Computer Science 3718  
Fall 2003  

Final Exam  

December 10, 2003  

Instructor:  
T. Wareham  

NAME: ______________________  STUDENT ID #: ____________

- This exam will be 120 minutes long and is out of 120 marks.
- This exam has 12 pages (including this cover page). There are two questions, both of which have multiple parts.
- Please answer all questions in the space provided on this exam; if you find it necessary to continue an answer on the back of a sheet of paper, that is fine, but please make a note on the front side, e.g., “answer cont’d on back”.

<table>
<thead>
<tr>
<th>Question</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. a) 6</td>
<td></td>
</tr>
<tr>
<td>b) 6</td>
<td></td>
</tr>
<tr>
<td>c) 16</td>
<td></td>
</tr>
<tr>
<td>d) 14</td>
<td></td>
</tr>
<tr>
<td>2. a) 14</td>
<td></td>
</tr>
<tr>
<td>b) 10</td>
<td></td>
</tr>
<tr>
<td>c) 18</td>
<td></td>
</tr>
<tr>
<td>d) 20</td>
<td></td>
</tr>
<tr>
<td>e) 16</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>
1. (42 marks)

   a) (6 marks) Sketch the UML diagram corresponding to the following situation:

   In a hospital ward, there are two types of supply cabinets – medical
cabinets (which contain drugs and medical supplies) and housekeeping

cabinets (which contain bed linens, pillows, and such). Each cabinet is

   located within a fixed ward and each ward has between 5 and 20 cabinets.

   Each room is assigned to a ward and also has an associated housekeeping

   cabinet. Each pharmaceutical cabinet has two associated pharmacists

   who are responsible for the contents of that cabinet. Each ward also has

   a pharmacist on call 24/7 to deal with drug-related emergencies; given

   these hours, a pharmacist can be on call for at most one ward.

   b) (6 marks) Consider the following Java method:

   ```java
   public static int FunkyTown(int a, char c){
       if (a < 2)
           if (c != 'c')
               a = (a * a * a);
           if (a < 42)
               a += 42;
           if (c == 'a')
               return(a);
       else
           return((int) c);
   }
   ```

   Give a set of test cases for this method relative to the structural (clear box) test-case selection criterion that ensures that each possible execution-path through

   this method is executed at least once.
c) (16 marks) Consider the following Java program:

```java
class XObj2 {
    public int a = 1, b = 2;
    public YObj2 w = new YObj2(3);

    public static void main(String[] arg) {
        XObj2 x = new XObj2(4);
        ZObj2 z = new ZObj2(5);
        int a = 6, b = 7;

        x.printVal(); z.printVal();
    }
}

class YObj2 {
    public int a = 9, b = 10;

    public YObj2(int a) {
        System.out.println("Initializing Y (" + a + ")");
        a = a;
    }

    public String val() {
        int b = 11;
        return ("a + " + b + ");
    }
}

class ZObj2 extends YObj2 {
    public YObj2 w = new YObj2(12);
    public int a = 13;

    public ZObj2(int a) {
        super(14);
        System.out.println("Initializing Z (" + a + ")");
    }

    public void printVal() {
        System.out.println("Zver: " + a + " + b + " + w.val());
    }
}
```

Please give the output produced by executing this program.
d) (14 marks) Consider the following Java program:

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

class tf_GUI extends JFrame {
    private JPanel displayPanel1, displayPanel2;

    public tf_GUI() {
        super("GUI: tf_GUI");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.getContentPane().setLayout(new BorderLayout());

        displayPanel1 = new JPanel();
        displayPanel1.setPreferredSize(new Dimension(300, 100));
        displayPanel1.setLayout(new FlowLayout());
        displayPanel1.add(new JLabel("Something for you: "));
        displayPanel1.add(new JButton("Push Me!"));

        displayPanel2 = new JPanel();
        displayPanel2.setPreferredSize(new Dimension(100, 100));
        displayPanel2.setLayout(new BorderLayout());
        displayPanel2.add(new JButton("N"), BorderLayout.NORTH);
        displayPanel2.add(new JButton("S"), BorderLayout.WEST);
        displayPanel2.add(new JButton("W"), BorderLayout.CENTER);

        this.getContentPane().add(displayPanel1, BorderLayout.EAST);
        this.getContentPane().add(displayPanel2, BorderLayout.SOUTH);
        this.getContentPane().add(new JButton("Here!"), BorderLayout.WEST);
    } // End of constructor method

    public void display() {
        this.pack();
        this.show();
    } // End of method display

    public static void main(String[] arg) {
        tf_GUI G;
        G = new tf_GUI();
        G.display();
    } // End of main method

} // End of class tf_GUI
```

Please sketch the frame(s) produced by executing this program.
2. (78 marks)

All parts of this question refer to the classes described in your exam handout. You may assume that there will never be more than one object of a given type with the same information, e.g., there will never be two Patient-objects in the system, each of which describes the patient Todd Wareham; hence, one can check equality of entities by reference equality, i.e., you do not need to check all object-fields for equality.

a) (14 marks) Give the appropriate comments and javadoc tags appearing in the places marked (A)–(D) in the code-skeleton below that are needed to reproduce the comments associated with class LifesignSet and the methods numLifesign, delLifesign, includesLifesign, and getIndLifesign in the printout of the javadoc-produced HTML file given in your exam handout.

```java
public class LifesignSet {

    private Lifesign[] LS = null;

    ...

    (A)

    public int numLifesigns(){
        ...
        ...

    (B)

    public void delLifesign(Lifesign L) throws LifesignAbsentExp {
        ...

    (C)

    public boolean includesLifesign(Lifesign L){
        ...

    (D)

    public int getIndLifesign(Lifesign L){
        ...

}  // End of class LifesignSet
```
b) (10 marks) Give Java code for class PatientInICU. All fields should be accessible only within this class, and all requested methods should be publicly available. You may assume the existence of all invoked exception-classes.
c) (18 marks) Give Java code for class Doctor. All fields should be accessible only within this class, and all requested methods should be publicly available. You may assume the existence of all invoked exception-classes.
d) (20 marks) Give **Java** code for class **DoctorSet**. All fields should be accessible only within this class, and all requested methods should be publicly available. You may assume the existence of all invoked exception-classes.
e) (16 marks) Please give the Java code “skeletons” (class definitions with all fields and the first line of each method) for each of the classes PersonMed, StudentMed, DoctorGP, and DoctorSpec relative to the following restrictions:

- Class PersonMed cannot have associated object-instances.
- Classes DoctorGP and DoctorSpec cannot have subclasses.
- All fields are accessible only within the classes in which they are defined.
- Method getMID is only accessible within class PersonMed and its subclasses.
- All remaining methods are publicly available.
- No method can be overridden.