1. Positive Or Negetive.
2. Even Or Odd.
3. Sum of First N Natural Number.
4. Sum of N Natural Number.
5. Sum of Number in a given range.
6. Greatest of the Two Number.
7. Greatest of the Three Number.
8. Leap Year or Not.
9. Prime Number.
10. Prime Number Within given Range.
11. Sum of digits of a Number.
12. Reverse of a Number.
13. Palindrom Number.
14. Armstrong Number.
15. Armstrong Number with in a given range.
16. Fibonacci Series upto nth term.
17. Find the Nth term Fibonacci Series.
18. Factorial of a Number.
19. Power Of a Number.
20. Factor of a Number.
21. Finding Prime Factors of a Number.
22. Strong Number.
23. Perfect Number.
24. Perfect Square.
25. Automorphic Number.
26. Harshad Number.
27. Abundant Number.
28. Frindly Pair.
29. Positive Or Negetive.

**class Q1\_Positive\_Or\_Negetive**

**{**

public static void main (String args [])

{

int num = 50;

positive\_Negetive(num);

}

**static void positive\_Negetive(int num)**

**{**

If (num == 0)

System.out.println("Number is Zero");

else if (num > 0)

System.out.println("Number is Positive");

else

System.out.println("Number is Negetive");

**}**

**}**

--------------------------------------------------------------------------------------------------------------

1. Even Or Odd.

**class Q2\_Even\_Or\_Odd**

**{**

public static void main(String args[])

{

int num = 51;

even\_Odd1(num);

even\_Odd2(num);

}

**static void even\_Odd1(int num)**

**{**

if(num % 2 == 0)

System.out.println("Number is Even");

else

System.out.println("Number is Odd");

**}**

**static void even\_Odd2(int number)**

**{**

if ((number & 1) == 1)

System.out.println ("Number is Odd");

else

System.out.println ("Number is Even");

**}**

**}**

1. Sum of First N Natural Number.

**class Q3\_Sum\_of\_First\_N\_Natural\_Number**

**{**

public static void main(String args[])

{

int num = 5;

sum\_First\_N\_Natural\_Number1(num);

sum\_First\_N\_Natural\_Number2(num);

sum\_First\_N\_Natural\_Number3(num, 0);

}

//Method 1: Using for Loop

**static void sum\_First\_N\_Natural\_Number1(int num)**

**{**

int count = 0;

for(int i = 1; i <= num; i++)

count+= i;

System.out.println("Sum of First N Natural Number : " + count);

**}**

// Method 2: Using Formula for the Sum of Nth Term

**static void sum\_First\_N\_Natural\_Number2(int num)**

**{**

int count = ( num \* (num+1) / 2 );

System.out.println("Sum of First N Natural Number : " + count );

**}**

// Method 3: Using Recursion

**static void sum\_First\_N\_Natural\_Number3(int num, int count)**

**{**

if(num == 0)

{

System.out.println("Sum of First N Natural Number : " + count );

return;

}

sum\_First\_N\_Natural\_Number3(num-1, count+num);

**}**

**}**

1. Sum of N Natural Number.

**class Q4\_Sum\_of\_N\_Natural\_Number**

**{**

public static void main(String args[])

{

int num = 5;

sum\_N\_Natural\_Number1(num);

sum\_N\_Natural\_Number2(num);

sum\_N\_Natural\_Number3(num, 0);

}

//Method 1: Using for Loop

**static void sum\_N\_Natural\_Number1(int num)**

**{**

int count = 0;

for(int i = 1; i <= num; i++)

count+= i;

System.out.println("Sum of First N Natural Number : " + count);

**}**

// Method 2: Using Formula

**static void sum\_N\_Natural\_Number2(int num)**

**{**

int count = ( num \* (num+1) / 2 );

System.out.println("Sum of First N Natural Number : " + count );

**}**

// Method 3: Using Recursion

**static void sum\_N\_Natural\_Number3(int num, int count)**

**{**

if(num == 0)

{

System.out.println("Sum of First N Natural Number : " + count );

return;

}

sum\_N\_Natural\_Number3(num-1, count+num);

}

}

1. Sum of Number in a given range.

**class Q5\_Sum\_of\_N\_Natural\_Number\_With\_In\_Range**

**{**

public static void main(String args[])

{

int num1 = 5, num2 = 10;

sum\_In\_Range1(num1, num2);

sum\_In\_Range2(num1, num2);

sum\_In\_Range3(num1, num2, 0);

}

// Method 1: Using Brute Force / Loop

**static void sum\_In\_Range1(int num1, int num2)**

**{**

int count = 0;

for(int i = num1; i <= num2; i++)

count+= i;

System.out.println("Sum of N Natural Number in Range : " + count);

**}**

// Method 2: Using the Formula

**static void sum\_In\_Range2(int num1, int num2)**

**{**

int count1 = ( num1\*(num1+1) / 2 );

int count2 = ( num2\*(num2+1) / 2 );

System.out.println("Sum of N Natural Number in Range : " + (count2 - count1 + num1 ) );

**}**

// Method 3: Using Recursion

**static void sum\_In\_Range3(int num1, int num2, int count)**

**{**

if(num1 > num2)

{

System.out.println("Sum of N Natural Number in Range : " + count);

return;

}

sum\_In\_Range3(num1+1, num2, (count+num1) );

**}**

**}**

1. Greatest of the Two Number.

**class Q6\_Greatest\_Two\_Number**

**{**

public static void main(String args[])

{

int num1 = 150, num2 = 50;

greatest\_Two\_Number1(num1, num2);

greatest\_Two\_Number2(num1, num2);

greatest\_Two\_Number3(num1, num2);

}

// Method 1: Using if-else Statements

static void greatest\_Two\_Number1(int num1, int num2)

{

if(num1 == num2)

System.out.println(num1 + " Both Are Equal");

else if(num1 > num2)

System.out.println(num1 + " : is a greatest.");

else

System.out.println(num2 + " : is a greatest.");

}

// Method 2: Using Ternary Operator

static void greatest\_Two\_Number2(int num1, int num2)

{

int temp = 0;

if(num1 == num2)

System.out.println(num1 + " Both Are Equal");

else

{ temp = num1 > num2 ? num1 : num2;

System.out.println(temp + " : is a greatest.");

}

}

// Method 3: Using inbuilt max Function

static void greatest\_Two\_Number3(int num1, int num2)

{

if(num1 == num2)

System.out.println(num1 + " Both Are Equal");

else

System.out.println( Math.max(num1, num2) + " : is a greatest.");

}

}

1. Greatest of the Three Number.

**class Q7\_Greatest\_Three\_Number**

**{**

public static void main(String args[])

{

int num1 = 150;

int num2 = 50;

int num3 = 200;

greatest\_Three\_Number(num1, num2, num3);

}

// Method 1: Using if-else Statements

**static void greatest\_Three\_Number(int num1, int num2, int num3)**

**{**

if(num1 == num2 && num1 == num3 )

System.out.println(num1 + " Both Are Equal");

else if(num1 > num2 && num1 > num3 )

System.out.println(num1 + " : is a greatest.");

else if(num2 > num1 && num2 > num3 )

System.out.println(num2 + " : is a greatest.");

else

System.out.println(num3 + " : is a greatest.");

**}**

**}**

1. Leap Year or Not.

class Q8\_Leap\_Year\_or\_Not

{

public static void main(String args[])

{

int num = 2020;

leap\_Year\_Or\_Not1(num);

leap\_Year\_Or\_Not2(num);

}

// Method 1: Using if-else Statements - 1

**static void leap\_Year\_Or\_Not1(int num)**

**{**

if( num % 400 == 0)

System.out.println(num + " is a Leap year");

else if( (num%4 == 0) && (num%100 != 0) )

System.out.println(num + " is a Leap year");

else

System.out.println(num + " is Not a Leap year");

**}**

// Method 2: Using if-else Statements - 2

**static void leap\_Year\_Or\_Not2(int num)**

**{**

if( (num % 400 == 0) || (num%4 == 0) && (num%100 != 0) )

System.out.println(num + " is a Leap year");

else

System.out.println(num + " is Not a Leap year");

**}**

**}**

1. Prime Number.

**class Q9\_prime\_Number**

**{**

    public static void main**(**String args**[])**

**{**

        int num **=** 317**;**

        check\_Prime\_Number1**(**num**);**

        check\_Prime\_Number2**(**num**);**

        check\_Prime\_Number3**(**num**);**

**}**

    // Method 1: Simple iterative solution

**static void check\_Prime\_Number1(int num)**

**{**

**if(**num **<** 2**)**

**{**

            System**.**out**.**println **(** num **+** " is not prime"**);**

            System**.**exit**(**0**);**

**}**

      int count **=** 0**;**

**for(** int i **=** 2**;** i **<=** num**;** i**++)**

**{**

**if(**num**%**i **==** 0**)**

                count **+=** 1**;**

**}**

**if(** count **>** 2**)**

            System**.**out**.**println **(** num **+** " is not prime"**);**

**else**

            System**.**out**.**println **(**num **+** " is prime"**);**

**}**

    // Method 2: Optimization by break condition

**static void check\_Prime\_Number2(int num)**

**{**

        boolean temp **=** **true;**

**if(** num **<** 2**)**

**{**

            temp **=** **false;**

**}**

**for(** int i **=** 2**;** i **<** num**;** i**++)**

**{**

**if(** num**%**i **==** 0**)**

**{**

                temp **=** **false;**

**break;**

**}**

**}**

**if(**temp**)**

            System**.**out**.**println **(**num **+** " is prime"**);**

**else**

            System**.**out**.**println **(** num **+** " is not prime"**);**

**}**

    // Method 3: Basic Recursion technique

**static void check\_Prime\_Number3(int num)**

**{**

**if(**num **==** 2**)**

**{**

            System**.**out**.**println**(**num **+** " is Prime."**);**

**return;**

**}**

        int i **=** 2**;**

        boolean temp **=** **true;**

        temp **=** check\_Prime**(**num**,** i**);**

**if(**temp**)**

**{**

            System**.**out**.**println**(**num **+** " is Prime."**);**

**}**

**else**

**{**

            System**.**out**.**println**(**num **+** " is not Prime."**);**

**}**

**}**

    static boolean check\_Prime**(**int num**,** int i**)**

**{**

**if(** num **==** i**)**

**return** **true;**

**if(**num **<** 2**)**

**return** **false;**

**if(**num **%** i **==** 0**)**

**{**

**return** **false;**

**}**

**return** check\_Prime**(**num**,** i**+=**1**);**

**}**

**}**

1. Prime Number Within given Range.

**class Q10\_prime\_Number\_With\_In\_Range**

**{**

public static void main(String args[])

{

int num1 = 1, num2 = 100;

prime\_With\_In\_Range1(num1, num2);

prime\_With\_In\_Range2(num1, num2);

}

// Method 1: Using inner loop Range as [2, number-1].

**static void prime\_With\_In\_Range1(int num1, int num2)**

**{**

if( num1 == num2)

return;

if(num1 == 1)

num1 += 1;

boolean temp = true;

for(int i = num1; i <= num2; i++)

{

temp = true;

for( int j = 2; j < i; j++)

{

if( i % j == 0)

{

temp = false;

break;

}

}

if(temp)

System.out.print( i + " ");

}

System.out.println();

}

// Method 2: Using Function as [2, number-1]. Optimization Code

static void prime\_With\_In\_Range2(int num1, int num2)

{

if(num1 == 1)

{

num1++;

}

for(int i = num1; i <= num2; i++)

{

if( check\_Prime(i) )

System.out.print(i + " ");

}

**}**

**static boolean check\_Prime(int num)**

**{**

for( int j = 2; j < num; j++)

{

if( num % j == 0)

{

return false;

}

}

return true;

**}**

**}**

1. Sum of digits of a Number.

**class Q11\_Sum\_Of\_Digits**

**{**

public static void main(String args[])

{

int num = 1234;

System.out.println("Sum of Digits : " + sum\_Of\_Digits(num) );

}

static int sum\_Of\_Digits(int num)

{

int count = 0;

while(num != 0)

{

count = count + num%10;

num = num/10;

}

return count;

**}**

**}**

1. Reverse of a Number.

**class Q12\_Reverse\_Number**

**{**

public static void main(String args[])

{

int num = 123456;

System.out.println("Sum of Digits : " + reverse\_Digits(num) );

}

**static int reverse\_Digits(int num)**

**{**

int temp = 0;

while(num != 0)

{

temp = temp\*10 + num%10;

num = num/10;

}

return temp;

**}**

**}**

1. Palindrom Number.

**class Q13\_Palindrom\_Number**

**{**

public static void main(String args[])

{

int num = 123454321;

if( palindrom\_Number(num) )

System.out.println("Yes, This is Palindrom");

else

System.out.println("No , This is Palindrom");

}

**static boolean palindrom\_Number(int num)**

**{**

int temp = num;

int count = 0;

while(num != 0)

{

count = count\*10 + num%10;

num = num/10;

}

if(temp == count)

return true;

return false;

**}**

**}**

1. Armstrong Number.

**class Q14\_Armstrong\_Number**

**{**

public static void main(String args[])

{

int num = 371;

if( armstrong\_Number(num) )

System.out.println("Yes, This is Armstrong");

else

System.out.println("No , This is Armstrong");

}

**static boolean armstrong\_Number(int num)**

**{**

int temp = num;

int count = 0;

int a = 1;

while(num != 0)

{

a = num%10;

count += a\*a\*a;

num = num/10;

}

if(temp == count)

return true;

return false;

**}**

**}**

1. Armstrong Number with in a given range.

**class Q15\_Armstrong\_With\_In\_Range**

**{**

public static void main(String args[])

{

int num1 = 1;

int num2 = 1000;

armstrong\_With\_In\_Range(num1, num2);

}

**static void armstrong\_With\_In\_Range(int num1, int num2)**

**{**

int num = 0, count = 0;

for(int i = num1; i <= num2; i++)

{

count = 0;

num = i;

while(num != 0)

{

int a = num%10;

count += a\*a\*a;

num = num/10;

}

if( i == count)

System.out.print( i + ", ");

}

**}**

**}**

1, 153, 370, 371, 407,

1. Fibonacci Series upto nth term.

**class Q16\_Fibonacci\_Series**

**{**

public static void main(String args[])

{

int index = 10;

fibonacci\_Series(index);

}

**static void fibonacci\_Series(int index)**

**{**

int num1 = 1;

int num2 = -1;

for(int i = 1; i <= index; i++)

{

int temp = num1+num2;

num2 = num1;

num1 = temp;

System.out.print( num1 + " ");

}

**}**

**}**

**Output : 0 1 1 2 3 5 8 13 21 34**

1. Find the Nth term Fibonacci Series.

**class Q17\_Fibonacci\_Series\_nth\_Term**

**{**

public static void main(String args[])

{

int index = 15;

fibonacci\_Series(index);

}

**static void fibonacci\_Series(int index)**

**{**

int num1 = 1;

int num2 = -1;

for(int i = 1; i <= index; i++)

{

int temp = num1+num2;

num2 = num1;

num1 = temp;

System.out.print( num1 + " ");

}

**}**

**}**

Output : 0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 , 144 , 233 , 377

1. Factorial of a Number.

Note : 0! = 1 and 1! = 1

Example : 5! = 1 \* 2 \* 3 \* 4 \* 5 = 120

**class Q18\_Factoria\_Of\_Number**

**{**

public static void main(String args[])

{

int index = 5;

factoria\_Of\_Number(index);

}

**static void factoria\_Of\_Number(int n)**

**{**

int count = 1;

for(int i = 2; i <= n; i++)

{

count \*= i;

}

System.out.print("Factoria Of Number : " + count);

**}**

**}**

1. Power Of a Number.

**class Q19\_Power\_of\_Number**

**{**

public static void main(String args[])

{

int num = 25;

int power = 2;

power\_of\_Number( num, power);

}

**static void power\_of\_Number(int n, int power)**

**{**

int count = 1;

for(int i = 1; i <= power; i++)

{

count \*= n;

}

System.out.print("Power Of Number : " + count);

**}**

**}**

1. Factor of a Number.

Input : 10

Output : 1, 2, 5

**class Q20\_Factor\_Of\_Number**

**{**

public static void main(String args[])

{

int num = 10;

factor\_Of\_Number( num );

}

**static void factor\_Of\_Number(int n)**

**{**

int i = 2;

if( n > 0)

{

System.out.print( 1 + " ");

}

while(n >= i)

{

if( n%i == 0)

{

System.out.print( i + " ");

}

i++;

}

**}**

**}**

1. Finding Prime Factors of a Number.

**class Q21\_Prime\_Factors\_Of\_Number**

**{**

public static void main(String args[])

{

int num = 90; //Output : 2 3 3 5

prime\_Factors\_Of\_Number( num );

}

**static void prime\_Factors\_Of\_Number(int n)**

**{**

int i = 2;

while(n != 1)

{

if( isPrime(i) && n%i == 0)

{

System.out.print( i + " ");

n = n/i;

}

else

i++;

}

**}**

**static boolean isPrime(int num)**

**{**

boolean temp = true;

if(num < 2)

temp = false;

for( int i = 2; i < num; i++)

{

if(num%i == 0)

{

temp = false;

break;

}

}

return temp;

}

}

1. Strong Number.

145 = 1! + 4! + 5!

1 + 24 + 120

Output : 145.

**class Q22\_Strong\_Number**

**{**

public static void main(String args[])

{

int num = 145;

strong\_Number( num );

}

**static void strong\_Number(int n)**

**{**

int temp = n, count = 0;

while(n != 0)

{

count += factorial\_Number(n%10);

n = n/10;

}

if(temp == count)

System.out.println("Yes, Strong Number.");

else

System.out.println("No , Strong Number.");

**}**

**static int factorial\_Number(int n)**

**{**

int count = 1;

for(int i = 2; i <= n; i++)

count \*= i;

return count;

}

}

1. Perfect Number.

**class Q23\_Perfact\_Number**

**{**

public static void main(String args[])

{

int num = 6; //28

perfact\_Number( num );

}

**static void perfact\_Number(int n)**

**{**

int count = 0;

for(int i = 1; i < n; i++)

{

if( n % i == 0)

count += i;

}

if( n == count)

System.out.println("Yes, Perfact Number.");

else

System.out.println("No , Perfact Number.");

**}**

**}**

1. Perfect Square.

**class Q24\_Perfect\_Square**

**{**

public static void main(String args[])

{

int num = 64;

perfect\_Square( num );

}

**static void perfect\_Square(int n)**

**{**

if( n <= 0)

{

System.out.println("No , A Perfact Square.");

return;

}

int temp = (int)Math.sqrt(n);

if( (temp\*temp) == n)

System.out.println("Yes, A Perfact Square.");

**}**

**}**

1. Automorphic Number.

5 =(5)2 = 25

6 = (6)2 = 36

25 = (25)2 = 625

76=(76)2 = 5776

**class Q25\_Automorphic\_Number**

**{**

public static void main(String args[])

{

int num = 25;

automorphic\_Number( num );

}

**static void automorphic\_Number(int n)**

**{**

int temp = n\*n;

if( temp % 10 == n)

System.out.println("Yes, A Automorphic Number.");

else

System.out.println("No , A Automorphic Number.");

**}**

**}**

1. Harshad Number.

Input : 21

count = 2 + 1;

(21 % 3 == 0) True

OutPut : Yes, This is Harshad Number

Input : 153 OutPut : Yes, This is Harshad Number

Input : 4991 OutPut : Yes, This is Harshad Number

**class Q26\_Harshad\_Number**

**{**

public static void main(String args[])

{

int num = 153;

harshad\_Number( num );

}

static void harshad\_Number(int n)

{

int value = n;

int temp = 0;

while(n != 0)

{

temp += n%10;

n = n/10;

}

if( value % temp == 0)

System.out.println("Yes, A Harshad Number.");

else

System.out.println("No , A Harshad Number.");

}

}

1. Abundant Number.

The Factors of number 18 are = 1, 2, 3, 6 and 9. (don't include number itself)

1 + 2 + 3 + 6 + 9 = 21

as the number 21>18

**class Q27\_Abundant\_Number**

**{**

public static void main(String args[])

{

int num = 12;

abundant\_Number( num );

}

**static void abundant\_Number(int n)**

**{**

int sum = 0;

for( int i = 1; i < n; i++)

{

if(n%i == 0)

sum += i;

}

if (sum > n)

{

System.out.println (n + " is an Abundant Number.\n

The Abundance is: " + (sum - n));

}

else

{

System.out.println (n + " is not an Abundant Number");

}

**}**

**}**