**Name: G. Vinay sai**

**MCA -DIV -B**

**P.en.No:2305112070003**

**Temp.en.No:23PG110387**

List of Practical

**1. Basic datatype and looping**

**a**. Write a program for swapping and find a factorial value. Perform swapping without using a third variable.

-import java.util.Scanner;

public class SwapAndFactorial

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

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

int num1 = scanner.nextInt();

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

int num2 = scanner.nextInt();

num1 = num1 + num2;

num2 = num1 - num2;

num1 = num1 - num2;

System.out.println("After swapping: num1 = " + num1 + ", num2 = " + num2);

System.out.print("Enter a number to find its factorial: ");

int number = scanner.nextInt();

long factorial = 1;

for (int i = 1; i <= number; i++)

{

factorial \*= i;

}

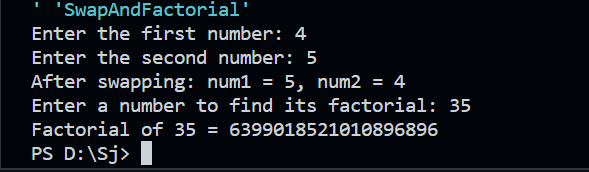
System.out.println("Factorial of " + number + " = " + factorial);

scanner.close();

}

}

Output:



**b.** Write a Java program to check whether a number is even or odd.

-public class EvenOdd

{

public static void main(String[] args)

{

int number = 15;

if (number % 2 == 0)

{

System.out.println(number + " is an even number.");

}

else

{

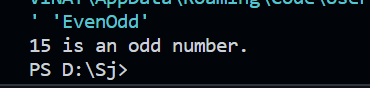
System.out.println(number + " is an odd number.");

}

}

}

Output:



**c.** Write a Java program to check whether a number is prime or not

-import java.util.Scanner;

public class primenumber

{

public static void main(String args[])

{

Scanner x = new Scanner(System.in);

System.out.println("Enter the number");

int num = x .nextInt();

int count = 0;

for (int i=2 ; i<num ; i++)

{

if (num%i==0)

{

count=count+1;

}

}

if(count==0)

{

System.out.println("Prime number");

}

else

{

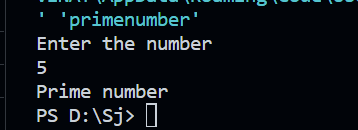
System.out.println("Not a Prime number");

}

}

}

Output:



**d.** Write a Java program to print a Fibonacci series with user input.

-import java.util.Scanner;

public class FibonacciSeries

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

int a = 0, b = 1;

System.out.print("Fibonacci Series: ");

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

{

System.out.print(a + " ");

int nextTerm = a + b;

a = b;

b = nextTerm;

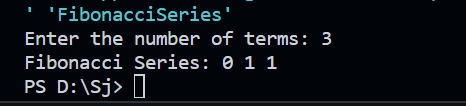
}

scanner.close();

}

}

Output:



**e**. Write a Java Program to Check the Armstrong Number

-import java.util.Scanner;

public class ArmstrongNumberCheck

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number to check for Armstrong: ");

int number = scanner.nextInt();

boolean isArmstrong = isArmstrong(number);

System.out.println(number + (isArmstrong ? " is an Armstrong number." : " is not an Armstrong number."));

scanner.close();

}

static boolean isArmstrong(int num)

{ {

int originalNumber = num, sum = 0;

while (num > 0)

{

int digit = num % 10;

sum += Math.pow(digit, 3);

num /= 10;

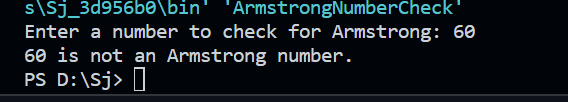
}

return sum == originalNumber;

}

}

Output:



**f.** Swapping Two Numbers in Java Using Third Variable

-public class SwapTwoNumbers

{

public static void main(String[] args)

{

int num1 = 5;

int num2 = 10;

System.out.println("Before swapping: num1 = " + num1 + ", num2 = " + num2);

int temp = num1;

num1 = num2;

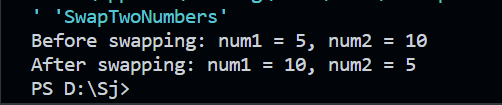
num2 = temp;

System.out.println("After swapping: num1 = " + num1 + ", num2 = " + num2);

}

}

Oputput:



2. **Looping and Control structure**

a. Write a program to accept a number from the user through the command line and display whether the given number is

palindrome or not

-public class PalindromeCheck

{

public static void main(String[] args)

{

if (args.length == 0)

{

System.out.println("Please provide a number as a command line argument.");

return;

}

int number = parseNumber(args[0]);

if (isPalindrome(number))

{

System.out.println(number + " is a palindrome.");

}

else

{

System.out.println(number + " is not a palindrome.");

}

}

static int parseNumber(String input)

{

try

{

return Integer.parseInt(input);

}

catch (NumberFormatException e)

{

System.out.println("Invalid input. Please provide a valid number.");

System.exit(1);

return 0;

}

}

static boolean isPalindrome(int num)

{

int original = num, reversed = 0;

while (num > 0)

{

reversed = reversed \* 10 + num % 10;

num /= 10;

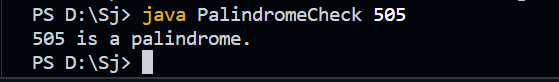
}

return original == reversed;

}

}

Output:



b. Java Pattern Programs

Number triangle Pattern

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

6 6 6 6 6 6

-public class NumberTrianglePattern

{

public static void main(String[] args)

{

int rows = 6;

for (int i = 1; i <= rows; i++)

{

for (int j = 1; j <= rows - i; j++)

{

System.out.print(" ");

}

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

System.out.print(i + " ");

}

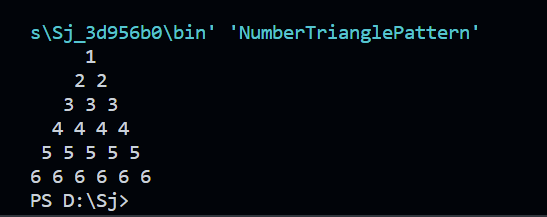
System.out.println();

}

}

}

Output:



Right Half Pyramid Pattern

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

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

-public class RightHalfPyramidPattern

{

public static void main(String[] args)

{

int rows = 6;

for (int i = 1; i <= rows; i++)

{

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

System.out.print("\*");

}

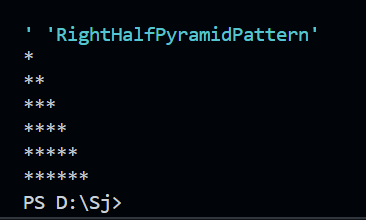
System.out.println();

}

}

}

Output:



Left Half Pyramid Pattern

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

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

-public class LeftHalfPyramidPattern

{

public static void main(String[] args)

{

int rows = 6;

for (int i = 1; i <= rows; i++)

{

for (int j = 1; j <= rows - i; j++)

{

System.out.print(" ");

}

for (int k = 1; k <= i; k++)

{

System.out.print("\*");

}

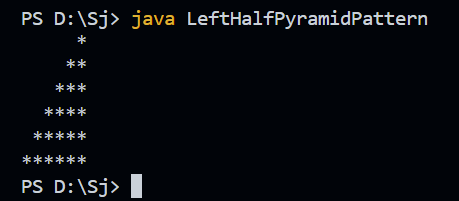
System.out.println();

}

}

}

Output:



Reverse Right Half Pyramid Pattern

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

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

-public class ReverseRightHalfPyramidPattern

{

public static void main(String[] args)

{

int rows = 6;

for (int i = 1; i <= rows; i++)

{

for (int j = rows; j >= i; j--) {

System.out.print("\*");

}

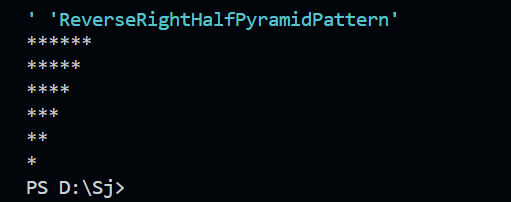
System.out.println();

}

}

}

Output:



Triangle Star Pattern

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

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

-public class TriangleStarPattern

{

public static void main(String[] args)

{

int rows = 6;

for (int i = 1; i <= rows; i++)

{

for (int j = 1; j <= rows - i; j++) {

System.out.print(" ");

}

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

System.out.print("\* ");

}

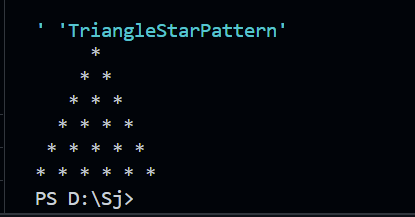
System.out.println();

}

}

}

Output:



**3. Array**

a. Write a program to accept an array of integers from the user through the command line and find prime numbers from the array

-public class PrimeNumbersInArray

{

public static void main(String[] args)

{

if (args.length == 0)

{

System.out.println("Please provide integers as command line arguments.");

return;

}

System.out.println("Prime numbers in the array:");

for (String arg : args)

{

try

{

int number = Integer.parseInt(arg);

if (isPrime(number))

{

System.out.print(number + " ");

}

}

catch (NumberFormatException e)

{

System.out.println("Invalid input. Please provide valid integers.");

return;

}

}

}

static boolean isPrime(int num)

{

if (num <= 1) return false;

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

{

if (num % i == 0) return false;

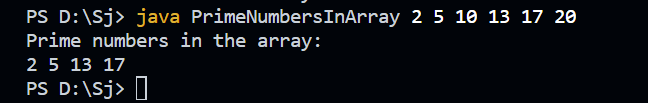
}

return true;

}

}

Output:



**4. Class**

a. Create a class Stack that defines an integer stack that can hold 10 values. Perform push and pop actions in a stack.

-public class Stack

{

private int maxSize = 10;

private int[] stackArray = new int[maxSize];

private int top = -1;

public void push(int value)

{

if (top < maxSize - 1) stackArray[++top] = value;

else System.out.println("Stack Overflow. Cannot push " + value + ".");

}

public int pop()

{

if (top >= 0) return stackArray[top--];

else

{

System.out.println("Stack Underflow. Cannot pop from an empty stack.");

return -1;

}

}

public void display()

{

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

for (int i = 0; i <= top; i++) System.out.print(stackArray[i] + " ");

System.out.println(top >= 0 ? "" : " (empty)");

}

public static void main(String[] args)

{

Stack stack = new Stack();

stack.push(5);

stack.push(10);

stack.push(15);

stack.display();

stack.pop();

stack.display();

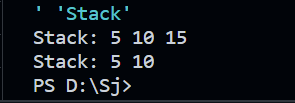
stack.pop();

stack.pop();

}

}

Output:



b. Write a program to print the area of two rectangles having sides (4,5) and (5,8) respectively by creating a class named 'Rectangle' with a method named 'Area' which returns the area and length and breadth passed as parameters to its constructor.

-class Rectangle

{

private int length;

private int breadth;

public Rectangle(int length, int breadth)

{

this.length = length;

this.breadth = breadth;

}

public int calculateArea()

{

return length \* breadth;

}

}

public class RectangleArea

{

public static void main(String[] args)

{

Rectangle rectangle1 = new Rectangle(4, 5);

Rectangle rectangle2 = new Rectangle(5, 8);

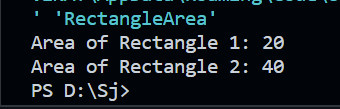
System.out.println("Area of Rectangle 1: " + rectangle1.calculateArea());

System.out.println("Area of Rectangle 2: " + rectangle2.calculateArea());

}

}

Output:



c. Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.

-import java.util.Scanner;

class Area

{

private int length;

private int breadth;

public Area(int length, int breadth)

{

this.length = length;

this.breadth = breadth;

}

public int returnArea()

{

return length \* breadth;

}

}

public class RectangleArea

{

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the rectangle: ");

int length = scanner.nextInt();

System.out.print("Enter the breadth of the rectangle: ");

int breadth = scanner.nextInt();

Area rectangle = new Area(length, breadth);

int area = rectangle.returnArea();

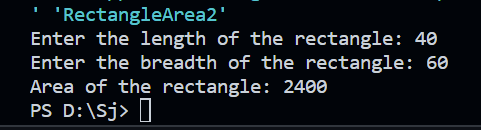
System.out.println("Area of the rectangle: " + area);

scanner.close();

}

}

Output:



**5. Inheritance**

Write a program to create a class Publisher with attributes publisher name and publisher id. Derive a subclass Book with attributes bookname, bookid and author name. All these data should be entered by the user. Create two methods getdata() and showdata() to display the details of book and publisher.

-import java.util.Scanner;

class Publisher

{

private String publisherName;

private int publisherId;

public void getData()

{

Scanner scanner = new Scanner(System.in);

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

this.publisherName = scanner.nextLine();

System.out.print("Enter Publisher ID: ");

this.publisherId = scanner.nextInt();

scanner.nextLine();

}

public void showData()

{

System.out.println("Publisher Name: " + publisherName);

System.out.println("Publisher ID: " + publisherId);

}

}

class Book extends Publisher

{

private String bookName;

private int bookId;

private String authorName;

public void getData()

{

super.getData();

Scanner scanner = new Scanner(System.in);

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

this.bookName = scanner.nextLine();

System.out.print("Enter Book ID: ");

this.bookId = scanner.nextInt();

scanner.nextLine();

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

this.authorName = scanner.nextLine();

}

public void showData()

{

super.showData();

System.out.println("Book Name: " + bookName);

System.out.println("Book ID: " + bookId);

System.out.println("Author Name: " + authorName);

}

}

public class BookPublisherDemo

{

public static void main(String[] args)

{

Book book = new Book();

book.getData();

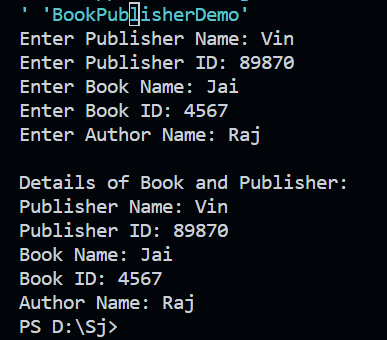
System.out.println("\nDetails of Book and Publisher:");

book.showData();

}

}

Output:



**6. Method Overloading**

a. Write a program to create a class with two methods with same name addfunc(), one accepting two integer parameters and other accepting two double parameters. When method is called, the appropriate method should be selected depending on parameters passed(method overloading.

-import java.util.Scanner;

class Publisher

{

private String publisherName;

private int publisherId;

public void getData()

{

Scanner scanner = new Scanner(System.in);

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

this.publisherName = scanner.nextLine();

System.out.print("Enter Publisher ID: ");

this.publisherId = scanner.nextInt();

scanner.nextLine();

}

public void showData()

{

System.out.println("Publisher Name: " + publisherName);

System.out.println("Publisher ID: " + publisherId);

}

}

class Book extends Publisher

{

private String bookName;

private int bookId;

private String authorName;

public void getData()

{

super.getData();

Scanner scanner = new Scanner(System.in);

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

this.bookName = scanner.nextLine();

System.out.print("Enter Book ID: ");

this.bookId = scanner.nextInt();

scanner.nextLine();

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

this.authorName = scanner.nextLine();

}

public void showData()

{

super.showData();

System.out.println("Book Name: " + bookName);

System.out.println("Book ID: " + bookId);

System.out.println("Author Name: " + authorName);

}

}

public class BookPublisherDemo

{

public static void main(String[] args)

{

Book book = new Book();

book.getData();

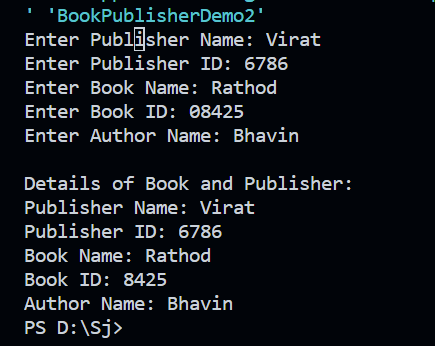
System.out.println("\nDetails of Book and Publisher:");

book.showData();

}

}

**Output:**



**7. Super and this keyword**

Declare a variable called x with integer as the data type in base class and subclass. Make a method named as show() which displays the value of x in the superclass and subclass.

**-class BaseClass**

**{**

**protected int x;**

**public BaseClass(int value)**

**{**

**this.x = value;**

**}**

**public void show()**

**{**

**System.out.println("Base Class - x: " + x);**

**}**

**}**

**class SubClass extends BaseClass**

**{**

**public SubClass(int value)**

**{**

**super(value);**

**}**

**public void show()**

**{**

**super.show();**

**System.out.println("Sub Class - x: " + x);**

**}**

**}**

**public class Main**

**{**

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

**{**

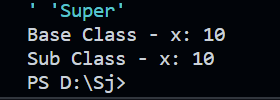
**SubClass obj = new SubClass(10);**

**obj.show();**

**}**

**}**

**Output:**



**8. Method Overriding**

**a.**Write a Java program to create a class called Shape with a method called getArea(). Create a subclass called Rectangle that overrides the getArea() method to calculate the area of a rectangle.

-class Shape

{

public double getArea()

{

return 0.0;

}

}

class Rectangle extends Shape

{

private double length;

private double width;

public Rectangle(double length, double width)

{

this.length = length;

this.width = width;

}

public double getArea()

{

return length \* width;

}

}

public class Main

{

public static void main(String[] args)

{

Rectangle rectangle = new Rectangle(5.0, 3.0);

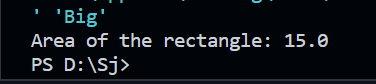
double area = rectangle.getArea();

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

}

}

Output:



**b.** Write a Java program to create a class called Employee with methods called work() and getSalary(). Create a subclass called HRManager that overrides the work() method and adds a new method called addEmployee().

-class Employee

{

private String name;

private double salary;

public Employee(String name, double salary)

{

this.name = name;

this.salary = salary;

}

public void work()

{

System.out.println(name + " is performing regular duties.");

}

public double getSalary()

{

return salary;

}

}

class HRManager extends Employee

{

public HRManager(String name, double salary)

{

super(name, salary);

}

public void work()

{

System.out.println(getName() + " is managing HR tasks.");

}

public void addEmployee()

{

System.out.println("HRManager is adding a new employee.");

}

}

public class Main

{

public static void main(String[] args)

{

Employee emp = new Employee("John Doe", 50000);

emp.work();

System.out.println("Salary: $" + emp.getSalary());

HRManager hrManager = new HRManager("Jane Smith", 70000);

hrManager.work();

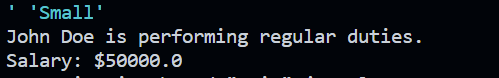
System.out.println("Salary: $" + hrManager.getSalary());

hrManager.addEmployee();

}

}

Output:



**c.** Write a Java program to create a class known as Person with methods called getFirstName() and getLastName(). Create a subclass called Employee that adds a new method named getEmployeeId() and overrides the getLastName() method to include the employee's job title.

-class Person

{

private String firstName;

private String lastName;

public Person(String firstName, String lastName)

{

this.firstName = firstName;

this.lastName = lastName;

}

public String getFirstName()

{

return firstName;

}

public String getLastName()

{

return lastName;

}

}

class Employee extends Person

{

private int employeeId;

private String jobTitle;

public Employee(String firstName, String lastName, int employeeId, String jobTitle)

{

super(firstName, lastName);

this.employeeId = employeeId;

this.jobTitle = jobTitle;

}

public int getEmployeeId()

{

return employeeId;

}

public String getLastName()

{

return super.getLastName() + " (" + jobTitle + ")";

}

}

public class Main

{

public static void main(String[] args)

{

Person person = new Person("John", "Doe");

System.out.println("Person: " + person.getFirstName() + " " + person.getLastName());

Employee employee = new Employee("Jane", "Smith", 12345, "Software Engineer");

System.out.println("Employee: " + employee.getFirstName() + " " + employee.getLastName() +

", Employee ID: " + employee.getEmployeeId());

}

}

Output:



**d.** Write a Java program to create a class called Shape with methods called getPerimeter() and getArea(). Create a subclass called Circle that overrides the getPerimeter() and getArea() methods to calculate the area and perimeter of a circle.

-class Shape

{

public double getPerimeter()

{

return 0.0;

}

public double getArea()

{

return 0.0;

}

}

class Circle extends Shape

{

private double radius;

public Circle(double radius)

{

this.radius = radius;

}

public double getPerimeter()

{

return 2 \* Math.PI \* radius;

}

public double getArea()

{

return Math.PI \* radius \* radius;

}

}

public class Main

{

public static void main(String[] args)

{

Circle circle = new Circle(5.0);

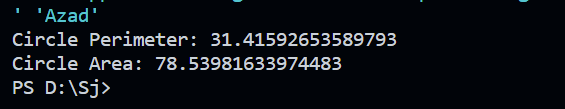
System.out.println("Circle Perimeter: " + circle.getPerimeter());

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

}

}

Output:



**9. Final class, abstract class, and interface**

a. Write a program to calculate the area, circumference and volume for all shapes. [Perform this application using final class, abstract class and interface]

-interface Shape

{

double getArea();

double getCircumference();

}

abstract class ThreeDimensionalShape implements Shape

{

abstract double getVolume();

}

final class Circle implements Shape

{

private double radius;

public Circle(double radius)

{

this.radius = radius;

}

public double getArea()

{

return Math.PI \* radius \* radius;

}

public double getCircumference()

{

return 2 \* Math.PI \* radius;

}

}

final class Sphere extends ThreeDimensionalShape

{

private double radius;

public Sphere(double radius)

{

this.radius = radius;

}

public double getArea()

{

return 4 \* Math.PI \* radius \* radius;

}

public double getCircumference()

{

return 0;

}

double getVolume() {

return (4.0 / 3.0) \* Math.PI \* Math.pow(radius, 3);

}

}

public class Main

{

public static void main(String[] args)

{

Circle circle = new Circle(5.0);

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

System.out.println("Circle Circumference: " + circle.getCircumference());

Sphere sphere = new Sphere(3.0);

System.out.println("Sphere Area: " + sphere.getArea());

System.out.println("Sphere Circumference: " + sphere.getCircumference());

System.out.println("Sphere Volume: " + sphere.getVolume());

}

}

Output:



b. w.a.p in java to create a class circle to calculate and display the area of the circle to implement Final Variable in Java

-public class Circle

{

private final double radius;

private final double PI = 3.14159;

public Circle(double radius)

{

this.radius = radius;

}

public double getArea()

{

return PI \* radius \* radius;

}

public static void main(String[] args)

{

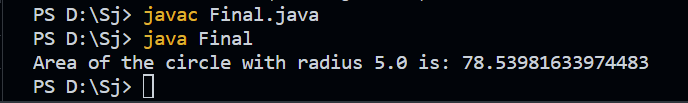
Circle circle = new Circle(5.0);

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

}

}

Output:



c. w.a.p in Java to create a class Animal **to Implement Final Methods** the general characteristics of an Animal along with the constant variables to store the number of legs, ears eyes, and whether the animal has a tail. Create a subclass wolf to display the additional characteristics.

-class Animal

{

final int numberOfLegs;

final int numberOfEars;

final int numberOfEyes;

final boolean hasTail;

public Animal(int numberOfLegs, int numberOfEars, int numberOfEyes, boolean hasTail)

{

this.numberOfLegs = numberOfLegs;

this.numberOfEars = numberOfEars;

this.numberOfEyes = numberOfEyes;

this.hasTail = hasTail;

}

final void displayGeneralCharacteristics()

{

System.out.println("Number of Legs: " + numberOfLegs);

System.out.println("Number of Ears: " + numberOfEars);

System.out.println("Number of Eyes: " + numberOfEyes);

System.out.println("Has Tail: " + (hasTail ? "Yes" : "No"));

}

}

Animal

class Wolf extends Animal

{

final String furColor;

public Wolf(int numberOfLegs, int numberOfEars, int numberOfEyes, boolean hasTail, String furColor)

{

super(numberOfLegs, numberOfEars, numberOfEyes, hasTail);

this.furColor = furColor;

}

final void displayGeneralCharacteristics()

{

super.displayGeneralCharacteristics();

System.out.println("Fur Color: " + furColor);

}

}

public class Main

{

public static void main(String[] args)

{

Animal genericAnimal = new Animal(4, 2, 2, true);

System.out.println("Generic Animal Characteristics:");

genericAnimal.displayGeneralCharacteristics();

System.out.println();

Wolf wolf = new Wolf(4, 2, 2, true, "Gray");

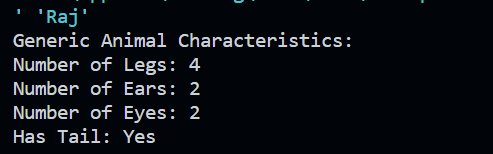
System.out.println("Wolf Characteristics:");

wolf.displayGeneralCharacteristics();

}

}

Output:



d. Write a Java program to create an abstract class BankAccount with abstract methods deposit() and withdraw(). Create subclasses: SavingsAccount and CurrentAccount that extend the BankAccount class and implement the respective methods to handle deposits and withdrawals for each account type.

-abstract class BankAccount

{

protected double balance;

public BankAccount(double initialBalance)

{

this.balance = initialBalance;

}

public abstract void deposit(double amount);

public abstract void withdraw(double amount);

public void displayBalance()

{

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

}

}

class SavingsAccount extends BankAccount

{

private double interestRate;

public SavingsAccount(double initialBalance, double interestRate)

{

super(initialBalance);

this.interestRate = interestRate;

}

public void deposit(double amount)

{

balance += amount + (amount \* interestRate);

System.out.println("Deposited $" + amount + " (including interest)");

}

public void withdraw(double amount)

{

if (amount <= balance)

{

balance -= amount;

System.out.println("Withdrawn $" + amount);

} else

{

System.out.println("Insufficient funds!");

}

}

}

class CurrentAccount extends BankAccount

{

private double overdraftLimit;

public CurrentAccount(double initialBalance, double overdraftLimit)

{

super(initialBalance);

this.overdraftLimit = overdraftLimit;

}

public void deposit(double amount)

{

balance += amount;

System.out.println("Deposited $" + amount);

}

public void withdraw(double amount)

{

if (amount <= balance + overdraftLimit)

{

balance -= amount;

System.out.println("Withdrawn $" + amount);

} else

{

System.out.println("Exceeds overdraft limit! Cannot withdraw.");

}

}

}

public class Main

{

public static void main(String[] args)

{

SavingsAccount savingsAccount = new SavingsAccount(1000.0, 0.05);

savingsAccount.displayBalance();

savingsAccount.deposit(500.0);

savingsAccount.displayBalance();

savingsAccount.withdraw(200.0);

savingsAccount.displayBalance();

System.out.println();

CurrentAccount currentAccount = new CurrentAccount(2000.0, 500.0);

currentAccount.displayBalance();

currentAccount.deposit(700.0);

currentAccount.displayBalance();

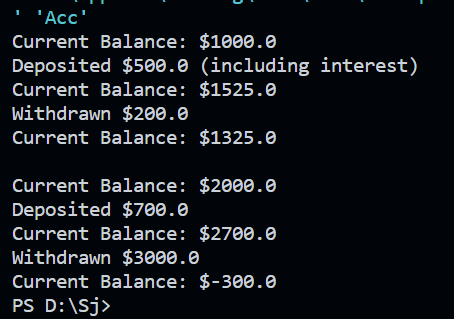
currentAccount.withdraw(3000.0);

currentAccount.displayBalance();

}

}

Output:



e. Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape.

-abstract class Shape3D

{

public abstract double calculateVolume();

public abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D

{

private double radius;

public Sphere(double radius)

{

this.radius = radius;

}

public double calculateVolume()

{

return (4.0 / 3.0) \* Math.PI \* Math.pow(radius, 3);

}

public double calculateSurfaceArea()

{

return 4 \* Math.PI \* Math.pow(radius, 2);

}

}

class Cube extends Shape3D

{

private double sideLength;

public Cube(double sideLength)

{

this.sideLength = sideLength;

}

public double calculateVolume()

{

return Math.pow(sideLength, 3);

}

public double calculateSurfaceArea()

{

return 6 \* Math.pow(sideLength, 2);

}

}

public class Main

{

public static void main(String[] args)

{

Sphere sphere = new Sphere(3.0);

System.out.println("Sphere Volume: " + sphere.calculateVolume());

System.out.println("Sphere Surface Area: " + sphere.calculateSurfaceArea());

System.out.println();

Cube cube = new Cube(2.0);

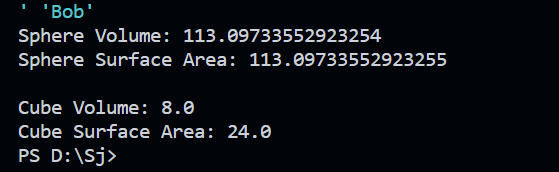
System.out.println("Cube Volume: " + cube.calculateVolume());

System.out.println("Cube Surface Area: " + cube.calculateSurfaceArea());

}

}

Output:



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--The End--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*