**Java Assignment-1**

**Q-1** Problem Statement: Write a Java program to analyze a string input by the user. The program should count the number of vowels, consonants, digits, and special characters in the string.

Input Example:

Enter a string: Hello World 2024!

Output Example:

Vowels: 3

Consonants: 7

Digits: 4

Special Characters: 3

Code:

import java.util.\*;

public class Find

{

    public static void main(String[] args) {

        String str="Hello World 123!!";

        int v=0,c=0,d=0,s=0;

        String newstr=str.toLowerCase();

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

        {

            char ch=newstr.charAt(i);

            if(ch>='a'&& ch<='z')

            {

            if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')

            {v++;

            }

            else{

                c++;

            }}

            else if(ch>='0' && ch<='9')

            {

                d++;

            }

            else{

                s++;

            }

        }

        System.out.println("Vowels:"+v);

        System.out.println("Consonants:"+c);

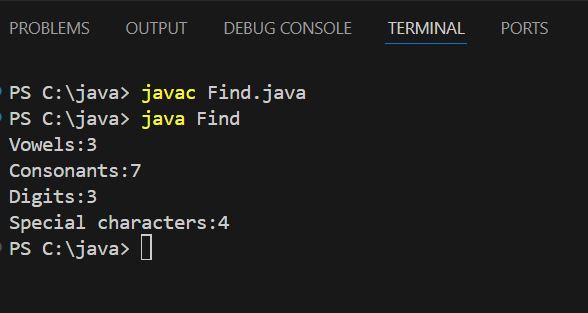
        System.out.println("Digits:"+d);

        System.out.println("Special characters:"+s);

    }

}

Output:

****

**Q-2** Problem Statement: Write a Java program to perform matrix operations (addition, subtraction, and multiplication) on two matrices provided by the user. The program should check the dimensions of the matrices to ensure valid operations.

Input Example: Output Example:

Matrix 1: Matrix 2: Addition: Subtraction: Multiplication:

1 2 5 6 6 8 -4 -4 19 22

3 4 7 8 10 12 -4 -4 43 50

Code:

import java.util.Scanner;

public class Matrix {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of rows and columns of the matrices: ");

        int rows = scanner.nextInt();

        int cols = scanner.nextInt();

        int[][] matrix1 = new int[rows][cols];

        int[][] matrix2 = new int[rows][cols];

        System.out.println("Enter elements of Matrix 1:");

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

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

                matrix1[i][j] = scanner.nextInt();

            }

        }

        System.out.println("Enter elements of Matrix 2:");

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

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

                matrix2[i][j] = scanner.nextInt();

            }

        }

        scanner.close();

        System.out.println("Addition:");

        int[][] sum = new int[rows][cols];

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

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

                sum[i][j] = matrix1[i][j] + matrix2[i][j];

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

            }

            System.out.println();

        }

        System.out.println("Subtraction:");

        int[][] difference = new int[rows][cols];

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

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

                difference[i][j] = matrix1[i][j] - matrix2[i][j];

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

            }

            System.out.println();

        }

        if (cols != rows) {

            System.out.println("Multiplication is not possible with these dimensions.");

            return;

        }

        System.out.println("Multiplication:");

        int[][] product = new int[rows][cols];

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

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

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

                    product[i][j] += matrix1[i][k] \* matrix2[k][j];

                }

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

            }

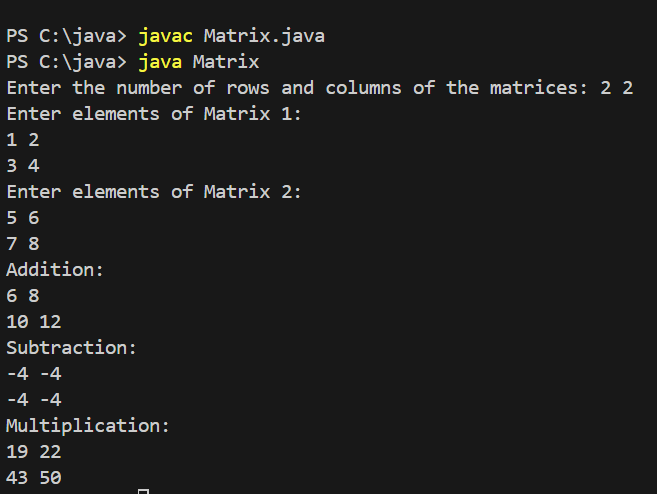
            System.out.println();

        }

    }

}

Output:



**Q-3** Problem Statement: Create a Java program to implement a basic banking system with the following features:

Account creation (Name, Account Number, Balance).

Deposit and withdrawal operations.

Prevent overdraft by checking the balance before withdrawal.

Input Example:

Create Account:

Name: John Doe

Account Number: 12345

Initial Balance: 1000

Deposit: 500

Withdraw: 2000

Output Example:

Deposit successful! Current Balance: 1500

Error: Insufficient funds. Current Balance: 1500

Code:

import java.util.Scanner;

class BankAccount {

    private String name;

    private String accountNumber;

    private double balance;

    public BankAccount(String name, String accountNumber, double balance) {

        this.name = name;

        this.accountNumber = accountNumber;

        this.balance = balance;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Deposit successful! Current Balance: " + balance);

        } else {

            System.out.println("Error: Invalid deposit amount.");

        }

    }

    public void withdraw(double amount) {

        if (amount > balance) {

            System.out.println("Error: Insufficient funds. Current Balance: " + balance);

        } else if (amount > 0) {

            balance -= amount;

            System.out.println("Withdrawal successful! Current Balance: " + balance);

        } else {

            System.out.println("Error: Invalid withdrawal amount.");

        }

    }

}

public class BankingSystem {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        String name = scanner.nextLine();

        System.out.print("Enter Account Number: ");

        String accountNumber = scanner.nextLine();

        System.out.print("Enter Initial Balance: ");

        double balance = scanner.nextDouble();

        BankAccount account = new BankAccount(name, accountNumber, balance);

        System.out.print("Enter Deposit Amount: ");

        double depositAmount = scanner.nextDouble();

        account.deposit(depositAmount);

        System.out.print("Enter Withdrawal Amount: ");

        double withdrawAmount = scanner.nextDouble();

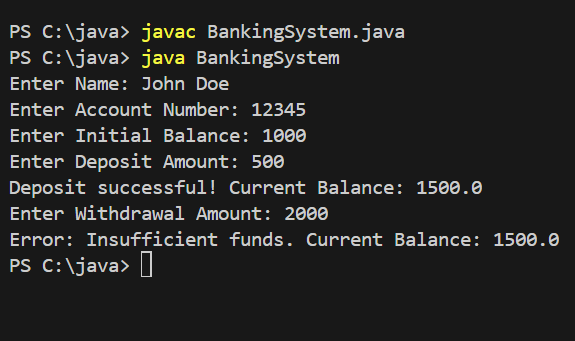
        account.withdraw(withdrawAmount);

        scanner.close();

    }

}

Output:



**Q-4** Problem Statement: Write a Java program to create a Product class with attributes id, name, and price. The program should demonstrate the use of constructors and methods to display product details

Input Example:

Product ID: 101

Name: Laptop

Price: 75000

Output Example:

Product Details:

ID: 101

Name: Laptop

Price: 75000

Code:

import java.util.Scanner;

class Product {

    private int id;

    private String name;

    private double price;

    public Product(int id, String name, double price) {

        this.id = id;

        this.name = name;

        this.price = price;

    }

    public void displayProductDetails() {

        System.out.println("Product Details:");

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

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

        System.out.println("Price: " + price);

    }

}

public class ProductDemo {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        int id = scanner.nextInt();

        scanner.nextLine();

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

        String name = scanner.nextLine();

        System.out.print("Enter Product Price: ");

        double price = scanner.nextDouble();

        Product product = new Product(id, name, price);

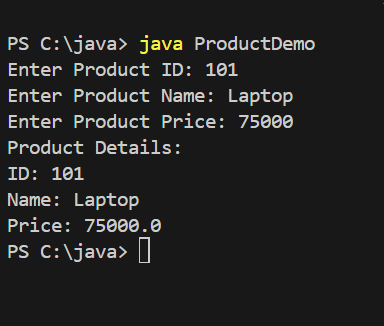
        product.displayProductDetails();

        scanner.close();

    }

}

Output:



**Q-5** Problem Statement: Write a Java program to implement a library management system. The program should:

Use a base class Book and derived classes Fiction and NonFiction.

Input Example: Output Example:

Book 1: Fiction Book Details:

Type: Fiction Title: Harry Potter

Title: Harry Potter Author: J.K. Rowling

Author: J.K. Rowling Price: 500

Price: 500

Book 2: Non-Fiction Book Details:

Type: Non-Fiction Title: Sapiens

Title: Sapiens Author: Yuval Noah Harari

Author: Yuval Noah Harari Price: 700

Price: 700

Code:

import java.util.Scanner;

class Book {

    protected String title;

    protected String author;

    protected double price;

    public Book(String title, String author, double price) {

        this.title = title;

        this.author = author;

        this.price = price;

    }

    public void displayDetails() {

        System.out.println("Title: " + title);

        System.out.println("Author: " + author);

        System.out.println("Price: " + price);

    }

}

class Fiction extends Book {

    public Fiction(String title, String author, double price) {

        super(title, author, price);

    }

    @Override

    public void displayDetails() {

        System.out.println("Fiction Book Details:");

        super.displayDetails();

    }

}

class NonFiction extends Book {

    public NonFiction(String title, String author, double price) {

        super(title, author, price);

    }

    @Override

    public void displayDetails() {

        System.out.println("Non-Fiction Book Details:");

        super.displayDetails();

    }

}

public class LibraryManagementSystem {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        System.out.print("Title: ");

        String fictionTitle = scanner.nextLine();

        System.out.print("Author: ");

        String fictionAuthor = scanner.nextLine();

        System.out.print("Price: ");

        double fictionPrice = scanner.nextDouble();

        scanner.nextLine();

        Fiction fictionBook = new Fiction(fictionTitle, fictionAuthor, fictionPrice);

        System.out.println("Enter details for Non-Fiction Book:");

        System.out.print("Title: ");

        String nonFictionTitle = scanner.nextLine();

        System.out.print("Author: ");

        String nonFictionAuthor = scanner.nextLine();

        System.out.print("Price: ");

        double nonFictionPrice = scanner.nextDouble();

        NonFiction nonFictionBook = new NonFiction(nonFictionTitle, nonFictionAuthor, nonFictionPrice);

        System.out.println();

        fictionBook.displayDetails();

        System.out.println();

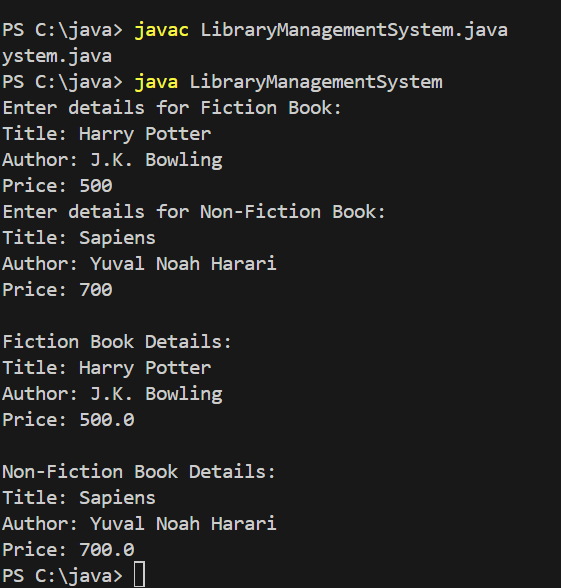
        nonFictionBook.displayDetails();

        scanner.close();

    }

}

Output:



**Q-6** Problem Statement: Design a student information system using Java with the following features: Use an abstract class Person with attributes name, age, and methods like displayDetails().

Create derived classes Student and Teacher to override displayDetails() and add unique attributes like rollNumber for students and subject for teachers.

Input Example:

Add Student:

Name: Alice

Age: 20

Roll Number: 101

Add Teacher:

Name: Mr. Smith

Age: 40

Subject: Mathematics

Output Example:

Student Details:

Name: Alice

Age: 20

Roll Number: 101

Teacher Details:

Name: Mr. Smith

Age: 40

Subject: Mathematics

Code:

import java.util.Scanner;

abstract class Person {

    protected String name;

    protected int age;

    public Person(String name, int age) {

        this.name = name;

        this.age = age;

    }

    abstract void displayDetails();

}

class Student extends Person {

    private int rollNumber;

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

        super(name, age);

        this.rollNumber = rollNumber;

    }

    @Override

    public void displayDetails() {

        System.out.println("Student Details:");

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

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

        System.out.println("Roll Number: " + rollNumber);

    }

}

class Teacher extends Person {

    private String subject;

    public Teacher(String name, int age, String subject) {

        super(name, age);

        this.subject = subject;

    }

    @Override

    public void displayDetails() {

        System.out.println("Teacher Details:");

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

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

        System.out.println("Subject: " + subject);

    }

}

public class StudentInformationSystem {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Add Student:");

        System.out.print("Name: ");

        String studentName = scanner.nextLine();

        System.out.print("Age: ");

        int studentAge = scanner.nextInt();

        System.out.print("Roll Number: ");

        int rollNumber = scanner.nextInt();

        scanner.nextLine();

        Student student = new Student(studentName, studentAge, rollNumber);

        System.out.println("Add Teacher:");

        System.out.print("Name: ");

        String teacherName = scanner.nextLine();

        System.out.print("Age: ");

        int teacherAge = scanner.nextInt();

        scanner.nextLine();

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

        String subject = scanner.nextLine();

        Teacher teacher = new Teacher(teacherName, teacherAge, subject);

        System.out.println();

        student.displayDetails();

        System.out.println();

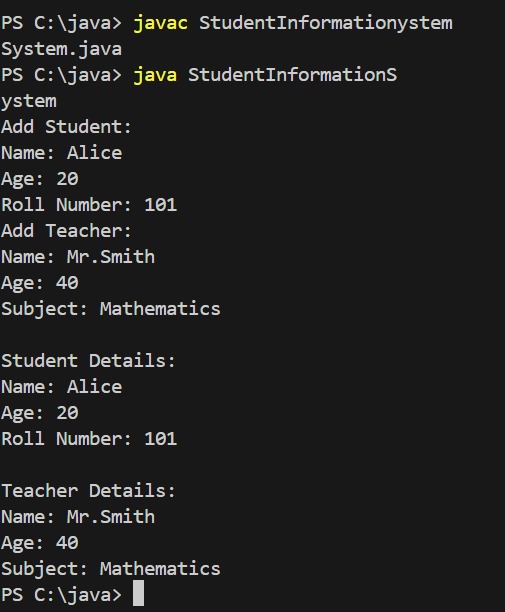
        teacher.displayDetails();

        scanner.close();

    }

}

Output:



**Q-7** Problem Statement: Write a Java program to calculate the square root of a number entered by the user. Use try-catch to handle invalid inputs (e.g., negative numbers or non-numeric values).

Input Example:

Enter a number: -16

Output Example:

Error: Cannot calculate the square root of a negative number.

Code:

import java.util.Scanner;

public class SquareRootCalculator {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        try {

            System.out.print("Enter a number: ");

            double number = scanner.nextDouble();

            if (number < 0) {

                throw new IllegalArgumentException("Error: Cannot calculate the square root of a negative number.");

            }

            double result = Math.sqrt(number);

            System.out.println("Square Root: " + result);

        } catch (IllegalArgumentException e) {

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

        } catch (Exception e) {

            System.out.println("Error: Invalid input. Please enter a numeric value.");

        } finally {

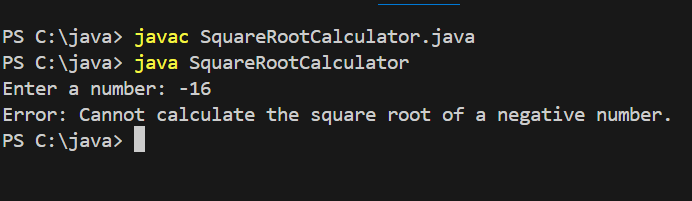
            scanner.close();

        }

    }

}

Output:



**Q-8** Problem Statement: Write a Java program to simulate an ATM withdrawal system. The program should:

Ask the user to enter their PIN.

Allow withdrawal if the PIN is correct and the balance is sufficient.

Throw exceptions for invalid PIN or insufficient balance.

Ensure the system always shows the remaining balance, even if an exception occurs.

Input Example:

Enter PIN: 1234

Withdraw Amount: 5000

Output Example:

Error: Insufficient balance. Current Balance: 3000

Code:

import java.util.Scanner;

class ATM {

    private static final int PIN = 1234;

    private double balance;

    public ATM(double initialBalance) {

        this.balance = initialBalance;

    }

    public void withdraw(int enteredPin, double amount) {

        try {

            if (enteredPin != PIN) {

                throw new SecurityException("Error: Invalid PIN.");

            }

            if (amount > balance) {

                throw new IllegalArgumentException("Error: Insufficient balance.");

            }

            balance -= amount;

            System.out.println("Withdrawal successful! Remaining Balance: " + balance);

        } catch (SecurityException | IllegalArgumentException e) {

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

        } finally {

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

        }

    }

}

public class ATMSystem {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        ATM atm = new ATM(3000);

        System.out.print("Enter PIN: ");

        int enteredPin = scanner.nextInt();

        System.out.print("Withdraw Amount: ");

        double amount = scanner.nextDouble();

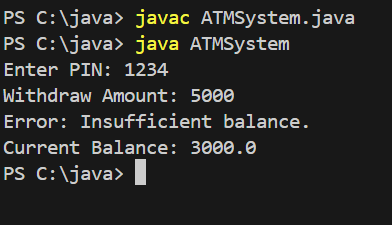
        atm.withdraw(enteredPin, amount);

        scanner.close();

    }

}

Output:



**Q-9** Problem Statement: Create a Java program for a university enrollment system with exception handling. The program should:

Allow students to enroll in courses.

Throw a CourseFullException if the maximum enrollment limit is reached.

Throw a PrerequisiteNotMetException if the student hasn’t completed prerequisite courses.

Input Example:

Enroll in Course: Advanced Java

Prerequisite: Core Java

Status: Prerequisite not completed

Output Example:

Error: PrerequisiteNotMetException - Complete Core Java before enrolling in Advanced Java.

Code:

import java.util.Scanner;

class CourseFullException extends Exception {

    public CourseFullException(String message) {

        super(message);

    }

}

class PrerequisiteNotMetException extends Exception {

    public PrerequisiteNotMetException(String message) {

        super(message);

    }

}

class Course {

    private String name;

    private String prerequisite;

    private int enrolledStudents;

    private int capacity;

    public Course(String name, String prerequisite, int capacity) {

        this.name = name;

        this.prerequisite = prerequisite;

        this.capacity = capacity;

        this.enrolledStudents = 0;

    }

    public void enrollStudent(boolean prerequisiteCompleted) throws CourseFullException, PrerequisiteNotMetException {

        if (enrolledStudents >= capacity) {

            throw new CourseFullException("Error: CourseFullException - Enrollment limit reached for " + name);

        }

        if (prerequisite != null && !prerequisiteCompleted) {

            throw new PrerequisiteNotMetException("Error: PrerequisiteNotMetException - Complete " + prerequisite + " before enrolling in " + name);

        }

        enrolledStudents++;

        System.out.println("Enrollment successful! You are now enrolled in " + name);

    }

}

public class UniversityEnrollmentSystem {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        Course advancedJava = new Course("Advanced Java", "Core Java", 2);

        System.out.print("Have you completed Core Java? (yes/no): ");

        boolean prerequisiteCompleted = scanner.next().equalsIgnoreCase("yes");

        try {

            advancedJava.enrollStudent(prerequisiteCompleted);

        } catch (CourseFullException | PrerequisiteNotMetException e) {

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

        }

        scanner.close();

    }

}

Output:

