Two hours

QUESTION PAPER MUST NOT BE REMOVED FROM THE EXAM ROOM

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

High-Level Programming (Java)

Date: Thursday 20th January 2011
Time: 14:00 - 16:00

Answer ALL Questions in Sections A and B and one question from Section C

For full marks your answers should be concise as well as accurate.
Marks will be awarded for reasoning and method as well as being correct.

This is a CLOSED book examination

The use of electronic calculators is permitted provided they are not programmable and do not store text.
Section A is a multiple choice section and is therefore restricted
SECTION B

COMPULSORY

(a) What is an accessor method? What is a mutator method? What are the commonly used naming conventions for accessor and mutator methods, respectively? (3 marks)

(b) Explain the difference between the output from

```java
String s;
System.out.println("s= " + s);
```

and

```java
String s = new String();
System.out.println("s= " + s);
```

(3 marks)

(c) The following class `Bus` is incomplete; a constructor and methods have yet to be implemented. Complete the class by implementing the constructor and four methods based on the specification described in comments.

```java
public class Bus {
    private int id;
    private double fuel;

    /**
     * Construct a specific bus of some amount fuel.
     * @param aPlateNumber the plate number for this bus
     * @param initialAmount the initial amount of fuel in its tank
     */
    public Bus(int aPlateNumber, double initialAmount) {
        fill in your code
    }

    /**
     * Get the plate number of this bus.
     * @return the plate number
     */
    public int getPlateNumber() {
        fill in your code
    }

    /**
     * Fill fuel into the tank of this bus.
     * @param amount the amount of fuel to add
     */
    public void addFuel(double amount) {
        fill in your code
    }

    /**
     * Consume fuel in the tank of this bus.
     * @param amount the amount of fuel to use
     */
    public void consumeFuel(double amount) {
        fill in your code
    }
}
```

[PTO]
(Question B(c) continues from the previous page)

```java
/**
   public void useFuel(double amount)
   {       fill in your code    }

   /**
   Get the state of fuel in the tank of this bus.
   @return the current amount of fuel
   */
   public double readMeter()
   {       fill in your code    }
}
```

(6 marks)

(d) What is the output from the following program? List all printed results in order.

```java
class ExcHandling {
   public static void main(String[] args) {
      // Here, numer is longer than denom.
      Int[] numer = { 4, 8, 16, 32, 64, 128, 256, 512 };  
      Int[] denom = { 2, 0, 4, 4, 0, 8 }; 

      for(int i=0; i<numer.length; i++) {
         try {
            System.out.println(numer[i] + " / " +
               denom[i] + " is " +
               numer[i]/denom[i]);
         }
         catch (ArithmeticException exc) {
            // catch the exception
            System.out.println("Can't divide by Zero!");
         }
         catch (ArrayIndexOutOfBoundsException exc) {
            // catch the exception
            System.out.println("No matching element found.");
         }
      }
   }
}
```

(8 marks)
SECTION C

Answer ONE of two questions

C1. (a) Given variables $x$ and $y$ of type `int`, the following statement sequence exchanges the values stored in $x$ and $y$.

```java
{ int temp = x;
  x = y;
  y = temp;
}
```

Illustrate the way in which the exchange takes place by showing what the values of each of the variables is after each of the three statements when, initially, $x = 2$ and $y = 5$.

(2 marks)

Using the same initial values, $x = 2$ and $y = 5$, explain, similarly, why the following simpler statement sequence does not have the same effect.

```java
{ x = y;
  y = x;
}
```

(1 mark)

The following method declaration uses the first of the above statement sequences in an attempt to exchange the values of its parameters.

```java
private static void swap(int x, int y)
{ int temp = x;
  x = y;
  y = temp;
}
```

Consider two variables $a$ and $b$ of type `int` with values 2 and 5 respectively. Explain why the statement `swap(a, b);` does not change the value of $a$ and $b$.

(2 marks)

Consider the class

```java
public class MyInt
{ int value;
  public MyInt(int val) { value = val; }
  public int getValue() { return value; }
}
```

The declarations

```java
MyInt c = new MyInt(2);
MyInt d = new MyInt(5);
```

[PTO]
(Question C1 continues from the previous page)

create objects c and d of class MyInt with c.getValue() equal to 2 and
d.getValue() equal to 5. Consider changing the method swap as follows, so that it uses
objects of class MyInt instead of items of the primitive type int:

```java
private static void swapMyInt(MyInt x, MyInt y)
{
    MyInt temp = x;
    x = y;
    y = temp;
}
```

Explain how objects are referenced by addresses and show how this may mean that
assignments involving objects of class MyInt may have after undesirable side-effects which
assignments using items of type int do not have. Explain why, despite such side-effects, the
method call swapMyInt(c, d); does not change the values stored in the objects c and d.

(3 marks)

Show how to write a method swapMyInt which does have the desired effect of swapping
the values stored by its actual parameters. (2 marks)

(b) Consider variables

```java
final static maxLength = 30;
static int n;
static int[] seq = new int[maxLength];
```

which have been initialised so that n > 0, n < maxLength, and seq stores a
sequence of integers (with repeated values permitted) in index positions from 0 to n-1.

Write a method

```java
public static void sort ()
```

which repeatedly passes through the sequence, examining each successive consecutive
pairs of items, exchanging them if the first is greater than the second, until no swaps are
made on such a pass. (5 marks)

Explain why your method terminates. (2 marks)

What can be said about the sequence after sort() has been called and why? (2 marks)

What is the relationship between the initial state of the sequence, before the method
call, and its final state, after the method call? (1 mark)
C2.

(a) Object orientation in JAVA is achieved by means of classes. Objects are obtained by instantiating classes. Illustrate this by giving a declaration of a class called ArmedMan with two public fields, String name and int number, together with a public constructor to create a new ArmedMan with given name and number. [You need not include the class in a package.]

Show how to use your constructor to create two objects: shvejk with name "Shvejk" and number 2011 and goodSoldier with the same name and number. Show how to access and/or update the fields of napoleon1. To what extent are shvejk and goodSoldier equal? (2 marks)

(b) Arguably, fields should be made private and methods should be provided to access and update them. Explain the reason which is usually given for this and show how to comply with this practice, using the ArmedMan class from part a). Why is extra care required when fields are objects rather than items of primitive types like int? (5 marks)

Write a method with heading

    public String toString()

which gives an appropriate String representation of an ArmedMan, e.g. if the name is "Shvejk" and the number is 2011 the string would be "Shvejk, number 2011". (1 mark)

(c) The essence of object orientation is inheritance. Illustrate this by means of a class Sailor which extends ArmedMan and has an extra public field String ship to record the name of the sailor’s ship. You should include declarations of a constructor to create a new Sailor with given name, number and ship, and a method toString like the one for ArmedMan, but this time including the name of the ship. (4 marks)

(d) In JAVA, all methods are said to be virtual. Illustrate what this means by considering a method which has a parameter of class ArmedMan and uses the parameter’s method toString in order to print details. Consider what happens when the method is passed an actual parameter of class Sailor. (2 marks)

(e) JAVA does not have multiple inheritance! Explain what this means and why absence of multiple inheritance allows the use super in JAVA based on two classes ArmedMan and Sailor described above. (4 marks)

END OF EXAMINATION