Purpose: This exercise is to expand your knowledge of Java applets by introducing you to Buttons, LayoutManagers and Panels. Buttons are simply the icons you see on a screen that you can click with the mouse. LayoutManagers control how objects such as Buttons are laid out on the screen. Panels are containers for holding objects such as Buttons. They help us subdivide our applet region into different areas so we can have more control over where objects are placed.

To Receive Credit: Show TA your working program before leaving class.

Activities:

1. Create a new project named with your Virginia Tech PID in CodeWarrior. You will create a Java applet named RoomMonitorInterface (be sure to select applet rather than application at the appropriate point). Remove the files TrivialAppletDebug.html, TrivialApplet.html, and TrivialApplet.java from the project sources.

2. Add the web page to drive the RoomMonitorInterface applet. Create a file named RoomMonitorInterface.html. Type the following html into the editor window that appears and add the file to your project.

   ```html
   <html>
   <head>
   <title>Lab 5</title>
   </head>
   <body>
   <hr>
   <applet codebase="Java Classes" code="RoomMonitorInterface.class" width=200 height=200></applet>
   <hr>
   </body>
   </html>
   ```

3. We will be creating an applet that looks like this:

   ![Applet View](image)

   This is an interface for a Room Monitor that displays the number of people in a room, the maximum number of people that can be in the room, and the total number of people that have entered the room. Our applet will have three buttons that can be clicked on: an “Enter” button that increments the number of people in the room as well as the total number of people that have entered the room, a “Reset” button that resets the number of people in the room to 0000, and an “Exit” button that decrements the number of people in the room. These buttons will not do anything in this version of the applet.
To get started, create a Java applet file named `RoomMonitorInterface.java` and type in the following code.

```java
import java.awt.*;
import java.applet.*;

public class RoomMonitorInterface extends Applet {
    public void init() {
        // declare and create Button and Label objects
        // add statements
    }
}
```

a. First, add declarations for three `Button` reference variables, and statements to create one `Button` for each of the actions enter, reset, and exit. Use the labels: “Enter”, “Reset”, and “Exit”, and choose descriptive names for the variables. Recall that the `Button` constructor takes a `String` object for the label.

b. Next create three `Label`s and give them text to display: “Current”, “Max”, and “Total”. A `Label` is created in the same way a `Button` is:

```java
Label currentLabel = new Label("Current");
```

c. To display the number of people currently in the room, the maximum that can be in the room and the total number of people that have entered the room, create 3 other `Label`s. Set these labels to contain 0080 for the current number of people in the room, 0150 for the max and 0245 for the total. Make sure you put the leading 0’s in front of the numbers so they’ll line up properly.

d. The steps above have created all of the `Label` and `Button` objects needed for the display. We can add these objects to the current applet by using the `add` command. For each `Button` `b` and `Label` `l` that you have declared, include lines like

```java
add(b);
add(l);
```

You should now compile and run the applet to see the layout of the objects on the screen. It should not look the same as the example. Pay attention to how the objects are presented relative to the order in which you added them to the applet.

4. The previous step builds an applet with the needed components for our program, but as you can see the layout of the buttons and labels is not organized as given in the example. To organize the objects in the display, `Panel` and `LayoutManager` objects can be used. Remove the `add` commands before continuing.

a. Declare two `Panel` variables and create two `Panel` objects: one to hold the labels and one to hold the buttons. The `Panel` constructor takes no arguments.

```java
Panel newPanel = new Panel();
```

Name the panel variables `displayPanel` and `controlPanel`. The `displayPanel` will contain the labels that will display the number of people in the room at various times and the `controlPanel` will contain the buttons that allow us to increment, decrement or reset that information.

b. The displays are organized using layout manager objects. In particular, you will need to organize the layout of the panels in the applet window, and the buttons and labels in the panels. To do this, use the following commands.

```java
setLayout(new BorderLayout());
displayPanel.setLayout(new GridLayout(3, 2));
```

The first command determines the layout of the applet window so that it has five regions: north, south, east, west, and center. The second command determines the layout of the panel of labels so that it consists of area of three rows and two columns on the screen.
c. The next step is to add the labels and buttons to the panels, and the panels to the applet. To add a button to a Panel, use a command similar to the following (you will need to use your variable names):
   
   ```java
   nameofPanel.add(buttonOrLabelToAdd);
   ```

   Add the labels you’ve created to the `displayPanel`. The GridLayout layout manager will add the labels row by row with two columns in each row as specified in the first two arguments to the constructor. So the first two labels that you add will go in the first row, the next two labels that you add in the second row and so on. We want the Labels displaying the information regarding the current number of people in the room to match up with the Label that indicates that what the information is. In other words, we want the Label labeled “Current” to be beside the label that displays the number of people currently in the room (0085).

d. Now that you have added the labels to the `displayPanel`, you can put the `displayPanel` on the applet. To do this, use the following code:

   ```java
   add(displayPanel, "Center");
   ```

   This will add the `displayPanel` to the center of the applet.

e. Now, add the Buttons you have created to the `controlPanel` in the appropriate order. By default the buttons will be displayed next to each other as desired and so you do not need to setup a layout manager.

f. Finally, add the `controlPanel` in the south portion of the applet using a command similar to that in step d.

If you run the applet, it should now look like the one displayed on page 1.