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 Business Management

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

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

First Year – Joint Degree of BA with Honours
Ancient History and Archaeology/Computer Studies
Computer Studies and French
Hispanic Studies and Computer Studies

06 18190

Software Workshop 1

Summer Examinations 2007

Time allowed: 3 hours

[Answer ALL Questions]

Turn Over

Section A.

1. Answer the following questions about the Java programming language.

   (a) What is the value of `x` after this line is executed? [1%]

   ```
   double x = 7 / 2 + 1.5;
   ```

   (b) What are the possible values of a boolean variable? [1%]

   (c) The following for loop has three parts missing from its first line. Write down that first line with the three parts filled in so that when the for loop runs it will print out the numbers 1, 2, 4, 8 and 16.

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

   [3%]

   (d) Suppose an interface is defined as follows:

   ```
   public interface Evaluator {
       public double evaluate();
   }
   ```

   Suppose also we want to use a declaration

   ```
   Evaluator e = new C();
   ```

   What words should appear in the gaps in the following sentences?

   Class `C` must have _____ as a method. It must also _____ `Evaluator`.

   [2%]

2. The following Java code has mistakes. It was intended to print the largest integer `i` such that `i*i <= n` – in other words, `i*i <= n` but `(i+1)*(i+1) > n`. For example, if `n` is 9 it should print 3, while if `n` is 8 it should print 2. It assumes that `n >= 0`. Line numbers are included for convenience. Remember that integer division rounds towards zero.

```
(1)  int left = 0;
(2)  int right = n;
(3)  while (left < right) {
(4)     int mid = (left + right)/2;
(5)                  //now left <= mid < right
(6)     if (mid*mid <= n) {
(7)        left = mid;
(8)     } else {
(9)        right = mid;
(10)    }
(11) }
(12) System.out.println(left);
```

(a) Suppose, in the NetBeans IDE, a breakpoint is put on line (6) and the above code (as part of a suitable main method) is run under the debugger with `n` having the value 9.

(i) For each of the first 5 hits, write down the values of `left`, `right` and `mid` that can be seen in the debugger. [5%]

(ii) What will be printed out? [1%]

(b) Consider the truth of the following conditions at the execution point immediately before the loop test is evaluated in line (3). (They are a loop invariant.)

```
left <= right
left*left <= n
(right+1)*(right+1) > n
```

(i) Why are these conditions true on each iteration? (Your answer should apply to every possible use of the code, not just to the example in part (a).) [3%]

(ii) If the conditions are still true when looping finishes, explain how they ensure that `left` is the answer required. [2%]

(c) For the loop to make progress, on each iteration either `left` or `right` must change.

(i) When does this fail to happen? [1%]

(ii) Suppose lines (4) and (5) are changed to

```
int mid = (left + right + 1)/2;
                  //now left < mid <= right
```

How does this affect the situation? [1%]

(iii) In addition to the change in (ii), what change to *one* other line is needed to make the code correct? After the changes, why are the conditions in part (b) still a loop invariant? [2%]

3 (a) Every array has a list of elements.

(i) Explain the difference between the *index* and the *value* of an array element.
(ii) If the array has length `n`, what indexes are allowed?
(iii) If the type of the array is array of double, what are the types of the indexes and of the values? [3%]

(b) A static method `newArray` is required, that constructs, initializes and returns an array. It has one parameter `n`, which will be the length of the array. The type of the returned result is array of integer. Its elements are initialized as {0, 1, 2, ...}: in other words, at index `i` the value must be `i`.

(i) Complete this definition of `newArray`: [5%]

```
public _______  ________  newArray (_____ n) {

   _______ result = ______;

   for (_____ ; _____ ; _____) {
      result[i] = i;
   }
   return result;
}
```

(ii) Write Java code that declares an integer array variable `a`, and then uses a call of `newArray` to assign to `a` an array with element values {0, 1, 2, 3, 4}.
(*Do not* use "`= {0,1,2,3,4}`" in your code.) [2%]

(c) (i) What does it mean if the header comment for `newArray` includes this line? [1%]

```
requires n >= 0
```

(ii) Write a complete *defensive* header comment for `newArray`, to specify that it throws an illegal argument exception if `n` < 0. What would you need to include in the Java code to make this happen?
(You will gain an extra mark if you use correct Javadoc. Remember that a header with a "requires" comment is *non*-defensive.) [4%]

4 (a) A class `C` is to have a private integer field `val`, a constructor that initializes `val` from a parameter, and public getter and setter methods `getVal` and `setVal`. It should also have a `toString` method that returns a string that shows the value of `val`.

Write a Java definition of the class `C`. [6%]

(b) A class `D` is to be a subclass of `C`. It has an extra private integer field `inc`, and a constructor that initializes `val` to 0 and `inc` from a parameter. It also has a public method `increment` that changes `val` by adding `inc` to it. It should override `toString` to show the values of both `val` and `inc`.

Write a Java definition of the class `D`. [5%]

In parts (a) and (b), remember that the fields are private.

(c) Consider the following piece of Java code.

```
C x1 = new C(3);
D x2 = new D(5);
C x3 = x2;
x2.increment();
System.out.println(x1.toString());
System.out.println(x2.toString());
System.out.println(x3.toString());
```

(i) How many instances of `C` and of `D` are created by this code? [1%]
(ii) What will the code print out? [3%]

5 In question 3, the method `newArray` could only initialize the array values as 0,1, 2, ... in that order. Suppose it is now desired to initialize the values in a more flexible way. The plan is this. `newArray` should have a second parameter `f` of type `Filler`, where `Filler` has a method `element`, with an integer parameter and integer result, and `f.element(i)` is used to calculate the initial array element value for index `i`.

(a) Write an *interface* `Filler`. [2%]

(b) Describe the changes needed in the code for `newArray`. [2%]

An *arithmetic progression* is a sequence of numbers (in our case integers) such as

4, 7, 10, 13, 16, 19, 22, ...

or

5, 3, 1, -1, -3, -5, -7, ...

that goes up (or down) by the same amount each time. So to describe an arithmetic progression, you just have to specify two integers: (i) the *start* of the sequence, and (ii) its *increment*, i.e. how much it goes up each time. (The second example has a negative increment, of –2.)

(c) Write a class `Linear`, implementing `Filler`, so that each instance `f` represents an arithmetic progression given by

```
f.element(0), f.element(1), f.element(2), ...
```

[6%]

(d) Write code that creates a suitable instance of `Linear`, uses it as a parameter in a call of `newArray` in order to create an array with elements {4,7,10,13,16}, and then assigns it to an array variable `a`. [2%]

(e) Write a class `ZeroStartLinear` that is a subclass of `Linear` and *inherits* (without overriding) its `element` method. Every `ZeroStartLinear` instance must have 0 as the start of its arithmetic progression. [3%]

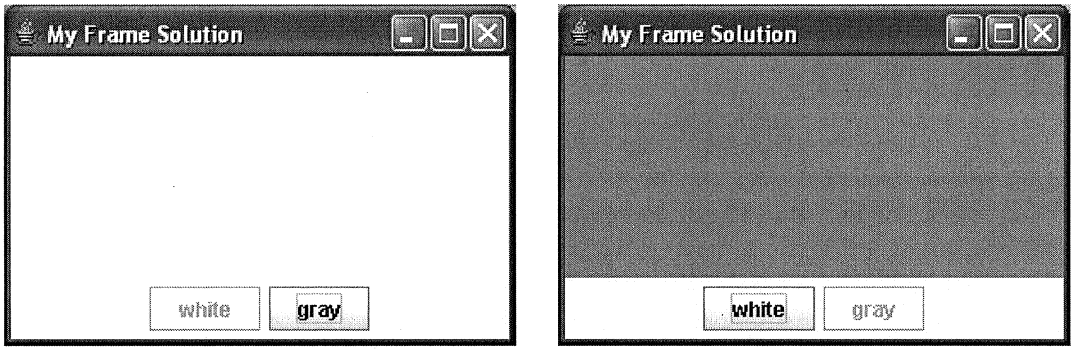
Section B

6. (a) The Java class `MyFrame` includes the code listed below. It describes a blank window that responds to mouse clicks:

```
public class MyFrame extends Jframe
                     implements MouseListener
{
   private Color currentColour = Color.white;
   private JPanel mainPanel;

   public MyFrame()
   {
      setTitle("Colour Frame");
      setSize(300, 200);

      mainPanel = new JPanel();
      mainPanel.setBackground(Color.white);
      mainPanel.addMouseListener(this);
      add(mainPanel);  // default is "Center"
   }

   public void mouseClicked(MouseEvent e)
   {
      if (currentColour.equals(Color.white))
      {  mainPanel.setBackground(Color.gray);
         currentColour = Color.gray;  }
      else
      {  mainPanel.setBackground(Color.white);
         currentColour = Color.white;  }
   }

   public void mouseEntered(MouseEvent e) {}
   public void mouseExited(MouseEvent e)  {}
   public void mousePressed(MouseEvent e) {}
   public void mouseReleased(MouseEvent e) {}
}
```

Use this class to explain how mouse events are handled in Java. Your answer should explain the roles of the interface `MouseListener`, the method `addMouseListener` and the `MouseListener` methods such as `mouseClicked`. [5%]

(b) Rewrite the above class so that the colouring of the background of the JPanel is now controlled by two buttons in a separate panel below the main panel. The window should appear in one of the two states illustrated below:



The colour of the main panel can now be changed by clicking the appropriate button. The button corresponding to the current panel colour should always be disabled and the other should be enabled.

[7%]

(c) Explain the differences between the following two fragments of code for reading 100 integers from a file and adding them together. Your explanation should include a description of the nature of the data in the file being read in each case.

(i)

```
int total=0;
BufferedReader in = new BufferedReader(
                        new FileReader("DataFile1"));
for (int count=1; count<=100; count++) {
   String line = in.readLine();
   int nextInt = Integer.parseInt(line);
   total += nextInt;
}
```

(ii)

```
int total=0;
DataInputStream in = new DataInputStream(
                    new FileInputStream("DataFile2"));
for (int i=1;  i<=100;  i++) {
   int nextInt = in.readInt();
   total += nextInt;
}
```

[4%]

7. A Java class is required that describes a data structure that can be used to store a collection of a large number of strings. The collection will be a *set* in the sense that it will not contain any duplicate entries (i.e. two strings for which the `compareTo` method returns 0). The following interface specifies the functionality required of our string set class and any class we define must implement this interface:

```
/** Describes the functionality required to store a
 *     set of strings.
 */
public interface StringSet
{
   /** Tests whether the set contains a given string
    */
   public boolean contains(String string);

   /** Adds a new string to the set,
    *      ONLY if not there already
    * @return true if the string was added,
    *         false if not.
    */
   public boolean add(String string);

   /** Reports the number of strings stored in the
    *    set */
   public int size();
}
```

(a) Given that we require a rapid response to calls of the `add` and `contains` methods, together with the additional requirement of a `toString()` method that lists the words in alphabetical order, explain, *without writing any code at this stage*, why a binary tree would be appropriate for implementing this data structure. [4%]

(b) Sketch a diagram illustrating what such a tree structure would look like when the following strings have been added to it in the following order:

```
"lion", "frog", "dog", "rat", "unicorn", "duck",
"snail", "wolf", "pig", "mouse", "hare", "cat"
```

[4%]

(c) You are given the following class for representing a node in an ordered binary tree:

```
class TreeCell
{
   String data;
   TreeCell leftBranch, rightBranch;

   TreeCell(String d, TreeCell l, TreeCell r)
   {  data = d;
      leftBranch = l;
      rightBranch = r;
   }

   TreeCell(String d)
   {  this(d,null,null);
   }
}
```

Using this class, write the code for a class TreeStringSet that implements the StringSet interface and uses an ordered binary tree to store the words. You need *not* write the toString method that was mentioned above. Do *not* use any of the Java Collection classes in your answer to this part.

[9%]

End of Paper