Ollscoil na hÉireann, Gaillimh
National University of Ireland, Galway

GX 1495

Spring Examinations, 2003/2004

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. P. Nixon
Internal Examiner(s)	Prof. G. Lyons Dr. D. Chambers

Instructions: Answer any 5 questions.
All questions will be marked equally.

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

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 and access 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.
   4 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. 4 MARKS

   c: Implement a class called HourlyWorker, derived from Employee. An hourly worker is paid a fixed wager per hour, so in any given week they will be paind for the number of hours worked in the past week. The class should include a constructor and implement the earnings() method. 4 MARKS

   d: Implement a class called CommissionWorker, derived from Employee. A commission worker is paid a base salary per week and an additional bonus based on the number of items sold during the past week. The class should include a constructor and earnings() method. 4 MARKS

   e: Write a short driver program that creates an object for each of the employee sub-classes, it then calls the earnings() method for each object and displays the results. 4 MARKS

2.a: Discuss briefly the differences between a process and a thread. Show (using simple code examples) how threads may be created (and started) using the following mechanisms:
   (i) Application class implements the Runnable interface.
   (ii) Application class extends the Thread class. 4 MARKS

   b: What is meant by the term *deadlock*? Using the example of the *Dining Philosophers Problem* (covered in class), discuss how deadlock might occur in this case and propose a solution to overcome the problem. 8 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.
   8 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**. 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 short driver program that could be used to test your class.

20 MARKS

4.a: Describe, using simple examples, the main differences between Java applets and standalone Java applications. 4 MARKS

b: The Java SDK 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. 8 MARKS

c: Write a simple Java program that creates an *ArrayList* object and adds a number of *String* objects to the *ArrayList*. It then serializes the *ArrayList* object to a file named *theList*. Finally, it reads in the saved *ArrayList* object from the file and reconstructs a new *ArrayList* object from the deserialised data.

8 MARKS

5.a: Identify and correct the errors in each of the following program segments. [*Note:* There may be more than one error in each piece of code]:

i:
```
String s[] = { "Hello", "World!"};
System.out.println(s[1], " ", s[2]);
```
2 MARKS

ii:
```
if (age > 65);
  System.out.println("Age 65 or greater");
else
  System.out.println("Age is less than 65");
```
2 MARKS

iii:
```
int b[] = int[10];
for (int i = 0; i <= b.length; i++)
  b[i] = 1;
```
2 MARKS

iv:
```
switch(n)
{
  case 1:
    System.out.println("The number is 1");
  case 2:
    System.out.println("The number is 2");
  default:
    System.out.println("The number is" n);
}
```
2 MARKS

v:
```
public class test1 expands generictest
{
  public static void main(args)
  {
    for (int i = 0; i < args.length; i++)
      System.out.println(args[i]);
  }
}
```
2 MARKS

b: Write a Java application that inputs a date as a string in the form 20-02-2004. 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. 10 MARKS

6.a: Explain briefly the purpose of Design Patterns. Give an example of a Design Pattern and where it could be used. 4 MARKS

b: Describe the dependency inversion principle. Illustrate using an example (which includes a class diagram). What are the consequences of applying the principle on small systems? 8 MARKS

c: Describe the various polymorphic algorithms provided in the JAVA Collections framework. In relation to these algorithms, what is the purpose of the following code idiom:

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

8 MARKS

7. Assume that a Sports Club 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 member details. The class should include methods for updating member details and querying their subscription status i.e. are they fully paid up club members. Each member of the club should also have a unique membership id number, this number is automatically assigned when the member object is created. 8 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 id number). 8 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. 4 MARKS