A20487

No calculator permitted in this examination

UNIVERSITY OF BIRMINGHAM

School of Computer Science

First Year – Degree of BSc with Honours
Artificial Intelligence and Computer Science
Computer Science
Computer Science with Study Abroad
Computer Science with Business Management
Natural Sciences

First Year – Degree of BEng/MEng with Honours
Computer Science/Software Engineering

First Year – Joint Degree of BEng/MEng with Honours
Electronic and Software Engineering

First Year – Joint Honours Degree of BSc/MSci with Honours
Mathematics and Computer Science

06 18190

Software Workshop 1

Summer Examinations 2008

Time allowed: 3 hours

[Answer ALL Questions]

Turn Over

Section A.

[Answer ALL Questions]

1 (a) The following for loop has three parts missing from its first line, each indicated by ???. Write down that first line with the three parts filled in so that when the for loop runs it will print out the numbers 10, 9, 8, ..., 1, 0.[3%]

```
for (??? ; ??? ; ??? ) {
    System.out.println(j);
}
```

(b) A class `C` has a private `int` field `n`, a public getter method `getN`, and a constructor that is intended to intialize the field from a parameter. The constructor is of the form

```
public C(final int n) {
  ...;
}
```

In four different versions of the class, the "..." line in the constructor has been written in four different ways as follows.

(i) `n = n;`
(ii) `this.n = n;`
(iii) `n = this.n;`
(iv) `this.n = this.n;`

For each version say, with reasons, what will be printed by the following code.

```
C x = new C(7);
System.out.println(x.getN());
```

Which version is correct? [4%]

2. The following method is intended to return an array whose elements are the first n Fibonacci numbers. For example, `fibArray(6)` should return an array whose elements are 0, 1, 1, 2, 3, 5. (Each Fibonacci number is the sum of the two previous ones.) Note that for `n` ≥ 47, the Fibonacci numbers are too big to be stored as `int` values. Line numbers have been included, for convenience.

```
/**
 * Make an array whose elements are a given number
 *   of Fibonacci numbers.
 * requires: 0 <= n <= 47
 * @param n number of Fibonacci numbers to use
 * @return the array
 */
public static int fibArray(final int n) // line 1
{                                       // line 2
    int[] a;                            // line 3
    for (i = 0; i <= n; i = i+1);       // line 4
    {                                   // line 5
        a[i] = a[i-1] + a[i-2];         // line 6
    }                                   // line 7
    return a;                           // line 8
};                                      // line 9
```

(a) The method contains *six* errors. For each one, explain what the error is, whether and how it would be detected either at compile time or at run time, and how to correct it. [10%]

(b) When you have corrected the method to match the Javadoc heading, what will it do when n is (i) too small, or (ii) too large?

Suppose an array of `double` is used instead of the array of `int`. Describe three ways in which this affects the accuracy with which Fibonacci numbers can be stored for large indexes. [5%]

3. Suppose a class `C` includes (amongst other things) a private field `arr` of type `int[]` and a method `sum` defined as follows (with line numbers given here for convenience).

```
/**
 * Sum the elements of arr
 * @return sum of elements of arr.
 */
public int sum() {
    int i = 0;
    int S = 0;
    /* loop invariant:
     *   0 <= i <= arr.length and
     *   S is sum of first i elements of arr.
     */
    while (i < arr.length) {    //loop test
        S = S+arr[i];
        i = i+1;
    }
    return S;
}
```

(a) Suppose (in NetBeans or using a similar debugger) a breakpoint has been set at the line containing the loop test. Execution will stop immediately before each evaluation of the loop test. The method `sum` is called on an instance of the class `C` in which `arr` has length 5, with elements 10, 20, 30, 40, 50 (in that order).

(i) How many times will the breakpoint be hit?
(ii) What will be the values of `i` and `S` on the first three times and on the last time that the breakpoint is hit?
(iii) Why does the loop invariant say `i <= arr.length`, while the loop test says `i < arr.length`?

[5%]

It is now required to write a new method `sumPart`, similar to `sum`, but summing only a region within the array with inclusive start "from" and exclusive end "to".

(b) The plan is, first, to write a non-defensive, private method `sumPartND`.

```
private int sumPartND(final int from, final int to)
```

Write a definition of `sumPartND`. It should use a while loop, as in `sum`, with a suitable loop invariant. Also include a Javadoc heading, with a requires condition. [5%]

(c) Now write a defensive, public version `sumPart`. It should call `sumPartND` but throw an `IllegalArgumentException` if the parameters are invalid. Do not include Javadoc, but describe how the Javadoc from part (b) needs to be modified. [5%]

4. Suppose someone has already written a Java class `Triangle` whose instances represent triangles, described by the lengths of their three sides. Its constructor takes the three side lengths as parameters, and throws an `IllegalArgumentException` if they do not form a triangle. It has various methods, including a method area that returns the area of the triangle. (Note – `IllegalArgumentException` is an unchecked exception.)

   (a) Write a *static* method `printArea` that takes three `doubles` as parameters, representing the side lengths for a triangle, and prints out (to `System.out`) the area of the triangle. It should work by creating an instance of `Triangle`, and calling its `area` method. If the side lengths are invalid, it should print `"Invalid sides"`. [3%]

   (b) Write the Java definition for a class `Equilateral` that extends `Triangle` and represents equilateral triangles (all three sides are equal in length). It should have its own private, final field `side`, of type `double`, which should be equal to the three sides stored for `Triangle`. Most of the methods of `Triangle` will be inherited, but you should override the `area` method so that it uses the formula

```
Math.sqrt(3)/4*side*side
```

   Include full Javadoc, and also an appropriate invariant condition for `side`. [6%]

   (c) An interface `DataItem` is defined as follows.

```
public interface DataItem {
    public double value();
}
```

   Write the Java definition for a class `AreaData` that implements `DataItem`. It should have a private, final field `theTriangle` of type `Triangle`. Its constructor should initialize that field from a parameter. Its `value` method calls `area` on `theTriangle`. [3%]

   (d) Consider the following code.

```
DataItem ad = new AreaData(new Equilateral(3));
System.out.println(ad.value());
```

   (i) What are the *type* of the variable `ad` and the *class* of the object it refers to? In that object, what are the type of the field `theTriangle`, and the class of the object it refers to? In each case, why are the type and class allowed to be different? [2%]

   (ii) In order to execute `ad.value()`, what other method is called and how is its definition found? [1%]

5. This question is about using Java objects to represent and evaluate *arithmetic expressions* such as $(x * y + 2.6)*(x + 3)$. All their classes will implement an interface `Expression`, defined by

```
public interface Expression {
  /**
   * Calculate the value of this expression.
   * @return value calculated
   */
  public double value();
}
```

Different classes will be needed for different operations. For example, the class `Multiply` will be for expressions where the final operation is a multiplication. The class has two fields (initialized from constructor parameters) for the two *operands* – i.e. the two subexpressions that are multiplied together. The expression just mentioned will be represented by an instance of `Multiply`, and its two operands will be objects representing $x * y + 2.6$ and $x + 3$. Many other operations (such as + and /) also have two operands, but some have only one (for example, sin or log).

(a) (i) Write a Java definition of the `Multiply` class. [5%]

(ii) How would you modify it for a class (e.g. `Sin`) for an operation with only one operand? [2%]

(iii) Write a Java definition for a class `Variable`, implementing `Expression`, in which `value` acts as a getter method for a number stored in a private field. A method `update` is to act as a setter method for that field. [4%]

(b) (i) Fill in the gaps (indicated by ???) in the following Java code so that `sinXInRadians` refers to an object representing the expression sin(x*π/180):

```
final Variable x = ???;
final Expression piBy180 = ???;          // for pi/180
final Expression xInRadians = ???;       // for x * pi/180
final Expression sinXInRadians = ???;    // for sin(x * pi/180)
```

You may use the classes `Multiply`, `Sin` and `Variable`, and also a class `Constant` that is similar to `Variable` but without the `update` method. (Use a single `Constant` instance for π/180.) [2%]

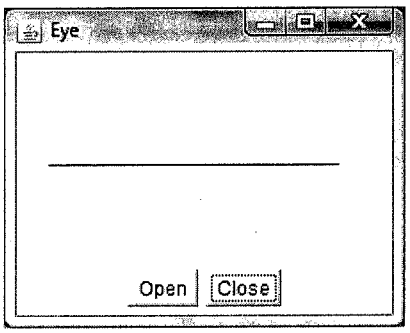
(ii) Write further code that updates `x` with successive values 0, 30, 60 and 90, and for each prints out the value of `sinXInRadians`. (*Note:* `x` has been made final. This is to prevent you trying to use reassignments `x = ...`.) [2%]

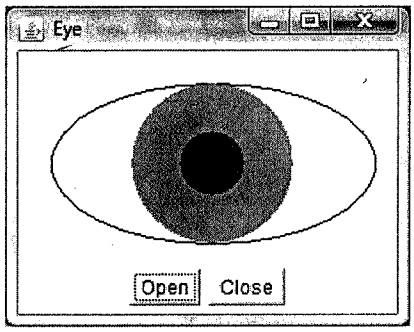
Section B

[Answer ALL Questions]

6. (a) Explain briefly the role of the interface `ActionListener`, its method `actionPerformed` and the `Button` method `addActionListener` in handling events generated by buttons in a Java Graphic User Interface. [3%]

(b) A window can be in one of two states as illustrated below.





The region of the window containing the eye is created using the code for the following `Canvas` class.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class EyeCanvas extends Canvas {

   private boolean open = false;

   public void paint(Graphics g) {
      if (open) {
         g.clearRect(10, 10, 220, 120);
         g.drawOval(20, 20, 200, 100);
         g.setColor(Color.GRAY);
         g.fillOval(70, 20, 100, 100);
         g.setColor(Color.BLACK);
         g.fillOval(100, 50, 40, 40);
      }
      else {
         g.clearRect(10, 10, 220,120);
         g.drawLine(20,70,200,70);
      }
   }

   public void setOpen(boolean open) {
      this.open = open;
      repaint();
   }

}
```

Write Java code for a `Frame` class that can be used to create windows with the appearance illustrated above, where the Open button changes the state of the eye to open and the Close button changes the state of the eye to closed. [8%]

(c) The following code outputs a sequence of random integers to a file:

```
DataOutputStream out =
          new DataOutputStream(
                  new FileOutputStream("DataFile"));
for (int i=1;  i<=100;  i++)
{
   int nextInt = (int)(Math.random()*1000000000);
   out.writeInt(nextInt);
}
out.close();
```

(i) Describe briefly the format of the data in this file. [2%]

(ii) Write a fragment of Java code that will read the integers from the file and calculate their average. [3%]

7. (a) The heading for the class `HashSet` in the Java library package `java.util` includes the following elements:

```
public class HashSet<E> ... implements Set<E>, ...
```

The method `add` defined in the class `HashSet` starts with:

```
public boolean add(E e) {
```

Explain the the use of the type `E` in the above code fragments. [3%]

(b) A small shop has a very simple stock control system. An item for sale is represented by a Java object constructed from the class:

```
public class StockItem
{
   private String stockCode, description;
   private double price; // price per item
   private int quantity; // no of items in stock

   public StockItem(String s, String d, double p, int q)
   {  setStockCode(s);  setDescription(d);
      setPrice(p);  setQuantityInStock(q);
   }

   // plus get and set methods for
   // stockCode, description, price, quantity

}
```

The entire stock is represented by the class `StockSet` whose heading is as follows:

```
import java.util.*;

public class StockSet extends HashSet<StockItem> { ...
```

Explain the use of the type `StockItem` in this class heading. [3%]

(c) Write, for inclusion in the class `StockSet`, a method `totalValue` that adds up the total value of all items currently in stock. [6%]

(d) The interface `Map` in the Java library package `java.util` includes the following:

```
public interface Map<K,V>
{
    V put(K key, V value);

    V get(Object key);
}
```

Instead of just storing a *set* of stock items as in (b) and (c), it is required to store the stock in such a way that a `stockCode String` can be used to look up the corresponding `StockItem` object. Demonstrate how a `HashMap` (which implements the `Map` interface) could be used to provide this functionality. [5%]

End of paper