bank account class that may be accessed by more than one thread of execution simultaneously. Show how the various methods of the class may be made thread safe. 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

4.a: Desribe the functionality provided by the following Java code. Would it have been possible to write similar code using an *Enumeration* instead of an *Iterator*? Explain your answer.

```
import java.util.*;

   static void filter(Collection c) {
      for (Iterator i = c.iterator(); i.hasNext(); )
            if (!cond(i.next()))
               i.remove();
      }
```
10 MARKS

b: Develop a simple GUI-based Java program that may be used to control a washing machine. Use suitable Swing components to allow the washing machine operator to perform the following functions:
1) Switch the machine on.
2) Choose a temperature from a list.
3) Spin speed selection buttons - can be 600, 800 or 122 RPM.
4) Display the current status of the wash cycle.

Show the top-level design of the GUI, including any Panels and related Layout Manager objects that you propose to use. For each of the components you've chosen above, write the code to construct the component, add the component to a container and then setup simple event handling for the component (for those that generate events). The event handlers need only print out a message indicating that they have been called. 15 MARKS

5. The following Java code provides the outline of a simple bank account class:

```
import java.io.*;
public class Account implement Serializable{
        protected HolderDetails holder;
        protected List transactions;
        protected float balance;

        // Add a suitable constructor here

        // Add methods to make deposits / withdrawls

        // Add a method to print out summary of account transactions

        // Add suitable accessor methods for retrieving / updating attributes
}
```

a: Complete the implementation of the Account class, providing a suitable constructor, attribute accessor methods, methods for making deposits / withdrawls and a method to print out a transaction summary.
7 MARKS

b: Provide implementations for the HolderDetails class and a suitable class to represent transactions. The HolderDetails class is used to store details about the account holder. The Transaction class contains details about past transactions.
7 MARKS

c: Define and implement a new class, called CurrentAccount, derived from Account, that allows withdrawls to proceed up to some overdraft limit.
5 MARKS

d: The attributes of class Account are defined as *protected*. What is the implication of this definition? 3 MARKS

e: What does the statement *implements Serializable* mean? What are the implications of this statement? 3 MARKS

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

b: 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 text string 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.
10 MARKS

c: Suppose that you've written a program that displays three messages, as follows:

```
public class NotI18N {

    static public void main(String[] args) {

        System.out.println("Hello.");
        System.out.println("How are you?");
        System.out.println("Goodbye.");
    }
}
```

You then decide that this program needs to display the same or similar messages for people living in France and Germany. Outline the steps needed to properely internationalise this program i.e. the hardcoded English language messages should be removed and replaced with a more flexible mechanism that will facilitate additional language support in the future. 10 MARKS