# Java Problem Statements Day 6

***By Kavuri Santosh Kumar***

**Methods :-**

Methods are the block of instructions in the form of statements used to perform various operations for each and every operation developer has to create different number of methods

- ➢ **Method:-**A method is a block of code that performs a specific task.
- A method is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a method.
- ➢ **Why use methods.?**
- To reuse code: define the code once, and use it many times.
- Reducing the total number of lines of code.

**Methods are classified into two parts:**

1.Method declaration /header

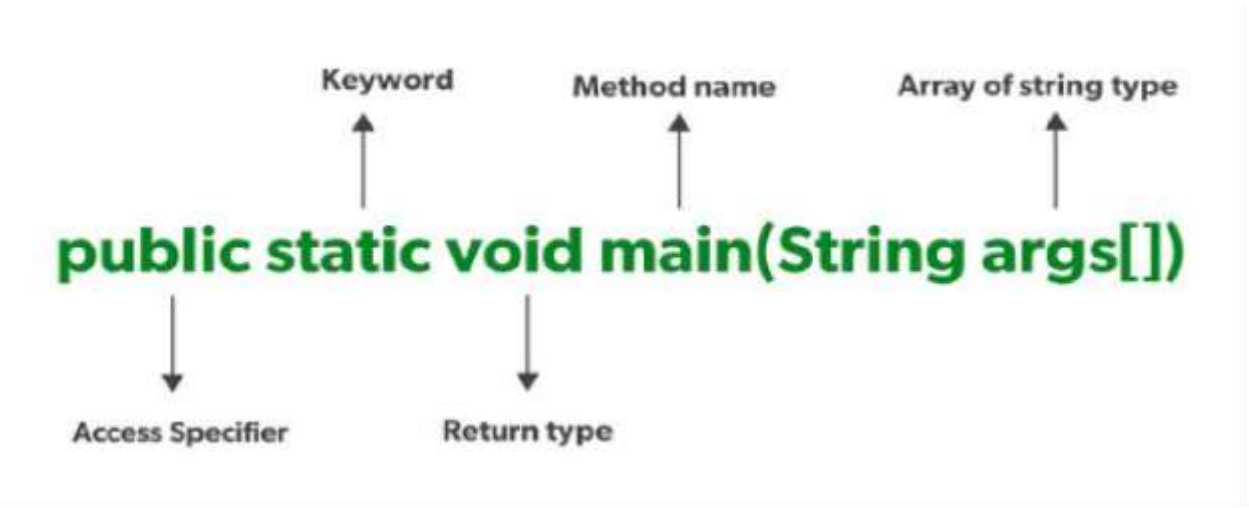
2.Methods implementation /context/body/block/boundary

**Syntax:-**

**1.method declaration/ header**

<Access-speclifier> <access-modifier><return type><method_name>(<variable declration>){

// statements/ **method implementation**/context/body/boundary
}



**Note:-**

-We can create any number of methods but JVM execute only main(String [] args).

-In order to execute other methods we need to use the signature of the method known as method calling statement
-Method calling statement can be written in any methods and without executing the method completely the control never returns to the method calling statements.

**Call a Method:-** To call a method in Java, write the method's name followed by two

parentheses () and a semicolon;

**Example:-**

```
class Method{
        static void myMethod{
                System.out.println(“I just got executed!”);
}
        public static void main(String args[]){
                myMethod();
}
}
```

- **Example:-**

```
class Main {
 static void myMethod() {
   System.out.println("I just got executed!");
 }
 public static void main(String[] args) {
   myMethod();-→ We can call the method n no of time to get the o/p
   myMethod();
   myMethod();
 }
}
```

**what are the function/methods are there in java .?**

1. Method without return type and without parameters
2. Method without return type and with parameters
3. Method with return type and without parameters
4. Method with return type and with parameters

**Static methods:-**

**1.Method without return type and without parameters**

**Syntax:-**

```
accessModifier void methodName( ) {
   // Method body
   // Statements to perform the task
}
```

**Example1:-**

```
public class Example {
   public static void sayHello() {
      System.out.println("Hello, Jenny !");
   }
   public static void main(String[] args) {
      sayHello(); // Calling the method
   }
}
```

**Example 1:-**
**CODE:-**

```
import java. util.Scanner;
class Sign{
        void findSign(){
        Scanner scan=new Scanner(System.in);
        System.out.println("Enter a number");
        int num=scan.nextInt();
        if(num>0){
        System.out.println("positive");
        }
        else if(num<0){
        System.out.println("negative");
        }
        else{
        System.out.println("zero");
```

```
        }
    }
        public static void main(String [] args){
        Sign obj =new Sign();
        obj.findSign();

        }
}
```

**Example 2:-**
**CODE:-**

```
import java. util.Scanner;
class Odd{
        void findOddEven(){
        Scanner scan=new Scanner(System.in);
        int num=scan.nextInt();
        if(num%2==0){
        System.out.println("Even number");
        }
        else{
        System.out.println("odd number");
        }
   }
        public static void main(String[] args){
        Odd obj =new Odd();
        obj.findOddEven();

        }
}
```

**2. Method without return type and with parameters**

**Syntax:-**

```
accessModifier void methodName(parameterType1 parameter1, parameterType2 parameter2, ...) {
   // Method body
   // Statements to perform the task using parameters
}
```

**Example:-**

**CODE:-**

```
class Example {
   public static void printSum(int a, int b) {
      int sum = a + b;
      System.out.println("Sum of " + a + " and " + b + " is: " + sum);
   }
   public static void main(String[] args) {
      printSum(3, 5); // Calling the method with arguments
      printSum(-2, 7); // Calling the method with different arguments
   }
}
```

```
import java. util.Scanner;
class Odd{
         void findOddEven(int num){
         if(num%2==0){
         System.out.println("Even number");
         }
         else{
         System.out.println("odd number");
         }
   }
         public static void main(String[] args){
         Odd obj =new Odd();
          Scanner scan=new Scanner(System.in);
         int num=scan.nextInt();
         obj.findOddEven(num);

         }
}
```

**3. Method with return type and without parameters**

**Example1:-**

**CODE:-**

```
public class Example {
   // Method with return type (int) and without parameters
   public static int getRandomNumber() {
      return 42; // Return a constant value (just for demonstration)
   }
   public static void main(String[] args) {
      // Calling the method and storing the returned value
      int randomNumber = getRandomNumber();
      // Printing the returned value
      System.out.println("Random Number: " + randomNumber);
   }
}
```

**Example 2:-**

**CODE:-**

```
import java. util.Scanner;
class Odd{
        int findOddEven(){
        Scanner scan=new Scanner(System.in);
        int scan=scan.nextInt();
        if(n%2==0){
                return 0;
        }
        else{
                return 1;
       }
        }
        public static void main(String[] args){
        Odd obj=new Odd();
        int result;
        result=obj.findOddEven();
        if(result==0){
                System. out.println("Even number");
        }
        else{
                System.out.println("odd number");
                }
        }
}
```

**4. Method with return type and with parameters**

**Example:-**

**CODE:-**

```
public class Example {
   // Method with return type (double) and parameters (width and height)
   public static double calculateRectangleArea(double width, double height) {
      double area = width * height;
      return area;
   }

   public static void main(String[] args) {
      double width = 5.0;
      double height = 3.0;

      // Calling the method with arguments and storing the returned value
      double rectangleArea = calculateRectangleArea(width, height);
      // Printing the returned value
      System.out.println("Rectangle Area: " + rectangleArea);
   }
}
```

**Example:-**

**CODE:-**

```
import java. util.Scanner;
class Odd{
        int findOddEven(int num){
        if(num%2==0){
                return 0;
        }
        else{
                return 1;
        }
 }
        public static void main(String [] args){
        Scanner scan =new Scanner(System.in);
        Odd obj=new Odd();
        int num=scan.nextInt();
        int result;
```

```
        result=obj.findOddEven(num);
        if(result==0){
                System.out.println("Even");
      }
        else{
                System.out.println("odd");
        }
}
```

**Non-static methods:-**

**Example:-**

```
public class Example {
   // Method with return type (double) and parameters (width and height)
   public double calculateRectangleArea(double width, double height) {
      double area = width * height;
      return area;
   }
   public static void main(String[] args) {
      Example example = new Example(); // Creating an instance of the class
      double width = 5.0;
      double height = 3.0;
      // Calling the method on an instance and storing the returned value
      double rectangleArea = example.calculateRectangleArea(width, height);
      // Printing the returned value
      System.out.println("Rectangle Area: " + rectangleArea);
   }
}
```

**Recursion:-**

Recursion is a programming technique where a method calls itself to solve a problem. In Java, you can implement recursion by having a method call itself within its own definition. Recursion is often used when a problem can be broken down into smaller, similar subproblems.

**Example:-**

```
class Example {
   public static int factorial(int n) {
      // Base case: if n is 0 or 1, return 1
      if (n == 0 || n == 1) {
         return 1;
      } else {
         // Recursive case: n! = n * (n-1)!
         return n * factorial(n - 1);
      }
   }
   public static void main(String[] args) {
      int number = 5;
      int result = factorial(number);
      System.out.println("Factorial of " + number + " is: " + result);
   }
}
```

**Explanation:**

- The **factorial** method takes an integer **n** as input and returns its factorial.
- The base case checks if **n** is 0 or 1. If it is, it returns 1, as 0! and 1! are defined as 1.
- The recursive case calculates the factorial of **n** by multiplying **n** with the factorial of **n-1**. This is the essence of recursion: breaking down a problem into smaller, similar sub-problems until reaching a base case.
- In the **main** method, we call the **factorial** method with the value **5**, and then print the result.

**Math Methods**

In Java, the `Math` class is part of the `java.lang` package and provides a variety of methods to perform basic mathematical operations.

## 1. Basic Arithmetic Operations:

- `Math.abs(x)` – Returns the absolute value of `x`.
- `Math.addExact(x, y)` – Adds two integers and throws an exception if overflow occurs.
- `Math.subtractExact(x, y)` – Subtracts two integers and throws an exception if overflow occurs.
- `Math.multiplyExact(x, y)` – Multiplies two integers and throws an exception if overflow occurs.
- `Math.incrementExact(x)` – Increments an integer and throws an exception if overflow occurs.
- `Math.decrementExact(x)` – Decrements an integer and throws an exception if overflow occurs.
- `Math.negateExact(x)` – Negates an integer and throws an exception if overflow occurs.

## 2. Rounding Methods:

- `Math.ceil(x)` – Returns the smallest integer greater than or equal to `x`.
- `Math.floor(x)` – Returns the largest integer less than or equal to `x`.
- `Math.round(x)` – Rounds `x` to the nearest integer (returns a `long`).
- `Math.rint(x)` – Returns the closest integer value to `x` as a `double`.

## 3. Exponential and Logarithmic Methods:

- `Math.exp(x)` – Returns Euler's number raised to the power of `x` (`e^x`).
- `Math.log(x)` – Returns the natural logarithm (base e) of `x`.
- `Math.log10(x)` – Returns the base 10 logarithm of `x`.
- `Math.pow(x, y)` – Returns `x` raised to the power of `y` (`x^y`).
- `Math.sqrt(x)` – Returns the square root of `x`.

## 4. Trigonometric Methods:

- `Math.sin(x)` – Returns the sine of the angle `x` (in radians).
- `Math.cos(x)` – Returns the cosine of the angle `x` (in radians).
- `Math.tan(x)` – Returns the tangent of the angle `x` (in radians).
- `Math.asin(x)` – Returns the arc sine of `x` in radians.
- `Math.acos(x)` – Returns the arc cosine of `x` in radians.
- `Math.atan(x)` – Returns the arc tangent of `x` in radians.
- `Math.atan2(y, x)` – Returns the angle (in radians) between the positive x-axis and the point (x, y).
- `Math.toRadians(x)` – Converts the angle `x` from degrees to radians.
- `Math.toDegrees(x)` – Converts the angle `x` from radians to degrees.

## 5. Utility Methods:

- `Math.max(x, y)` – Returns the larger of `x` and `y`.
- `Math.min(x, y)` – Returns the smaller of `x` and `y`.
- `Math.random()` – Returns a random `double` value between 0.0 (inclusive) and 1.0 (exclusive).
- `Math.random()` – Generates a random number.

- `Math.copySign(x, y)` – Returns a value with the magnitude of `x` and the sign of `y`.
- `Math.signum(x)` – Returns the sign of the value `x` (-1, 0, or 1).

## 6. Constants:

- `Math.PI` – The value of Pi (approximately 3.14159).
- `Math.E` – The value of Euler's number (approximately 2.71828).

**PROGRAMS**

**1. Write a Java method that takes an integer as input and returns true if it is a prime number, otherwise returns false.**
**2. Write a Java method to calculate the factorial of a given number.**
**3. Write a Java method to generate the Fibonacci series up to a specified number of terms.**
**4. Write a Java method to calculate the power of a number raised to an exponent.**
**5. Write a Java method to calculate the greatest common divisor (GCD) of two numbers.**
**6. Write a Java method to calculate the least common multiple (LCM) of two numbers.**
**7. Write a Java method to check if a given number is an Armstrong number.**
**8. Write a Java method to check if a given number is a perfect number.**

## ANSWERS

**1. Write a Java method that takes an integer as input and returns true if it is a prime number, otherwise returns false.**

**CODE:-**

```
import java.util.Scanner;

class Main {
    public boolean isPrime(int n) {
        // Check if the number is less than or equal to 1
        if (n <= 1) {
            return false; // Numbers less than or equal to 1 are not prime
        }
        // Check divisibility from 2 to the square root of n
        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0) {
                return false; // If divisible by any number, it's not prime
            }
        }
        // If no divisors found, it's prime
        return true;
    }
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt(); // Read the input number
        Main obj = new Main(); // Create an object of Main class
        boolean result = obj.isPrime(n); // Call the isPrime method
        System.out.println(result); // Print the result (true if prime, false otherwise)
    }
}
```

**2. Write a Java method to calculate the factorial of a given number.?**

**CODE:-**

```
import java.util.Scanner;
class FactorialCalculator {
    // Method to calculate factorial
    public static int factorial(int n) {
        int result = 1;
        for (int i = 1; i <= n; i++) {
            result *= i; // Multiply result by i for each iteration
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Create scanner object for user input
        System.out.print("Enter a number to calculate its factorial: ");
        int n = scanner.nextInt(); // Read the number from the user
        // Call factorial method and store the result
        int result = factorial(n);
        // Display the result
        System.out.println("Factorial of " + n + " is: " + result);
    }
}
```

**3. Write a Java method to generate the Fibonacci series up to a specified number of terms.?**

**CODE:-**

```
import java.util.Scanner;
class Main {
   public int mainName(int n){
       int a=0;
      int b=1;
      int c;
      for(int i=1;i<=n;i++){
         System.out.println(a);
         c=a+b;
         a=b;
         b=c;
      }
      return b;

   }
   public static void main(String[] args) {
      Scanner scan=new Scanner(System.in);
      int n=scan.nextInt();
      Main obj=new Main();
      int result=obj.mainName(n);
      System.out.println(result);
   }
}
```

**4. Write a Java method to calculate the power of a number raised to an exponent.?**

To calculate the power of a number raised to an exponent, you can use the formula:

$$\text{Power} = \text{Base}^{\text{Exponent}}$$

For example, if you want to calculate $2^3$ (2 raised to the power of 3):

$$2^3 = 2 \times 2 \times 2 = 8$$

Here's a quick breakdown for another example:

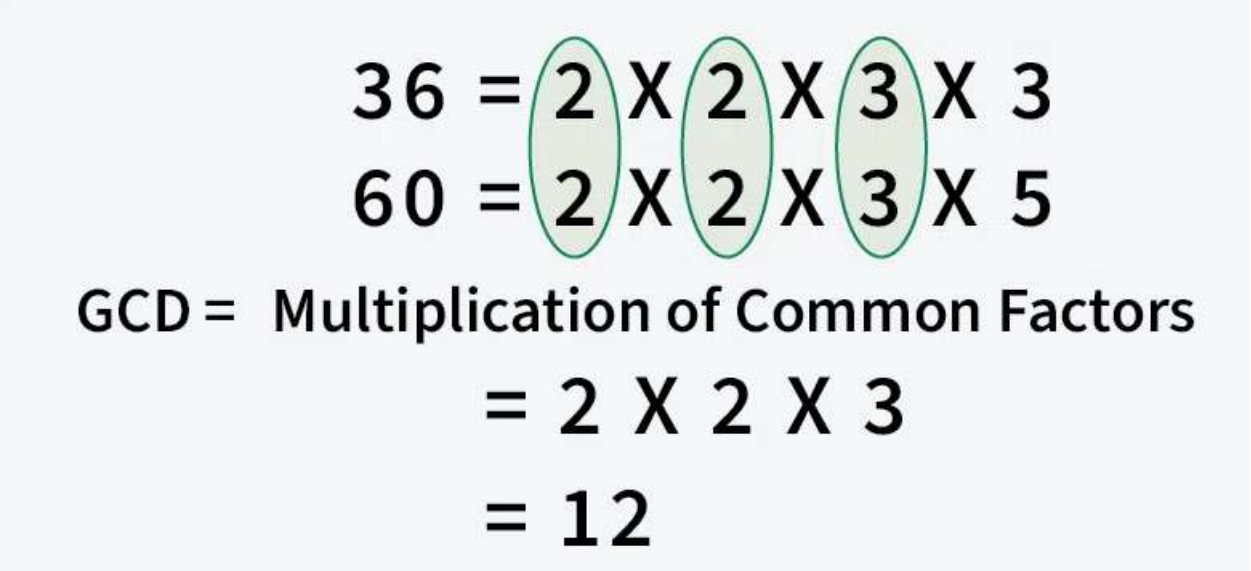
Let's calculate $5^4$:

$$5^4 = 5 \times 5 \times 5 \times 5 = 625$$

**CODE:-**

```
import java.util.Scanner;
class PowerCalculator {
    // Method to calculate power
    public static double calculatePower(double base, double exponent) {
        return Math.pow(base, exponent); // Using Math.pow to compute the result
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input the base and the exponent
        System.out.print("Enter the base: ");
        double base = scanner.nextDouble();
        System.out.print("Enter the exponent: ");
        double exponent = scanner.nextDouble();
        // Call the method and get the result
        double result = calculatePower(base, exponent);
        // Display the result
        System.out.println(base + " raised to the power of " + exponent + " is: " + result);
    }
}
```

**5. Write a Java method to calculate the greatest common divisor (GCD) of two numbers.?**



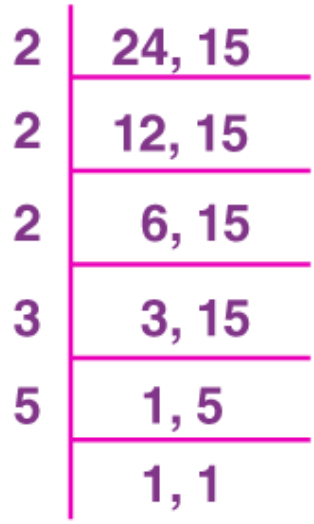

***Input:*** *a = 20, b = 28*
***Output:*** *4*
***Explanation:*** *The factors of 20 are 1, 2, 4, 5, 10 and 20. The factors of 28 are 1, 2, 4, 7, 14 and 28. Among these factors, 1, 2 and 4 are the common factors of both 20 and 28. The greatest among the common factors is 4.*
***Input:*** *a = 60, b = 36*
***Output:*** *12*

**CODE:-**

```
import java.util.Scanner;

class GCDCalculator {

    // Method to calculate GCD using the Euclidean algorithm

    public static int calculateGCD(int a, int b) {

        while (b != 0) {

            int temp = b;

            b = a % b;

            a = temp;

        }

        return a;

    }
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input two integers
        System.out.print("Enter the first number: ");
        int num1 = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int num2 = scanner.nextInt();
        // Call the GCD method and display the result
        int gcd = calculateGCD(num1, num2);
        System.out.println("The GCD of " + num1 + " and " + num2 + " is: " + gcd);
    }
}
```

**6. Write a Java method to calculate the least common multiple (LCM) of two numbers.?**




Here, the LCM of 24, 15 will be $2\times2\times2\times3\times5=2^3\times3\times5=120$

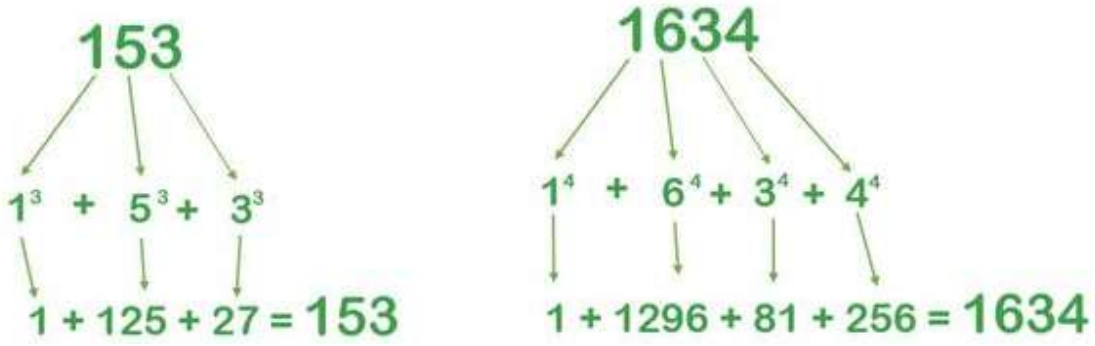
L.C.M. = a×b/ gcd(a,b)

For example, for 15 and 24, the GCF will be 3. So, the LCM will be (15 × 24) / 3 = 3.

**CODE:-**

```
import java.util.Scanner;
class LCMCalculator {
    // Method to calculate GCD using the Euclidean algorithm
    public static int calculateGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
    // Method to calculate LCM
    public static int calculateLCM(int a, int b) {
        return Math.abs(a * b) / calculateGCD(a, b); // Using GCD to calculate LCM
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input two integers
        System.out.print("Enter the first number: ");
        int num1 = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int num2 = scanner.nextInt();
        // Call the LCM method and display the result
        int lcm = calculateLCM(num1, num2);
        System.out.println("The LCM of " + num1 + " and " + num2 + " is: " + lcm);
    }
}
```

**7. Write a Java method to check if a given number is an Armstrong number.?**

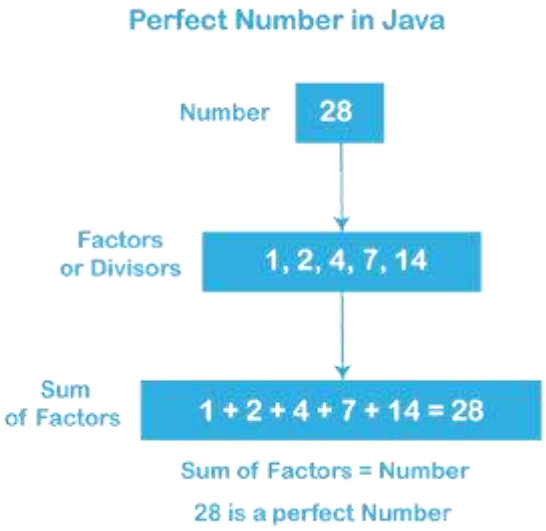



**CODE:-**

```
import java.util.Scanner;
class ArmstrongNumberChecker {
    // Method to check if a number is an Armstrong number without using strings
    public static boolean isArmstrong(int number) {
        int originalNumber = number;
        int sum = 0;
        int digits = 0;
        // Calculate the number of digits in the number
        int temp = number;
        while (temp > 0) {
            temp /= 10;
            digits++;
        }
```

```
        // Calculate the sum of the powers of each digit
        temp = number;
        while (temp > 0) {
            int digit = temp % 10;  // Extract the last digit
            sum += Math.pow(digit, digits);  // Add the power of the digit to the sum
            temp /= 10;  // Remove the last digit
        }
        // Check if the sum matches the original number
        return sum == originalNumber;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input a number
        System.out.print("Enter a number to check if it is an Armstrong number: ");
        int num = scanner.nextInt();
        // Check and display the result
        if (isArmstrong(num)) {
            System.out.println(num + " is an Armstrong number.");
        } else {
            System.out.println(num + " is not an Armstrong number.");
        }
    }
}
```

**8. Write a Java method to check if a given number is a perfect number.?**




**CODE:-**

```
import java.util.Scanner;
class PerfectNumberChecker {
    // Method to check if a number is a perfect number
    public static boolean isPerfectNumber(int number) {
        int sum = 0;
        // Find all divisors of the number and add them
        for (int i = 1; i <= number / 2; i++) {
            if (number % i == 0) {
                sum += i; // Add the divisor to sum
            }
        }
```

```
        // A number is perfect if the sum of its divisors equals the number itself
        return sum == number;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input a number
        System.out.print("Enter a number to check if it is a perfect number: ");
        int num = scanner.nextInt();
        // Check and display the result
        if (isPerfectNumber(num)) {
            System.out.println(num + " is a perfect number.");
        } else {
            System.out.println(num + " is not a perfect number.");
        }
    }
}
```