NUMBER-RELATED PROGRAMS IN JAVA

There are many number-related programs in Java, covering different mathematical concepts. Here are some categories along with examples:

Basic Number Programs:

1. Check if a number is even or odd
2. Check if a number is prime
3. Find the sum of digits of a number
4. Reverse a number
5. Find the factorial of a number

Special Number Programs:

6. Palindrome Number (121 → same when reversed)
7. Armstrong Number (153 → $1^3 + 5^3 + 3^3 = 153$)
8. Perfect Number (28 → sum of factors = 28)
9. Strong Number (145 → sum of factorial of digits = 145)
10. Harshad Number (18 → 18 is divisible by 1+8 = 9)

Mathematical Series Programs:

11. Fibonacci Series
12. Factorial using recursion
13. LCM and GCD of two numbers
14. Sum of first N natural numbers
15. Sum of squares of first N numbers

Number Conversion Programs:

16. Decimal to Binary conversion
17. Binary to Decimal conversion
18. Decimal to Octal conversion
19. Decimal to Hexadecimal conversion
20. Roman Number to Integer conversion

Sorting and Searching Number Programs:

21. Find the largest/smallest number in an array
22. Sort an array using Bubble Sort, Quick Sort, etc.
23. Search an element in an array (Linear/Binary Search)
24. Find the second largest number in an array
25. Find duplicate numbers in an array

Random and Advanced Number Programs:

26. Generate random numbers
27. Check if a number is power of 2
28. Generate prime numbers within a range
29. Find HCF (Highest Common Factor) of two numbers
30. Check if a number is a Fibonacci number

BASIC NUMBER PROGRAMS:

CHECK IF A NUMBER IS EVEN OR ODD:

If a number is evenly divisible by 2 with no remainder, then it is even. You can calculate the remainder with the modulo operator % like this num % 2 == 0. If a number divided by 2 leaves a remainder of 1, then the number is odd. You can check for this using num % 2 == 1.

Coding:

```
import java.util.*;
class OddEven {
   public static void main(String[] args) {
      Scanner in =new Scanner(System.in);
      System.out.println("Enter The Number:-");
      int num=in.nextInt();
      if(num%2==0){
         System.out.println(num+" is even number");
      }else {
         System.out.println(num+" is Odd Number");
      }
   }
}
```

CHECK IF A NUMBER IS PRIME

To check if a number is prime, you need to determine if it's only divisible by 1 and itself. A number is considered prime if it has exactly two distinct positive divisors: 1 and the number itself.

Coding:

```
import java.util.Scanner;
class Main{
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.println("Enter The Number : -");
      int num=in.nextInt();
      int count=0;
      if(num<2){
         System.out.println("Low Value");
      }
      for(int i=1;i<=num;i++){
         if(num%i==0){
            count+=1;
         }
      }
      if(count>2){
         System.out.println("The Given Number "+num+" is Not Prime Number");
      }else{
         System.out.println("The Given Number " +num+" is Prime Number");
      }
   }
}
```

FIND THE SUM OF DIGITS OF A NUMBER

To find the sum of digits of a number, you separate each digit and then add them together individually. For example, in the number 123, you would separate the digits to get 1, 2, and 3, and then add them: 1 + 2 + 3 = 6.

Here's a more detailed explanation:

- **Separate the Digits:** Identify each individual digit in the number.
- **Add the Digits:** Sum up all the separated digits.
- **Example:**

Number: 786

Separate Digits: 7, 8, 6

Sum of Digits: 7 + 8 + 6 = 21

If the sum of digits is greater than nine, repeat the process: 2 + 1 = 3

Therefore, the digit sum of 786 is 3.

Coding :

```
import java.util.*;
class SumOfDigit{
   public static void main(String SANTHOSH[]) {
      Scanner in = new Scanner(System.in);
      int num;
      int sum=0;
      System.out.println("Enter The Number:-");
      num=in.nextInt();
      while (num!=0){
         sum=sum+(num%10);
         num=num/10;
      }
      System.out.println("The Given Number Sum Of Digit Is : "+sum);
   }
}
```

REVERSE A NUMBER

To reverse a number, you rearrange its digits in the opposite order, like turning 123 into 321. You can achieve this using various methods, including string manipulation, mathematical operations, or built-in functions in programming languages.

Coding:

```
import java.util.*;
class NumberReverse {
    public static void main(String SANTHOSH[]) {
        Scanner in = new Scanner(System.in);
        int num;
        int sum=0;
        System.out.println("Enter The Number:-");
        num=in.nextInt();
        while (num!=0){
            sum=sum*10+num%10;
            num=num/10;
        }
        System.out.println("The Reversed Number Is : "+sum);
    }
}
```

FIND THE FACTORIAL OF A NUMBER

In math, the factorial of a number is the product of all positive integers less than or equal to that number. It is represented by the number followed by an exclamation point. For example, 7! means 1 × 2 × 3 × 4 × 5 × 6 × 7.

How to calculate a factorial

To calculate a factorial, start with the number and multiply it by each previous integer until you reach 1.

For example, 3! is 3 * 2 * 1.

Coding:

```
import java.util.*;
class Factorial{
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.println("Enter The Number:");
      int num=in.nextInt();
      int fact=1;
      for(int i=1;i<=num;i++){
         fact=fact*i;
      }
      System.out.println("The Given Number "+num+" is Factorial is  "+fact);
   }
}
```

Output:

Enter The Number:

10

The Given Number 10 is Factorial is 3628800

SPECIAL NUMBER PROGRAMS

Palindrome Number (121 → same when reversed)

A palindromic number is a number that remains the same when its digits are reversed. If we take the original number and reverse it i.e. 121 and reverse it, we still get 121. We can say more examples like 11, 131, 242 etc which when we reverse we get the same number.

Coding:

```
import java.util.*;
class  Palindrome{
   public static void main(String SANTHOSH[]) {
      Scanner in=new Scanner(System.in);
      System.out.println("Enter The Number :-");
      int num=in.nextInt();
      int palindrome=num;
      int sum=0;
      while (num!=0){
         sum=sum*10+num%10;
         num=num/10;
      }
      System.out.println(sum);
      if(sum==palindrome){
         System.out.println("The Given Number "+sum+" is Palindrome");
      }
      else {
         System.out.println("The Given Number "+sum+ " is Not Palindrome");
      }
   }
}
```

Output:

212

The Given Number 212 is Palindrome

ARMSTRONG NUMBER (153 → $1^3 + 5^3 + 3^3 = 153$)

An Armstrong number is a number that is equal to the sum of its digits raised to the power of the number of digits. For example, 153 is an Armstrong number because $1^3 + 5^3 + 3^3 = 153$.

Examples of Armstrong numbers: 153, 370, and 371.

Coding

```
import java.util.*;
class Main{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the Number");
        int num= in.nextInt();
        int real=num;
        int temp ;
        int cheack=0;
        while (num>0){
            temp=num%10;
            cheack=cheack+(temp*temp*temp);
            num=num/10;
        }
        if(real==cheack){
            System.out.println(cheack+" it is a Armstrong Number");
        }else {
            System.out.println(cheack+" it is Not a Armstrong Number");
        }
    }
}
```

Out Put:

Enter the Number

370

370 it is a Armstrong Number

PERFECT NUMBER (28 → SUM OF FACTORS = 28)

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For example, 6 is a perfect number because 1 + 2 + 3 = 6.

Examples of perfect numbers: 6, 28, 496, and 8128.

Coding:

```
import java.util.*;
class PerfectNumber {
   public static void main(String[] Santhosh) {
      Scanner in = new Scanner(System.in);
      int num = in.nextInt();
      int sum=0;
      for(int i =1;i<=num/2;i++){
         if(num%i==0){
            sum+=i;
         }
      }
      if(sum==num){
         System.out.println("The Given Number "+sum+" is Perfect Number");
      }else {
         System.out.println("The Given Number "+sum+" is Not Perfect Number");
      }
   }
}
```

STRONG NUMBER (145 → SUM OF FACTORIAL OF DIGITS = 145)

A strong number is one in which the factorial of each digit of a number equals the original number's sum. Example of strong number in C: 145 is a strong number. To see if 145 is a strong number, we must compute and add the factorials of its digits:

1! + 4! + 5! = 1 + 24 + 120 = 145

Coding:

```
import java.util.*;
class StrongNumber {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the Number:");
        int num = in.nextInt();
        int rem = 0;
        int sum = 0;
        int num2 = num;
        int fact = 0;
        while (num != 0) {
            rem = num % 10;
            fact = 1;
            for (int i = 1; i <= rem; i++) {
                fact = fact * i;
            }
            sum = sum + fact;
            num = num / 10;
        }
            if (sum == num2) {
                System.out.println("The Given Number "+sum+ " is Strong Number");
            } else {
                System.out.println("The Given Number "+num2 + " is NOT Strong Number");

            }
    }
}
```

HARSHAD NUMBER (18 → 18 IS DIVISIBLE BY 1+8 = 9)

A Harshad number is a positive integer where the number itself is exactly divisible by the sum of its digits.

Examples:

12 is a Harshad number because 1 + 2 = 3, and 12 is divisible by 3.

18 is a Harshad number because 1 + 8 = 9, and 18 is divisible by 9.

Coding:

```
import java.util.*;
class HarshadNumber{
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.println("Enter The Number:");
      int num=in.nextInt();
      int num1=num;
      int sum=0;
      int rem;
      while (num1!=0){
        rem=num%10;
         sum=sum+rem;
         num1=num1/10;
      }
      if(num%sum==0){
         System.out.println("The Given Number "+num+" is Harshad Number");
      }else {
         System.out.println("The Given Number "+num+" is NoT Harshad Number");
      }
   }
}
```

MATHEMATICAL SERIES PROGRAMS:

Fibonacci Series

```
import java.util.*;
class Fibonacci{
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int num;
      int a=0,b=1;
      System.out.println(a+" , "+b+" , ");
      num=in.nextInt();
      int nextTerm;
      for(int i=2;i<num;i++){
         nextTerm=a+b;
         a=b;
         b=nextTerm;
         System.out.print(nextTerm+", ");
      }
   }
}
```

THAT'S A LOT OF PROGRAMS! I'LL WRITE A SI -UNDERSTAND JAVA CODE COVERING ALL THE NUMB FOR. THE CODE WILL BE STRUCTURED S FOR CLARITY.

JAVA NUMBER

Coding:

```java
import java.util.*;

public class Main {

    // 1. Check if a number is Even or Odd
    static void checkEvenOdd(int num) {
        if (num % 2 == 0) System.out.println(num + " is Even");
        else System.out.println(num + " is Odd");
    }

    // 2. Check if a number is Prime
    static boolean isPrime(int num) {
        if (num < 2) return false;
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) return false;
        }
        return true;
    }

    // 3. Sum of digits of a number
    static int sumOfDigits(int num) {
        int sum = 0;
        while (num > 0) {
            sum += num % 10;
            num /= 10;
        }
        return sum;
    }

    // 4. Reverse a number
    static int reverseNumber(int num) {
        int rev = 0;
        while (num > 0) {
            rev = rev * 10 + num % 10;
```

```
        num /= 10;
    }
    return rev;
}

// 5. Factorial of a number (Recursion)
static int factorial(int num) {
    return (num == 0 || num == 1) ? 1 : num * factorial(num - 1);
}

// 6. Check Palindrome Number
static boolean isPalindrome(int num) {
    return num == reverseNumber(num);
}

// 7. Check Armstrong Number
static boolean isArmstrong(int num) {
    int sum = 0, temp = num, digits = String.valueOf(num).length();
    while (temp > 0) {
        sum += Math.pow(temp % 10, digits);
        temp /= 10;
    }
    return sum == num;
}

// 8. Check Perfect Number
static boolean isPerfect(int num) {
    int sum = 0;
    for (int i = 1; i < num; i++) {
        if (num % i == 0) sum += i;
    }
    return sum == num;
}
```

```java
    // 9. Fibonacci Series up to n terms
    static void fibonacci(int n) {
        int a = 0, b = 1;
        for (int i = 0; i < n; i++) {
            System.out.print(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
        System.out.println();
    }

    // 10. Decimal to Binary
    static String decimalToBinary(int num) {
        return Integer.toBinaryString(num);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();

        checkEvenOdd(num);
        System.out.println("Is Prime: " + isPrime(num));
        System.out.println("Sum of Digits: " + sumOfDigits(num));
        System.out.println("Reversed Number: " + reverseNumber(num));
        System.out.println("Factorial: " + factorial(num));
        System.out.println("Is Palindrome: " + isPalindrome(num));
        System.out.println("Is Armstrong: " + isArmstrong(num));
        System.out.println("Is Perfect: " + isPerfect(num));
        System.out.print("Fibonacci Series (first 10 terms): "); fibonacci(10);
        System.out.println("Binary Representation: " + decimalToBinary(num));
    }
}
```