```
====================MathOperation.java============
package com.ameya.interfaces;

public interface MathOperation {
    int operation(int a,int b);
}
====================GreetingService.java===============
package com.ameya.interfaces;

public interface GreetingService {
    void sayMessage(String message);
}
==================TestLambda.java=====================
package com.ameya.test;

import com.ameya.interfaces.GreetingService;
import com.ameya.interfaces.MathOperation;

public class TestLambda {
    private int operate(int a,int b,MathOperation mathOperation) {
        return mathOperation.operation(a, b);
    }

    public static void main(String[] args) {
        TestLambda tester=new TestLambda();
        MathOperation addition = (int a,int b) -> a+b;
        MathOperation subtraction = (a,b) -> a-b;
        MathOperation multiplication = (int a, int b) -> {
            int c=a*b;
            return c;
        };
        MathOperation division = (int a,int b) -> a/b;

        int result =0;
        result=tester.operate(10, 5, addition);
        System.out.println("RESULT :: "+result);
        result=tester.operate(10, 5, subtraction);
        System.out.println("RESULT :: "+result);
        result=tester.operate(10, 5, multiplication);
        System.out.println("RESULT :: "+result);
        result=tester.operate(10, 5, division);
        System.out.println("RESULT :: "+result);

        GreetingService gs1 = message -> {
            String uppercaseMsg=message.toUpperCase();
            System.out.println("Hello "+uppercaseMsg);
        };
        GreetingService gs2 = (message) -> System.out.println("Hello "+message);
        gs1.sayMessage("ameya");
        gs2.sayMessage("Avani");

    }

}
====================================TestStreamMethods.java=========================
package com.ameya.test;

import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

public class TestStreamMethods {

    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(2,3,4,5,2);
        List<Integer> squares=numbers.
                                  stream().
                                  map(x -> x*x).
                                  collect(Collectors.toList());
        System.out.println("==========Displaying Squares LIST===========");
        System.out.println(squares);

```

```
74          List<String> names=Arrays.asList("Reflection","Collection","Stream","String");
75          List<String> result = names.
76                               parallelStream().
77                               filter(s -> s.startsWith("S")).
78                               collect(Collectors.toList());
79          System.out.println("=============Displaying Filtered Strings");
80          System.out.println(result);
81
82          Set<Integer> intSet=numbers.
83                  stream().
84                  map(x -> x*x).
85                  collect(Collectors.toSet());
86          System.out.println("==========Displaying Squares SET===========");
87          System.out.println(intSet);
88
89          System.out.println("=========Printing the numbers list========");
90          numbers.stream().map(x -> x*x).forEach(y->System.out.println(y));
91
92          int sum=numbers.parallelStream().
93                  filter(x -> x%2==0).reduce(0, (ans,i)->ans+i);
94          System.out.println("SUM :: "+sum);
95      }
96
97  }
98
99
```