**EXPERIMENT 1:**

**Use Eclipse or Net bean platform and acquaint yourself with the various menus. Create a test project, add a test class, and run it. See how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods, and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.**

1)Creating test class in test project:

**package** test;

**public** **class** test {

**public** **static** **void** main(String[] args)

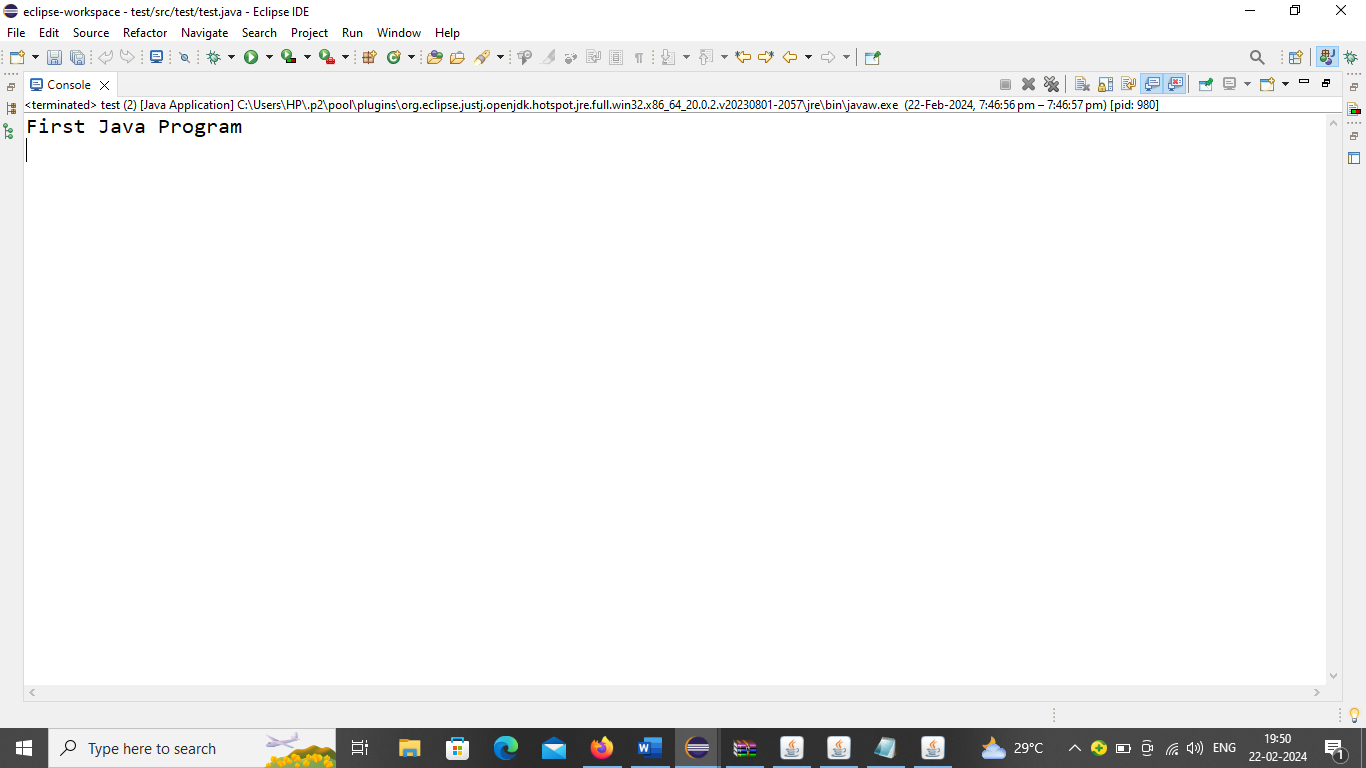
{

System.***out***.println("First Java Program");

}

}

**OUTPUT:**



2)Prime Numbers:

**package** Package1;

**public** **class** Prime{

**public** **static** **void** main(String[] args) {

**int** n=10;

System.***out***.println("Prime numbers from 1 to " + n + ":");

**for**(**int** i=1;i<=10;i++)

{

**int** count=0;

**for**(**int** j=1;j<=i;j++)

{

**if**(i%j==0)

{

count++;

}}

**if**(count==2)

{

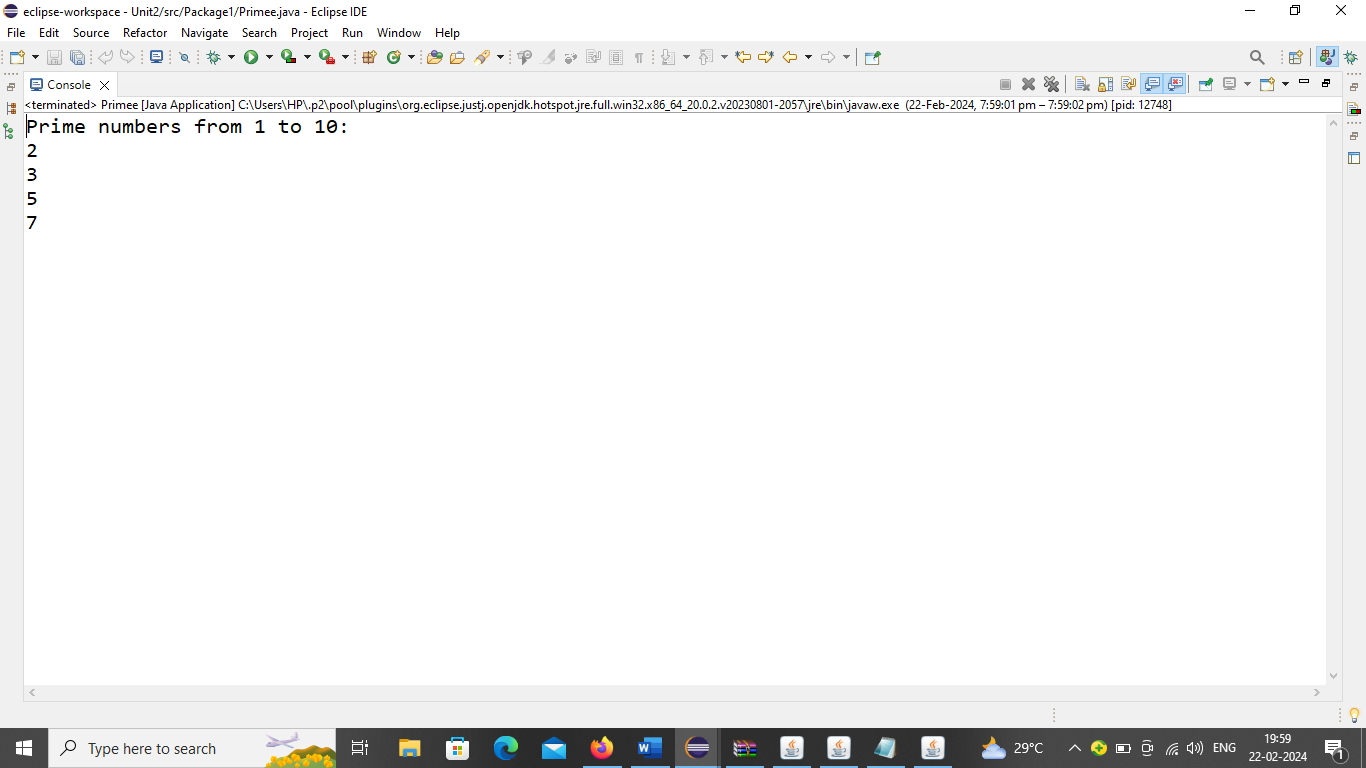
System.***out***.println(i);

n++;

}

}}}

**OUTPUT:**



3)Fibonacci Series:

**package** Package1;

**public** **class** fib {

**static** **int** fib(**int** n)

{

**if**(n<=1)

**return** n;

**return** *fib*(n-1)+*fib*(n-2);

}

**public** **static** **void** main(String args[])

{

**int** N=10;

**for**(**int** i=0;i<N;i++)

{

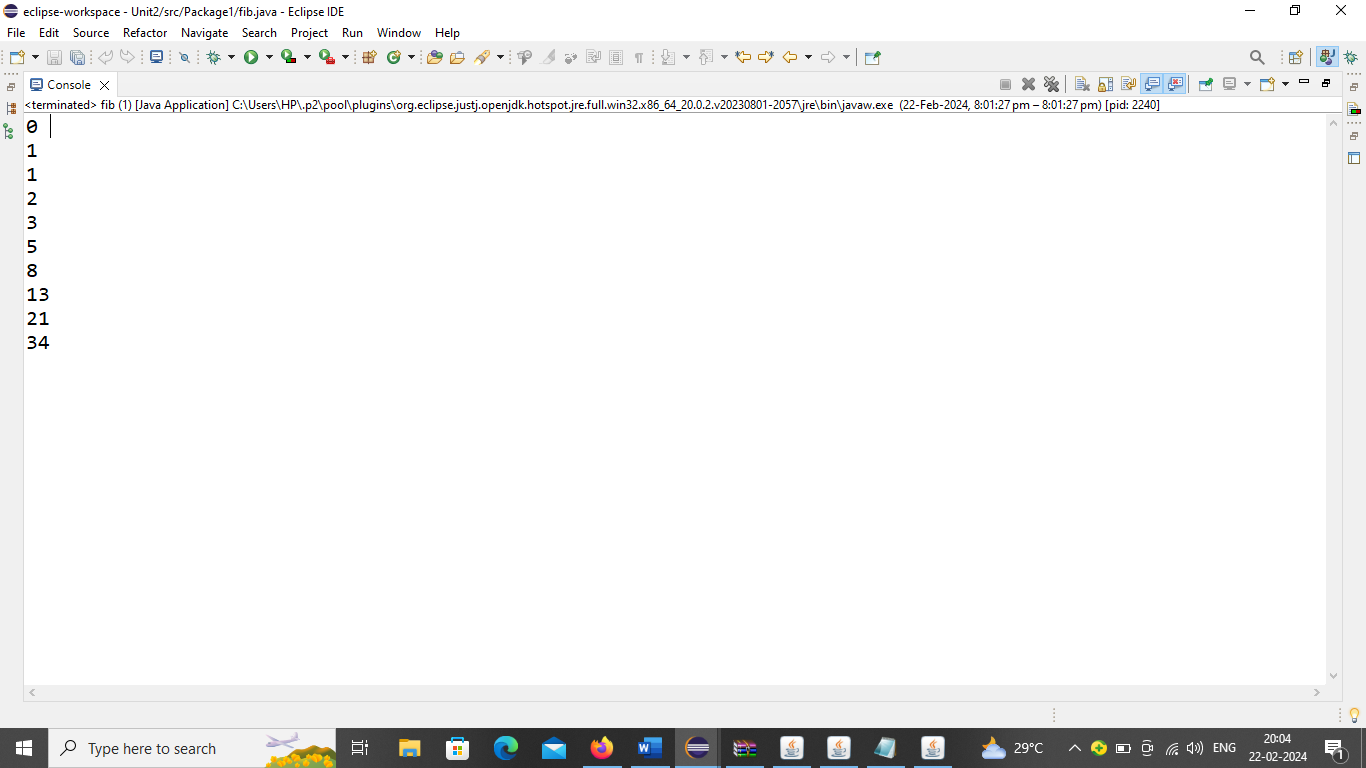
System.***out***.println(*fib*(i)+" ");

}

}

}

**OUTPUT:**



**EXPERIMENT 2:**

**Write a Java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired (Use Adapter classes).**

package package2;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.SwingUtilities;

public class Mouseeventexample extends JFrame{

public Mouseeventexample()

{ JFrame f=new JFrame();

f.setTitle("Mouse Event Demo");

f.setSize(200,300);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setLocationRelativeTo(null);

f.add(new MouseEventhandler());

f.setVisible(true);

}

private class MouseEventhandler extends JPanel{

public MouseEventhandler()

{

addMouseListener(new MouseAdapter()

{

public void mouseEntered(MouseEvent e)

{

displayEvent("Mouse entered");

}

public void mousePressed(MouseEvent e)

{

displayEvent("Mouse pressed");

}

public void mouseExited(MouseEvent e)

{

displayEvent("Mouse Exited");

}

});}

private void displayEvent(String msg)

{

JOptionPane.showMessageDialog(this, msg, "Mouse event", JOptionPane.INFORMATION\_MESSAGE);

} }

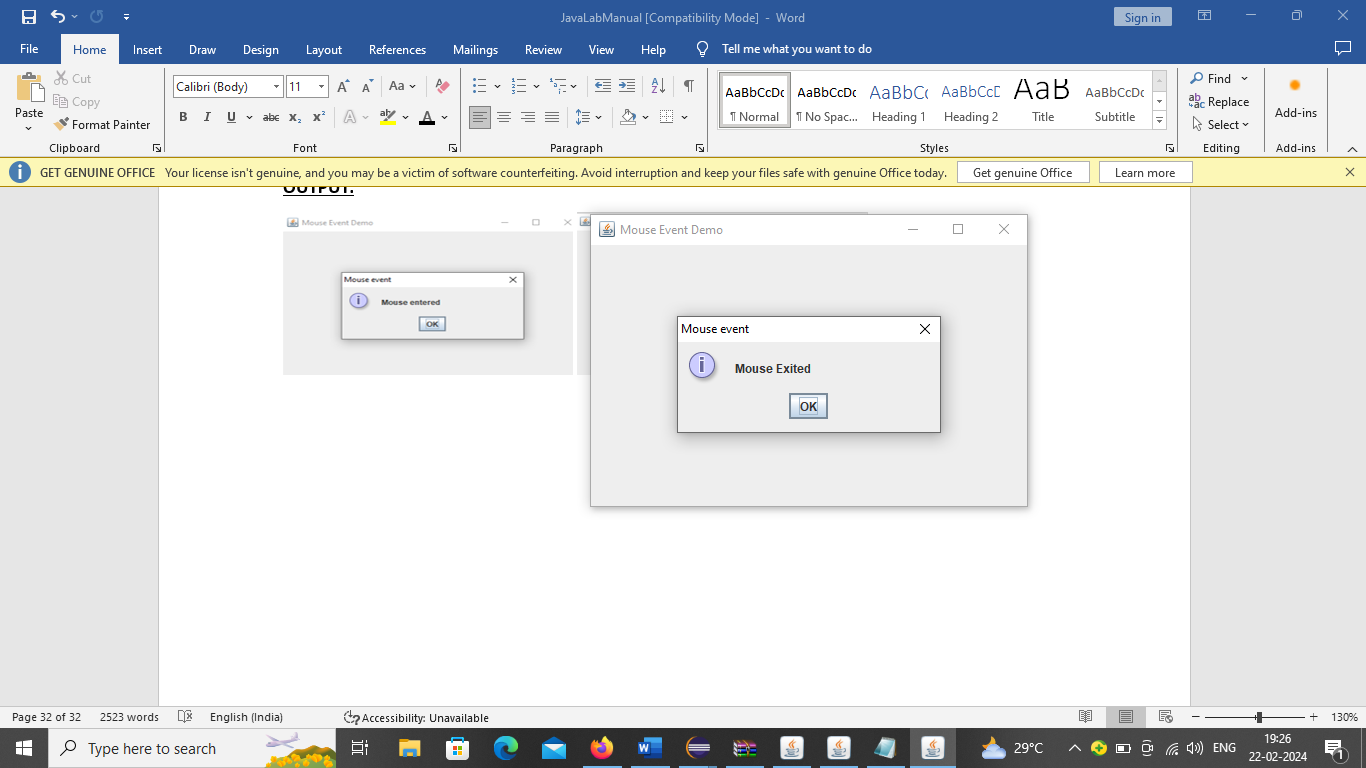
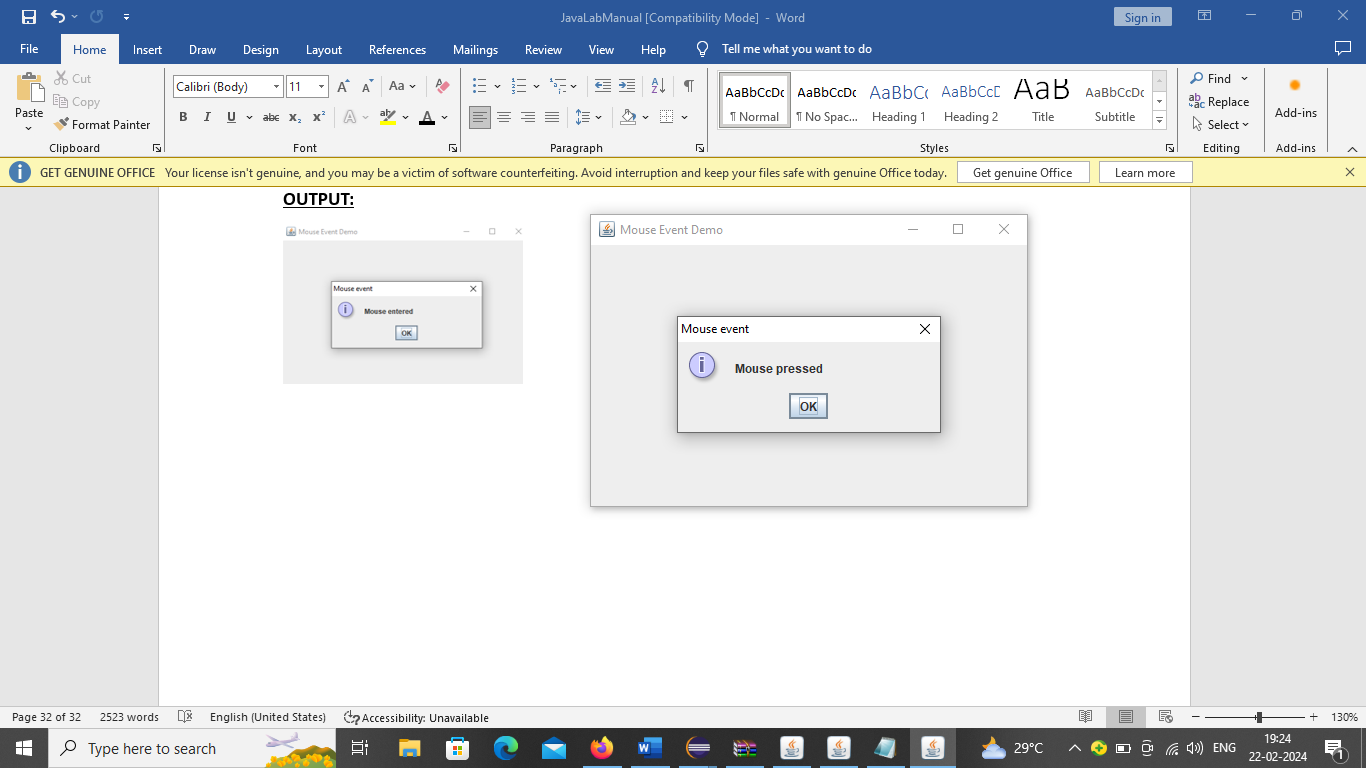
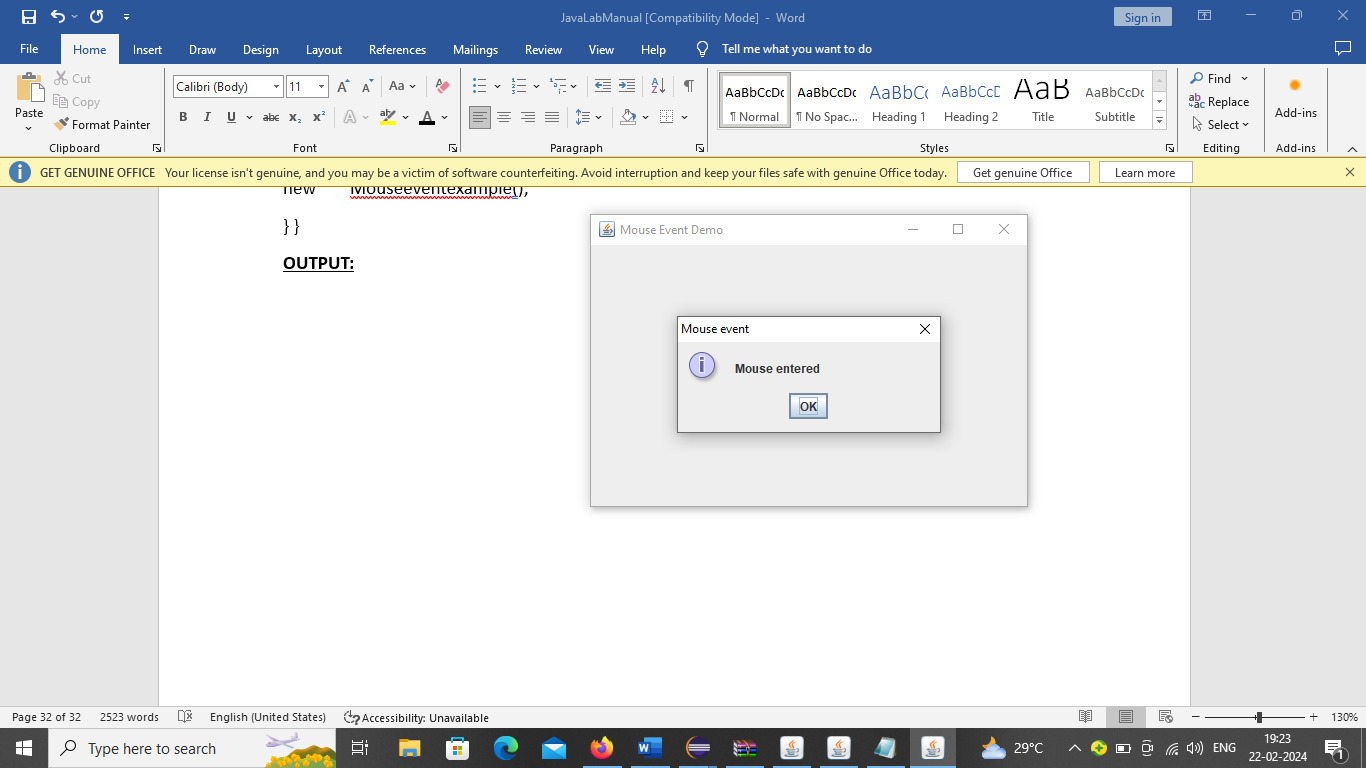
public static void main(String args[])

{

new Mouseeventexample();

} }

**OUTPUT:**



**EXPERIMENT 3:**

**Write a Java program to list all the files in a directory including the files present in all itssubdirectories.**

**package** package2;

**import** java.io.File;

**public** **class** ListFiles {

**public** **static** **void** main(String[] args) {

String directoryPath = "C://Users//HP//Desktop//test";

*listFiles*(**new** File(directoryPath));

}

**private** **static** **void** listFiles(File directory) {

File[] files = directory.listFiles();

**if** (files != **null**) {

**for** (File file : files) {

**if** (file.isDirectory()) {

*listFiles*(file);

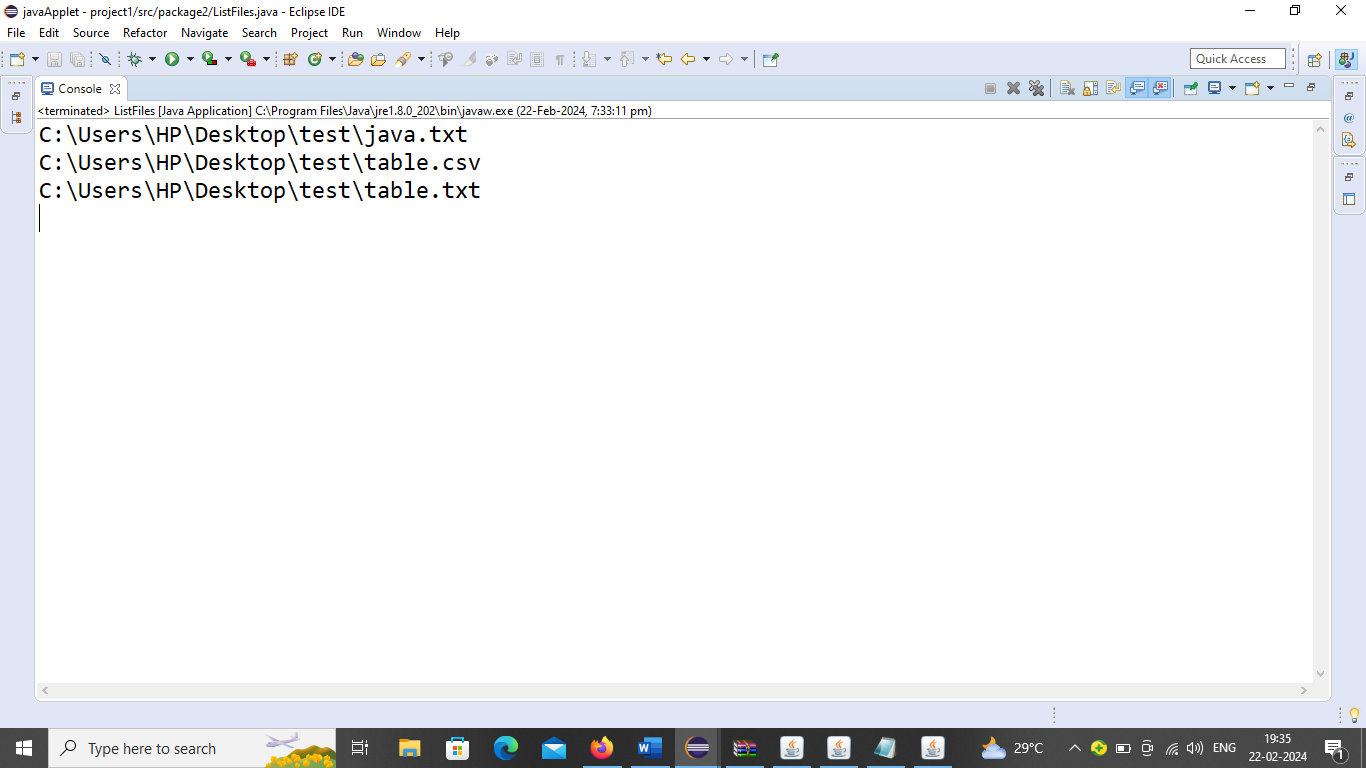
} **else** {

System.***out***.println(file.getAbsolutePath());

}}}

}}

**OUTPUT:**



**EXPERIMENT 4:**

**Write a Java program that correctly implements the producer – consumer problem using the**

**concept of inter thread communication.**

**package** Package1;

**class** Q

{

**int** n;

**boolean** valueSet=**false**;

**synchronized** **public** **void** get()

{

**if**(!valueSet)

{

**try**{wait();

}

**catch**(Exception e) {}

}

System.***out***.println("GET :"+n);

**try**{

Thread.*sleep*(2000);

}**catch**(Exception e) {}

valueSet=**false**;

notify();

}

**synchronized** **public** **void** put(**int** nn)

{

**if**(valueSet)

{

**try**{wait();

}

**catch**(Exception e) {}

}

n=nn;

System.***out***.println("PUT :"+n);

**try**{Thread.*sleep*(1000);}**catch**(Exception e) {};

valueSet=**true**;

notify();

}

}

**class** Producer **extends** Thread

{

Q q;

**public** Producer(Q qq)

{

q=qq;

}

@Override

**public** **void** run()

{

**int** i=0;

**while**(**true**)

{

q.put(i++);

}

}

}

**class** Consumer **extends** Thread

{

Q q;

**public** Consumer(Q qq)

{

q=qq;

}

@Override

**public** **void** run()

{

**while**(**true**)

{

q.get();

}

}

}

**public** **class** Test3

{

**public** **static** **void** main(String[] args)

{

Q q=**new** Q();

Producer p=**new** Producer(q);

Consumer c=**new** Consumer(q);

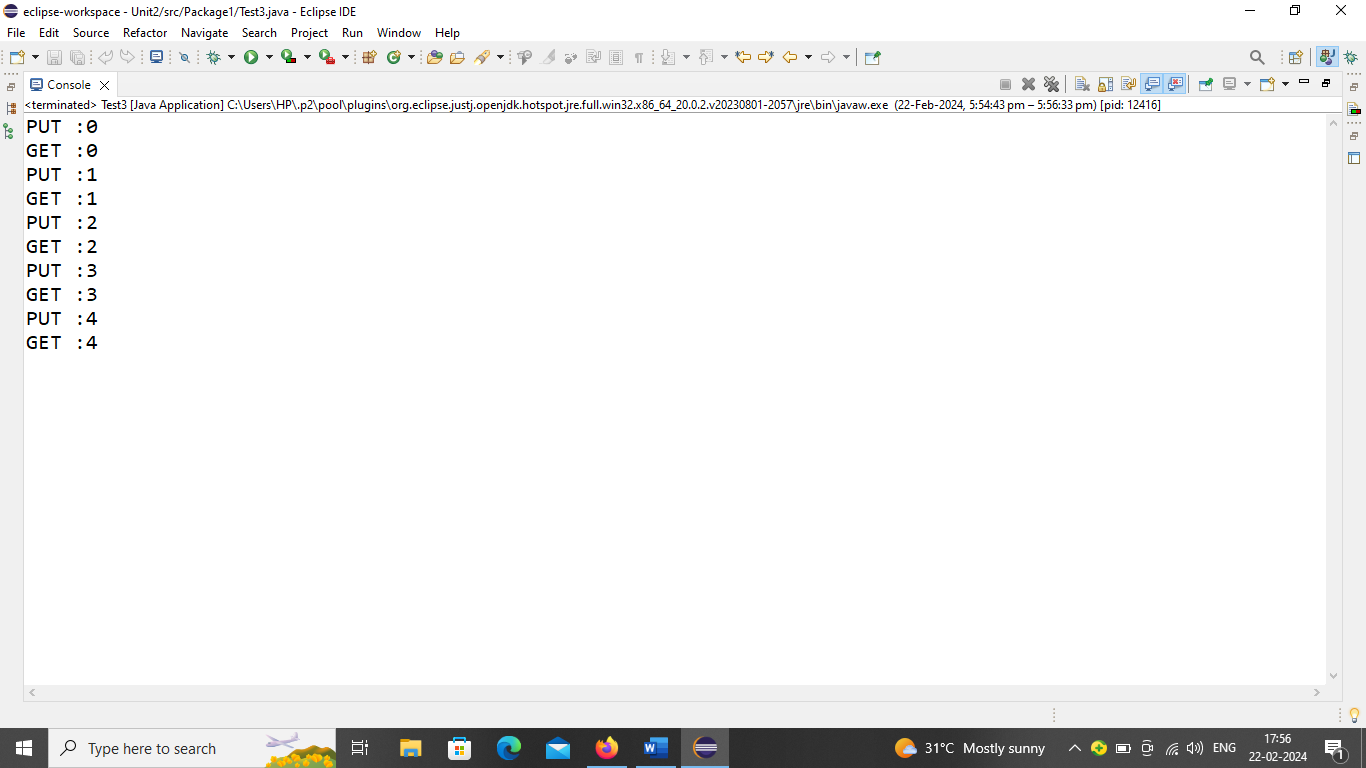
p.start();

c.start();

}

}

**OUTPUT:**



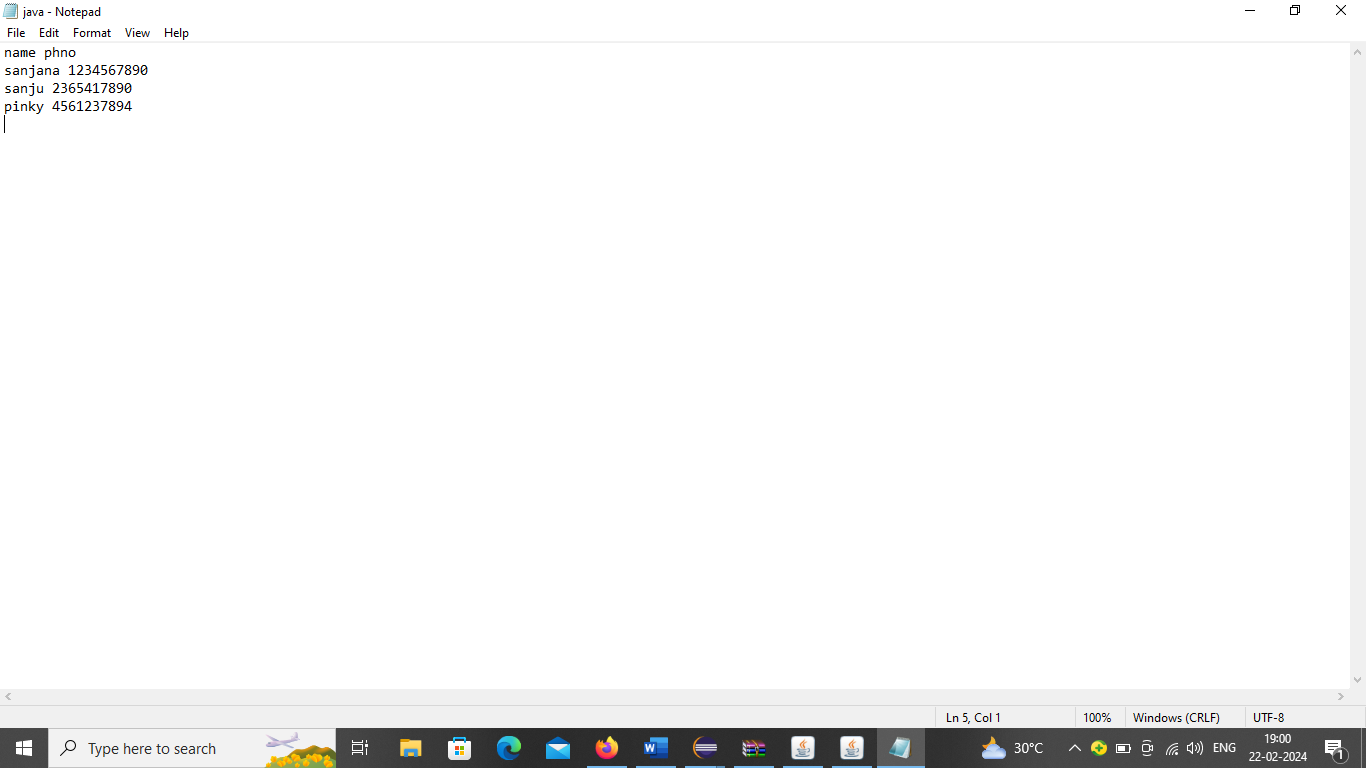
**EXPERIMENT 5:**

**Write a Java program that loads names and phone numbers from a text file where the data is**

**organized as one line per record and each field in a record are separated by a tab (\t). It takes a**

**name or phone number as input and prints the corresponding other value from the hash table (hint: use hash tables).**

**NOTE:** Create a file in notepad as shown here:



package package2;

import java.util.HashMap;

import java.util.Map;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.FileReader;

import java.util.Scanner;

public class PhoneBook

{

private Map<String,String>honebook;

public PhoneBook()

{

honebook=new HashMap<>();

}

public void load(String honeboo)throws IOException

{

try(BufferedReader reader=new BufferedReader(new FileReader(honeboo)))

{

String line;

while((line=reader.readLine())!=null)

{

String[] fields=line.split(“\\s+”);

if(fields.length==2)

{

String name=fields[0];

String phoneNumber=fields[1];

honebook.put(name, phoneNumber);

}

}

}

}

public String search(String input)

{

for(Map.Entry<String, String>entry:honebook.entrySet())

{

if(entry.getValue().equals(input))

{

return entry.getKey();

}

}

return honebook.get(input);

}

public static void main(String[] args)

{

// TODO Auto-generated method stub

PhoneBook honebook=new PhoneBook();

try

{

honebook.load(“C://Users//HP//Desktop//test//java.txt”);//Enter file path

}

catch(IOException e)

{

e.printStackTrace();

return;

}

Scanner sc=new Scanner(System.in);

while(true)

{

System.out.println(“Enter name or phonenumber(or ‘exit’ to quit):”);

String input=sc.nextLine();

if(input.equalsIgnoreCase(“exit”))

{

break;

}

String result=honebook.search(input);

if(result!=null)

{

if(input.matches(“\\d+”))

{

System.out.println(“Name:”+result);

}

else

{

System.out.println(“phonenumber:”+result);

}

}

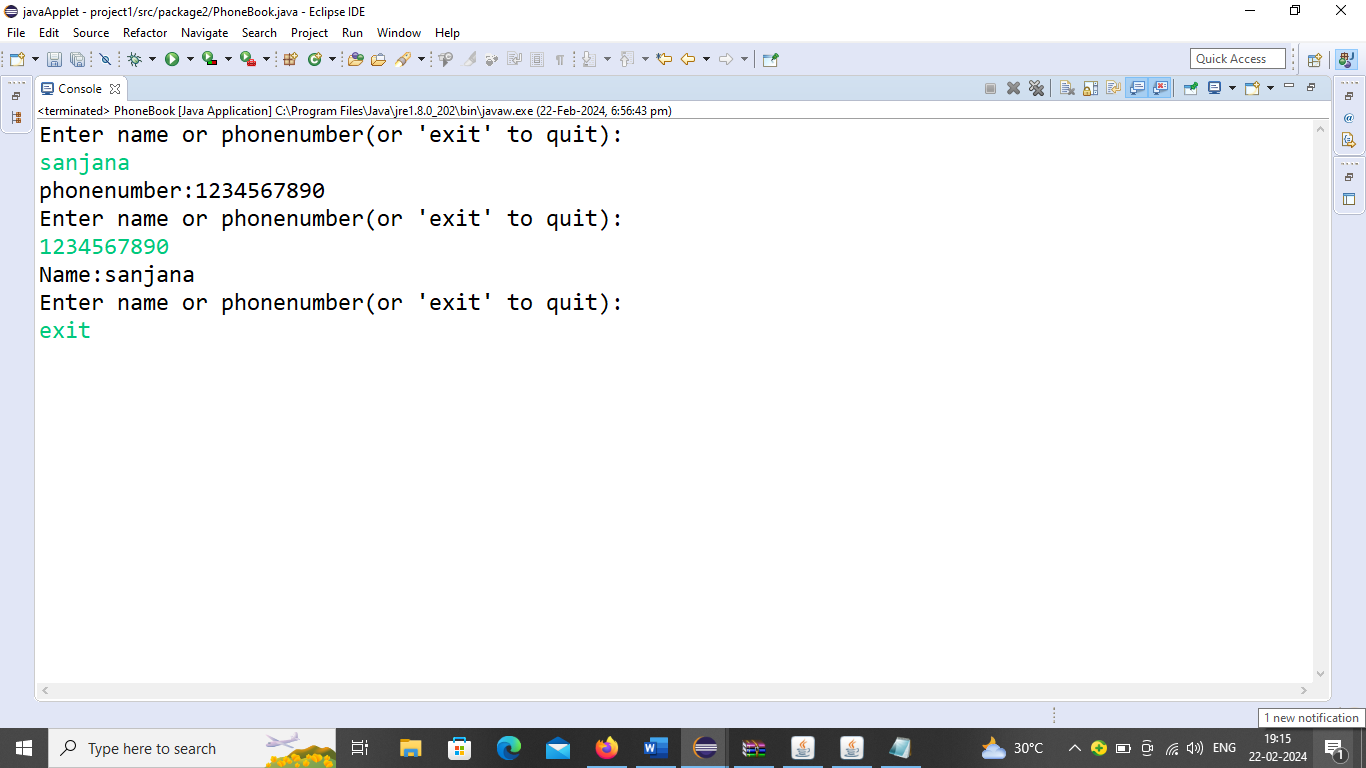
else

{

System.out.println(“entry not found”);

}}}}

**OUTPUT:**



**COMPLEX PROGRAMS**

**EXPERIMENT 6:**

**Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.**

**package** Package1;

**import** java.util.Random;

**class** square **extends** Thread

{

**int** x;

square(**int** n)

{

x=n;

}

**public** **void** run()

{

**int** sq=x\*x;

System.***out***.println("Square is"+sq);

}

}

**class** cube **extends** Thread{

**int** x;

cube(**int** n)

{

x=n;

}

**public** **void** run()

{

**int** cb=x\*x\*x;

System.***out***.println("Cube is"+cb);

}

}

**class** Rnumber **extends** Thread

{

**public** **void** run()

{

Random r=**new** Random();

**for**(**int** i=0;i<3;i++)

{

**int** randomInt=r.nextInt(30);

System.***out***.println("Random integer:"+randomInt);

square s=**new** square(randomInt);

s.start();

cube c=**new** cube(randomInt);

c.start();

**try** {

Thread.*sleep*(2000);

}

**catch**(InterruptedException e)

{

System.***out***.println(e);

}

}

}

}

**public** **class** threadexample {

**public** **static** **void** main(String[] args)

{

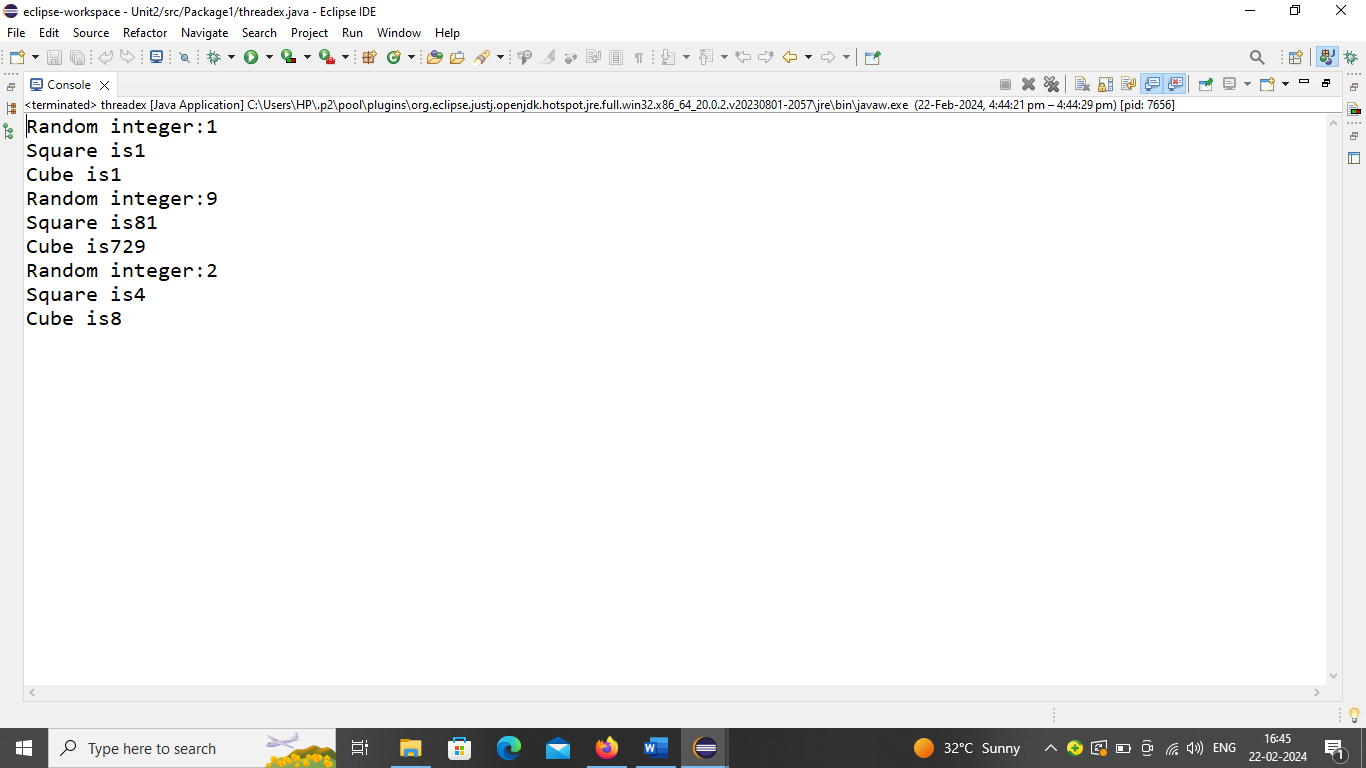
Rnumber r1=**new** Rnumber();

r1.start();

}

}

**OUTPUT:**



**EXPERIMENT 7:**

**Write a Java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle, and Circle suchthat each one of the classes extends the class Shape. Each one of the classes contains only the methodprint Area () that prints the area of the given shape.**

**package** Package1;

**abstract** **class** shapes {

**static** **int** *x*=8, *y*=4;

**abstract** **void** area();

}

**class** rect **extends** shape

{

@Override

**void** area() {

System.***out***.println("AREA OF RECTANGLE:"+*x*\**y*);

}

}

**class** tri **extends** shape

{

@Override

**void** area() {

System.***out***.println("AREA OF TRIANGLE:"+0.5\**x*\**y*);

}

}

**class** circle **extends** shape

{

@Override

**void** area() {

System.***out***.println("AREA OF CIRCLE:"+3.14\**x*\**x*);

}

}

**public** **class** shapes

{ **public** **static** **void** main(String[] args)

{

rect r=**new** rect();

r.area();

tri t=**new** tri();

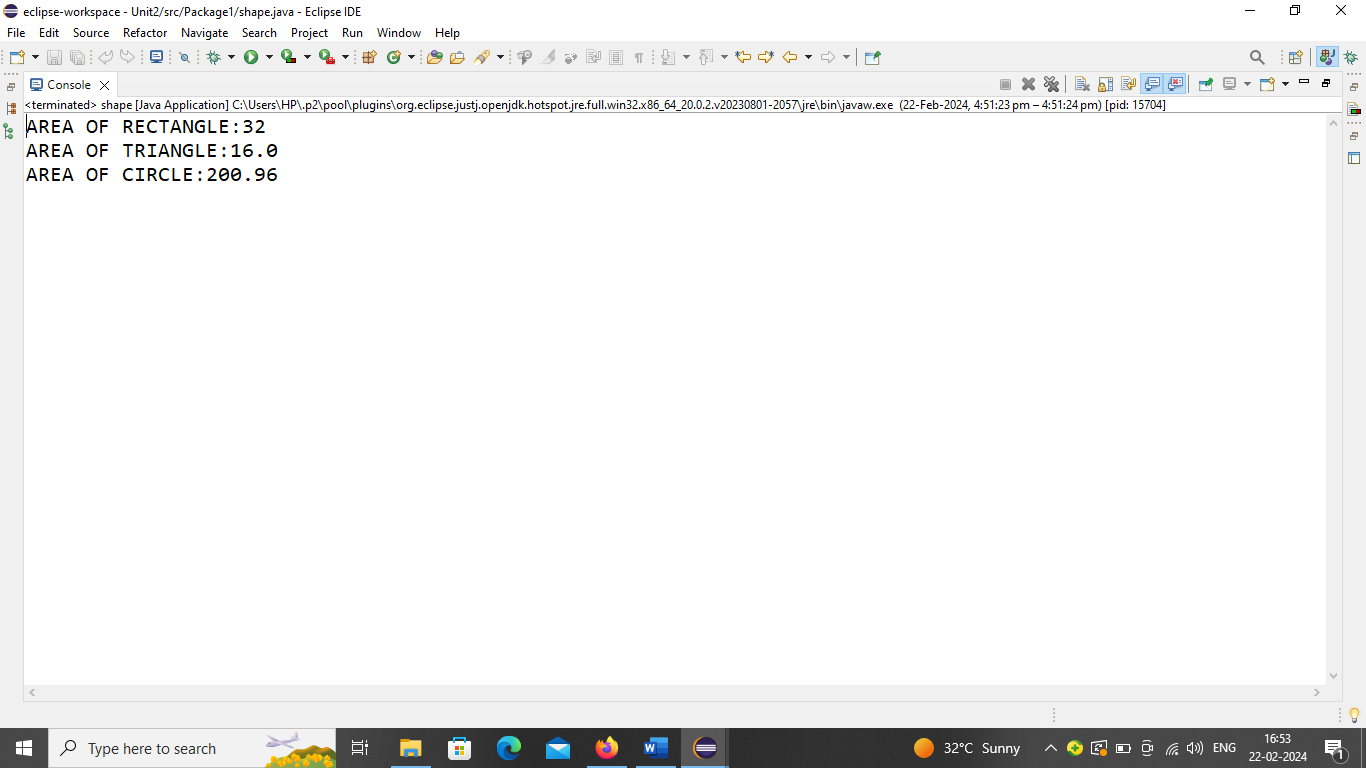
t.area();

circle c=**new** circle();

c.area();

}}

**OUTPUT:**



**EXPERIMENT 8:**

**Write a Java program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num 2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception. Display the exception in a message dialog box.**

package package2;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class IntegerDivision extends JFrame {

private JTextField num1, num2, result;

public IntegerDivision() {

setTitle("Integer Division Calculator");

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

num1 = new JTextField(10);

num2 = new JTextField(10);

result = new JTextField(10);

result.setEditable(false);

JButton divideButton = new JButton("Divide");

divideButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

performDivision();

}

});

setLayout(new GridLayout(4, 2));

add(new JLabel("Num1:"));

add(num1);

add(new JLabel("Num2:"));

add(num2);

add(new JLabel("Result:"));

add(result);

add(divideButton);

setVisible(true);

}

private void performDivision() {

try {

int n1 = Integer.parseInt(num1.getText());

int n2 = Integer.parseInt(num2.getText());

if (n2 == 0) {

throw new ArithmeticException("Cannot divide by zero!");

}

int res = n1 / n2;

result.setText(String.valueOf(res));

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Please enter valid integers for Num1 and Num2.",

"Number Format Exception", JOptionPane.ERROR\_MESSAGE);

} catch (ArithmeticException e) {

JOptionPane.showMessageDialog(this, "Cannot divide by zero.",

"Arithmetic Exception", JOptionPane.ERROR\_MESSAGE);

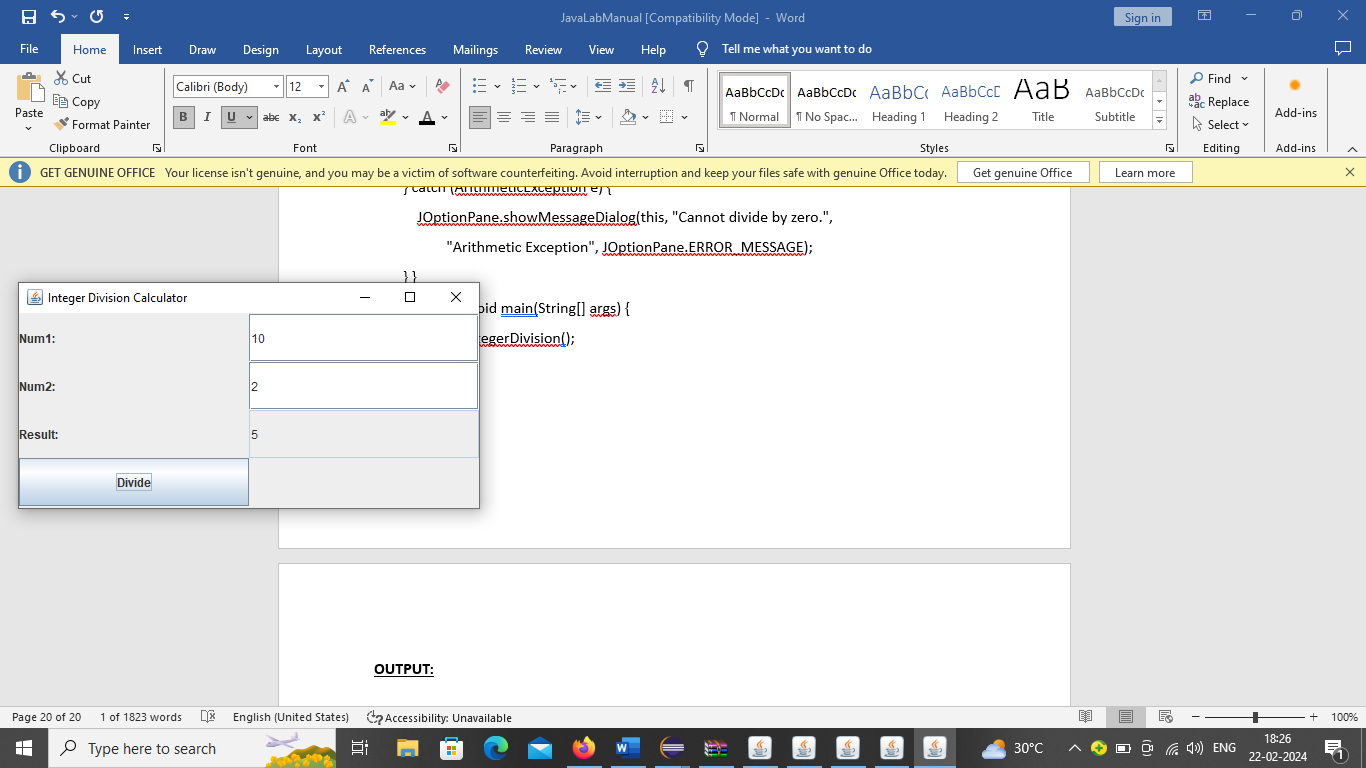
} }

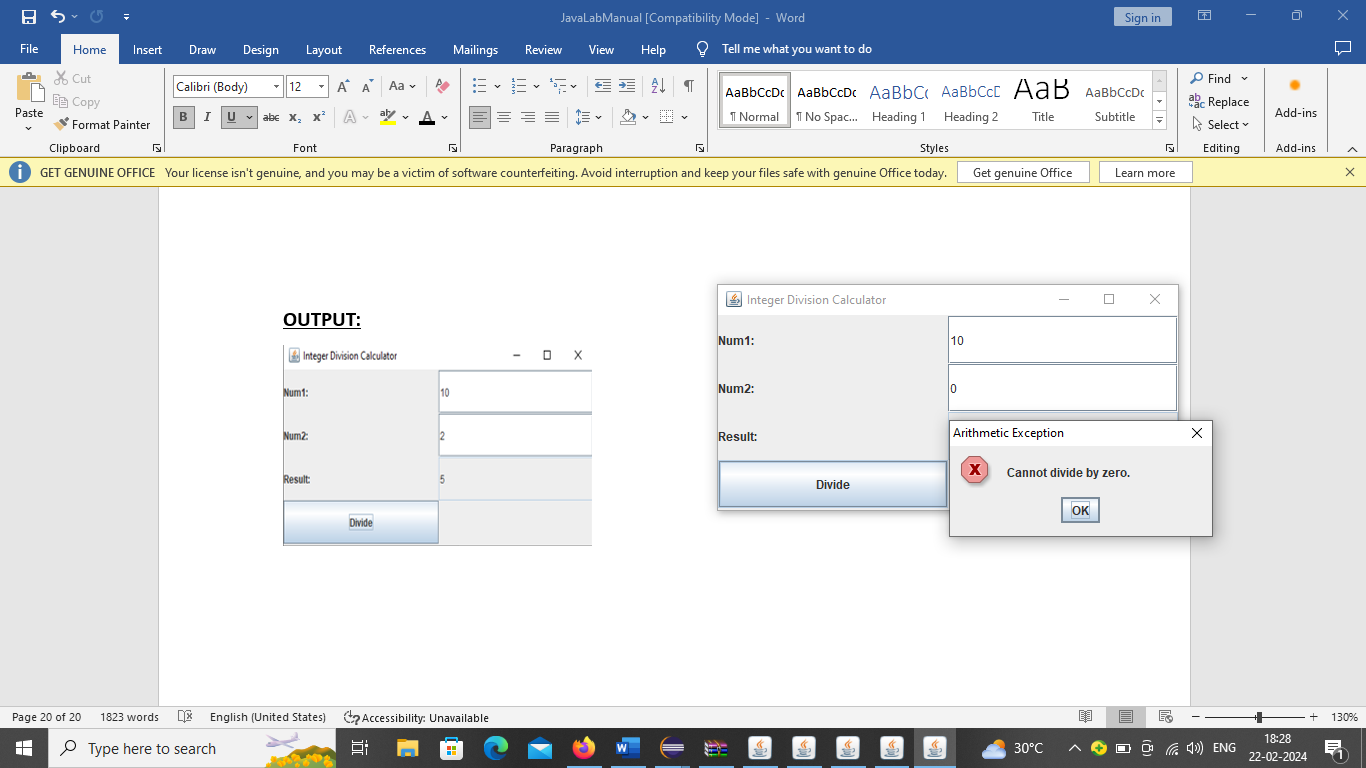
public static void main(String[] args) {

new IntegerDivision();

}}

**OUTPUT:**



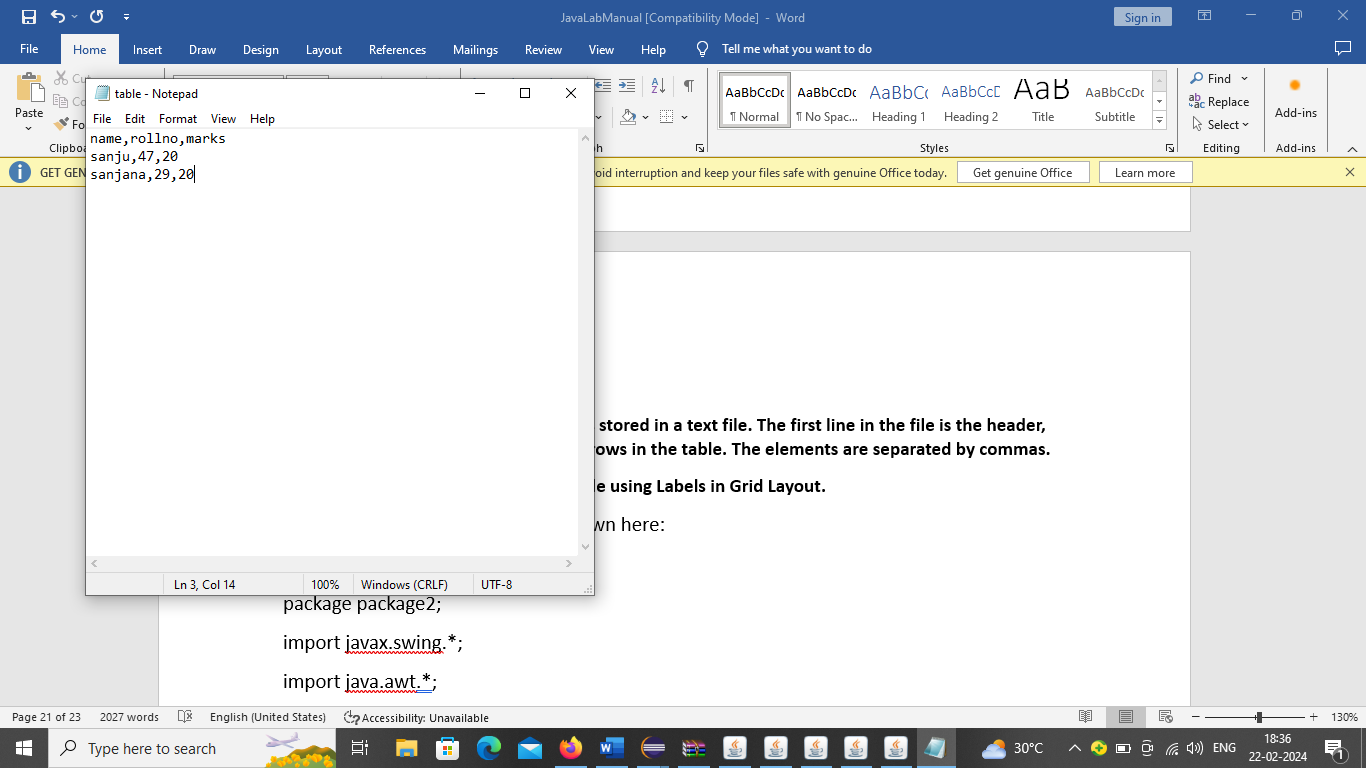


**EXPERIMENT 9:**

**Suppose that a table named Table.txt is stored in a text file. The first line in the file is the header, and the remaining lines correspond to rows in the table. The elements are separated by commas.**

**Write a java program to display the table using Labels in Grid Layout.**

NOTE:Create a file in notepad as shown here:



package package2;

import javax.swing.\*;

import java.awt.\*;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class TableDisplay extends JFrame {

public TableDisplay() {

setTitle("Table Display");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

setLayout(new GridLayout(3, 2));

try {

List<String[]> l1 = read("C://Users//HP//Desktop//test//table.txt"); //Enter file path

if (l1 != null) {

String[] s1 = l1.get(0);

for (String s2 : s1) {

add(new JLabel(s2));

}

for (int i = 1; i < l1.size(); i++) {

String[] s3 = l1.get(i);

for (String s4 : s3) {

add(new JLabel(s4));

}

}

}

} catch (IOException e) {

System.out.println(e);

}

setVisible(true);

}

private List<String[]> read(String table) throws IOException {

List<String[]> l1 = new ArrayList<>();

try (BufferedReader b1 = new BufferedReader(new FileReader(table))) {

String line;

while ((line = b1.readLine()) != null) {

String[] row = line.split(",");

l1.add(row);

}

}

return l1;

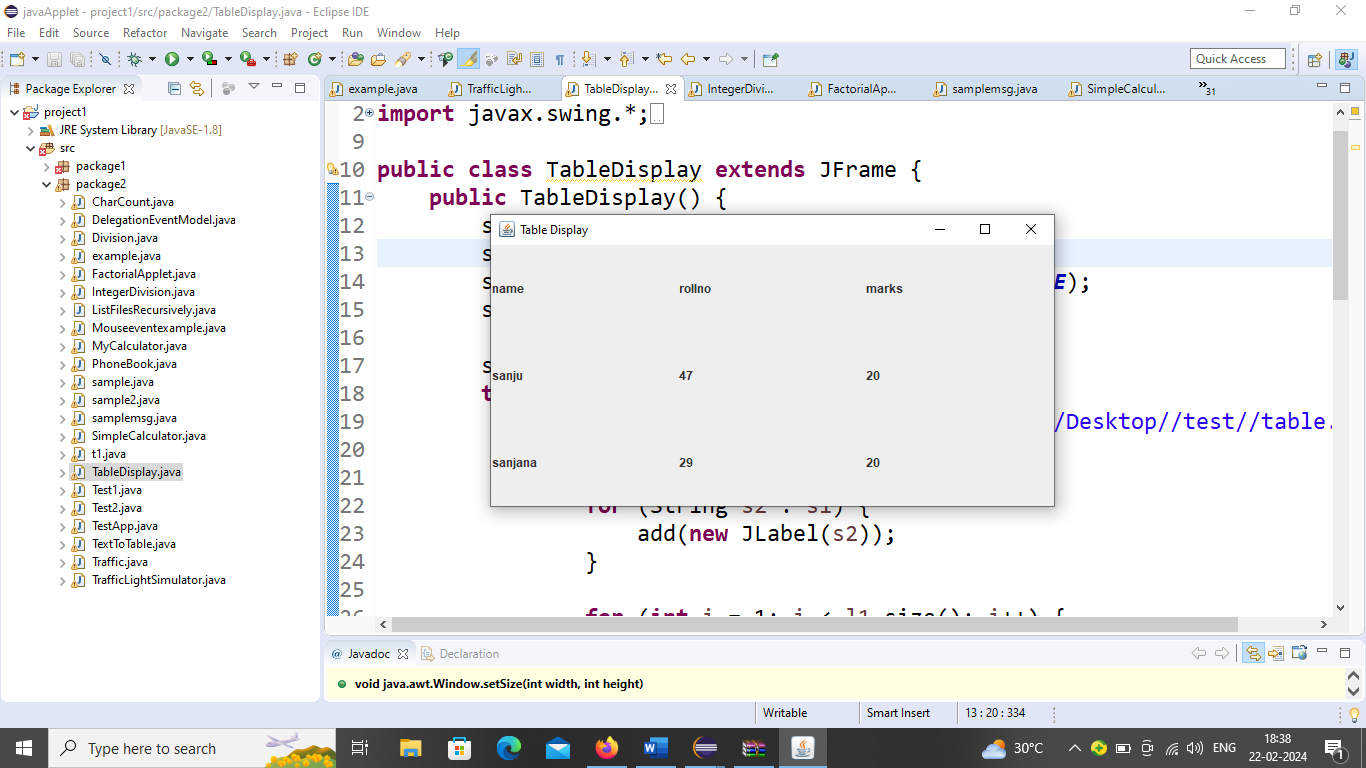
}

public static void main(String[] args) {

new TableDisplay();

}}

**OUTPUT:**



**EXPERIMENT 9:**

**Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with “Stop” or “Ready” or “Go” should appear above the buttons in the selected color. Initially, there is no message shown.**

package package2;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Traffic extends JFrame{

private JLabel message;

private JRadioButton red,yellow,green;

public void showmessage(String msg,Color color)

{

message.setText(msg);

message.setForeground(color);

}

public Traffic()

{

setTitle("Traffic Light Simulator");

setSize(300, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

message=new JLabel("",JLabel.CENTER);

red=new JRadioButton("Red");

yellow=new JRadioButton("Yellow");

green=new JRadioButton("Green");

ButtonGroup b=new ButtonGroup();

b.add(red);

b.add(yellow);

b.add(green);

red.addActionListener(new ActionListener()

{ @Override

public void actionPerformed(ActionEvent e) {

showmessage("Stop",Color.RED);

}

});

yellow.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e) {

showmessage("Ready",Color.YELLOW);

}

});

green.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e) {

showmessage("Go",Color.GREEN);

}

});

add(message,BorderLayout.CENTER);

JPanel p=new JPanel(new FlowLayout());

p.add(red);

p.add(yellow);

p.add(green);

add(p,BorderLayout.NORTH);

setVisible(true);

}

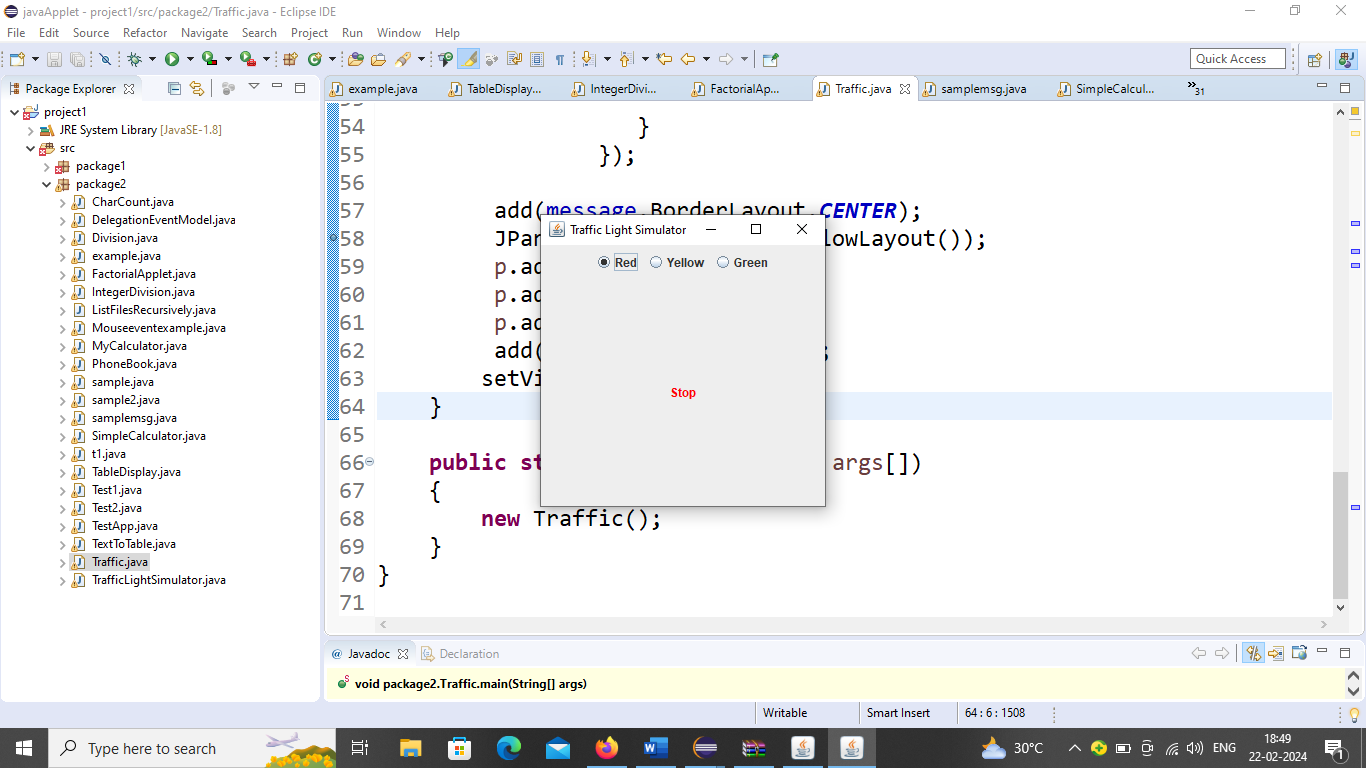
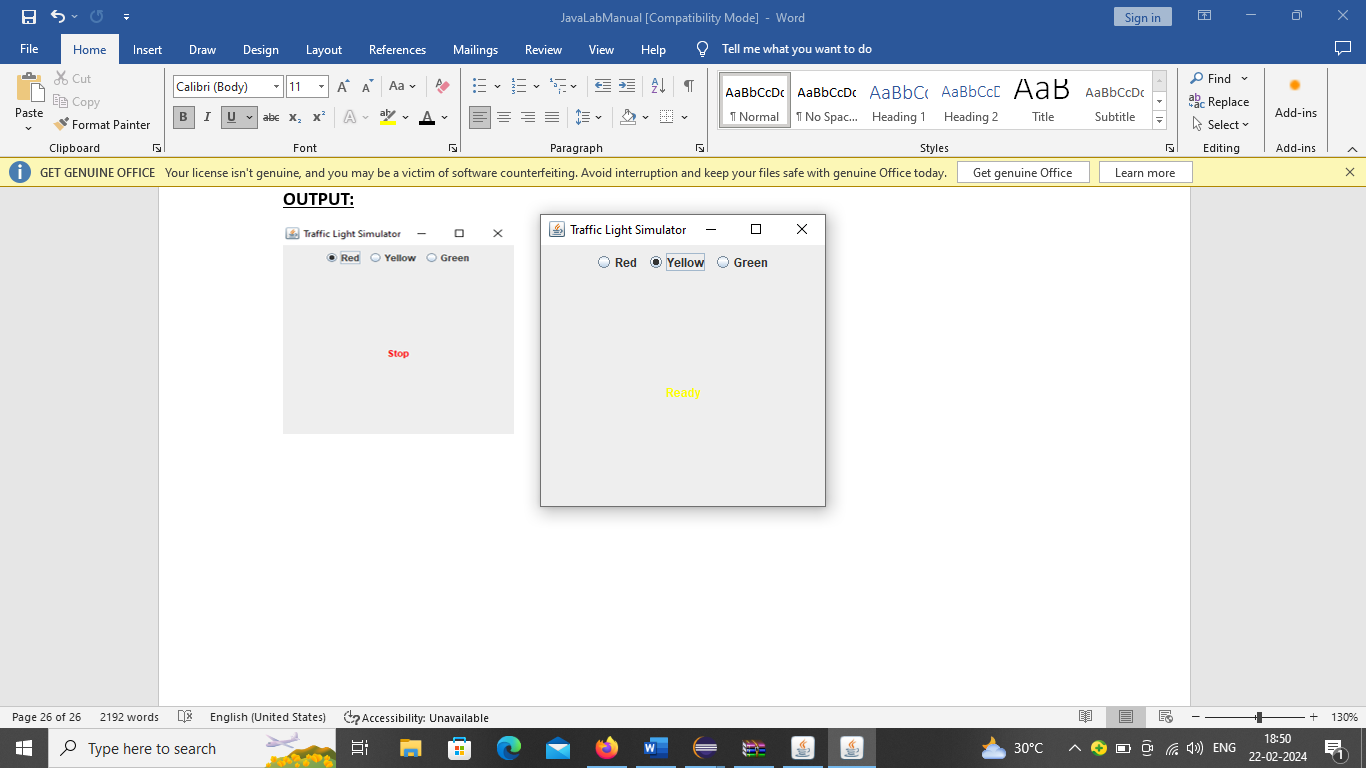
public static void main(String args[])

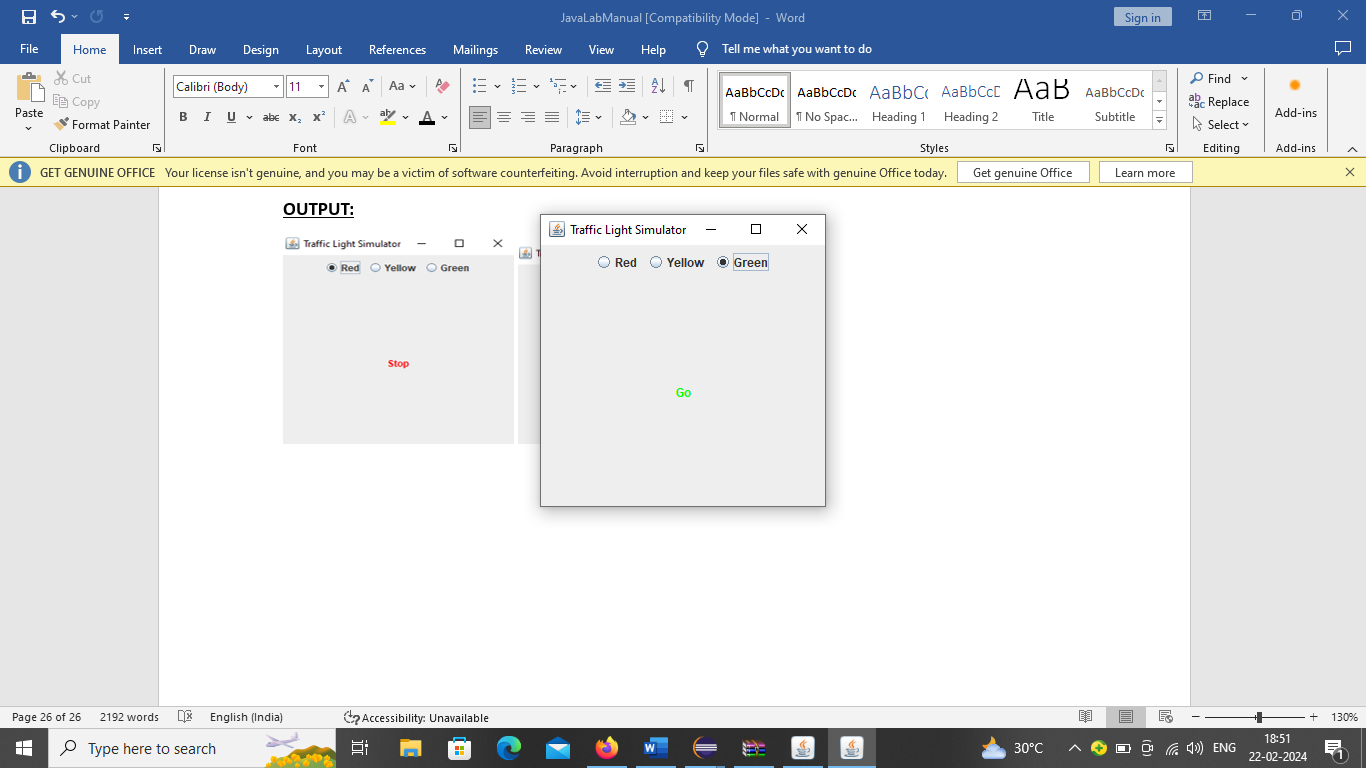
{

new Traffic();

}}

**OUTPUT:**



**ADDITIONAL QUESTIONS**

11.Java program to calculate average of numbers using array

import java.util.Scanner;

public class AverageCalculator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of elements: ");

int n = sc.nextInt(), sum = 0;

System.out.println("Enter the numbers:");

for (int i = 0; i < n; i++) sum += sc.nextInt();

System.out.println("Average: " + (double) sum / n);

sc.close();

}

}

12. Write a java program for interface.

// Define the interface

interface Animal {

void sound(); // Abstract method

}

// Implementing the interface in Dog class

class Dog implements Animal {

@Override

public void sound() {

System.out.println("Dog barks");

}

}

// Implementing the interface in Cat class

class Cat implements Animal {

@Override

public void sound() {

System.out.println("Cat meows");

}

}

// Main class

public class InterfaceExample {

public static void main(String[] args) {

Animal dog = new Dog();

Animal cat = new Cat();

dog.sound(); // Output: Dog barks

cat.sound(); // Output: Cat meows

}

}

13.Write a java program for Overriding.

// Parent class

class Animal {

void sound() {

System.out.println("Animals make sounds");

}

}

// Child class

class Dog extends Animal {

@Override

void sound() {

System.out.println("Dog barks");

}

}

// Main class

public class OverridingExample {

public static void main(String[] args) {

Animal animal = new Animal(); // Parent class object

animal.sound(); // Output: Animals make sounds

Animal dog = new Dog(); // Child class object using parent reference

dog.sound(); // Output: Dog barks

}

}

14. [How to Get the Id of a Current Running Thread in Java?](https://www.geeksforgeeks.org/how-to-get-the-id-of-a-current-running-thread-in-java/)

To get the ID of the currently running thread in Java, you can use the Thread class's currentThread() method, which returns a reference to the currently executing thread. Then, call the getId() method on that thread object to retrieve its ID.

Here’s a simple example:

**Java Code:**

java

Copy code

public class CurrentThreadId {

public static void main(String[] args) {

// Get the current thread

Thread currentThread = Thread.currentThread();

// Get the ID of the current thread

long threadId = currentThread.getId();

// Print the thread ID

System.out.println("The ID of the current thread is: " + threadId);

}

}

**Explanation:**

1. **Thread.currentThread()**:
   * Returns the Thread object representing the thread currently executing this code.
2. **getId()**:
   * Returns a unique ID for the thread. This ID is assigned by the JVM and remains constant during the thread's lifetime.

**Example Output:**

arduino

Copy code

The ID of the current thread is: 1

Note: The main thread typically has the ID 1, but thread IDs may vary depending on the JVM implementation and the context in which the program is run.

#### 15. **Write a program to create three buttons with caption OK, SUBMIT, CANCEL.**

import javax.swing.\*; // Import Swing package

import java.awt.\*; // Import AWT for layout management

public class ButtonExample {

public static void main(String[] args) {

// Create a JFrame (window)

JFrame frame = new JFrame("Button Example");

// Set frame size and close operation

frame.setSize(300, 150);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create a JPanel to hold buttons

JPanel panel = new JPanel();

panel.setLayout(new FlowLayout()); // Use FlowLayout for arranging buttons

// Create buttons

JButton okButton = new JButton("OK");

JButton submitButton = new JButton("SUBMIT");

JButton cancelButton = new JButton("CANCEL");

// Add buttons to the panel

panel.add(okButton);

panel.add(submitButton);

panel.add(cancelButton);

// Add panel to the frame

frame.add(panel);

// Make the frame visible

frame.setVisible(true);

}

}