ABSTRACT

Road infrastructure has seen consistent improvement in the last few years. Connectivity has improved and road transportation has become a focus of rapid development. Roads are providing better access to services, ease of transportation and freedom of movement to people. But in metropolitan cities traffic congestion is increasing rapidly, it results in chronic situation in dense downtown areas. Traffic signals play a significant role in the urban transportation system. They control the movement of traffic on urban streets by determining the appropriate signal timing settings. Adaptive traffic signal controllers as the principle part of intelligent transportation systems has a primary role to effectively reduce traffic congestion by making a real time adaptation in response to the changing traffic network dynamics. Many methods used for traffic signal timing optimization under different criteria. In this paper different methods are proposed by reviewing different research papers for traffic signal control, which gives best adaptability & optimization ideas in traffic signal control.

In this project we are going to create a Button Group such that it contains three radio buttons Red, Yellow, Green when red is clicked it need to display “Stop!”, when yellow is clicked it need to display “Get Ready to go!”, when the green button is clicked Display “Go!!” and also, we are going to create a sample Traffic signal demo using rectangle and ovals using 2D graphics. Using this software automatically on and off operations are performed based on the timing we set in the software. Green, Red, and orange are lights which are used for traffic controller. Traffic control systems are widely used to monitor and control the flow of automobiles through the junction of many roads.

OBJECTIVE OF THE PROJECT

Roads without any supervision or guidance can lead to traffic conflicts and accidents. Traffic signals are required for an orderly flow of traffic. A traffic signal is used as an instructing device that indicates the road user to act as per the displayed sign. Traffic lights allow everyone to cross the intersection point one by one, reducing conflicts between vehicles entering intersection points from different directions. It provides road safety, also helps to solve traffic in simple manners. There are different colors in traffic lights. Each light has a meaning, and these lights tell drivers what to do. The lights used to monitor the flow of traffic are traffic lights or traffic signals. They are located at intersections and intersections of the road. The various colors of lights instruct riders what can be done next. Light traffic signals play a major role in the traffic flow. In order to preserve decorum, such traffic signals control the movement and flow of traffic.

Red light ON- A driver should stop.

Yellow light ON- A driver has to slow down and be ready to stop.

Greenlight ON- A driver can start driving or keep driving

DEFINITIONS OF THE ELEMENTS THAT ARE USED IN THE PROJECT

JFrame: The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation() method.

JLabel: The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

JButton: The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

JComboBox: JComboBox is a part of Java Swing package. JComboBox inherits JComponent class. JComboBox shows a popup menu that shows a list and the user can select an option from that specified list. JComboBox can be editable or read- only depending on the choice of the programmer.

setBounds( ): Moves and resizes this component. The new location of the top-left corner is specified by x and y, and the new size is specified by width and height.

setDefaultCloseOperation( ): Sets the operation that will happen by default when the user initiates a "close" on this frame. You must specify one of the following choices:

- **DO_NOTHING_ON_CLOSE (defined in WindowConstants):** Don't do anything; require the program to handle the operation in the windowClosing method of a registered WindowListener object.
- **HIDE_ON_CLOSE (defined in WindowConstants):** Automatically hide the frame after invoking any registered WindowListener objects.
- DISPOSE_ON_CLOSE (defined in WindowConstants): Automatically hide and dispose the frame after invoking any registered WindowListener objects.
- **EXIT_ON_CLOSE (defined in WindowConstants):** Exit the application using the System exit method. Use this only in applications.

setLayout( ): Sets the LayoutManager. Overridden to conditionally forward the call to the contentPane.

setVisible(): Shows or hides this Window depending on the value of parameter Boolean b.

ActionListener( ): The listener interface for receiving action events. The class that is interested in

processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

add( ): This method changes layout-related information, and therefore, invalidates the component hierarchy. If the container has already been displayed, the hierarchy must be validated thereafter in order to display the added component.

repaint( ): Repaints this component. If this component is a lightweight component, this method causes a call to this component's paint method as soon as possible. Otherwise, this method causes a call to this component's update method as soon as possible.

addItem( ): Adds an item to the item list. This method works only if the JComboBox uses a mutable data model.

setText( ): Defines the single line of text this component will display. If the value of text is null or empty string, nothing is displayed.

DESIGN

SCREENS:

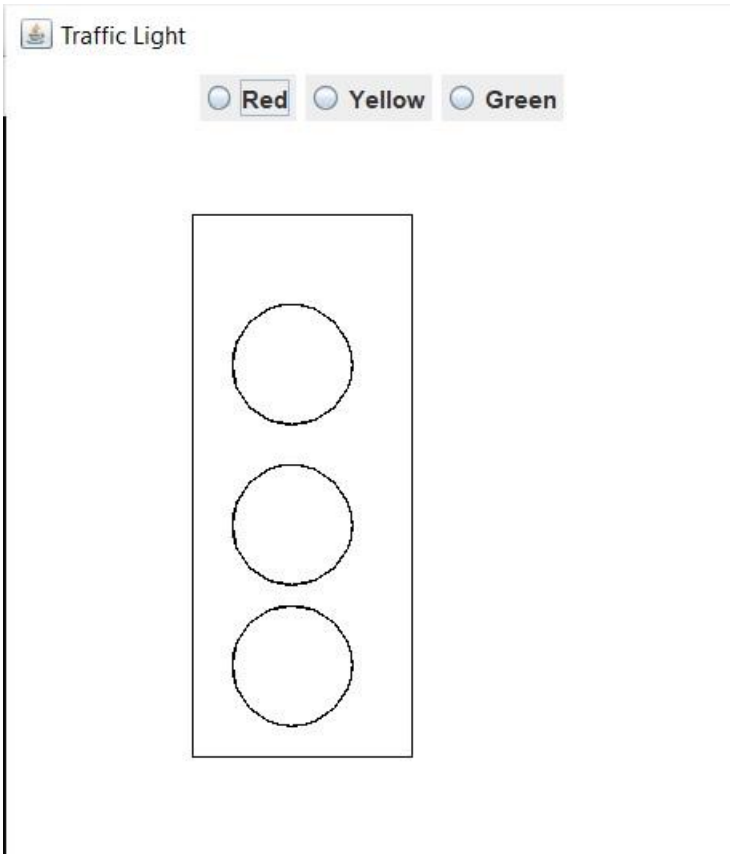

Fig: Initial output of project

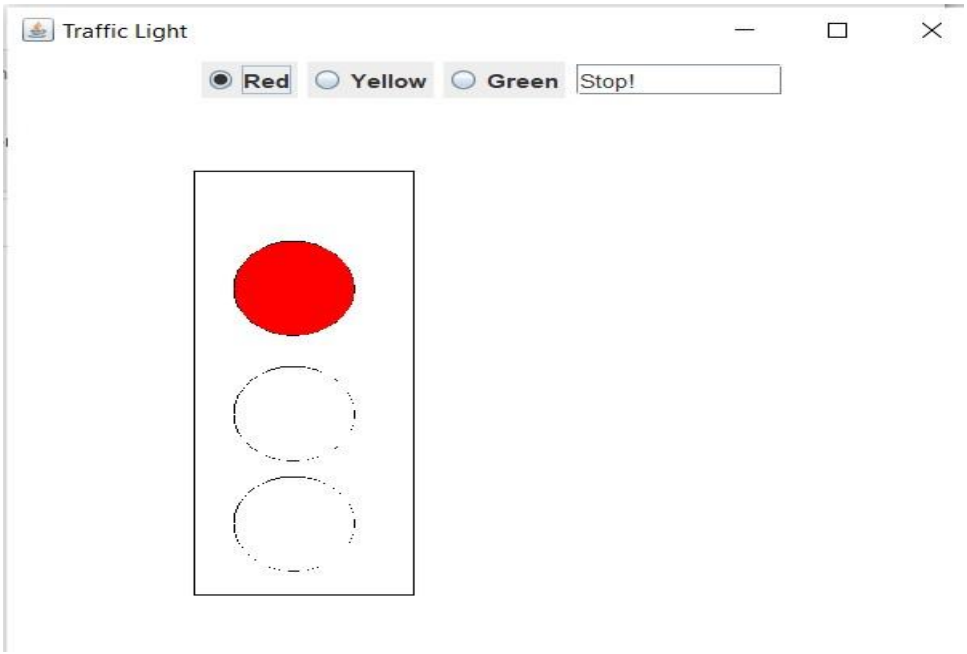

Fig: Output which shows the RED signal

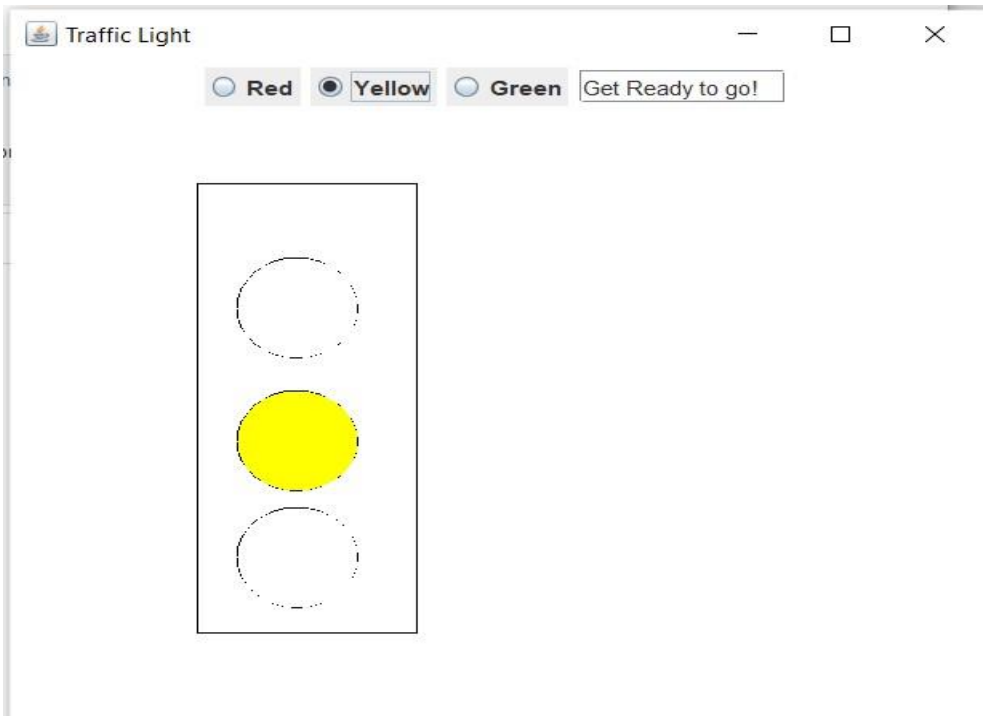

Fig: Output which shows the YELLOW signal

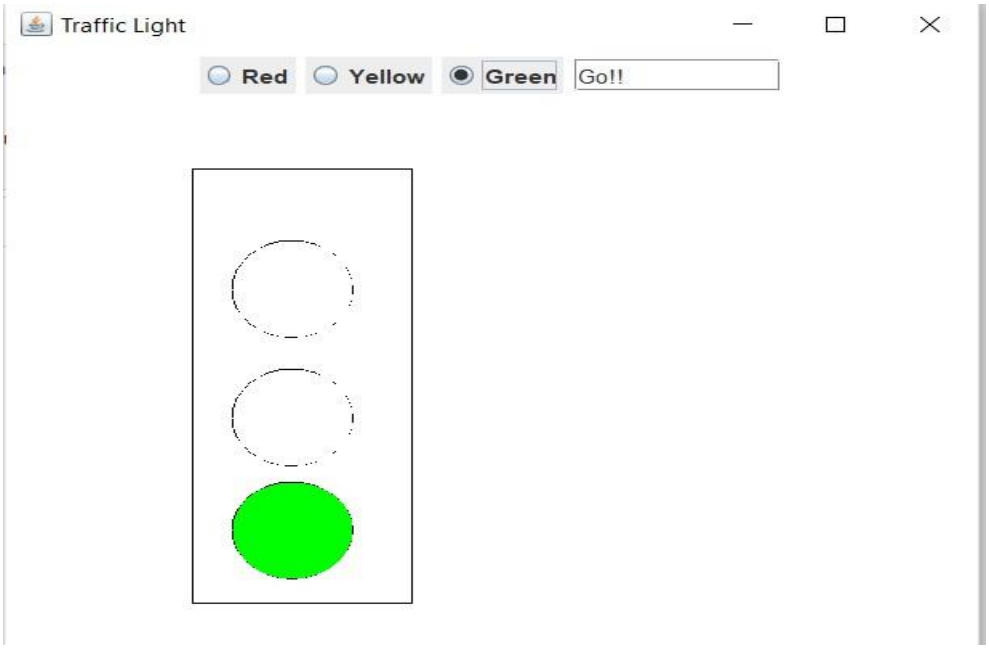

Fig: Output which shows the GREEN signal

IMPLEMENTATION

CODE:

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
// Main class
// Extending JFrame class and
// Implementing ItemListener interface
public class Traffic_Signal
 extends JFrame implements ItemListener {
 // Setting the buttons for the layout
 JRadioButton jr1;
 JRadioButton jr2;
 JRadioButton jr3;
 // Setting the field area
 JTextField j1 = new JTextField(10);
 ButtonGroup b = new ButtonGroup();
 String msg = " ";
// Initially setting the co-ordinates to 0,0,0
 int x = 0, y = 0, z = 0;
 public Traffic_Signal(String msg)
 {
 super(msg);
 setLayout(new FlowLayout());

// Assigning name to the button declared above
 // with help of JRadioButton class
 jr1 = new JRadioButton("Red");
 jr2 = new JRadioButton("Yellow");
 jr3 = new JRadioButton("Green");
 jr1.addItemListener(this);
 jr2.addItemListener(this);
 jr3.addItemListener(this);
 add(jr1);
```

```
add(jr2);
add(jr3);
b.add(jr1);
b.add(jr2);
b.add(jr3);
add(j1);
// Method 1
// To add a window
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e)
{
// It haults here itself
System.exit(0);
}
});

}
// Method 2
// To change colors of traffic signal
public void itemStateChanged(ItemEvent ie)
{
// If it is red
if (ie.getSource() == jr1) {
if (ie.getStateChange() == 1) {
// Then display message- Stop
msg = "Stop!";
x = 1;
// Repainting the box with original one
// Practically black
repaint();
}
else {
msg = "";
}
}
// If state is Orange or technically jr2
```

```
if (ie.getSource() == jr2) {
if (ie.getStateChange() == 1) {
// Then display message-
// Get ready in waiting state

msg = "Get Ready to go!";
y = 1;
// Again repainting the button
repaint();
}
else {
msg = "";
}
}
// If state is Green
if (ie.getSource() == jr3) {
if (ie.getStateChange() == 1) {
// Then display message- Go
msg = "Go!!";
z = 1;
repaint();
}
else {
msg = "";
}
}
j1.setText(msg);
}
// Method 3

// handling the paint graphics and
// dimensions of the buttons via
// setting co-ordinates
public void paint(Graphics g)
{
g.drawRect(100, 105, 110, 270);
```

```
g.drawOval(120, 150, 60, 60);
g.drawOval(120, 230, 60, 60);
g.drawOval(120, 300, 60, 60);
// Case: Red
if (x == 1) {
g.setColor(Color.RED);
g.fillOval(120, 150, 60, 60);
g.setColor(Color.WHITE);
g.fillOval(120, 230, 60, 60);
g.setColor(Color.WHITE);
g.fillOval(120, 300, 60, 60);
x = 0;
}
// Case: Orange
if (y == 1) {
g.setColor(Color.WHITE);
g.fillOval(120, 150, 60, 60);
g.setColor(Color.YELLOW);

g.fillOval(120, 230, 60, 60);
g.setColor(Color.WHITE);
g.fillOval(120, 300, 60, 60);
y = 0;
}
// Case: Green
if (z == 1) {
g.setColor(Color.WHITE);
g.fillOval(120, 150, 60, 60);
g.setColor(Color.WHITE);
g.fillOval(120, 230, 60, 60);
g.setColor(Color.GREEN);
g.fillOval(120, 300, 60, 60);
z = 0;
}
}
// Method 4
```

```
 // Main driver method
 public static void main(String args[])
 {
 // Creating object of Jframe class inside main()
 // method
 JFrame jf = new Traffic_Signal("Traffic Light");

// Setting size and making traffic signal
 // operational using setVisible() method
 jf.setSize(500, 500);
 jf.setVisible(true);
 }
}
```

RESULT SCREEN

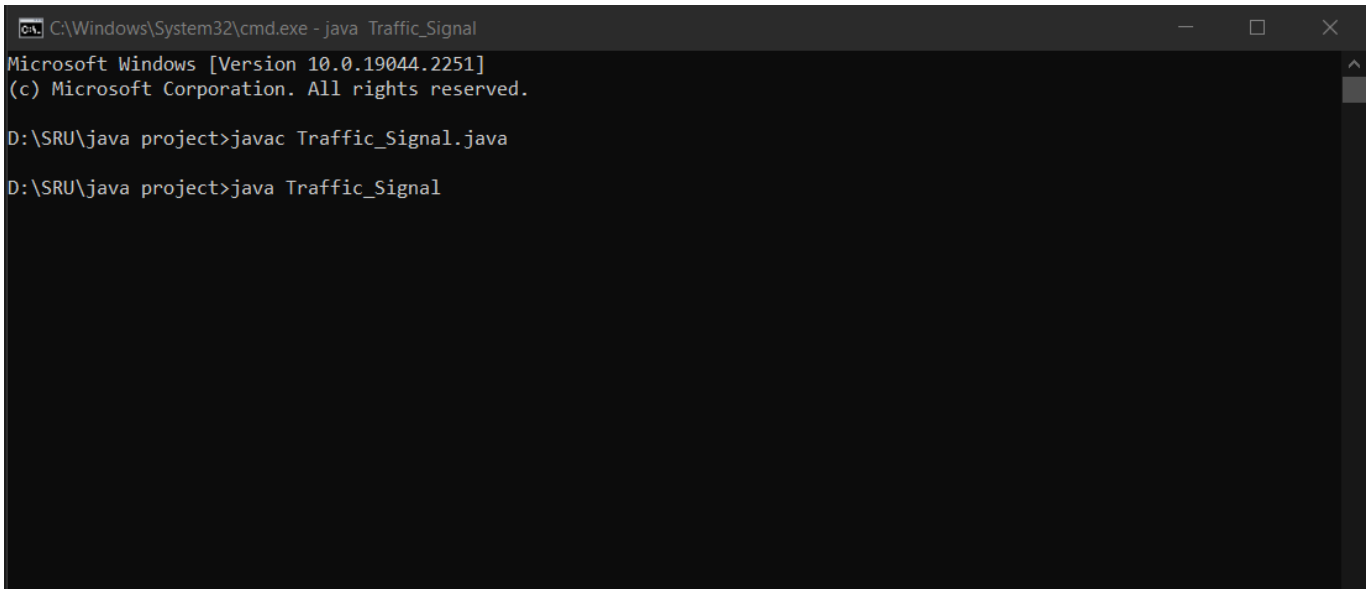

Fig: Compilation of project

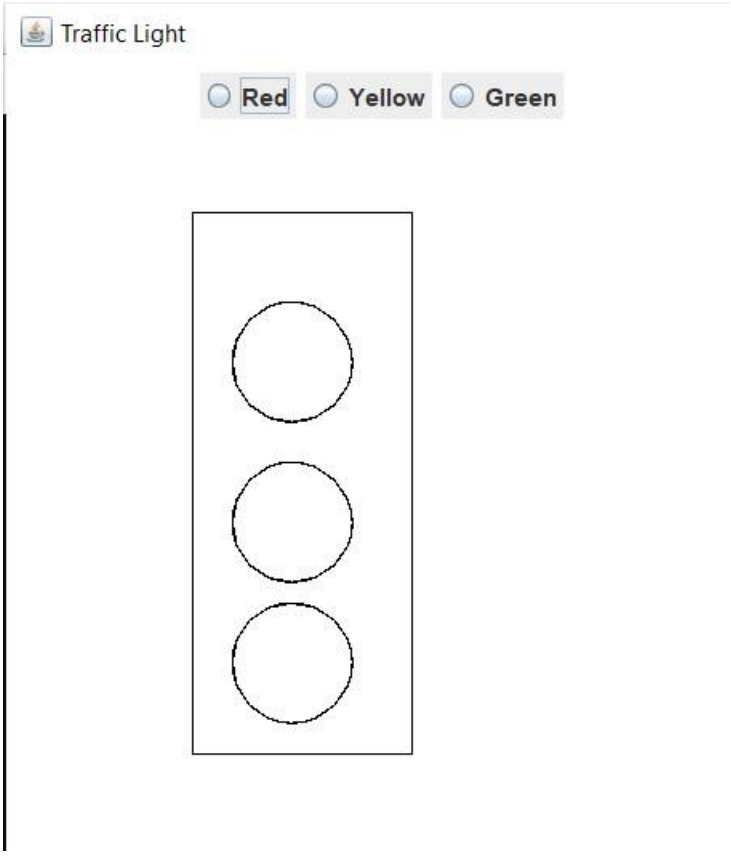

Fig: Initial output of project

This is the output of project, when code is compiled. Until we give any signal it remains in the same way

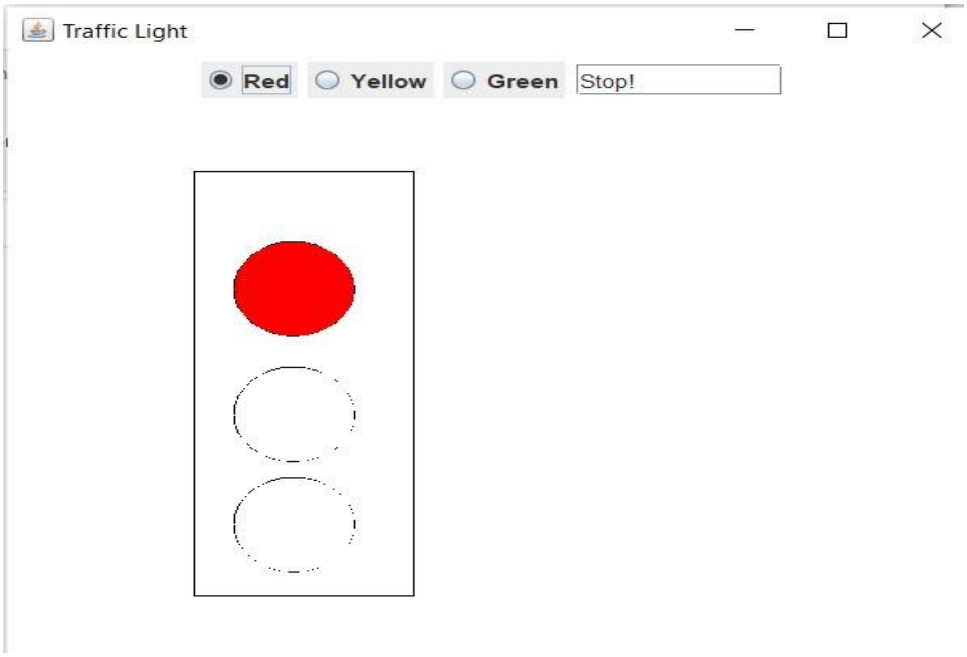

Fig: Output which shows the RED signal

The above diagram shows us the red signal. Which means that, it tells us to stop! It is the intimation for travelers to stop for the time being

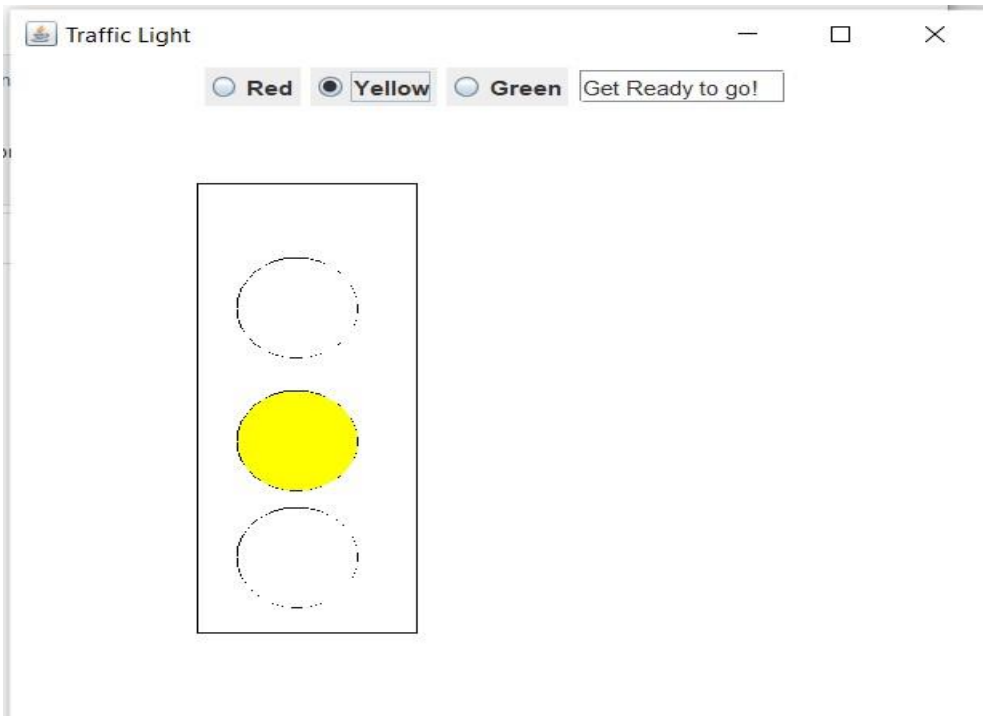

Fig: Output which shows the YELLOW signal

The above diagram shows us the yellow signal. Which means that, it is instructing us to get ready for going.

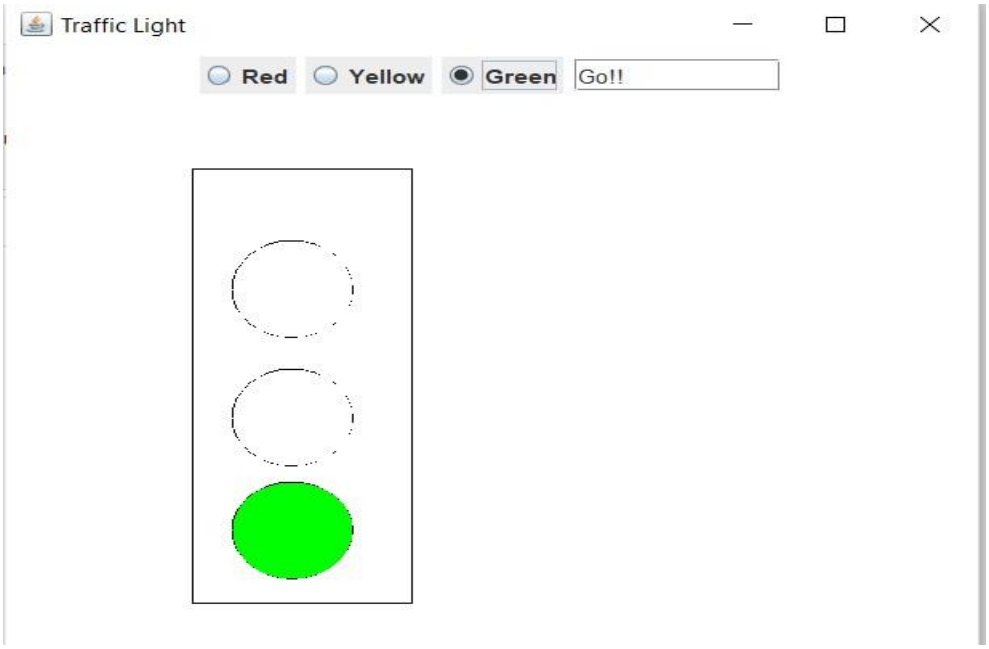

Fig: Output which shows the GREEN signal

The above diagram shows us the green signal. Which means that, it tells us to go!!. It is the intimation for travelers to carry on their work.

CONCLUSION

Traffic jam is the common phenomena in city area. Traffic jam is obstructing for trade and commerce also waste valuable time. The main reason of traffic jam can be not maintained traffic rules, faulty traffic signaling systems. Faulty traffic signaling systems, inadequate manpower and narrow road spaces and overtaking tendency of drivers create pro-longed traffic congestions and intensify sufferings of commuters keeping people motionless as well as creating suffocating condition in the streets. Also, there are bus terminals not authorized by the traffic department and drivers do not go by traffic rules. VIP protocol maintaining is another reason for frequent traffic jams in the streets and divider problem in the city's different important roads also causes congestion. So, if we want to overcome this problem, we must install a modern traffic controller system also grow up the tendency of maintaining traffic rules.