1. Traffic Control

In this exercise you will develop a traffic light application. The application draws three lights red (top), yellow (middle), and green (bottom) that are turned on in the sequence when a change button is clicked. The application consists of three classes: TrafficControlPanel, TrafficLight, TrafficLightPanel. TrafficLight is the driver class that creates an object of TrafficControlPanel when the application is executed. TrafficControlPanel waits for an event to occur and change the state of the traffic light when an event is received. TrafficLightPanel draws the traffic light and changes the light when an object of TrafficControlPanel is created. A panel of TrafficLightPanel should be placed in a panel of TrafficControlPanel, that is, a traffic control panel is a nested panel.

1. Implement the traffic light application. In the implementation, place the traffic light in the middle and the button at the bottom of the panel (Hint: use BorderLayout.CENTER to add the light panel and BorderLayout.SOUTH for the button). When the application is executed, the initial state of the traffic light should be red. Implement the TrafficControlPanel class with an inner listener class named ChangeListener as shown in section 4.7. Complete file TrafficLightPanel.java. Set the color of lights to darkgray when they are turned off. You will need to use if or switch statements to determine the light to turn on.

2. An inner class is used to facilitate communication between two classes when they have an intimate relationship. Make the inner class (ChangeListener) a regular class by taking it out from TrafficControlPanel and observe the complication involved in making them communicate and the intimacy of the two classes.
public class TrafficLightPanel extends JPanel {
    private int currentState = 0;
    private final int NUM_LIGHTS = 3; // the number of lights
    private final int X = 50, Y = 10, WIDTH = 50, HEIGHT = 130; // box size
    private final int DIAMETER = 30; // light diameter
    private final int X_OFFSET = 10, Y_OFFSET = 10; // offsets to position the lights in the box
    private final int PANEL_WIDTH = 150, PANEL_HEIGHT = 230; // the size of a traffic control panel

    // Creates the traffic light panel
    public void TrafficLightPanel() {
        setPreferredSize(new Dimension(PANEL_WIDTH, PANEL_HEIGHT));
    }

    public void paintComponent(Graphics page) {
        super.paintComponent(page);
        int lightOn = currentState % NUM_LIGHTS;
        setBackground(Color.white);
        page.setColor(Color.lightGray);
        page.fillRect(X, Y, WIDTH, HEIGHT);

        // add code to determine the light to turn on and turn on the light
    }

    public void change() {
        currentState++;
        repaint();
    }
}
Example screen:

Deliverables

- A printout of the complete program and the final execution in question 1.
- A printout of the ChangeListener and TrafficControlPanel classes and the final execution in question 2.
2. Counting Characters

The file Count.java contains the skeleton of a program to read in a string (a sentence or phrase) and count the number of blank spaces in the string. The program currently has the declarations and initializations and prints the results. All it needs is a loop to go through the string character by character and count (update the countBlank variable) the characters that are the blank space. Since we know how many characters there are (the length of the string) we use a count controlled loop—for loops are especially well-suited for this.

1. Add the for loop to the program. Inside the for loop you need to access each individual character—the charAt method of the String class lets you do that. NOTE: You could also directly use phrase.charAt(i) in your if (without assigning it to a variable).

2. Test your program on several phrases to make sure it is correct.

3. Now modify the program so that it will count several different characters, not just blank spaces. To keep things relatively simple we'll count the a's, e's, s's, and t's (both upper and lower case) in the string. You need to declare and initialize four additional counting variables (e.g. countA and so on). Your current if could be modified to cascade but another solution is to use a switch statement. Replace the current if with a switch that accounts for the 9 cases we want to count (upper and lower case a, e, s, t, and blank spaces). The cases will be based on the value of the ch variable.

4. Add statements to print out all of the counts.

5. It would be nice to have the program let the user keep entering phrases rather than having to restart it every time. To do this we need another loop surrounding the current code. That is, the current loop will be nested inside the new loop. Add an outer while loop that will continue to execute as long as the user does NOT enter the phrase quit. Modify the prompt to tell the user to enter a phrase or quit to quit. Note that all of the initializations for the counts should be inside the while loop (that is we want the counts to start over for each new phrase entered by the user). All you need to do is to add the while statement (and think about placement of your reads so the loop works correctly). Be sure to go through the program and properly indent after adding code—with nested loops the inner loop should be indented.

Deliverables

- A printout of the complete program and the final execution.
import java.util.Scanner;

public class Count {
    public static void main (String[] args) {
        String phrase;    // a string of characters
        int countBlank;   // the number of blanks (spaces) in the phrase
        int length;       // the length of the phrase
        char ch;          // an individual character in the string
        Scanner scan = new Scanner(System.in);

        // Print a program header
        System.out.println();
        System.out.println("Character Counter");
        System.out.println();

        // Read in a string and find its length
        System.out.print("Enter a sentence or phrase: ");
        phrase = scan.nextLine();
        length = phrase.length();

        // Initialize counts
        countBlank = 0;

        // a for loop to go through the string character by character
        // and count the blank spaces

        // Print the results
        System.out.println();
        System.out.println("Number of blank spaces: " + countBlank);
        System.out.println();
    }
}

2. Loan Calculation

Download [www.secs.oakland.edu/~l2lu/230/LoanCalculation.zip](http://www.secs.oakland.edu/~l2lu/230/LoanCalculation.zip) and unzip it. You will find a sub-directory named Step3 which contains source codes for LoanCalculator and Loan classes that resulted from the third step during the top-down development of the application. Finish the last two steps. For each step, provide your design, source code and snapshots from test runs.
PROJECT 4 DELIVERABLES

Submit **hard and soft copies to lab assistant.**

- A cover page with the project number, due date, and the names of your Project Team Members.
- Deliverables from exercises 1--3.
- This page, with the appropriate signature and date, indicating that the project has been completely and correctly demonstrated in lab.

LABORATORY SIGNATURE

PROJECT TEAM MEMBERS:

STUDENT NAME _______________________________

STUDENT NAME _______________________________

STUDENT NAME _______________________________

STUDENT NAME _______________________________

LAB INSTRUCTOR SIGNATURE _______________________________

DATE
