MethodReference

1. Define your own class **with** an instance method "int factorial(int n)" which should return the factorial of the given input "n".

Define your own functional interface **to** refer this instance method and invoke it to get the factorial result.

**CODE:**

@FunctionalInterface

interface FindFactorial{

    int compute(int n);

}

class Factorial{

    int findFact(int num){

        if(num<=1)

        return 1;

        else

        return num\*findFact(num - 1);

    }

}

public class FindFact {

    public static void main(String args[]){

        Factorial fact = new Factorial();

        FindFactorial ref = fact::findFact;

        int result = ref.compute(5);

        System.out.println("Factorial is : "+ result);

    }

}

**OUTPUT:**

Factorial is: 120

1. Define your own class with a static method "int digitCount(int n)" which should return the number of digits in a given input "n".

Define your own functional interface to refer this static method and invoke it to get the number of digits.

**CODE:**@FunctionalInterface

interface Count{

    int Countdigits(int n);

}

class DigitsCount{

    static int digitcount(int n){

        if(n==1) return 1;

        int count = 0;

        while(n != 0){

            n = n / 10;

            count++;

            }

        return count;

    }

}

public class DigitCount {

    public static void main(String args[]){

        Count c = DigitsCount::digitcount;

        int result = c.Countdigits(4242);

        System.out.println("No of digits :  "+ result);

    }

}

**OUTPUT:**

No of digits : 4

1. Define your own class and a parameterized constructor with one integer argument. It should check the argument and display "Prime" or "Not Prime".

Define your own functional interface to refer this constructor and invoke it to check whether the given number is Prime or Not.

**CODE:**

@FunctionalInterface

interface PrimeChecker {

    PrimeClass check(int num);

}

class PrimeClass {

    PrimeClass(int n) {

        if (isPrime(n)) {

            System.out.println(n + " is Prime");

        } else {

            System.out.println(n + " is Not Prime");

        }

    }

    boolean isPrime(int n) {

        if (n <= 1) return false;

        for (int i = 2; i <= Math.sqrt(n); i++) {

            if (n % i == 0) return false;

        }

        return true;

    }

}

public class PrimeOrNotUsingCOnstructor {

    public static void main(String[] args) {

        PrimeChecker checker = PrimeClass::new;

        checker.check(7);

        checker.check(12);

    }

}

**OUTPUT:**

7 is Prime

12 is Not Prime