Name: Pallavi kallur

Roll.no: A54

**Experiment No. 6**

**Aim: Implement basic java programs based on abstract keyword, final keyword and interfaces.**

1. Write a program to demonstrate use of ‘abstract’ keyword.

**CODE:**

import java.io.\*;

abstract class Data

{

abstract void display();

void show()

{

System.out.println("Hi");

}

}

class Demo extends Data

{

void display()

{

System.out.println("Welcome");

}

public static void main(String args[])

{

Demo d = new Demo();

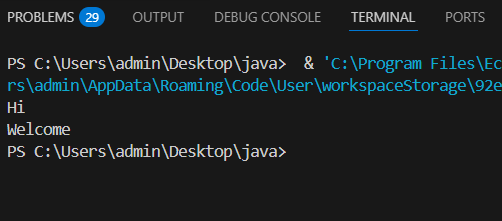
d.show();

d.display();

}

}

**OUTPUT:**

****

2. Write a program to demonstrate use of ‘final’ keyword.

**CODE:**

import java.io.\*;

final class Data

{

final int a=10;

final void display()

{

System.out.println("Welcome");

}

}

class Demo extends Data

{

void display()

{ a=20;

System.out.println("Hi");

}

public static void main(String args[])

{

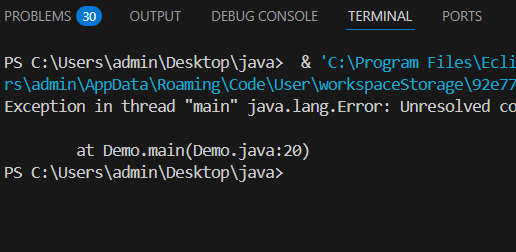
Data d = new Data();

d.display();

}

}

**OUTPUT:**



3. Write a program to demonstrate use of Interfaces.

**CODE:**

import java.io.\*;

interface A

{

int a=10;

void display();

}

class Data implements A

{

public void display()

{

System.out.println(a);

System.out.println("Welcome");

}

}

class Run {

public static void main(String args[])

{

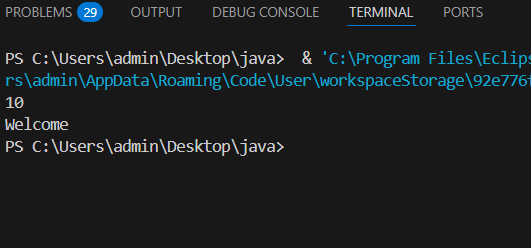
Data d = new Data();

d.display();

}

}

**OUTPUT:**



1. Payment Processing System

Imagine a payment processing system that supports multiple payment methods (e.g., credit card, PayPal, and cryptocurrency). You could define an interface for payment processing

interface PaymentProcess

{

void makePayment(double amount);

}

class CreditCard implements PaymentProcess

{

@Override

public void makePayment(double amount)

{

System.out.println("Paid using credit card "+amount);

}

}

class Paypal implements PaymentProcess

{

@Override

public void makePayment(double amount)

{

System.out.println("Paid using paypal "+amount);

}

}

class Cryptocurrency implements PaymentProcess

{

@Override

public void makePayment(double amount)

{

System.out.println("Paid using Cryptocurrency "+amount);

}

}

class Payment

{

public static void main(String[] args)

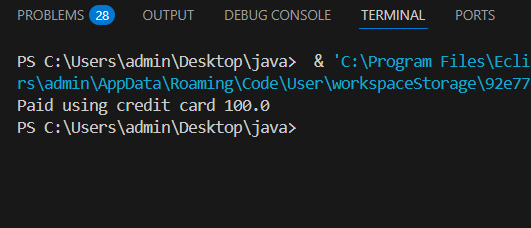
{

PaymentProcess p = new CreditCard();

p.makePayment(100);

}

}



2. Shape Area Calculation

In a graphics application, you may want to calculate the area of different shapes. You can define a Shape interface

interface Shape

{

double calculateArea();

}

class Rectangle implements Shape

{

private double length, width;

public Rectangle(double length, double width)

{

this.length = length;

this.width = width;

}

@Override

public double calculateArea()

{

return length \* width;

}

}

class Circle implements Shape

{

private double radius;

public Circle(double radius)

{

this.radius = radius;

}

@Override

public double calculateArea()

{

return Math.PI \* radius \* radius;

}

}

class Triangle implements Shape

{

private double base, height;

public Triangle(double base, double height) {

this.base = base;

this.height = height;

}

@Override

public double calculateArea() {

return 0.5 \* base \* height;

}

}

public class Demo

{

public static void main(String[] args) {

Shape rectangle = new Rectangle(10, 5);

Shape circle = new Circle(7);

Shape triangle = new Triangle(8, 4);

System.out.println("Rectangle Area: " + rectangle.calculateArea());

System.out.println("Circle Area: " + circle.calculateArea());

System.out.println("Triangle Area: " + triangle.calculateArea());

}

}

