0. Name (2 pts):

CS 2334: Programming Structures and Abstractions

Midterm Exam II

Solution Set

Monday, November 8, 2010

Problem	Topic	Max	Grade
0	Name	2	
1	Binary I/O	20	
2	Graphical User Interfaces	35	
3	Event-Driven Programming	15	
4	Collections Framework	30	
Total			

1. **Binary I/O** (20 pts)

(a) (10 pts) True or False and briefly explain: an ArrayList is always Serializable.

False. The components of the ArrayList must also be Serializable for this to be true.

(b) (10 pts) List two types of information that are written to an ObjectOutputStream when storing an object.

i. Object meta data: names and types of each of the object's properties.
ii. Object data: the values of the properties.

2. **Graphical User Interfaces** (35 pts)

Consider the following code for a Panel implementation:

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

public class MyPanel extends JPanel
{
    JButton b1 = new JButton("Red");
    JButton b2 = new JButton("Blue");
    JLabel label = new JLabel("Color");
    MyPanel self = this;

    public MyPanel() {
        super();

        // Configure panel
        setLayout(new GridLayout(0,3));

        // Event Listener
        ActionListener listener = new ActionListener() {
                public void actionPerformed(ActionEvent e){
                    String name = e.getActionCommand();
                    if(name.equalsIgnoreCase("Red")) {
                        self.setBackground(new Color(200, 0, 0));
                    }else{
                        self.setBackground(new Color(0, 255, 0));
                    };
                }
        };

        // Attach listener
        b1.addActionListener(listener);
        b2.addActionListener(listener);
    };
}
```

This panel contains two buttons and a text label. You may assume that an instance of this panel is properly attached to a JFrame and is made visible.

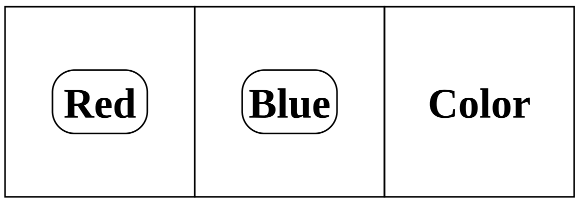
(a) (10 pts) In one sentence, describe the *intended* behavior of this panel (note that there are a couple bugs).

Button presses are responded to by setting the background color of the panel.

(b) (10 pts) For proper behavior, three lines of code are missing from the constructor. What are they?

```
    add(b1);
    add(b2);
    add(label);
```

(c) (5 pts) Assuming the above code addition, draw the layout of the panel including all child components.



(d) (10 pts) True or False and explain: a graphical user interface can make use of several different types of layout managers.

True. Each container has its own layout manager and these can be of different types.

3. **Event-Driven Programming** (15 pts)

(a) (10 pts) (Mad Events) Complete the following sentences:

A listener registers with a source.

A source notifies a listener.

A listener receives/processes an event.

(b) (5 pts) True or False: All Events are related to actions performed by the user.

False. (others are possible: e.g., Timers)

4. Java Collections Framework (30 pts)

Consider the following code:

```
public class Inhabitant implements Comparable<Inhabitant>
{
    public int ID;
    public String name;

    public Inhabitant(int ID, String name) {
        this.ID = ID;
        this.name = name;
    };

    public String toString() {
        return(name + ", " + ID);
    };

    public int compareTo(Inhabitant i) {
        return(this.name.compareTo(i.name));
    };

    // Used for containment checks
    public boolean equals(Object i) {
        return(((Inhabitant)i).ID == this.ID);
    };
};
```

```
import java.util.*;

public class driver {

    public static void displayCollection(Collection c) {
        System.out.println("Inhabitants: ");
        for(Object o: c) {
             System.out.println(o);
        };
    };

    public static void main(String[] args) {
        Collection<Inhabitant> c = new TreeSet<Inhabitant>();
        c.add(new Inhabitant(1138, "THX"));
        c.add(new Inhabitant(3417, "LUH"));
        c.add(new Inhabitant(1042, "THX"));
        c.add(new Inhabitant(5241, "SEN"));
        displayCollection(c);

        ArrayList<Inhabitant> list = new ArrayList<Inhabitant>(c);

        Collections.sort(list, new Comparator<Inhabitant>() {
                             public int compare(Inhabitant p1, Inhabitant p2){
                                 if(p1.ID < p2.ID) return(1);
                                 if(p1.ID > p2.ID) return(-1);
                                 return(0);
                             };
             });
        displayCollection(list);
        System.out.println("Search: " + list.contains(new Inhabitant(3417, "SEN")));
    };
}
```

(a) (20 pts) What output does the program produce?

```
Inhabitants:
LUH, 3417
SEN, 5241
THX, 1138
Inhabitants:
SEN, 5241
LUH, 3417
THX, 1138
Search: true
```

(b) (10 pts) Briefly explain a reason for using a **LinkedList** instead of a **TreeSet** for Collection **c**.

In general, names may not be unique. A LinkedList would preserve all entries that are added to the collection.