**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF TECHNOLOGY**

**PANDIT DEENDAYAL ENERGY UNIVERSITY**

**SESSION 2024-25**

****

**SUBMITTED BY**

|  |  |  |
| --- | --- | --- |
| **NAME** | **:** | **Samarth Vala** |
| **ROLL NO.** | **:** | **23BCP422** |
| **DIVISION** | **:** | **Div-7 G13** |
| **COURSE NAME** | **:** | **Object Oriented Programming with JAVA LAB** |
| **COURSE CODE** | **:** | **23CP201P** |

**SUBMITTED TO**

**Dr. Nandini Modi**

**Assistant Professor**

**Department of Computer Science and Engineering**

**Pandit Deendayal Energy University**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SR NO. | LIST OF EXPERIMENTS | DATE OF COMPLETION | Pages | SIGN |
| 1 | EXPERIMENT 1 | 23/7/24 | 3 |  |
| 2 | EXPERIMENT 2 | 6/8/24 | 5 |  |
| 3 | EXPERIMENT 3 | 13/8/24 | 17 |  |
| 4 | EXPERIMENT 4 | 20/8/24 | 28 |  |
| 5 | EXPERIMENT 5 | 3/8/24 | 36 |  |
| 6 | EXPERIMENT 6 | 10/9/24 | 39 |  |
| 7 | EXPERIMENT 7 | 8/10/24 | 48 |  |
| 8 | EXPERIMENT 8 | 22/11/24 | 55 |  |
| 9 | EXPERIMENT 9 | 12/11/24 | 73 |  |
| 10 | EXPERIMENT 10 | 12/11/4 | 85 |  |

EXPERIMENT – 1

Q-1 Install JDK, setup Java environment and write a program to print ―CODING IS FUN, ENJOY IT!.

Code: -

public class L1Q1 {

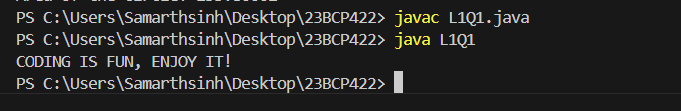
public static void main(String[] args) {

System.out.println("CODING IS FUN, ENJOY IT!");

}

}

Output: -



Q-2 Write a Java program to print the sum of two numbers.

Code: -

public class l1q2 {

public static void main(String[] args) {

int a = 10, b = 20;

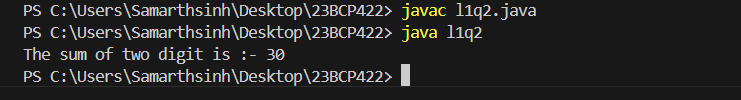
int sum = a+b;

System.out.println("The sum of two digit is :- " + sum);

}

}

Output: -



EXPERIMENT – 2

Q-1 You are developing a mathematical tool that requires generating a list of prime numbers. How would you implement a Java program to generate the first n prime numbers?

Code: -

import java.util.Scanner;

public class l2q1 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the number of primes to print:");

int n = scan.nextInt();

int p = 2;

int count = 0;

while (count < n) {

boolean isPrime = true;

for (int i = 2; i <= Math.sqrt(p); i++) {

if (p % i == 0) {

isPrime = false;

break;

}

}

if (isPrime) {

System.out.println(p);

count++;

}

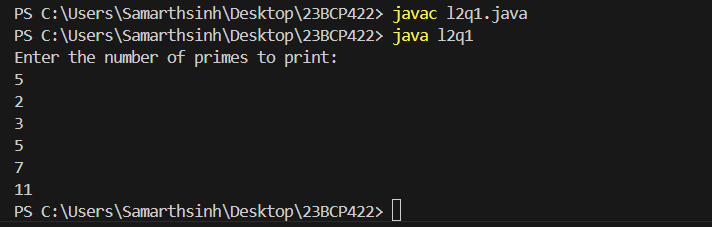
p++;

}

}

}

Output: -



Q-2 Write a program to enter two numbers and perform mathematical operations on them.

Code: -

import java.util.Scanner;

public class l2q2 {

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

System.out.println("Enter the first number:- ");

int a = scan.nextInt();

System.out.println("Enter the second number:- ");

int b = scan.nextInt();

int sum = a+b;

int multi = a\*b;

int divi = a/b;

int sub = a-b;

System.out.println("The sum is :- " + sum);

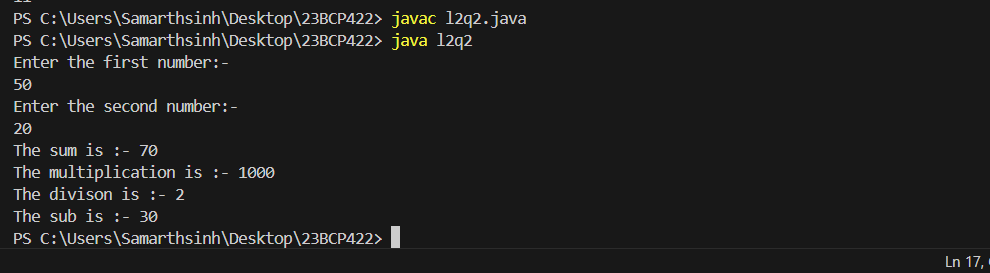
System.out.println("The multiplication is :- " + multi);

System.out.println("The divison is :- " + divi);

System.out.println("The sub is :- " + sub);

}

}

Output: - Q-3 Write a program in Java to find maximum of three numbers using conditional operator.

Code: -

import java.util.Scanner;

public class l2q3 {

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

System.out.println("Enter the first number:- ");

int a = scan.nextInt();

System.out.println("Enter the second number:- ");

int b = scan.nextInt();

System.out.println("Enter the Third number:- ");

int c = scan.nextInt();

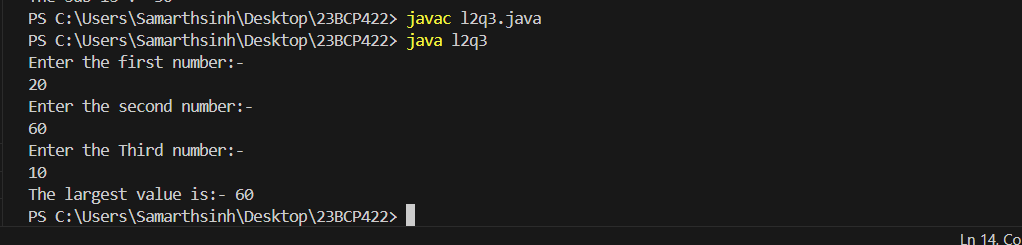
int large = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);

System.out.println("The largest value is:- " + large);

}

}

Output: -



Q-4 You're working on a text analysis feature that counts the number of vowels and consonants in a given line of text. Write a program to accept a line and check how many consonants and vowels are there in line

Code: -

import java.util.Scanner;

public class l2q4 {

public static void main(String[] args){

Scanner s = new Scanner(System.in);

System.out.print("Enter the sentance:- ");

String sen = s.nextLine();

int v = 0 , c = 0;

for(int i =0 ; i<sen.length();i++){

if(sen.charAt(i) == 'a' || sen.charAt(i) == 'u' ||sen.charAt(i) == 'o' ||sen.charAt(i) == 'i' ||sen.charAt(i) == 'e' ||sen.charAt(i) == 'A' ){

v++;

}

else{

c++;

}

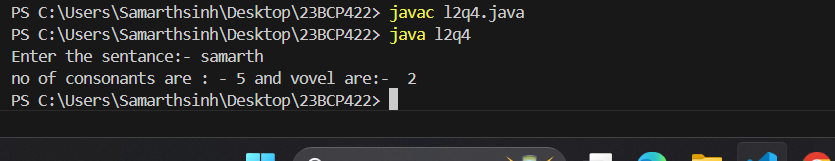
}

System.out.println("no of consonants are : - " + c + " and vovel are:- " + v);

}

}

Output: -



Q-5 Write an interactive program to print a string entered in a pyramid form. For instance, the string “stream” has to be displayed as follows:

S

S t

S t r

S t r e

S t r e am

Code: -

public class l2q5 {

public static void printPyramid(int n) {

String name = "stream";

for (int i = 0; i < n; i++) {

for (int j = n - i; j > 1; j--) {

System.out.print(" ");

}

for (int j = 0; j <= i; j++) {

System.out.print(name.charAt(j));

}

for (int j = n - i; j > 1; j--) {

System.out.print(" ");

}

System.out.println();

}

}

public static void main(String args[]) {

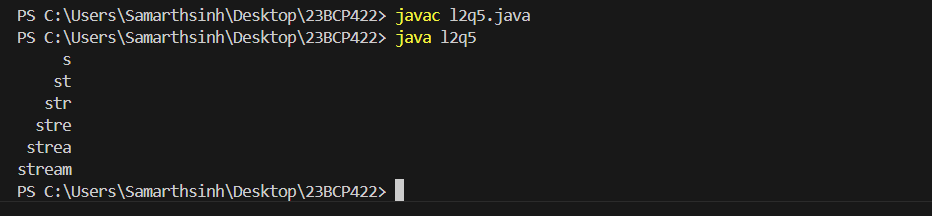
int n = 6;

printPyramid(n);

}

}

Output: -



Q-6 Java Program to Find Largest Number in an array

Code: -

import java.util.Scanner;

public class l2q6 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.print("Enter the size of array : - ");

int size = scan.nextInt();

int[] arr = new int[size];

int max = 0;

for(int i = 0; i< size ; i++){

System.out.print("Enter the " + i + " Element of array ");

arr[i] = scan.nextInt();

}

for(int i = 0; i < size; i ++ ){

if (arr[max]<arr[i]) {

max = i;

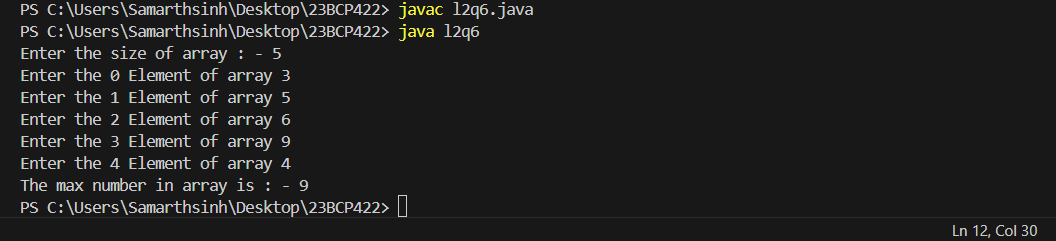
}

}

System.out.print("The max number in array is : - " + arr[max]);

}

}

Output: - 

Q-7 Write a java program to perform addition and multiplication of Two

Matrices

Code: -

import java.util.Scanner;

public class l2q7 {

public static void main(String arg[]) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter th Total row and columl:- ");

int row,col;

row = sc.nextInt();

col = sc.nextInt();

int[][] mat1 = new int[row][col];

int[][] mat2 = new int[row][col];

int[][] add = new int[row][col];

int[][] multi = new int[row][row];

System.out.println("Enter the element of matrix 1 ");

for(int i = 0; i<row ; i++ ){

for(int j= 0 ; j < col; j++){

System.out.print("Enter the element in row " + i + "colummn " + j + ":- ");

mat1[i][j] = sc.nextInt();

}

}

System.out.println("Enter the element of matrix 2");

for(int i = 0; i<row ; i++ ){

for(int j= 0 ; j < col; j++){

System.out.print("Enter the element in row " + i + "colummn " + j + ":- ");

mat2[i][j] = sc.nextInt();

}

}

System.out.println("the mat 1 is :- ");

for(int i = 0; i<row ; i++ ){

for(int j= 0 ; j < col; j++){

System.out.print(mat1[i][j]);

}

System.out.println();

}

System.out.println("the mat is :- ");

for(int i = 0; i<row ; i++ ){

for(int j= 0 ; j < col; j++){

System.out.print(mat2[i][j]+ " ");

}

System.out.println();

}

System.out.println("the Addition is:- ");

for(int i = 0; i<row ; i++ ){

for(int j= 0 ; j < col; j++){

add[i][j] = mat1[i][j] + mat2[i][j];

System.out.print(add[i][j]+ " " );

}

System.out.println();

}

System.out.println("the Multiplication is:- ");

for(int i = 0; i<row ; i++ ){

for(int j= 0 ; j < col; j++){

for(int k = 0; k< row;k++){

multi[i][j] += mat1[i][k] \* mat2[k][j];

}

System.out.print(multi[i][j] + " ");

}

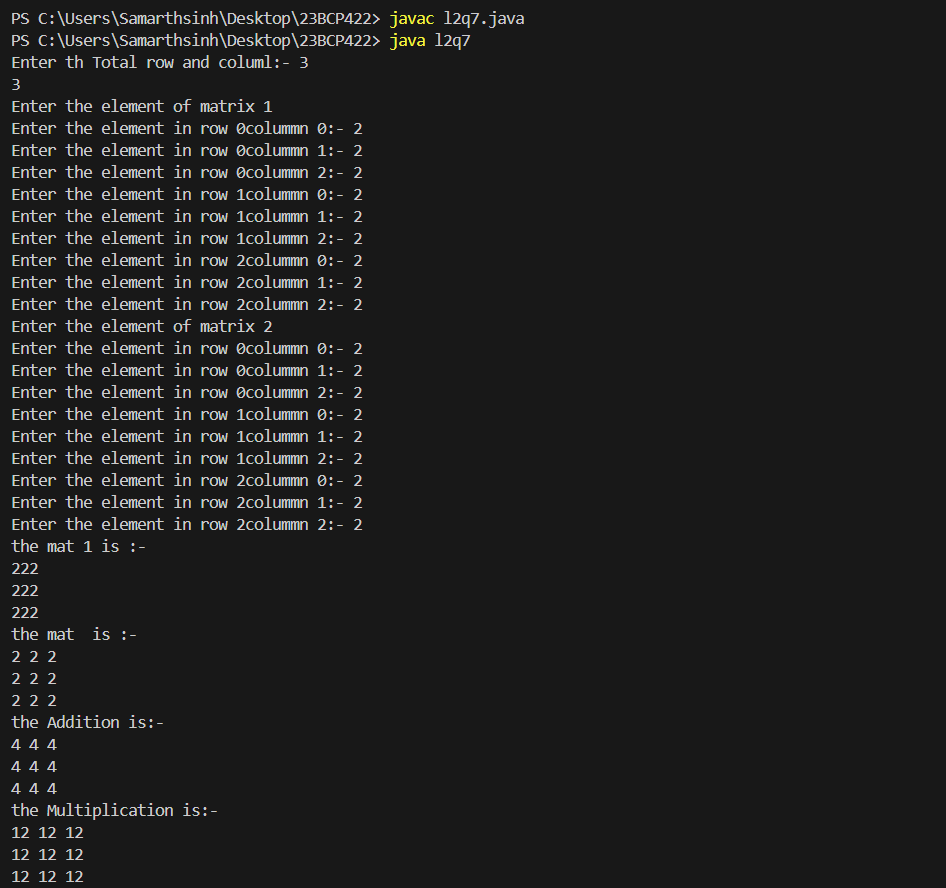
System.out.println();

}

}

}

Output: -



EXPERIMENT – 3

Q-1 Write a program to create a “distance” class with methods where distance is computed in terms of feet and inches, how to create objects of a class

Code: -

class distance{

void convert(int n){

int feet = n\*3;

int inch = n\*39;

System.out.println("distance in meter into feet is:- "+ feet + " and into inch is:- " + inch);

}

}

public class l3q1 {

public static void main(String[] args) {

int n = 10;

System.out.println("The distance in meter is:- " + n);

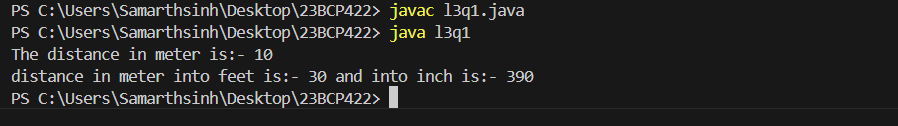
distance d = new distance();

d.convert(n);

}

}

Output: -



Q-2 Modify the “distance” class by creating constructor for assigning values (feet and inches) to the distance object. Create another object and assign second object as reference variable to another object reference variable.Further create a third object which is a clone of the first object.

Code: -

class Distance implements Cloneable {

private int feet;

private int inches;

public Distance(int feet, int inches) {

this.feet = feet;

this.inches = inches;

}

public void display() {

System.out.println("Distance: " + feet + " feet " + inches + " inches");

}

public Distance clone() throws CloneNotSupportedException {

return (Distance) super.clone();

}

}

public class l3q2 {

public static void main(String[] args) {

try {

Distance dist1 = new Distance(5, 10);

dist1.display();

Distance dist2 = new Distance(8, 6);

Distance ref = dist2;

ref.display();

Distance dist3 = dist1.clone();

dist3.display();

} catch (CloneNotSupportedException e) {

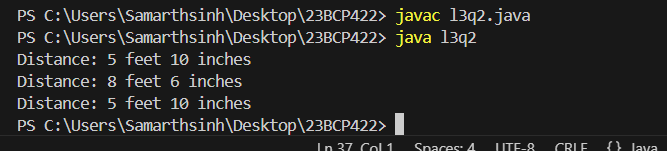
e.printStackTrace();

}

}

}

Output: -



Q-3 Write a program to show the difference between public and private access specifiers. The program should also show that primitive data types are passed by value and objects are passed by reference and to learn use of final keyword

Code: -

class AccessSpecifiers {

// Public variable - accessible from anywhere

public int publicVar = 10;

// Private variable - accessible only within the class

private int privateVar = 20;

// Method to display private variable value

public void displayPrivateVar() {

System.out.println("Private Variable: " + privateVar);

}

}

class PassByValueReference {

int value;

PassByValueReference(int value) {

this.value = value;

}

}

public class l3q3{

public static void main(String[] args) {

// Access Specifiers Example

AccessSpecifiers obj = new AccessSpecifiers();

// Accessing public variable

System.out.println("Public Variable: " + obj.publicVar);

// Access private variable using public method

obj.displayPrivateVar();

// Pass by Value (Primitive Type)

int primitiveValue = 5;

System.out.println("Before modifying primitive value: " + primitiveValue);

modifyPrimitive(primitiveValue);

System.out.println("After modifying primitive value: " + primitiveValue); // remains unchanged

// Pass by Reference (Object Type)

PassByValueReference objectValue = new PassByValueReference(10);

System.out.println("Before modifying object value: " + objectValue.value);

modifyObject(objectValue);

System.out.println("After modifying object value: " + objectValue.value); // gets changed

// Final Keyword Example

final int finalVar = 100; // final variable, cannot be changed

System.out.println("Final Variable: " + finalVar);

final PassByValueReference finalObject = new PassByValueReference(30);

System.out.println("Before modifying final object's value: " + finalObject.value);

finalObject.value = 40;

System.out.println("After modifying final object's value: " + finalObject.value);

}

// Method to demonstrate pass by value

public static void modifyPrimitive(int value) {

value = 20; // Does not change the original value

}

// Method to demonstrate pass by reference

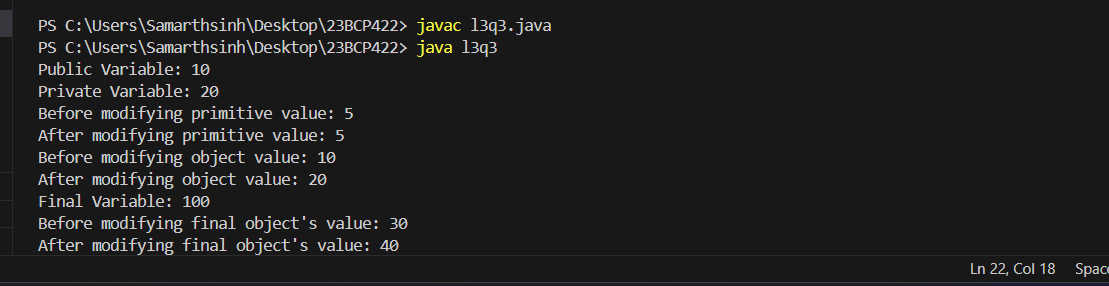
public static void modifyObject(PassByValueReference obj) {

obj.value = 20; // Changes the value inside the object

}

}

Output: -



Q-4 Write a program that implements two constructors in the class. We call the other constructor using ‘this’ pointer, from the default constructor of the class

Code: -

class sam{

int age;

String name;

public void para(int age, String name){

this.age = age;

this.name = name;

System.out.println("The name is:- " + this.name + " age is:- " + this.age );

}

public void para(){

this.para(42,"samarth");

}

}

public class l3q4 {

public static void main(String[] args) {

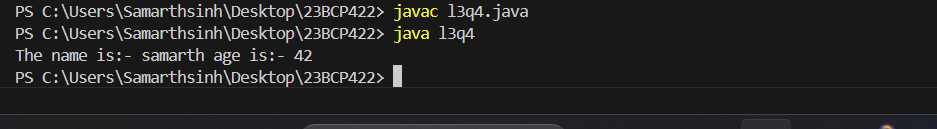
sam n = new sam();

n.para();

}

}

Output: -



Q-5 Write a program in Java in which a subclass constructor invokes the constructor of the super class and instantiate the values.

Code: -

class Vehicle {

String brand;

int year;

public Vehicle(String brand, int year) {

this.brand = brand;

this.year = year;

System.out.println("Vehicle constructor called.");

}

public void displayInfo() {

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

System.out.println("Year: " + year);

}

}

class Car extends Vehicle {

String model;

public Car(String brand, int year, String model) {

super(brand, year);

this.model = model;

System.out.println("Car constructor called.");

}

public void displayInfo() {

super.displayInfo();

System.out.println("Model: " + model);

}

}

public class l3q5 {

public static void main(String[] args) {

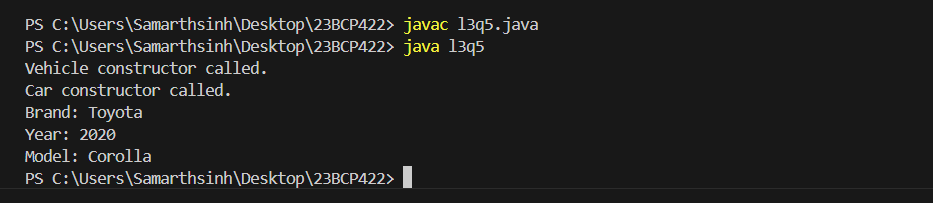
Car myCar = new Car("Toyota", 2020, "Corolla");

myCar.displayInfo();

}

}

Output: -



Q-6 Write a program in Java to develop overloaded constructor. Also develop the copy constructor to create a new object with the state of the existing object.

Code: -

class student{

int id;

String name;

int age;

student(int id, String name){

this.id = id;

this.name = name;

}

student(int id, String name, int age){

this.id = id;

this.name = name;

this.age = age;

}

void display(){System.out.println(id+" "+name+" "+age);}

}

public class l3q6 {

public static void main(String[] args){

student s1 = new student(1, "Alice");

student s2 = new student(2, "Bob", 20);

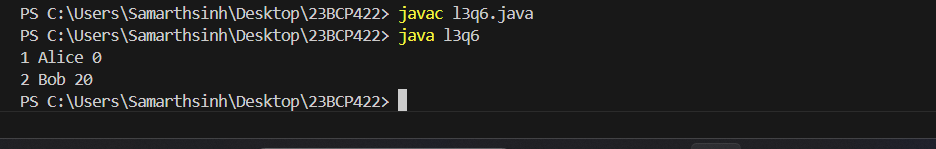
s1.display();

s2.display();

}

}

Output: -



EXPERIMENT – 4

Q-1 Write a program in Java to demonstrate single inheritance, multilevel inheritance and hierarchical inheritance.

Code: -

Single

class animal{

void eat(){System.out.println("Eating....");}

}

class dog extends animal {

void bark(){System.out.println("barking....");}

}

class lab4{

public static void main(String[] args){

dog d = new dog();

d.eat();

d.bark();

}

}

Multilevel

class A {

void methodA() {

System.out.println("Method A");

}

}

class B extends A {

void methodB() {

System.out.println("Method B");

}

}

class C extends B {

void methodC() {

System.out.println("Method C");

}

}

public class Main {

public static void main(String[] args) {

C obj = new C();

obj.methodA();

obj.methodB();

obj.methodC();

}

}

Hierarchical

class animal{

void eat(){System.out.println("Eating....");}

}

class dog extends animal {

void bark(){System.out.println("bark....");}

}

class cat extends animal{

void meow(){System.out.println("meow.....");}

}

class lab4q13{

public static void main(String[] args){

dog d = new dog();

d.eat();

d.bark();

cat c = new cat();

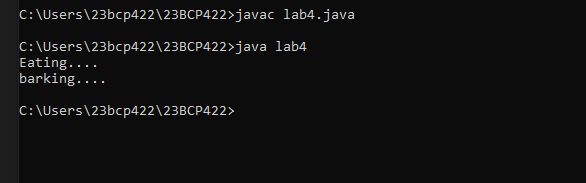
c.eat();

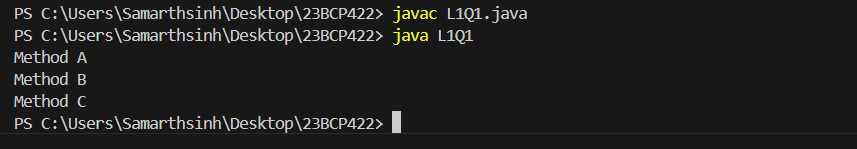
c.meow();

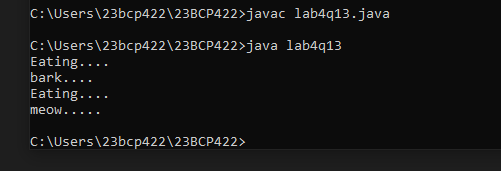
}

}

Output: -







Q-2 Java Program to demonstrate the real scenario (e.g., bank) of Java Method Overriding where three classes are overriding the method of a parent class. Creating a parent class.Code: -

class Bank {

double getInterestRate() {

return 0.0;

}

}

class BankA extends Bank {

double getInterestRate() {

return 5.0;

}

}

class BankB extends Bank {

double getInterestRate() {

return 6.5;

}

}

class BankC extends Bank {

double getInterestRate() {

return 7.0;

}

}

public class l4q2{

public static void main(String[] args) {

Bank bankA = new BankA();

Bank bankB = new BankB();

Bank bankC = new BankC();

System.out.println("Bank A Interest Rate: " + bankA.getInterestRate() + "%");

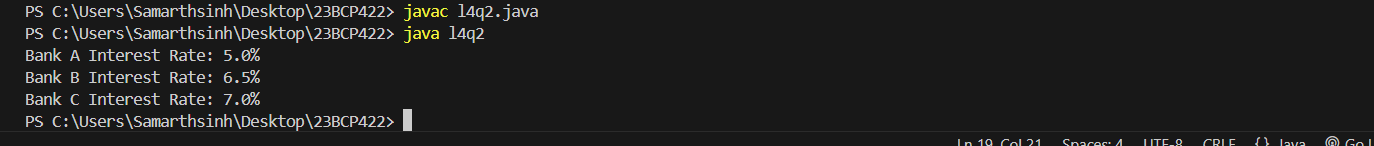
System.out.println("Bank B Interest Rate: " + bankB.getInterestRate() + "%");

System.out.println("Bank C Interest Rate: " + bankC.getInterestRate() + "%");

}

}

Output: -



Q-3 Write a java program for the use of super and this keyword.

Code: -

// Parent class

class Animal {

String name;

Animal(String name) {

this.name = name;

}

void display() {

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

}

}

class Dog extends Animal {

String breed;

Dog(String name, String breed) {

super(name);

this.breed = breed;

}

void display() {

super.display();

System.out.println("Dog breed: " + breed);

}

}

public class labb4q3{

public static void main(String[] args) {

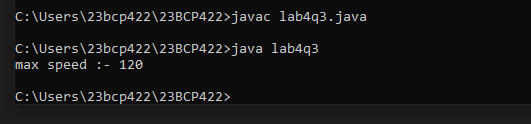
Dog dog = new Dog("Buddy", "Golden Retriever");

dog.display();

}

}

Output: -



Q-4 Write a java program for the use of final keyword

Code: -

class Animal {

final String name;

Animal(String name) {

this.name = name;

}

void displayName() {

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

}

}

class Dog extends Animal {

Dog(String name) {

super(name);

}

void bark() {

System.out.println("The dog is barking");

}

}

public class lab4q4 {

public static void main(String[] args) {

Dog dog = new Dog("Buddy");

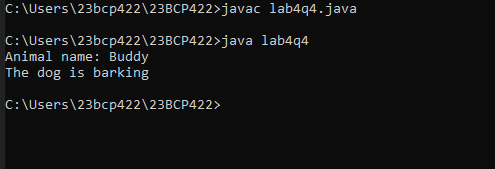
dog.displayName();

dog.bark();

}

}

Output: -



EXPERIMENT – 5

Q-1 Write a program that implements simple example of Runtime Polymorphism with multilevel inheritance. (e.g., Animal or Shape)

Code: -

class animal{

void speak(){

System.out.println("Animal is speaking");

}

}

class dog extends animal{

void speak(){

System.out.println("Dog is speaking");

}

}

class puppy extends dog{

void speak(){

System.out.println("Puppy is speaking");

}

}

class l5q1{

public static void main(String[] args){

animal a = new animal();

dog d = new dog();

puppy p = new puppy();

animal r;

r = a;

r.speak();

r = d;

r.speak();

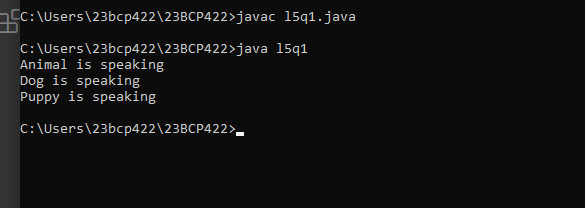
r = p;

r.speak();

}

}

Output: -



Q-2 Write a program to compute if one string is a rotation of another. For example, pit is rotation of tip as pit has same character as tip

Code: -

import java.util.Scanner;

abstract class StringChecker {

abstract boolean check(String s1, String s2);

}

class ReverseChecker extends StringChecker {

boolean check(String s1, String s2) {

if (s1.length() != s2.length()) {

return false;

}

String concate=s1+s2;

return concate.contains(s2);

}

}

class StringRotate {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String s1 = sc.next();

String s2 = sc.next();

StringChecker checker = new ReverseChecker();

if (checker.check(s1, s2)) {

System.out.println(s2 + " is the rotation of " + s1);

} else {

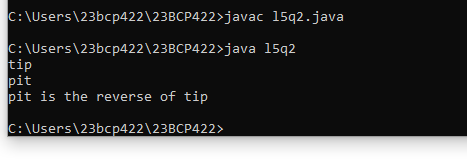
System.out.println(s2 + " is NOT the rotation of " + s1);

}

}

}

Output: -



EXPERIMENT – 6

Q-1 Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e. area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.

Code: -

import java.util.Scanner;

abstract class shape {

abstract double area();

}

class triangle extends shape {

int height, base;

triangle(int height, int base) {

this.height = height;

this.base = base;

}

double area() {

return 0.5 \* height \* base;

}

}

class rectangle extends shape {

int length, breadth;

rectangle(int length, int breadth) {

this.length = length;

this.breadth = breadth;

}

double area() {

return length \* breadth;

}

}

class circle extends shape {

int radius;

circle(int radius) {

this.radius = radius;

}

double area() {

return Math.PI \* radius \* radius;

}

}

public class l6q1 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int l, bre, r, h, b;

System.out.println("Enter the height and base of the triangle: ");

h = scan.nextInt();

b = scan.nextInt();

System.out.println("Enter the length and breadth of the rectangle: ");

l = scan.nextInt();

bre = scan.nextInt();

System.out.println("Enter the radius of the circle: ");

r = scan.nextInt();

triangle t = new triangle(h, b);

rectangle rect = new rectangle(l, bre);

circle c = new circle(r);

System.out.println("Area of the triangle: " + t.area());

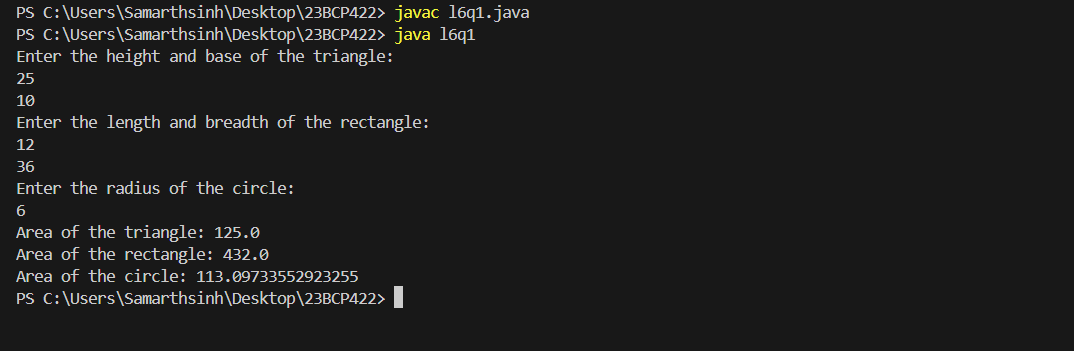
System.out.println("Area of the rectangle: " + rect.area());

System.out.println("Area of the circle: " + c.area());

}

}

Output: -



Q-2 Write a Java program to create an abstract class Employee with abstract methods calculateSalary() and displayInfo(). Create subclasses Manager and Programmer that extend the Employee class and implement the respective methods to calculate salary and display information for each

role.

Code: -

import java.util.Scanner;

abstract class Employee {

String name;

int employeeId;

double baseSalary;

Employee(String name, int employeeId, double baseSalary) {

this.name = name;

this.employeeId = employeeId;

this.baseSalary = baseSalary;

}

abstract double calculateSalary();

abstract void displayInfo();

}

class Manager extends Employee {

Manager(String name, int employeeId, double baseSalary) {

super(name, employeeId, baseSalary);

}

double calculateSalary() {

return baseSalary;

}

void displayInfo() {

System.out.println("Manager Info:");

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

System.out.println("Employee ID: " + employeeId);

System.out.println("Base Salary: " + baseSalary);

System.out.println("Total Salary: " + calculateSalary());

}

}

class Programmer extends Employee {

double incentive;

Programmer(String name, int employeeId, double baseSalary, double incentive) {

super(name, employeeId, baseSalary);

this.incentive = incentive;

}

double calculateSalary() {

return baseSalary + incentive;

}

void displayInfo() {

System.out.println("Programmer Info:");

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

System.out.println("Employee ID: " + employeeId);

System.out.println("Base Salary: " + baseSalary);

System.out.println("Incentive: " + incentive);

System.out.println("Total Salary: " + calculateSalary());

}

}

public class l6q2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter details for Manager:");

System.out.print("Name: ");

String managerName = scanner.nextLine();

System.out.print("Employee ID: ");

int managerId = scanner.nextInt();

System.out.print("Base Salary: ");

double managerBaseSalary = scanner.nextDouble();

Manager manager = new Manager(managerName, managerId, managerBaseSalary);

scanner.nextLine();

System.out.println("\nEnter details for Programmer:");

System.out.print("Name: ");

String programmerName = scanner.nextLine();

System.out.print("Employee ID: ");

int programmerId = scanner.nextInt();

System.out.print("Base Salary: ");

double programmerBaseSalary = scanner.nextDouble();

System.out.print("Incentive: ");

double programmerIncentive = scanner.nextDouble();

Programmer programmer = new Programmer(programmerName, programmerId, programmerBaseSalary, programmerIncentive);

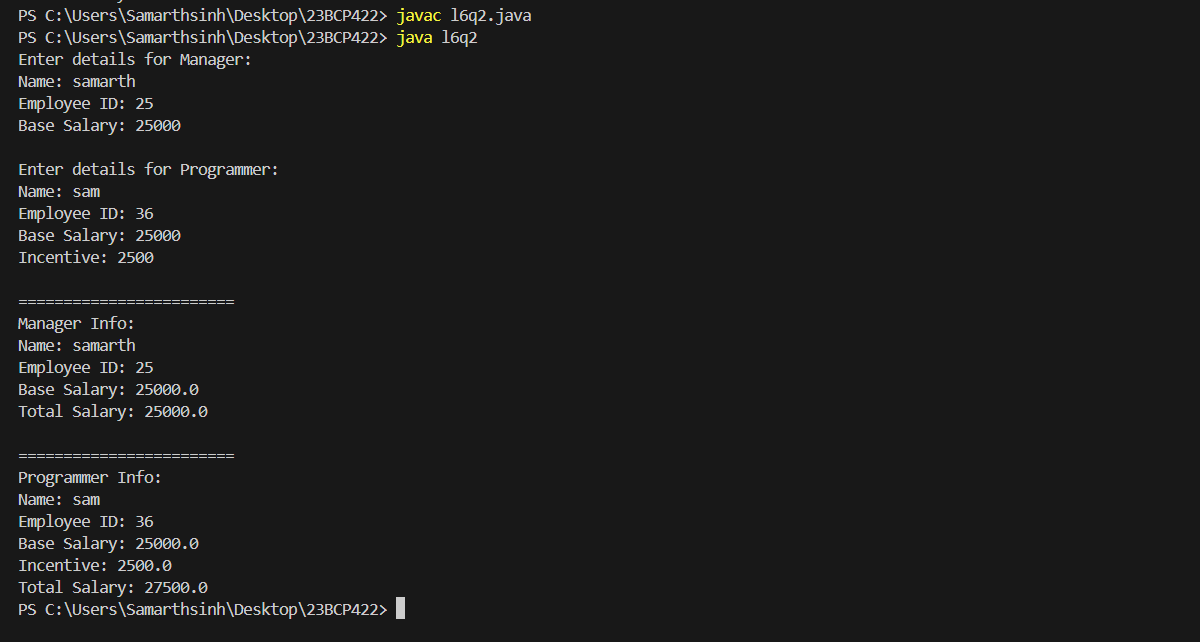
System.out.println("\n========================");

manager.displayInfo();

System.out.println("\n========================");

programmer.displayInfo();

}

}Output: - 

Q-3 Write a Java program to create an interface Shape with the getArea() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getArea() method for each of the three classes.

Code: -

interface Shape {

float pi = 3.14f;

float getArea();

}

class Rectangle implements Shape {

int l, bre;

Rectangle(int l, int bre) {

this.l = l;

this.bre = bre;

}

public float getArea() {

return l \* bre;

}

}

class Triangle implements Shape {

int h, b;

Triangle(int h, int b) {

this.h = h;

this.b = b;

}

public float getArea() {

return 0.5f \* h \* b;

}

}

class Circle implements Shape {

int r;

Circle(int r) {

this.r = r;

}

public float getArea() {

return pi \* r \* r;

}

}

public class l6q3 {

public static void main(String[] args) {

Rectangle rect = new Rectangle(10, 5);

Triangle tri = new Triangle(8, 6);

Circle circ = new Circle(7);

System.out.println("Area of the rectangle: " + rect.getArea());

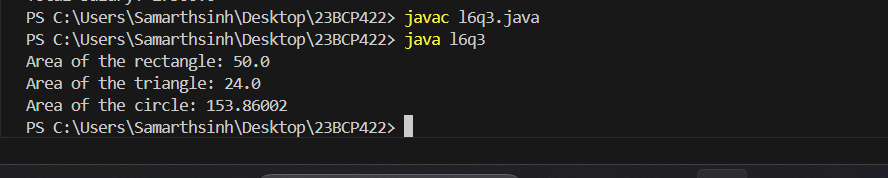
System.out.println("Area of the triangle: " + tri.getArea());

System.out.println("Area of the circle: " + circ.getArea());

}

}

Output: -



EXPERIMENT – 7

Q-1 Write a Java program for try-catch block in exception handling

Code: -

public class l7q1 {

public static void main(String[] args){

try{

int data = 50/0;

System.out.println("Rest of the code");

}

catch(ArithmeticException e){

System.out.println(e);

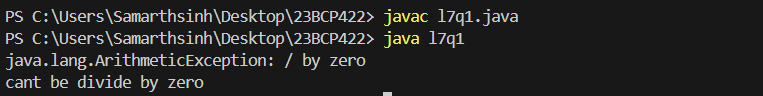
System.out.println("cant be divide by zero");

}

}

}

Output: -



Q-2 Write a Java for multiple catch block in exception handling.

Code: -

public class l7q2 {

public static void main(String[] args){

try{

int a[] = new int[5];

a[5] = 30/0;

System.out.println("the array is "+ a[10]);

//String s = Null

//print(s.length())

}

catch(ArithmeticException e){

System.out.println(e);

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println(e);

}

catch(Exception e){

System.out.println("parent exception occured");

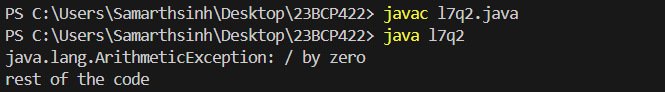
}

System.out.println("rest of the code ");

}

}

Output: -



Q-3 Write a java program for nested of try in exception handling

Code: -

public class l7q3 {

public static void main(String[] args){

try{

try{

System.out.println("going to dvide by zero ");

int data = 30/0;

}

catch(ArithmeticException e){

System.out.println(e);

}

try{

System.out.println("trying for array");

int a[] = new int[5];

System.out.println(a[10]);

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println(e);

}

System.out.println("Other statement ");

}

catch(Exception e){

System.out.println("handled exception (outer catch )");

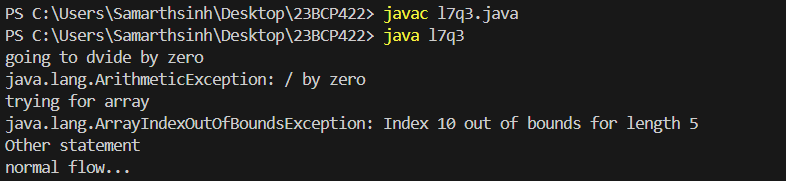
}

System.out.println("normal flow... ");

}

}

Output: -



Q-4 Write a small application in Java to develop Banking Application in which user deposits the amount Rs 1000.00 and then start withdrawing of Rs 400.00, Rs 300.00 and it throws exception “Not Sufficient Fund” when user withdraws Rs. 500 thereafter

Code: -

import java.util.Scanner;

public class l7q4 {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

int balance = 0;

int i = 0;

do {

System.out.println("\n1. Deposit");

System.out.println("2. Withdraw");

System.out.println("3. Exit");

System.out.print("Choose an option: ");

i = s.nextInt();

switch (i) {

case 1:

// Deposit logic

System.out.print("Enter deposit amount: Rs ");

int amount = s.nextInt();

balance = balance + amount;

System.out.println("Deposited: Rs " + amount);

System.out.println("Current Balance: Rs " + balance);

break;

case 2:

// Withdraw logic

System.out.print("Enter withdrawal amount: Rs ");

int withAmount = s.nextInt();

if (withAmount > balance) {

System.out.println("Not Sufficient Fund to withdraw Rs " + withAmount);

} else {

balance -= withAmount;

System.out.println("Withdrawn: Rs " + withAmount);

System.out.println("Current Balance: Rs " + balance);

}

break;

case 3:

System.out.println("Thank you for banking with us!");

break;

default:

System.out.println("Invalid option! Please choose again.");

break;

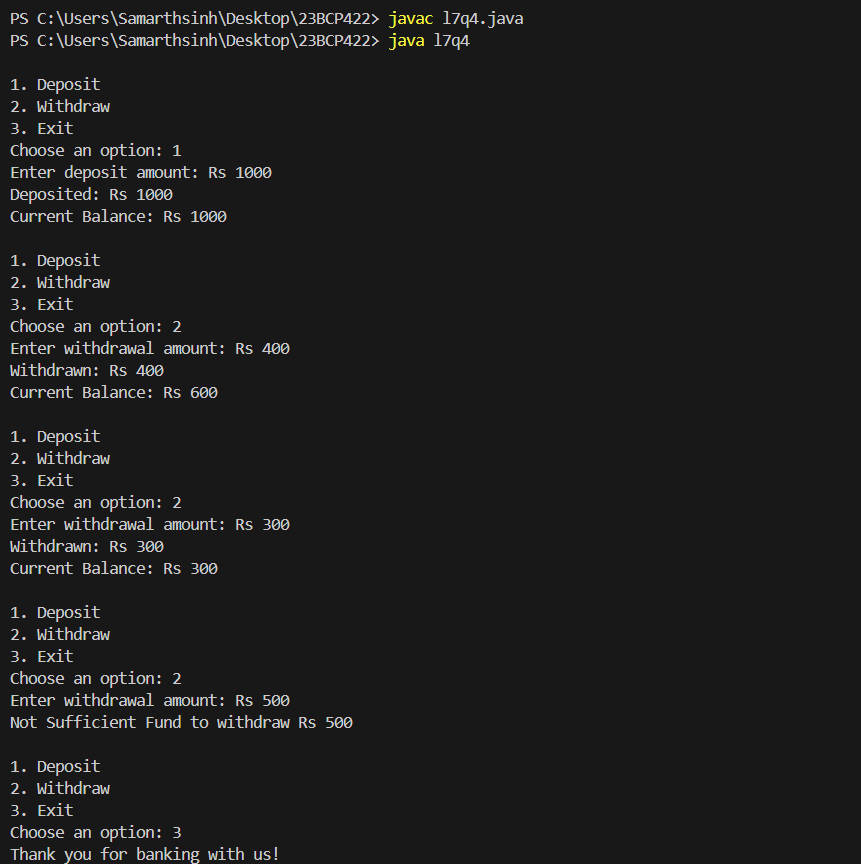
}

} while (i != 3);

}

}

Output: -



Q-5 Write a java program for finally block in exception handling

Code: -

public class l7q5 {

public static void main(String[] args){

try{

int data = 25/0;

}

catch(ArithmeticException e){

System.out.println("error is handled ");

System.out.println(e);

}

finally{

System.out.println("try catch is complete and final is excuted ");

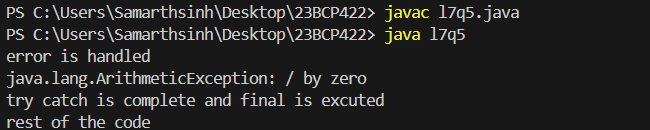
}

System.out.println("rest of the code ");

}

}

Output: -



EXPERIMENT – 8

Q-1 Read a content from file: calculate number of sentences, words and characters from the file.

Code: -

import java.io.\*;

public class l8q1 {

public static void main(String[] args){

String filePath = "1.txt";

int senc =0;

int charc = 0;

int wordc = 0;

try(BufferedReader reader = new BufferedReader(new FileReader(filePath))){

String line;

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

charc = line.length() + 1;

// Counting words using space as a delimiter

String[] words = line.split("\\s+");

wordc += words.length;

// Counting sentences using common sentence-ending punctuation

String[] sentences = line.split("[.!?]+");

senc += sentences.length;

}

System.out.println("Number of sentences: " + senc);

System.out.println("Number of words: " + wordc);

System.out.println("Number of characters: " + charc);

}

catch(IOException e ){

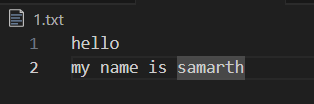
System.out.println("cant read the file");

}

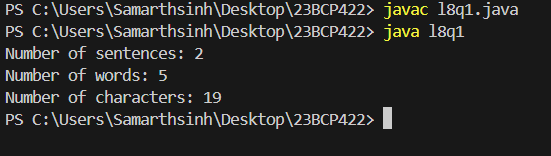
}

}

Input: -



Output: -



Q-2 Read content from a file convert it to uppercase and save it into another file.

Code: -

import java.io.\*;

public class l8q2 {

public static void main(String[] args) {

// Declare file paths

String inputFilePath = "input.txt";

String outputFilePath = "output.txt";

// Read from input file, convert to uppercase, and write to output file

try (

BufferedReader reader = new BufferedReader(new FileReader(inputFilePath));

BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilePath))

) {

System.out.println("\nReading content of the input file:");

String line;

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

System.out.println(line); // Print each line of the input file

// Convert to uppercase and write to output file

String upperc = line.toUpperCase();

writer.write(upperc);

writer.newLine();

}

// Close writer to ensure content is written to output file

writer.flush();

// Print content of the output file

System.out.println("\nContent of the output file (in uppercase):");

try (BufferedReader oreader = new BufferedReader(new FileReader(outputFilePath))) {

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

System.out.println(line); // Print each line of the output file

}

}

} catch (FileNotFoundException e) {

System.out.println("Input file not found: " + e.getMessage());

} catch (IOException e) {

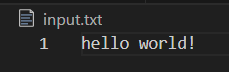
System.out.println("An error occurred: " + e.getMessage());

}

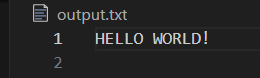
}

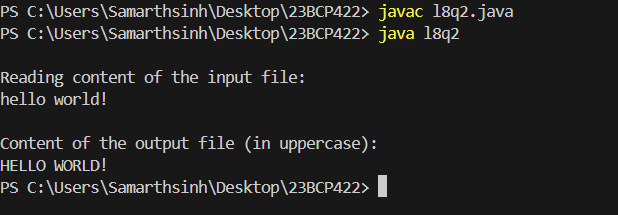
}

Input: -



Output: -





Q-3 Remove duplicate lines from a File.

Code: -

import java.io.\*;

import java.util.HashSet;

import java.util.Set;

public class l8q3 {

public static void main(String[] args) {

String inputFilePath = "input.txt"; // Input file with potential duplicates

String outputFilePath = "output.txt"; // Output file to store unique lines

Set<String> uniqueLines = new HashSet<>(); // Set to store unique lines

// Read from the input file and collect unique lines

try (BufferedReader reader = new BufferedReader(new FileReader(inputFilePath))) {

String line;

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

uniqueLines.add(line); // Add line to the set (duplicates are ignored)

}

} catch (IOException e) {

System.out.println("An error occurred while reading the file: " + e.getMessage());

return; // Exit if there's an error reading the file

}

// Write unique lines to the output file

try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilePath))) {

for (String uniqueLine : uniqueLines) {

writer.write(uniqueLine);

writer.newLine(); // Write each unique line followed by a newline

}

System.out.println("Duplicates removed. Unique lines written to " + outputFilePath);

} catch (IOException e) {

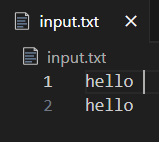
System.out.println("An error occurred while writing to the file: " + e.getMessage());

}

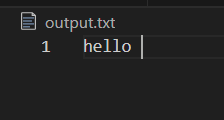
}

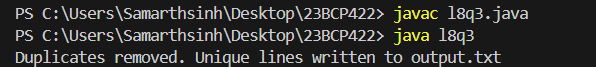
}

Input: -



Output: -





Q-4 Create a class called Student. Write a student manager program to manipulate the student information from files by using FileInputStream and FileOutputStream

Code: -

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

class Student {

private int id;

private String name;

private int age;

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public int getId() { return id; }

public String getName() { return name; }

public int getAge() { return age; }

@Override

public String toString() {

return id + "," + name + "," + age;

}

public static Student fromString(String line) {

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

int id = Integer.parseInt(parts[0].trim());

String name = parts[1].trim();

int age = Integer.parseInt(parts[2].trim());

return new Student(id, name, age);

}

}

class StudentManager {

private List<Student> students;

private String filePath;

public StudentManager(String filePath) {

this.filePath = filePath;

this.students = new ArrayList<>();

loadStudents();

}

public void addStudent(Student student) {

students.add(student);

saveStudents();

}

public void displayStudents() {

for (Student student : students) {

System.out.println("Student ID: " + student.getId() + ", Name: " + student.getName() + ", Age: " + student.getAge());

}

}

private void saveStudents() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

for (Student student : students) {

writer.write(student.toString());

writer.newLine();

}

System.out.println("Student data saved successfully to " + filePath);

} catch (IOException e) {

System.out.println("An error occurred while saving student data.");

e.printStackTrace();

}

}

private void loadStudents() {

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line;

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

students.add(Student.fromString(line));

}

System.out.println("Student data loaded successfully from " + filePath);

} catch (FileNotFoundException e) {

System.out.println("No previous student data found. Starting with an empty list.");

} catch (IOException e) {

System.out.println("An error occurred while loading student data.");

e.printStackTrace();

}

}

}

public class l8q4{

public static void main(String[] args) {

String filePath = "student.txt";

StudentManager manager = new StudentManager(filePath);

manager.addStudent(new Student(1, "Alice", 20));

manager.addStudent(new Student(2, "Bob", 22));

manager.addStudent(new Student(3, "Charlie", 19));

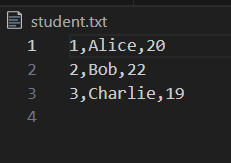
System.out.println("All Students:");

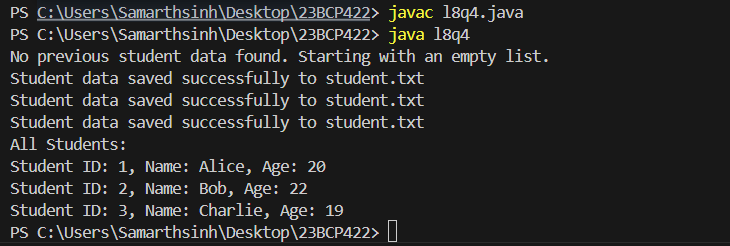
manager.displayStudents();

}

}

Output: -





Q-5 Refine the student manager program to manipulate the student information from files by using the BufferedReader and BufferedWriter

Code: -

import java.io.\*;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

class Student {

private int id;

private String name;

private int age;

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public int getId() { return id; }

public String getName() { return name; }

public int getAge() { return age; }

public void setName(String name) { this.name = name; }

public void setAge(int age) { this.age = age; }

@Override

public String toString() {

return id + "," + name + "," + age;

}

public static Student fromString(String line) {

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

int id = Integer.parseInt(parts[0].trim());

String name = parts[1].trim();

int age = Integer.parseInt(parts[2].trim());

return new Student(id, name, age);

}

}

class StudentManager {

private List<Student> students;

private String filePath;

public StudentManager(String filePath) {

this.filePath = filePath;

this.students = new ArrayList<>();

loadStudents();

}

public void addStudent(Student student) {

students.add(student);

saveStudents();

}

public void displayStudents() {

for (Student student : students) {

System.out.println("ID: " + student.getId() + ", Name: " + student.getName() + ", Age: " + student.getAge());

}

}

public void removeStudent(int id) {

Iterator<Student> iterator = students.iterator();

while (iterator.hasNext()) {

Student student = iterator.next();

if (student.getId() == id) {

iterator.remove();

saveStudents();

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

return;

}

}

System.out.println("Student with ID " + id + " not found.");

}

public void updateStudent(int id, String newName, int newAge) {

for (Student student : students) {

if (student.getId() == id) {

student.setName(newName);

student.setAge(newAge);

saveStudents();

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

return;

}

}

System.out.println("Student with ID " + id + " not found.");

}

private void saveStudents() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

for (Student student : students) {

writer.write(student.toString());

writer.newLine();

}

System.out.println("Student data saved successfully to " + filePath);

} catch (IOException e) {

System.out.println("An error occurred while saving student data.");

e.printStackTrace();

}

}

private void loadStudents() {

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line;

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

students.add(Student.fromString(line));

}

System.out.println("Student data loaded successfully from " + filePath);

} catch (FileNotFoundException e) {

System.out.println("No previous student data found. Starting with an empty list.");

} catch (IOException e) {

System.out.println("An error occurred while loading student data.");

e.printStackTrace();

}

}

}

public class l8q5{

public static void main(String[] args) {

String filePath = "student.txt";

StudentManager manager = new StudentManager(filePath);

manager.addStudent(new Student(1, "Alice", 25));

manager.addStudent(new Student(2, "Bob", 26));

manager.addStudent(new Student(3, "Charlie", 19));

System.out.println("All Students:");

manager.displayStudents();

manager.removeStudent(2);

manager.updateStudent(3, "Charlie Brown", 20);

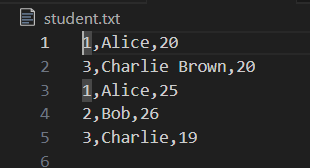
System.out.println("Updated Student List:");

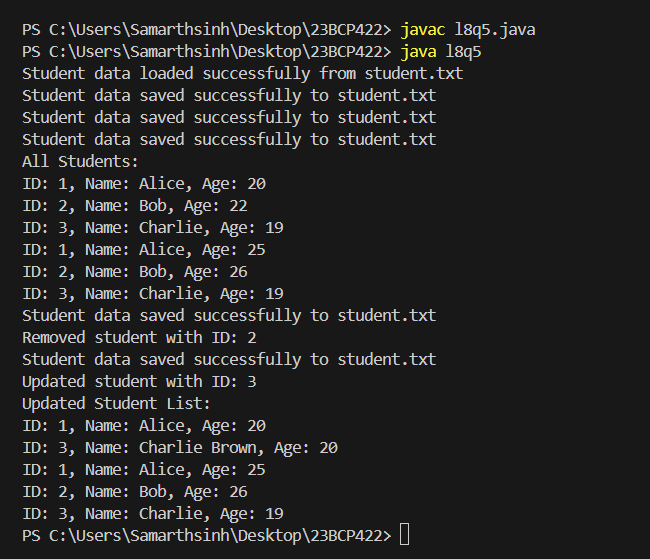
manager.displayStudents();

}

}

Output: -





Q-6 Write a program to manipulate the information from files by using the Reader and Writer class. Assume suitable data.

Code: -

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

class Student {

private int id;

private String name;

private int age;

public Student(int id, String name, int age) {

this.id = id;

this.name = name;

this.age = age;

}

public int getId() { return id; }

public String getName() { return name; }

public int getAge() { return age; }

@Override

public String toString() {

return id + "," + name + "," + age;

}

public static Student fromString(String line) {

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

int id = Integer.parseInt(parts[0].trim());

String name = parts[1].trim();

int age = Integer.parseInt(parts[2].trim());

return new Student(id, name, age);

}

}

class StudentManager {

private List<Student> students;

private String filePath;

public StudentManager(String filePath) {

this.filePath = filePath;

this.students = new ArrayList<>();

loadStudents();

}

public void addStudent(Student student) {

students.add(student);

saveStudents();

}

public void displayStudents() {

for (Student student : students) {

System.out.println("Student ID: " + student.getId() + ", Name: " + student.getName() + ", Age: " + student.getAge());

}

}

private void saveStudents() {

try (Writer writer = new FileWriter(filePath)) {

for (Student student : students) {

writer.write(student.toString() + "\n");

}

System.out.println("Student data saved successfully.");

} catch (IOException e) {

System.out.println("An error occurred while saving student data.");

e.printStackTrace();

}

}

private void loadStudents() {

try (Reader reader = new FileReader(filePath);

BufferedReader bufferedReader = new BufferedReader(reader)) {

String line;

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

students.add(Student.fromString(line));

}

System.out.println("Student data loaded successfully.");

} catch (FileNotFoundException e) {

System.out.println("No previous student data found. Starting with an empty list.");

} catch (IOException e) {

System.out.println("An error occurred while loading student data.");

e.printStackTrace();

}

}

}

public class Exp8\_6 {

public static void main(String[] args) {

String filePath = "students.txt";

StudentManager manager = new StudentManager(filePath);

manager.addStudent(new Student(1, "Alice", 20));

manager.addStudent(new Student(2, "Bob", 22));

manager.addStudent(new Student(3, "Charlie", 19));

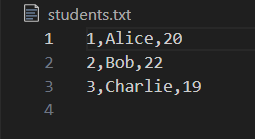
System.out.println("All Students:");

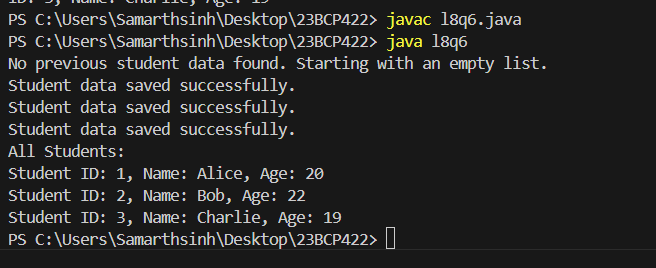
manager.displayStudents();

}

}

Output: -





EXPERIMENT – 9

Q-1 Write a Java program to demonstrate how to create and start a thread using both theThread class and the Runnable interface.

Code: -

// Thread by extending the Thread class

class A extends Thread {

public void run() {

for (int i = 0; i <= 5; i++) {

System.out.println("\tFrom thread A: " + i);

}

System.out.println("Exiting from thread A");

}

}

// Thread by implementing the Runnable interface

class B implements Runnable {

public void run() {

for (int i = 0; i <= 5; i++) {

System.out.println("\tFrom thread B: " + i);

}

System.out.println("Exiting from thread B");

}

}

public class l9q1 {

public static void main(String[] args) {

// Starting thread A

A threadA = new A();

threadA.start();

// Starting thread B

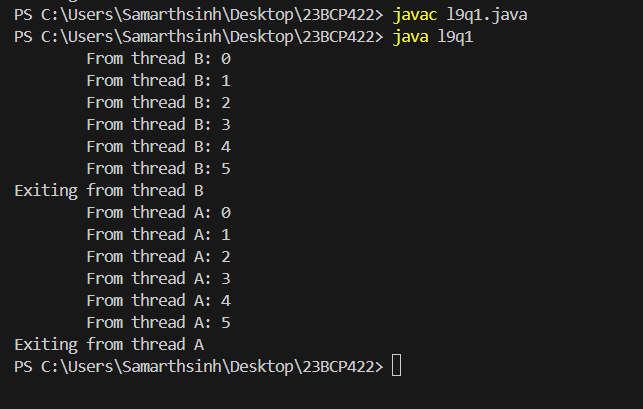
Thread threadB = new Thread(new B());

threadB.start();

}

}

Output: -



Q-2 Write a Java program that illustrates thread synchronization by ensuring multiple threads can safely access a shared resource without causing data

inconsistency.

Code: -

class Counter {

private int count = 0;

// Synchronized method to safely increment the counter

public synchronized void increment() {

count++;

System.out.println(Thread.currentThread().getName() + " incremented count to: " + count);

}

public int getCount() {

return count;

}

}

class CounterThread extends Thread {

private final Counter counter;

public CounterThread(Counter counter) {

this.counter = counter;

}

@Override

public void run() {

for (int i = 0; i < 5; i++) {

counter.increment(); // Safely increment the shared counter

}

}

}

public class l9q2 {

public static void main(String[] args) {

Counter sharedCounter = new Counter();

// Create two threads that share the same Counter instance

CounterThread thread1 = new CounterThread(sharedCounter);

CounterThread thread2 = new CounterThread(sharedCounter);

// Start both threads

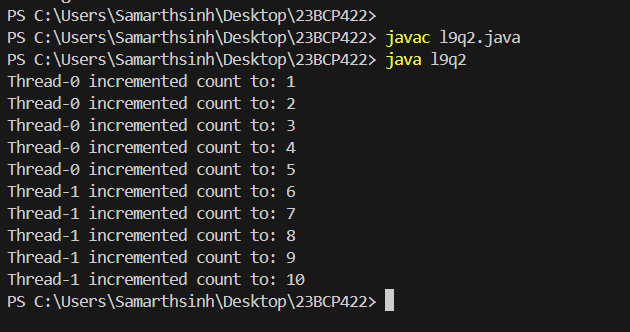
thread1.start();

thread2.start();

}

}

Output: -



Q-3 Write a Java program to demonstrate inter-thread communication using wait(), notify(), and notifyAll() methods, allowing threads to communicate and coordinate their actions

Code: -

class SharedResource {

private int data;

private boolean isDataAvailable = false;

// Method to produce data

public synchronized void produce(int value) {

// Wait if data is already available

while (isDataAvailable) {

try {

wait(); // Producer waits if data is already produced

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

data = value; // Produce new data

System.out.println("Produced: " + data);

isDataAvailable = true;

notify(); // Notify the consumer that data is available

}

// Method to consume data

public synchronized void consume() {

// Wait if no data is available

while (!isDataAvailable) {

try {

wait(); // Consumer waits if no data is produced yet

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

System.out.println("Consumed: " + data); // Consume data

isDataAvailable = false;

notify(); // Notify the producer that data has been consumed

}

}

class Producer extends Thread {

private final SharedResource resource;

public Producer(SharedResource resource) {

this.resource = resource;

}

public void run() {

for (int i = 1; i <= 5; i++) {

resource.produce(i); // Produce data from 1 to 5

try {

Thread.sleep(500); // Simulate time delay

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

}

class Consumer extends Thread {

private final SharedResource resource;

public Consumer(SharedResource resource) {

this.resource = resource;

}

public void run() {

for (int i = 1; i <= 5; i++) {

resource.consume(); // Consume data

try {

Thread.sleep(1000); // Simulate time delay for consumption

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

}

public class l9q3{

public static void main(String[] args) {

SharedResource resource = new SharedResource();

Producer producer = new Producer(resource);

Consumer consumer = new Consumer(resource);

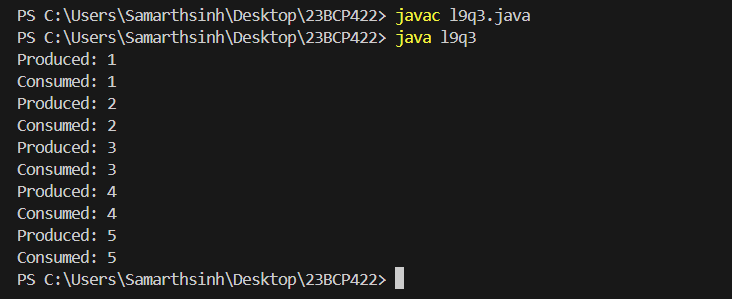
producer.start(); // Start producer thread

consumer.start(); // Start consumer thread

}

}

Output: -



Q-4 Write a Java program to show how thread priority affects the execution order of threads, highlighting the use of setPriority() and getPriority() methods

Code: -

class MyThread extends Thread {

public MyThread(String name) {

super(name);

}

public void run() {

System.out.println(getName() + " with priority " + getPriority() + " is running.");

}

}

public class l9q4 {

public static void main(String[] args) {

MyThread thread1 = new MyThread("Thread 1");

MyThread thread2 = new MyThread("Thread 2");

MyThread thread3 = new MyThread("Thread 3");

thread1.setPriority(10);

thread2.setPriority(5);

thread3.setPriority(1);

thread1.start();

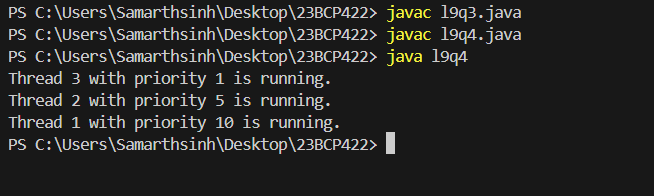
thread2.start();

thread3.start();

}

}

Output: -



Q-5 Write a Java program to implement the producer-consumer problem, ensuring the handling of potential deadlock conditions using proper synchronization techniques.

Code: -

class SharedResource {

private int data;

private boolean isDataAvailable = false;

// Produce method (called by producer)

public synchronized void produce(int value) throws InterruptedException {

while (isDataAvailable) {

wait();

}

data = value;

System.out.println("Produced: " + data);

isDataAvailable = true;

notify();

}

// Consume method (called by consumer)

public synchronized void consume() throws InterruptedException {

while (!isDataAvailable) {

wait();

}

System.out.println("Consumed: " + data);

isDataAvailable = false;

notify();

}

}

class Producer extends Thread {

private final SharedResource resource;

public Producer(SharedResource resource) {

this.resource = resource;

}

@Override

public void run() {

try {

for (int i = 1; i <= 5; i++) {

resource.produce(i);

Thread.sleep(1000);

}

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

class Consumer extends Thread {

private final SharedResource resource;

public Consumer(SharedResource resource) {

this.resource = resource;

}

@Override

public void run() {

try {

for (int i = 1; i <= 5; i++) {

resource.consume();

Thread.sleep(1500);

}

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

public class l9q5 {

public static void main(String[] args) {

SharedResource resource = new SharedResource();

Producer producer = new Producer(resource);

Consumer consumer = new Consumer(resource);

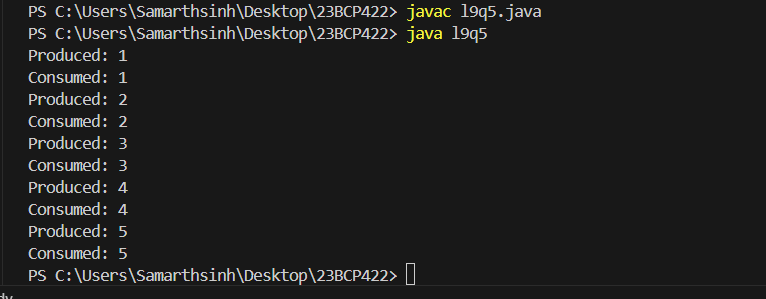
producer.start();

consumer.start();

}

}

Output: -



EXPERIMENT – 10

Q-1 Write a Java program to demonstrate various window handling events such as windowOpened(), windowClosing(), windowClosed(), windowIconified(), windowDeiconified(), windowActivated(), and

windowDeactivated().

Code: -

import java.awt.\*;

import java.awt.event.\*;

public class l10q1 extends Frame {

public l10q1() {

// Set the title and size of the window

setTitle("Window Event Demo");

setSize(400, 400);

// Add a window listener to handle window events

addWindowListener(new WindowAdapter() {

// Event when the window is opened

public void windowOpened(WindowEvent we) {

// Display message in the frame window

Label openedLabel = new Label("Window opened.");

add(openedLabel);

setLayout(new FlowLayout());

revalidate();

}

// Event when the window is closing

public void windowClosing(WindowEvent we) {

System.out.println("Window closing in terminal."); // Print to terminal

System.exit(0); // Close the application

}

// Event when the window is closed

public void windowClosed(WindowEvent we) {

System.out.println("Window closed.");

}

// Event when the window is minimized (iconified)

public void windowIconified(WindowEvent we) {

System.out.println("Window iconified.");

}

// Event when the window is restored from minimized state

public void windowDeiconified(WindowEvent we) {

System.out.println("Window deiconified.");

}

// Event when the window is activated

public void windowActivated(WindowEvent we) {

// Display message in the frame window

Label activatedLabel = new Label("Window activated.");

add(activatedLabel);

setLayout(new FlowLayout());

revalidate();

}

// Event when the window is deactivated

public void windowDeactivated(WindowEvent we) {

System.out.println("Window deactivated.");

}

});

}

public static void main(String[] args) {

// Create the window

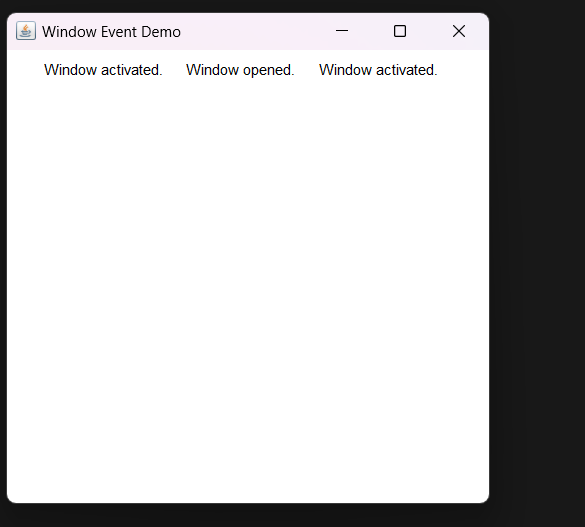
l10q1 window = new l10q1();

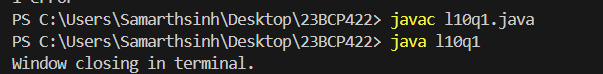
window.setVisible(true); // Make the window visible

}

}

Output: -





Q-2 Write a Java program to demonstrate various mouse handling events including mouseClicked(), mouseEntered(), mouseExited(), mousePressed(), mouseReleased(), and mouseDragged()

Code: -

import java.awt.\*;

import java.awt.event.\*;

public class l10q2 extends Frame implements MouseListener, MouseMotionListener {

// Constructor to set up the frame

public l10q2() {

setTitle("Mouse Event Demo");

setSize(400, 400);

// Add mouse listeners to the frame

addMouseListener(this);

addMouseMotionListener(this);

// Set the layout and make the window visible

setLayout(null);

setVisible(true);

}

// MouseListener methods

public void mouseClicked(MouseEvent e) {

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

}

public void mouseEntered(MouseEvent e) {

System.out.println("Mouse entered the frame.");

}

public void mouseExited(MouseEvent e) {

System.out.println("Mouse exited the frame.");

}

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() + ")");

}

// MouseMotionListener methods

public void mouseDragged(MouseEvent e) {

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

}

public void mouseMoved(MouseEvent e) {

// This event is triggered when the mouse is moved without clicking

// We can print the mouse position if needed

}

public static void main(String[] args) {

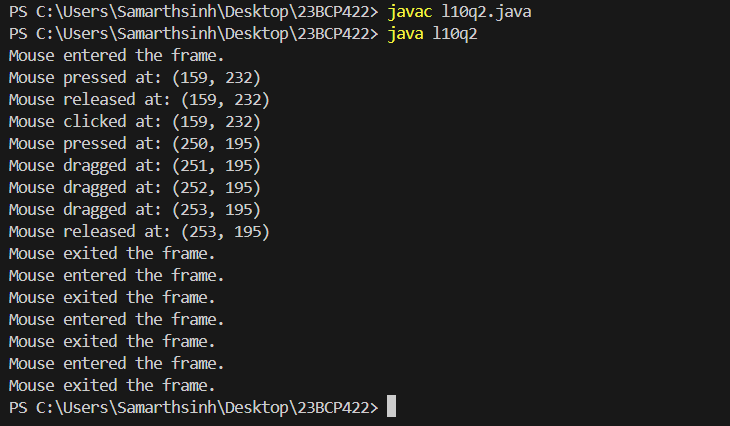
// Create the frame to show mouse events

new l10q2();

}

}

Output: -



Q-3 Write a Java program to demonstrate different keyboard handling events such as keyPressed(),

keyReleased(), and keyTyped().

Code: -

import java.awt.\*;

import java.awt.event.\*;

public class l10q3 extends Frame implements KeyListener {

// Label to display key events

Label keyLabel;

// Constructor to set up the frame

public l10q3() {

setTitle("Keyboard Event Demo");

setSize(400, 400);

// Set layout to null for absolute positioning

setLayout(null);

// Create and add a label to display key events

keyLabel = new Label("Press any key...");

keyLabel.setBounds(100, 150, 200, 30);

add(keyLabel);

// Add key listener to the frame

addKeyListener(this);

// Set the frame visible

setVisible(true);

}

// KeyListener methods

public void keyPressed(KeyEvent e) {

keyLabel.setText("Key pressed: " + e.getKeyChar());

}

public void keyReleased(KeyEvent e) {

keyLabel.setText("Key released: " + e.getKeyChar());

}

public void keyTyped(KeyEvent e) {

keyLabel.setText("Key typed: " + e.getKeyChar());

}

public static void main(String[] args) {

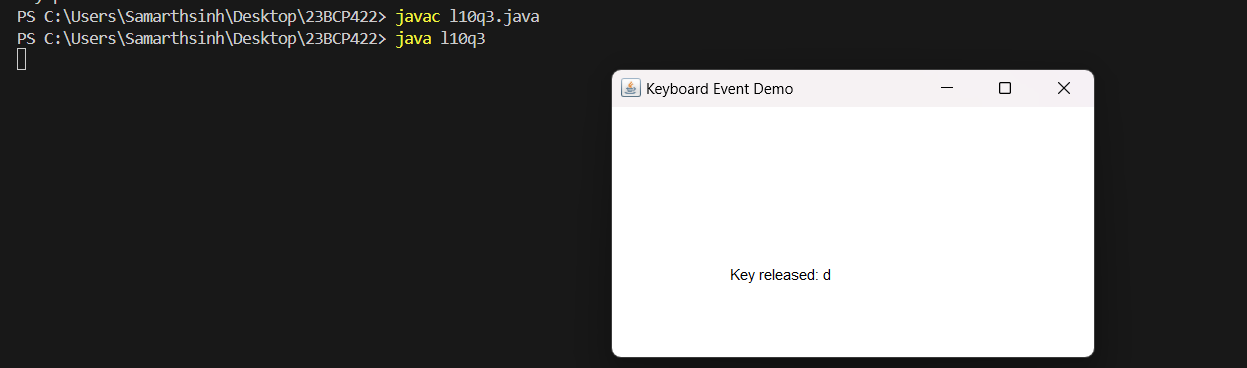
// Create the frame to show keyboard events

new l10q3();

}

}

Output: -



Q-4 Write a Java program to create a simple GUI that includes a button and a label. When the button is clicked, the text of the label should change accordingly

Code: -

import java.awt.\*;

import java.awt.event.\*;

public class l10q4 extends Frame {

Label label;

Button button;

// Constructor to set up the frame and components

public l10q4() {

// Set the title of the window

setTitle("Simple GUI Example");

// Set the layout manager

setLayout(new FlowLayout());

// Create a label and set initial text

label = new Label("Click the button to change text");

add(label);

// Create a button

button = new Button("Click Me");

add(button);

// Add a button click event listener

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

label.setText("Text changed after button click!");

}

});

setSize(300, 150);

setVisible(true);

}

public static void main(String[] args) {

new l10q4();

}

}

Output: -

