A20487

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator permitted in this examination

THE UNIVERSITY OF BIRMINGHAM

Degree of BSc with Honours
Artificial Intelligence and Computer Science. First Examination
Computer Science with Business Management. First Examination
Computer Science. First Examination

Degree of BEng with Honours
Computer Science/Software Engineering. First Examination

Joint Degree of BEng/MEng with Honours
Electronic and Software Engineering. First Examination
Computer Science and Civil Engineering. Final Examination

Combined Degree of BA with Honours
Ancient History and Archaeology and Computer Studies. First Examination
Computer Studies and French. First Examination

06 18190

Software Workshop 1

Thursday 25th May 2006 1400 hrs – 1700 hrs

[Answer ALL Questions]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator

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%]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator

2 (a) The following method is intended to calculate the maximum value in an integer array, assumed non-empty. It compiles without error.

```
static int max(int[] a) {
    int i = 0;
    int m = 0;

    /* loop invariant:
     * 0 <= i <= a.length
     * m = maximum of first i elements of a
     */
    while (i <= a.length) {
        if (a[i+1] > m) {
            m = a[i];
        }
        i = i+1;
    }
    return m;
}
```

The method contains **three** mistakes that stop it calculating the correct result. What are those mistakes, and how should they be corrected?
For full marks, you should explain how the mistakes conflict with the loop invariant and your corrected version should be compatible with the loop invariant as given.

[11%]

(b) Suppose that in some class a method has been declared as

```
public static int value(int n)
```

and sometimes the method throws an `IllegalArgumentException`.

Write a method

```
static void printValue(int n)
```

This should print the result of `value(n)` to `System.out`, unless `value(n)` throws an `IllegalArgumentException`, in which case it should instead print `"exception thrown"`.

[4%]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator

3. It is intended to sum the elements of an integer array "a" by adding them in order (starting with the smallest index) into a variable "sum". A variable "i" will be used to count the number of elements that have been added into sum so far. A suitable loop invariant can be written in the form –

```
0 <= i <= a.length
sum == sum of first i elements of a
```

(a) (i) When i elements have been added into sum (as the invariant states), what is the index of the *next* element to be added in? [1%]

(ii) What is a suitable loop test for a while loop based on this invariant? [1%]

(iii) Explain how the loop invariant tells you that, when the looping has finished, sum must be the required answer. [1%]

(b) Write Java code for a static method sumArray that uses a while loop to implement this algorithm. It should take a as parameter and return the sum as result. Include a Javadoc header, and a comment for the loop invariant. [8%]

(c) Suppose you wished to use the debugger to test the truth of the invariant on each repetition.

(i) On which line would you put the breakpoint? [1%]

(ii) Suppose you did that, and used the debugger to run a main method that just executes the following line, using an empty array.

```
System.out.println(sumArray(new double[0]));
```

How many times would the breakpoint be hit, what would be the values of i and sum each time, and what would be eventually printed out? [3%]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator

4. A public class `Counter` is to have a private integer field "`n`", initialized from a constructor parameter, a public getter `getN` for `n`, and a public method `inc` that adds 1 to `n` and returns no result.

(a) Write the entire Java definition of the class `Counter`. [4%]

(b) Answer the following questions.
(i) What is the name of the file that will contain this class definition? [1%]
(ii) What does it mean to say that `n` is "private"? [1%]
(iii) Write Java code, to be used ouside the file of `Counter`, that will declare a variable `c` of type `Counter`, initialize it to a new instance of `Counter` with `n` taking value 100, increment its `n` field twice, and finally read the value of its `n` field and print it to `System.out`. [3%]

(c) A subclass `BoundedCounter` is required, partly written as

```
public class BoundedCounter extends Counter {

    public static final int MAX_N = 1000;
    // invariant: n <= MAX_N;

    //constructor and inc needed here

}
```

It should behave just like `Counter`, except that it never allows `n` to exceed 1000. Complete this by writing Java definitions for the constructor and `inc`. [5%]

(d) How does your answer to (c) ensure that the invariant condition is always true? [1%]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator

5. A *permutation* of a string is a string with the same characters, but possibly in a different order. For example, the permutations of `"fly"` are

```
"fly", "fyl", "lfy","lyf","yfl","ylf"
```

Each permutation of `w` starts with one of the characters (could be any) in `w`, and then follows with a permutation of the remaining characters in `w`. For `"fly"` we have:

- two permutations with `'f'` followed by each permutation of `"ly"`
- two with `'l'` followed by each permutation of `"fy"`
- two with `'y'` followed by each permutation of `"fl"`

This suggests using a recursive algorithm.

(a) Write the whole definition for a class `Perms` that contains static methods as follows. Include Javadoc comments for the class and methods.

`prefixPerms(String prefix, String w)` displays all the permutations of `w` on `System.out`, each one preceded by `prefix`. For example, `prefixPerms("abc", "fly")` should display six strings

```
"abcfly", "abcfyl", "abclfy", ...
```

It works by calling, recursively, `prefixPerms("abcf", "ly")`, `prefixPerms("abcl", "fy")` and `prefixPerms("abcy", "fl")`.

`perms(String w)` uses `prefixPerms` to display all the permutations of `w`. [11%]

(b) Explain how you know `prefixPerms` works. [4%]

For reference, here are some API entries for `String` methods. Individual characters in a string are of type `char`. String addition, +, can be used to add strings and chars.

`public int` **`length`**`()` - Returns the length of this string.

`public char` **`charAt`**`(int index)`
Returns the `char` value at the specified index. An index ranges from 0 to `length() - 1`, as for array indexing.

`public String` **`substring`**`(int beginIndex)`
Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

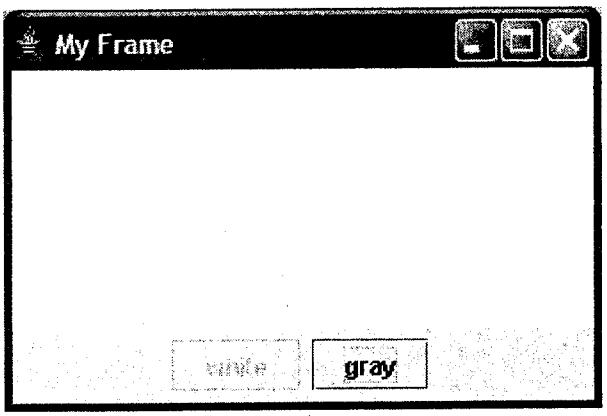
`public String` **`substring`**`(int beginIndex, int endIndex)`
Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator

Section B

6. (a) The Java class partially listed below is for windows like the following:




```
public class MyFrame extends JFrame implements ActionListener {
   JPanel mainPanel;
   JButton whiteButton, grayButton, lastButtonClicked;

   public MyFrame() {
      ... //some code omitted here
      whiteButton = new JButton("white");
      grayButton = new JButton("gray");
      whiteButton.addActionListener(this);
      grayButton.addActionListener(this);
      mainPanel = new JPanel();
      mainPanel.setBackground(Color.white);
      whiteButton.setEnabled(false);
      lastButtonClicked = whiteButton;
      ... //more code omitted here
   }

   public void actionPerformed(ActionEvent e) {
      lastButtonClicked.setEnabled(true);
      if (e.getActionCommand().equals("white"))
      {  mainPanel.setBackground(Color.white);
         lastButtonClicked = whiteButton;  }
      else if (e.getActionCommand().equals("gray"))
      {  mainPanel.setBackground(Color.gray);
         lastButtonClicked = grayButton;  }
      lastButtonClicked.setEnabled(false);
   }
}
```

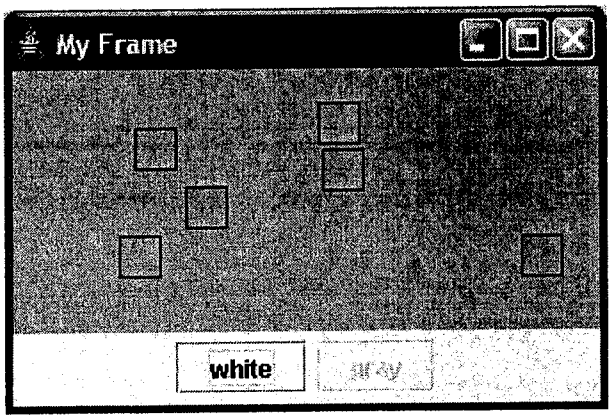
Using this class to illustrate your answer, explain how button events are handled in Java. Your answer should explain the roles of the interface `ActionListener`, the method `addActionListener` and the method `actionPerformed`.

[5%]

(b) Write a new class, `MyPanel`, to be used in place of the Java Swing class `JPanel` in the assignment:

```
mainPanel = new JPanel();
```

When the user of the program clicks the mouse at a point on the panel described by your class, the program should draw a square of size 20×20 pixels centred on the point at which the mouse was clicked. After several mouse clicks on the `MyPanel` object, the `MyFrame` object might look like this:



[7%]

(c) Explain the difference between the outputs of the following two fragments of code for outputting an int `i` to a file

(i)
```
PrintWriter outfile = new PrintWriter(
                      new FileWriter("ints.txt"));
outfile.print(i);
```
(ii)
```
DataOutputStream out = new DataOutputStream(
                       new FileOutputStream("ints.dat"));
out.writeInt(i);
```

[4%]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

No calculator

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. The following interface specifies the functionality required of our string set class. 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) What is a *hash table*? [4%]

(b) Sketch a diagram illustrating what a hash table would look like if used to implement `StringSet`. [4%]

(c) In this part, you may assume that a class called `LinkedListStringSet` has already been written. A `LinkedListStringSet` object is a linked list of Strings and it provides all the functionality specified by our `StringSet` interface. The class has the heading:

```
public class LinkedListStringSet implements StringSet
```

Write a class `HashStringSet` that implements the `StringSet` interface and uses a hash table to store the strings. Do *not* use any of the Java Collection classes in your answer to this part.

[9%]