Implement a Java program that demonstrates the three key OOP principles: Abstraction, Encapsulation, and Inheritance.

Problem Statement:
You are developing a Vehicle Management System. The system should support different types of vehicles like Cars and Bikes while following OOP principles.

Tasks:
1. Abstraction:Create an abstract class Vehicle with the following:
   - String brand (Encapsulated, use getter/setter)
   - An abstract method void startEngine();

2. Inheritance:
   - Create two classes, Car and Bike, that inherit from Vehicle.
   - Implement the startEngine() method in both subclasses with a unique message.

3. Encapsulation:
   - The brand field in Vehicle should be private, and accessible via public getter/setter methods.

Objective:
Implement a Java program that demonstrates method overloading (compile-time polymorphism) and method overriding (runtime polymorphism).

Problem Statement:
You are developing a Calculator application that can perform various operations. The calculator should support different ways of calculating the sum of numbers (method overloading) and different types of calculations based on the calculator type (method overriding).

Tasks:
1. Method Overloading:
Create a class Calculator with multiple sum methods:

- sum(int a, int b) - Adds two integers.
- sum(double a, double b) - Adds two doubles.
- sum(int a, int b, int c) - Adds three integers.

Create a main method to call each of these methods.

2.Method Overriding:

- Create a class AdvancedCalculator that extends Calculator.
- Override the sum(int a, int b) method to include a custom message.

3. Polymorphism:
    - In the main method, demonstrate runtime polymorphism by using a reference of type Calculator pointing to an AdvancedCalculator object.

Question Title:
Perform Basic Operations on an Integer Array

Problem Statement:
Write a Java program to do the following:

1. Accept n integers from the user and store them in an array.

2. Print the array elements.

3. Find and display:

    - The **maximum** number in the array.

    - The **minimum** number in the array.

    - The **average** of the numbers.

Write a Java program to perform division of two integers entered by the user.
Use **try-catch blocks** to handle the following situations:

- If the user enters a non-integer value, handle `InputMismatchException`.

- If the user tries to divide by zero, handle `ArithmeticException`.

Write a Java program that demonstrates:

1. **Types of inheritance** (single, multilevel, or hierarchical).
2. **The `final` keyword** (Apply it to a variable, method, and class).
3. **The `super` keyword** (Use it to call a parent class constructor and method).