**DESH BHAGAT UNIVERSITY**

**MANDI GOBINDGHAR, AMLOH**



**DEPARTMENT OF**

**COMPUTER SCIENCE AND ENGINEERING**

**PRACTICAL FILE OF**

**JAVA PROGRAMMING LAB**

|  |
| --- |
| * **Submitted by:- Karan Kumar** * **Class:- B.tech (CSE) 5th semester** * **Roll no:- 21320147018** |

|  |
| --- |
| * **Submitted to:- Dr. Navneet Kaur Sandhu** |

1. **Write a Java Program to define a class, describe its constructor, overload the Constructors and instantiate its object.**

**INPUT**:  
**Program Code:**

import java.lang.\*;

class student

{

String name;

int regno;

int marks1,marks2,marks3;

// null constructor

student()

{

name="Sudhanshu";

regno=12345;

marks1=56;

marks2=47;

marks3=78;

}

// parameterized constructor

student(String n,int r,int m1,int m2,int m3)

{

name=n;

regno=r;

marks1=m1;

marks2=m2;

marks3=m3;

}

// copy constructor

student(student s)

{

name=s.name;

regno=s.regno;

marks1=s.marks1;

marks2=s.marks2;

marks3=s.marks3;

}

void display()

{

System.out.println(name + "\t" +regno+ "\t" +marks1+ "\t" +marks2+ "\t" + marks3);

}

}

class studentdemo

{

public static void main (String arg [])

{

student s1=new student();

student s2=new student("Sachin", 34266, 58, 96, 84);

student s3=new student(s1);

s1.display();

s2.display();

s3. display();

}

}

**OUTPUT:**

Sudhanshu 12345  56  47  78

Sachin 34266 58  96  84

Sudhanshu 12345 56 47 78

2. **Write a Java Program to define a class, define instance methods for setting and Retrieving values of instance variables and instantiate its object.**

**INPUT:**

**Program Code:**

import java.lang.\*;

class emp

{

String name;

int id;

String address;

void getdata(String name,int id,String address)

{

this.name=name;

this.id=id;

this.address=address;

}

void putdata()

{

System.out.println("Employee details are :");

System.out.println("Name :" +name);

System.out.println("ID :" +id);

System.out.println("Address :" +address);

}

}

class empdemo

{

public static void main(String arg[])

{

emp e=new emp();

e.getdata("Ramesh",768,"Pune");

e.putdata()

}

}

**Output:**

Employee details are :

Name: Ramesh

ID: 768

Address : Pune

**3. Write a Java Program to define a class, define instance methods and overload them and use them for dynamic method invocation.**

**INPUT:**

**Program Code:**

import java.lang.\*;

class add

{

void display(int a,int b)

{

int c=a+b;

System.out.println("The sum of " + a + " & " + b + " is " + c);

}

void display(double a,double b)

{ double c=a+b;

System.out.println("The sum of " + a + " & " + b + " is " + c);

}

}

class add\_demo

{

public static void main(String arg[])

{

add obj=new add();

obj.display(10,20);

obj.display(10.2,20.2);

}

}

**Output:**

The sum of 10 & 20 is 30

The sum of 10.2 & 20.2 is 30.4

**4. Write a Java Program to demonstrate use of subclass**

**INPUT:**

// Superclass

class Vehicle {

private String brand;

public Vehicle(String brand) {

this.brand = brand;

}

public void start() {

System.out.println("Starting the vehicle.");

}

public void stop() {

System.out.println("Stopping the vehicle.");

}

public void displayInfo() {

System.out.println("Brand: " + brand);

}

}

// Subclass 1

class Car extends Vehicle {

private int numberOfDoors;

public Car(String brand, int numberOfDoors) {

super(brand);

this.numberOfDoors = numberOfDoors;

}

public void honk() {

System.out.println("Honk! Honk!");

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Number of Doors: " + numberOfDoors);

}

}

// Subclass 2

class Motorcycle extends Vehicle {

private boolean hasSideCar;

public Motorcycle(String brand, boolean hasSideCar) {

super(brand);

this.hasSideCar = hasSideCar;

}

public void wheelie() {

System.out.println("Performing a wheelie!");

}

@Override

public void displayInfo() {

super.displayInfo();

System.out.println("Has Sidecar: " + hasSideCar);

}

}

// Main class

public class SubclassExample {

public static void main(String[] args) {

// Creating objects of the subclasses

Car myCar = new Car("Toyota", 4);

Motorcycle myMotorcycle = new Motorcycle("Harley Davidson", false);

// Calling methods on objects

myCar.start();

myCar.honk();

myCar.displayInfo();

myCar.stop();

System.out.println(); // Separating outputs

myMotorcycle.start();

myMotorcycle.wheelie();

myMotorcycle.displayInfo();

myMotorcycle.stop();

}

}

**OUTPUT:**

Starting the vehicle. Honk!

Honk! Brand: Toyota

Number of Doors: 4

Stopping the vehicle.

Starting the vehicle.

Performing a wheelie!

Brand: Harley Davidson

Has Sidecar: false

Stopping the vehicle.

**5. Write a Java Program to demonstrate use of nested class**

**INPUT:**

// outer class

**class** OuterClass {

    // static member

**static** **int** outer\_x = 10;

**int** outer\_y = 20;

**private** **static** **int** outer\_private = 30;

**static** **class** StaticNestedClass {

**void** display()

        {

            System.out.println("outer\_x = " + outer\_x);

            System.out.println("outer\_private = "

                               + outer\_private);

              OuterClass out = **new** OuterClass();

              System.out.println("outer\_y = " + out.outer\_y);

        }

    }

}

**public** **class** StaticNestedClassDemo {

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

    {

        OuterClass.StaticNestedClass nestedObject

            = **new** OuterClass.StaticNestedClass();

        nestedObject.display();

    }

}

**OUTPUT:**

outer\_x = 10

outer\_private = 30

outer\_y = 20

**6. Write a Java Program to implement array of objects**

**INPUT:**

// Person class representing individual persons

class Person {

String name;

int age;

// Constructor

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Method to display information about the person

public void displayInfo() {

System.out.println("Name: " + name + ", Age: " + age);

}

}

public class ArrayOfObjectsExample {

public static void main(String[] args) {

// Creating an array of Person objects

Person[] peopleArray = new Person[3];

// Initializing each element of the array with a Person object

peopleArray[0] = new Person("Alice", 25);

peopleArray[1] = new Person("Bob", 30);

peopleArray[2] = new Person("Charlie", 22);

// Displaying information about each person using a loop

System.out.println("Information about people:");

for (Person person : peopleArray) {

person.displayInfo();

}

}

}

**OUTPUT:**

Information about people:

Name: Alice, Age: 25

Name: Bob, Age: 30

Name: Charlie, Age: 22

**7. Write a Java program to practice using String class and its methods**

**INPUT:**

public class StringPractice {

public static void main(String[] args) {

// Creating strings

String str1 = "Hello";

String str2 = "World";

// Concatenation

String concatenatedString = str1 + " " + str2;

System.out.println("Concatenated String: " + concatenatedString);

// Length

int length = concatenatedString.length();

System.out.println("Length of the String: " + length);

// Substring

String substring = concatenatedString.substring(6, 11);

System.out.println("Substring: " + substring);

// Uppercase and lowercase

String uppercaseString = concatenatedString.toUpperCase();

String lowercaseString = concatenatedString.toLowerCase();

System.out.println("Uppercase: " + uppercaseString);

System.out.println("Lowercase: " + lowercaseString);

// IndexOf

int indexOfWorld = concatenatedString.indexOf("World");

System.out.println("Index of 'World': " + indexOfWorld);

// Replace

String replacedString = concatenatedString.replace("Hello", "Hola");

System.out.println("Replaced String: " + replacedString);

// Trim

String stringWithSpaces = " Trim Me ";

String trimmedString = stringWithSpaces.trim();

System.out.println("Original String: '" + stringWithSpaces + "'");

System.out.println("Trimmed String: '" + trimmedString + "'");

}

}

**OUTPUT :**

Concatenated String: Hello World

Length of the String: 11

Substring: World

Uppercase: HELLO WORLD

Lowercase: hello world

Index of 'World': 6

Replaced String: Hola World

Original String: ' Trim Me '

Trimmed String: 'Trim Me'

**8. Write a Java Program to implement multilevel inheritance by applying various access controls to its data members and methods.**

**INPUT:**

// Grandparent class with different access modifiers

class Grandparent {

public String publicField = "Public field in Grandparent";

protected String protectedField = "Protected field in Grandparent";

String defaultField = "Default field in Grandparent";

private String privateField = "Private field in Grandparent";

public void publicMethod() {

System.out.println("Public method in Grandparent");

}

protected void protectedMethod() {

System.out.println("Protected method in Grandparent");

}

void defaultMethod() {

System.out.println("Default method in Grandparent");

}

private void privateMethod() {

System.out.println("Private method in Grandparent");

}

}

// Parent class inheriting from Grandparent

class Parent extends Grandparent {

// Additional fields and methods in Parent

public String parentField = "Public field in Parent";

protected String parentProtectedField = "Protected field in Parent";

public void parentPublicMethod() {

System.out.println("Public method in Parent");

}

protected void parentProtectedMethod() {

System.out.println("Protected method in Parent");

}

}

// Child class inheriting from Parent

class Child extends Parent {

// Additional fields and methods in Child

public String childField = "Public field in Child";

public void childPublicMethod() {

System.out.println("Public method in Child");

}

}

public class MultilevelInheritanceExample {

public static void main(String[] args) {

// Create an object of Child class

Child childObject = new Child();

// Accessing fields and methods from different inheritance levels

System.out.println(childObject.publicField);

System.out.println(childObject.protectedField); // Accessible due to protected modifier

// System.out.println(childObject.defaultField); // Not accessible from outside the package

// System.out.println(childObject.privateField); // Not accessible in Child class

childObject.publicMethod();

childObject.protectedMethod(); // Accessible due to protected modifier

// childObject.defaultMethod(); // Not accessible from outside the package

// childObject.privateMethod(); // Not accessible in Child class

System.out.println(childObject.parentField);

System.out.println(childObject.parentProtectedField); // Accessible due to protected modifier

childObject.parentPublicMethod();

childObject.parentProtectedMethod(); // Accessible due to protected modifier

System.out.println(childObject.childField);

childObject.childPublicMethod();

}

}

**OUTPUT :**

Public field in Grandparent

Protected field in Grandparent

Public method in Grandparent

Protected method in Grandparent

Public field in Parent

Protected field in Parent

Public method in Parent

Protected method in Parent

Public field in Child

Public method in Child

**9. Write a program to demonstrate use of implementing interfaces**

**INPUT**

// Interface defining the contract

interface Shape {

double calculateArea(); // Method to calculate the area

double calculatePerimeter(); // Method to calculate the perimeter

}

// Class representing a Circle implementing the Shape interface

class Circle implements Shape {

private double radius;

// Constructor

public Circle(double radius) {

this.radius = radius;

}

// Implementing methods from the Shape interface

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

@Override

public double calculatePerimeter() {

return 2 \* Math.PI \* radius;

}

}

// Class representing a Rectangle implementing the Shape interface

class Rectangle implements Shape {

private double length;

private double width;

// Constructor

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

// Implementing methods from the Shape interface

@Override

public double calculateArea() {

return length \* width;

}

@Override

public double calculatePerimeter() {

return 2 \* (length + width);

}

}

// Main class to demonstrate the use of implementing interfaces

public class InterfaceExample {

public static void main(String[] args) {

// Creating objects of classes implementing the Shape interface

Circle = new Circle(5.0);

Rectangle = new Rectangle(4.0, 6.0);

// Calculating and displaying area and perimeter of the circle

System.out.println("Circle - Area: " + circle.calculateArea() + ", Perimeter: " + circle.calculatePerimeter());

// Calculating and displaying area and perimeter of the rectangle

System.out.println("Rectangle - Area: " + rectangle.calculateArea() + ", Perimeter: " + rectangle.calculatePerimeter());

}

}

**OUTPUT:**

Circle - Area: 78.53981633974483, Perimeter: 31.41592653589793

Rectangle - Area: 24.0, Perimeter: 20.0

**10. Write a Java Program to demonstrate Keyboard event**

**INPUT:**

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import javax.swing.JFrame;

import javax.swing.JTextField;

public class KeyEventsDemo extends JFrame {

JTextField textField;

public KeyEventsDemo() {

textField = new JTextField();

textField.addKeyListener(new KeyListener() {

public void keyTyped(KeyEvent e) {

// Invoked when a key is typed

System.out.println("Key Typed: " + e.getKeyChar());

}

public void keyPressed(KeyEvent e) {

// Invoked when a key is pressed

System.out.println("Key Pressed: " + e.getKeyChar());

}

public void keyReleased(KeyEvent e) {

// Invoked when a key is released

System.out.println("Key Released: " + e.getKeyChar());

}

});

add(textField);

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

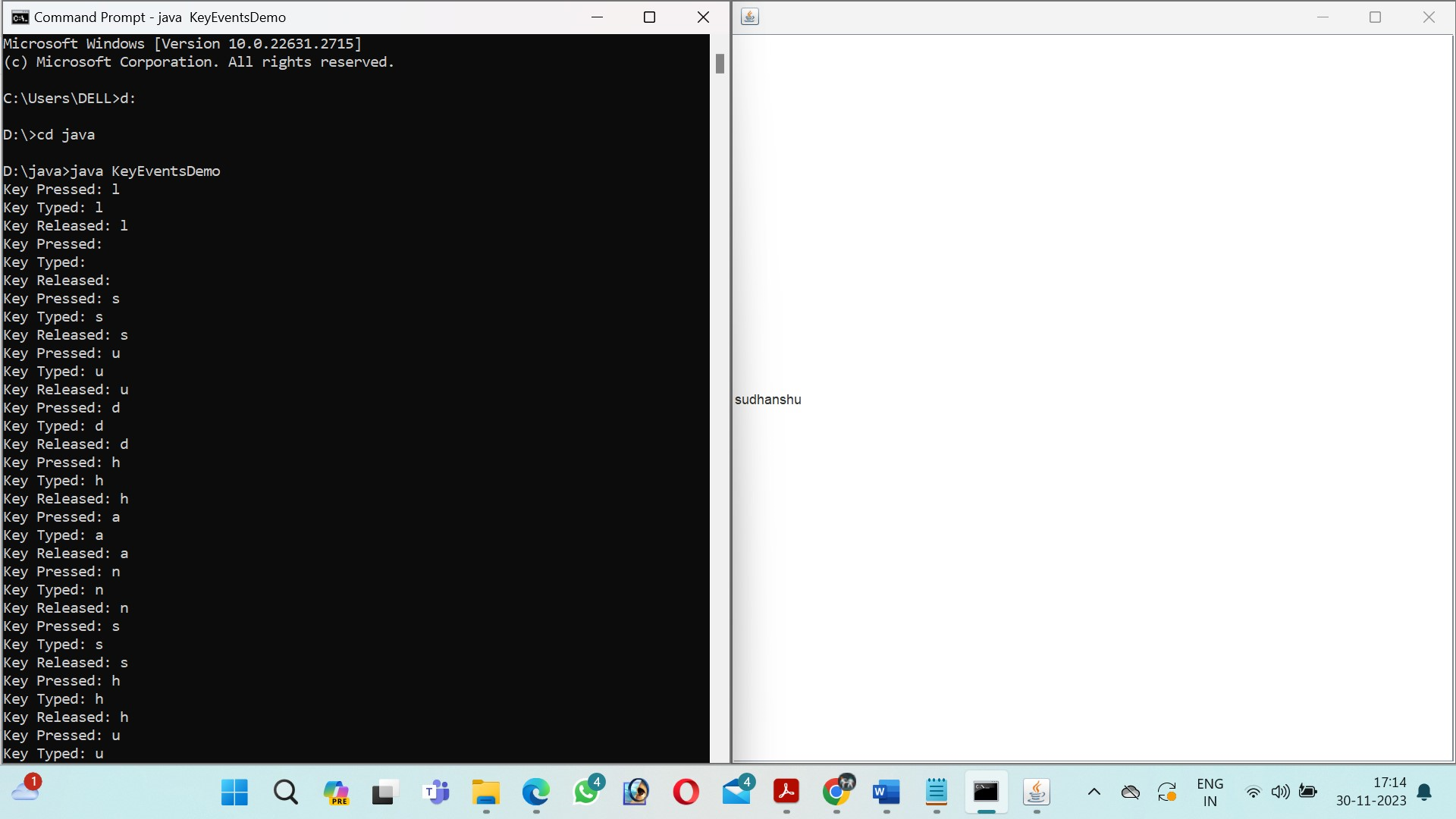
public static void main(String[] args) {

new KeyEventsDemo();

}

}

**OUTPUT:**



**11. Write a Java Program to demonstrate Mouse events**

**INPUT:**

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class MouseEventsDemo extends JFrame {

public MouseEventsDemo() {

JPanel panel = new JPanel();

panel.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

System.out.println("Mouse Clicked at (" + e.getX() + ", " + e.getY() + ")");

}

public void mousePressed(MouseEvent e) {

System.out.println("Mouse Pressed at (" + e.getX() + ", " + e.getY() + ")");

}

public void mouseReleased(MouseEvent e) {

System.out.println("Mouse Released at (" + e.getX() + ", " + e.getY() + ")");

}

public void mouseEntered(MouseEvent e) {

System.out.println("Mouse Entered");

}

public void mouseExited(MouseEvent e) {

System.out.println("Mouse Exited");

}

});

add(panel);

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

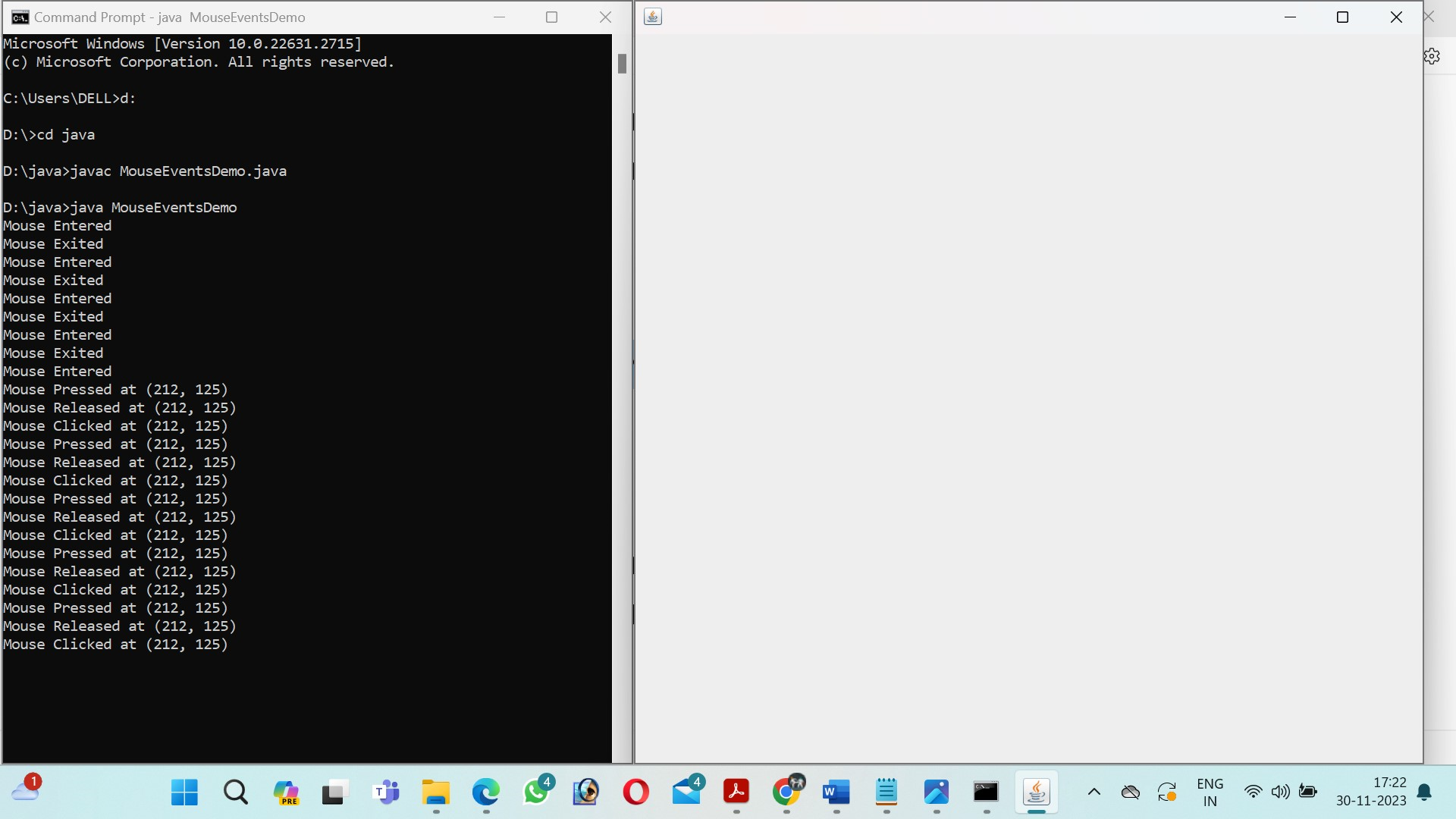
}

public static void main(String[] args) {

new MouseEventsDemo();

}

}

**OUTPUT:** 

**12. Write programs for using Graphics class**

**i) to display basic shapes and fill them**

**ii) draw different items using basic shapes**

**INPUT:**

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class BasicShapesDemo extends JFrame {

public BasicShapesDemo() {

add(new MyPanel());

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

class MyPanel extends JPanel {

public void paintComponent(Graphics g) {

super.paintComponent(g);

// Drawing basic shapes

g.setColor(Color.RED);

g.drawRect(20, 20, 50, 50); // Rectangle

g.setColor(Color.BLUE);

g.drawOval(100, 20, 50, 50); // Oval

g.setColor(Color.GREEN);

g.drawLine(200, 20, 250, 70); // Line

// Filling shapes

g.setColor(Color.YELLOW);

g.fillRect(20, 100, 50, 50); // Filled Rectangle

g.setColor(Color.CYAN);

g.fillOval(100, 100, 50, 50); // Filled Oval

}

}

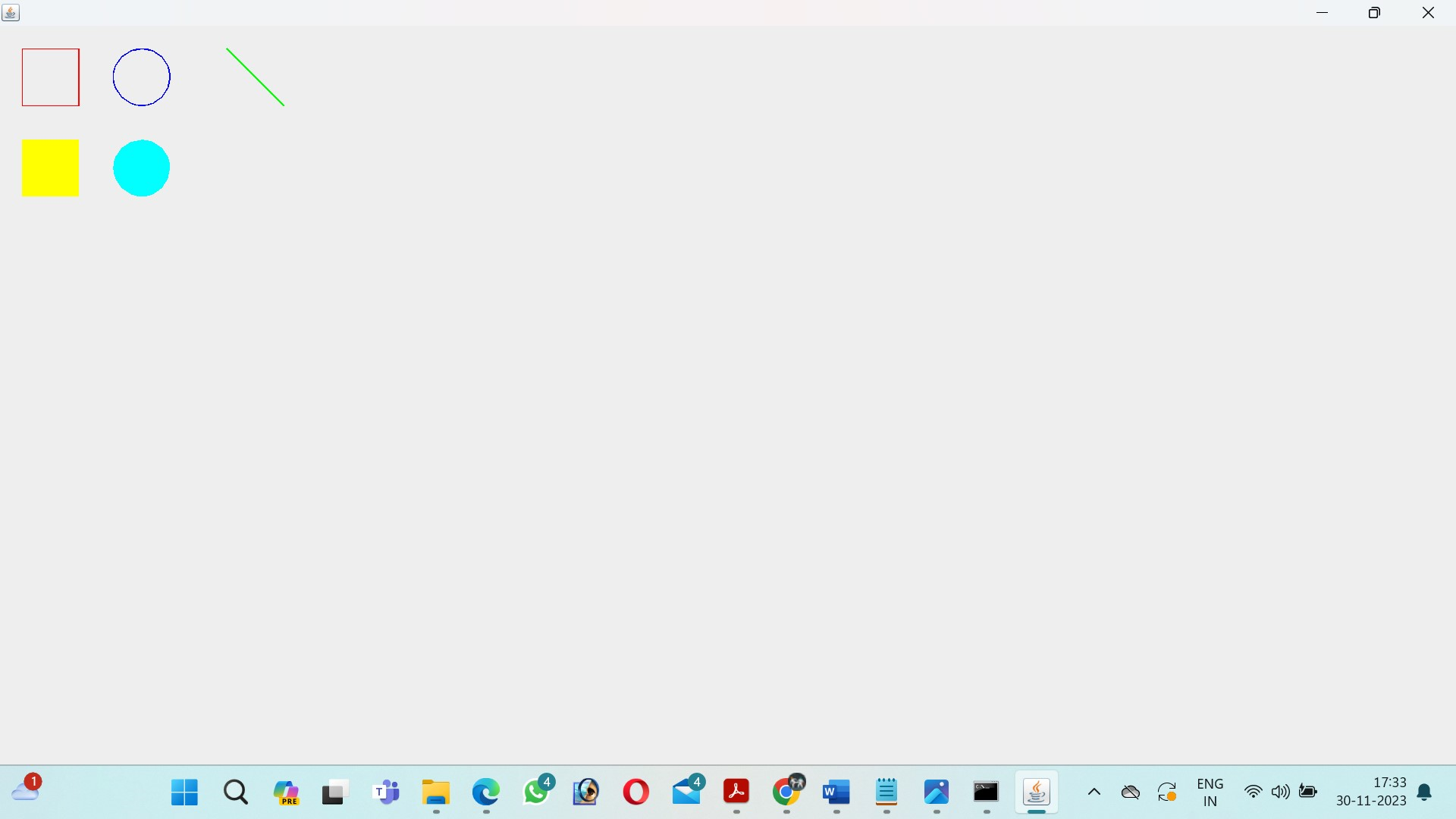
public static void main(String[] args) {

new BasicShapesDemo();

}

}

**OUTPUT:**



**iii) set background and foreground colors**

**INPUT:**

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class ColorsDemo extends JFrame {

public ColorsDemo() {

add(new MyPanel());

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

class MyPanel extends JPanel {

public void paintComponent(Graphics g) {

super.paintComponent(g);

// Setting background color

setBackground(Color.RED);

// Setting foreground color

g.setColor(Color.BLUE);

// Drawing a message

g.drawString("Hello, Colors!", 20, 50);

}

}

public static void main(String[] args) {

new ColorsDemo();

}

}

**OUTPUT:**

