1. Write a Java program to check if the given number is Armstrong number or not.

import java.util.Scanner;

public class armstrongFinder {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number to check if it is an Armstrong number: ");

int number = scanner.nextInt();

if (isArmstrong(number)) {

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

} else {

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

}

scanner.close();

}

public static boolean isArmstrong(int number) {

int originalNumber = number;

int sum = 0;

int digits = countDigits(number);

while (number > 0) {

int digit = number % 10;

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

number /= 10;

}

return sum == originalNumber;

}

public static int countDigits(int number) {

int count = 0;

while (number > 0) {

count++;

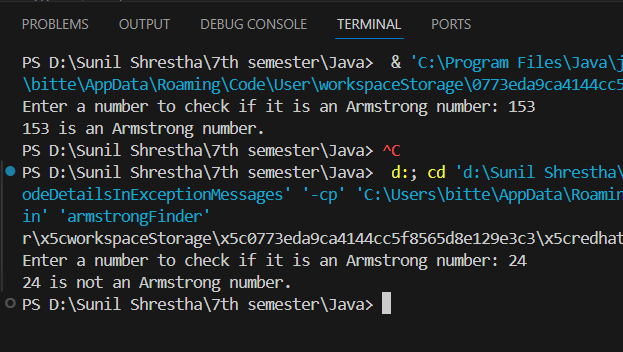
number /= 10;

}

return count;

}

}



2. Write a Java program to check the given number is palindrome or not.

import java.util.Scanner;

public class palindromeChecker {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        int originalNumber = scanner.nextInt();

        if (isPalindrome(originalNumber)) {

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

        } else {

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

        }

    }

    public static boolean isPalindrome(int number) {

        int original = number;

        int reversed = 0;

        while (number != 0) {

            int digit = number % 10;

            reversed = reversed \* 10 + digit;

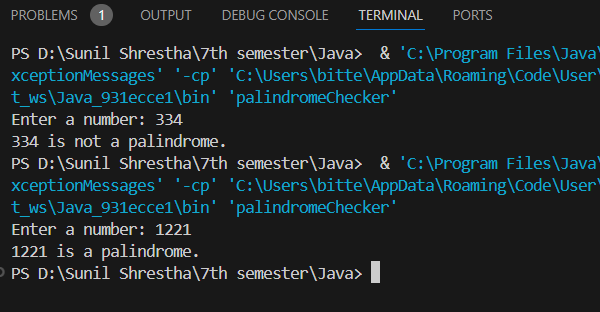
            number /= 10;

        }

        return original == reversed;

    }

}



3. Write a Java program that demonstrate the try, catch and finally block to handle exception.

public class ExceptionDemo {

    public static void main(String[] args) {

        try {

            System.out.println("Attempting to perform a division operation.");

            int result = 10 / 0;

        } catch (ArithmeticException e) {

            System.out.println("Oops! Something went wrong: couldn't divide by zero " + e.getMessage());

        } finally {

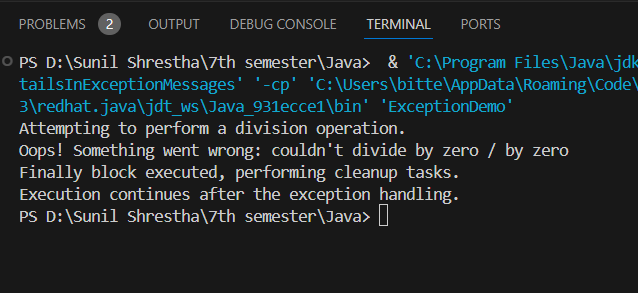
            System.out.println("Finally block executed, performing cleanup tasks.");

        }

        System.out.println("Execution continues after the exception handling.");

    }

}



4. Write a program to find second minimum and second maximum number from the array.

import java.util.Arrays;

public class MinMaxFinder {

public static void main(String[] args) {

int[] numbers = {12, 35, 1, 10, 34, 1};

int[] result = findSecondMinMax(numbers);

System.out.println("Second Minimum: " + result[0]);

System.out.println("Second Maximum: " + result[1]);

}

public static int[] findSecondMinMax(int[] numbers) {

Arrays.sort(numbers);

int secondMin = -1, secondMax = -1;

for (int i = 1; i < numbers.length; i++) {

if (numbers[i] != numbers[0]) {

secondMin = numbers[i];

break;

}

}

for (int i = numbers.length - 2; i >= 0; i--) {

if (numbers[i] != numbers[numbers.length - 1]) {

secondMax = numbers[i];

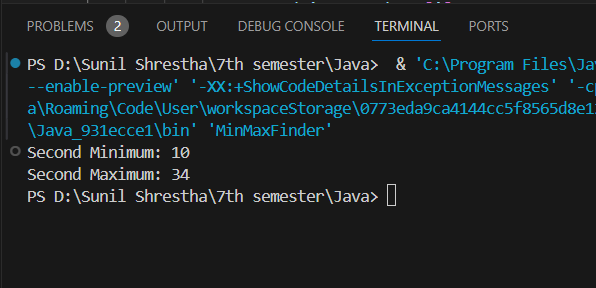
break;

}

}

return new int[] { secondMin, secondMax };

}



5. Write a Java program that creates two threads: one for printing even numbers and another for printing odd numbers. The program should ensure that the threads print the numbers alternatively, starting from 1 up to a given limit (e.g., 20) (Hint : You may use the synchronized keyword, wait(), and notify() )

public class EvenOddThread {

static class PrintEvenOdd {

private int number = 1;

public synchronized void printOdd() throws InterruptedException {

while (number <= 20) {

if (number % 2 == 0) {

wait();

}

if (number <= 20) {

System.out.println("Odd: " + number);

number++;

notify();

}

}

}

public synchronized void printEven() throws InterruptedException {

while (number <= 20) {

if (number % 2 != 0) {

wait();

}

if (number <= 20) {

System.out.println("Even: " + number);

number++;

notify();

public static void main(String[] args) {

PrintEvenOdd obj = new PrintEvenOdd();

Thread t1 = new Thread(() -> {

try {

obj.printOdd();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

});

Thread t2 = new Thread(() -> {

try {

obj.printEven();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

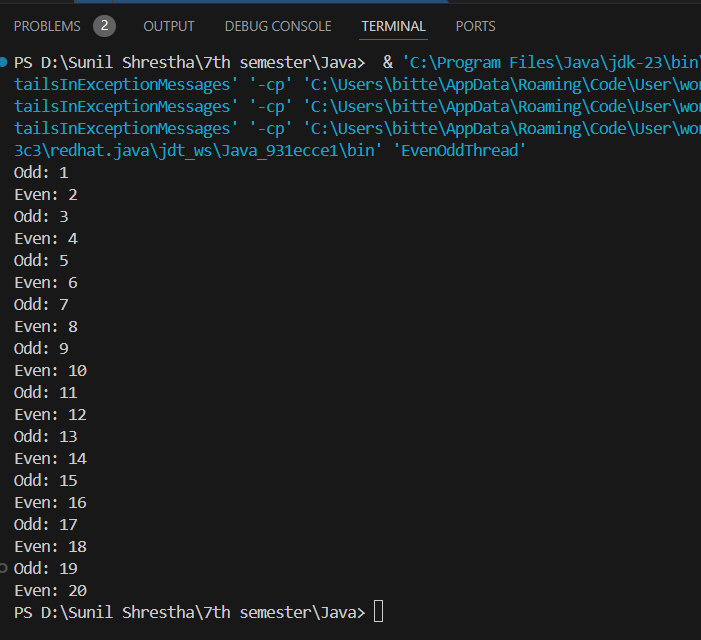
});

t1.start();

t2.start();

}

}



6. Write a Java program that defines a Person class with the following attributes: name, age, and gender. Include a constructor to initialize these attributes. Create two instances of the Person class and display their details.

**Person.java**

public class Person {

    String name;

    int age;

    String gender;

    public Person(String name, int age, String gender) {

        this.name = name;

        this.age = age;

        this.gender = gender;

    }

    public void displayDetails() {

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

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

        System.out.println("Gender: " + gender);

    }

}  
**Main.Java**

public class Main {

    public static void main(String[] args) {

        Person person1 = new Person("Sunil", 23, "Male");

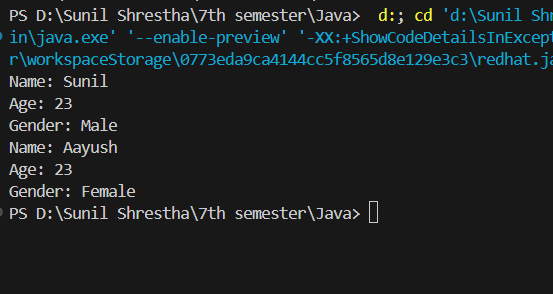
        Person person2 = new Person("Aayush", 23, "Female");

        person1.displayDetails();

        person2.displayDetails();

    }

}



7. Create a base class called Shape with methods calculateArea() and calculatePerimeter(). Derive two classes, Circle and Rectangle, from the Shape class. Implement the necessary methods in each derived class and demonstrate their usage.

abstract class Shape {

    public abstract double calculateArea();

    public abstract double calculatePerimeter();

}

class Circle extends Shape {

    private double radius;

    public Circle(double radius) {

        this.radius = radius;

    }

    @Override

    public double calculateArea() {

        return Math.PI \* radius \* radius;

    }

    @Override

    public double calculatePerimeter() {

        return 2 \* Math.PI \* radius;

    }

}

class Rectangle extends Shape {

    private double length;

   private double width;

    public Rectangle(double length, double width) {

        this.length = length;

        this.width = width;

    }

    @Override

    public double calculateArea() {

        return length \* width;

    }

    @Override

    public double calculatePerimeter() {

        return 2 \* (length + width);

    }

}

public class ShapesExample {

    public static void main(String[] args) {

        Shape circle = new Circle(5);

        Shape rectangle = new Rectangle(4, 7);

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

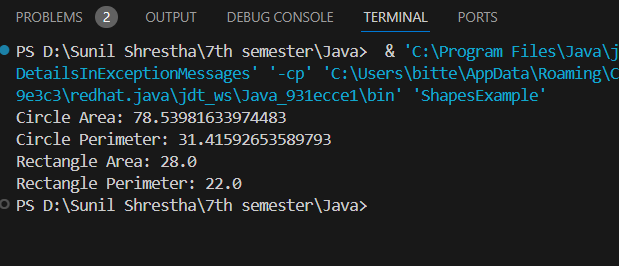
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());

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

        System.out.println("Rectangle Perimeter: " + rectangle.calculatePerimeter());

    }

}



8. Design a BankAccount class with private attributes accountNumber, accountHolder, and balance. Include methods to deposit and withdraw funds, and a method to display the account details. Ensure proper encapsulation by providing public accessors and mutators.

public class BankAccount {

    private String accountNumber;

    private String accountHolder;

    private double balance;

    public BankAccount(String accountNumber, String accountHolder, double initialBalance) {

        this.accountNumber = accountNumber;

        this.accountHolder = accountHolder;

        this.balance = initialBalance;

    }

    public String getAccountNumber() {

        return accountNumber;

    }

    public void setAccountNumber(String accountNumber) {

        this.accountNumber = accountNumber;

    }

    public String getAccountHolder() {

        return accountHolder;

    }

    public void setAccountHolder(String accountHolder) {

       this.accountHolder = accountHolder;

    }

    public double getBalance() {

        return balance;

    }

    public void setBalance(double balance) {

        this.balance = balance;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

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

        } else {

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

        }

    }

    public void withdraw(double amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

            System.out.println("Withdrew: " + amount);

        } else if (amount > balance) {

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

        } else {

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

        }

    }

    public void displayAccountDetails() {

        System.out.println("Account Number: " + accountNumber);

        System.out.println("Account Holder: " + accountHolder);

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

    }

    public static void main(String[] args) {

        BankAccount account = new BankAccount("123456789", "Sunil Shrestha", 1000.00);

        account.displayAccountDetails();

        account.deposit(500);

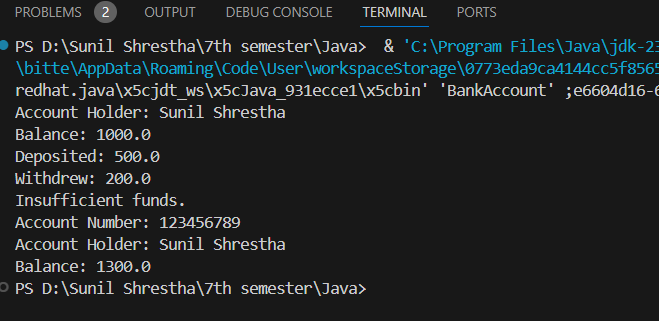
        account.withdraw(200);

        account.withdraw(1500);

        account.displayAccountDetails();

    }

}



9. Create an interface called Drawable with a method draw(). Implement this interface in two classes, Circle and Square. Create an array of Drawable objects and invoke the draw() method for each object.

**Drawable.java**

public interface Drawable {

    void draw();

}

**Circle.java**

public class Circle implements Drawable {

    @Override

    public void draw() {

        System.out.println("Drawing a Circle");

    }

}

**Square.java**

public class Square implements Drawable {

    @Override

    public void draw() {

        System.out.println("Drawing a Square");

    }

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        Drawable[] drawables = new Drawable[2];

        drawables[0] = new Circle();

        drawables[1] = new Square();

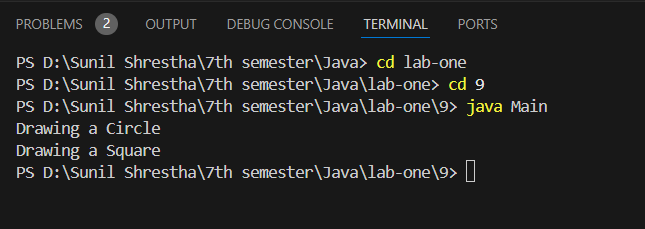
        for (Drawable drawable : drawables) {

            drawable.draw();

        }

    }

}



10. Design an abstract class called Animal with abstract methods makeSound() and move(). Create two subclasses, Dog and Bird, that extend the Animal class. Implement the abstract methods in each subclass and demonstrate their usage.

**Animal.java**

package lab10;

public abstract class Animal {

    public abstract void makeSound();

    public abstract void move();

}

**Dog.java**

package lab10;

public class Dog extends Animal {

    @Override

    public void makeSound() {

        System.out.println("The dog barks: Woof! Woof!");

    }

    @Override

    public void move() {

        System.out.println("The dog runs on four legs.");

    }

}

**Bird.java**

package lab10;

public class Bird extends Animal {

    @Override

    public void makeSound() {

        System.out.println("The bird chirps: Tweet! Tweet!");

    }

    @Override

    public void move() {

        System.out.println("The bird flies in the sky.");

    }

}

**Main.java**

package lab10;

public class Main {

    public static void main(String[] args) {

        Animal dog = new Dog();

        Animal bird = new Bird();

        System.out.println("Dog:");

        dog.makeSound();

        dog.move();

        System.out.println();

        System.out.println("Bird:");

        bird.makeSound();

        bird.move();

    }

