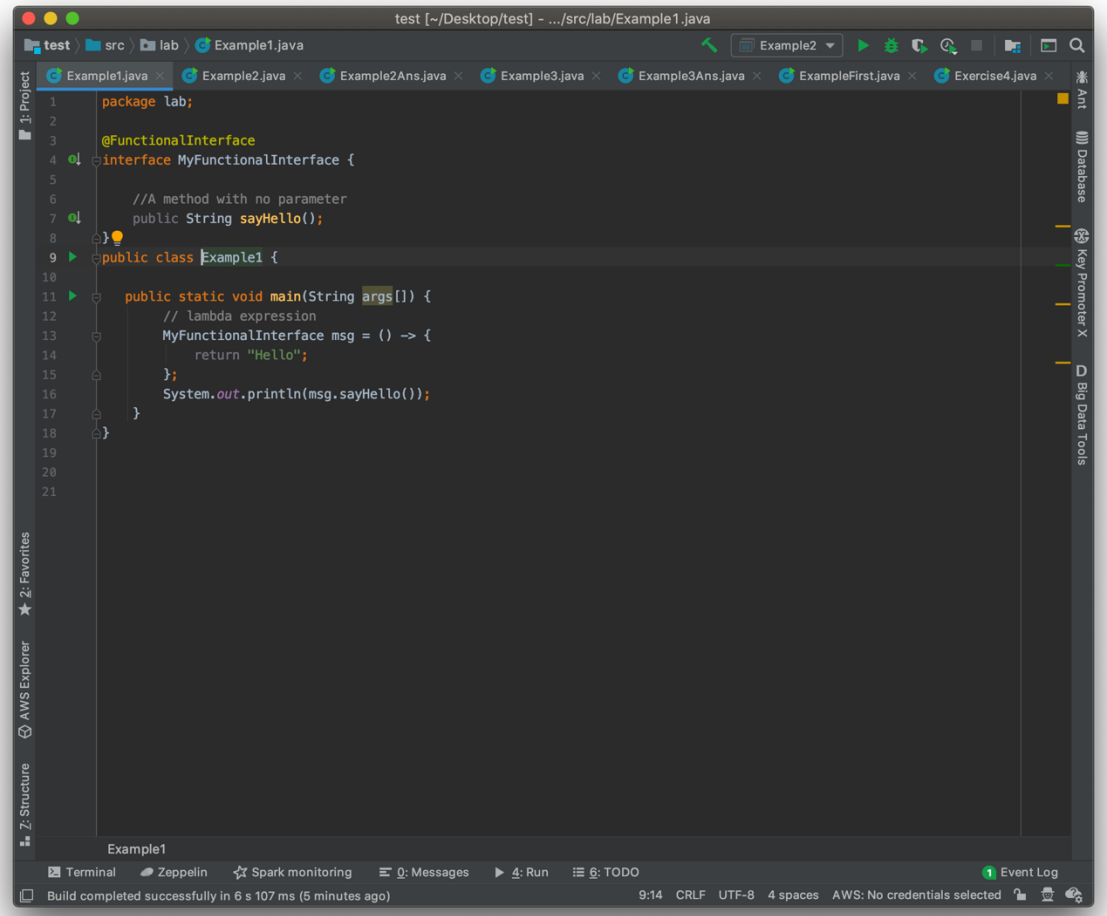
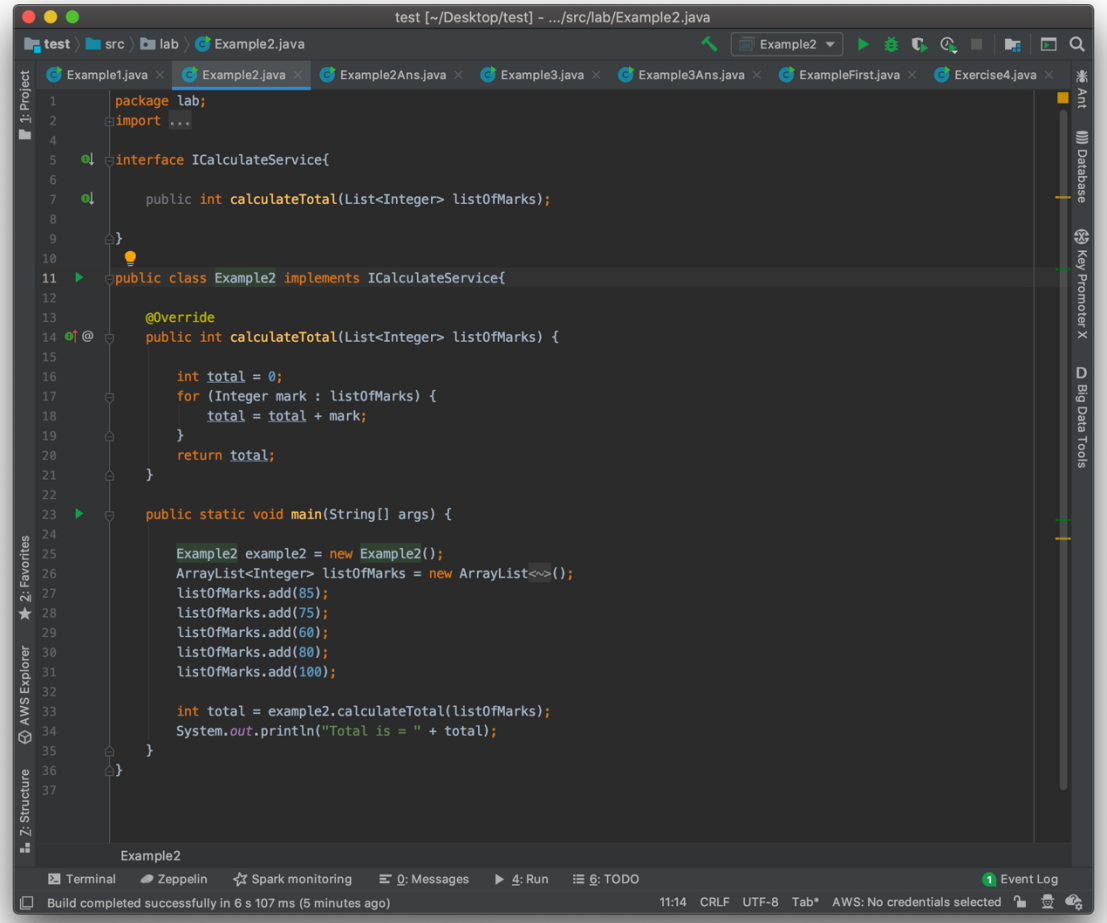
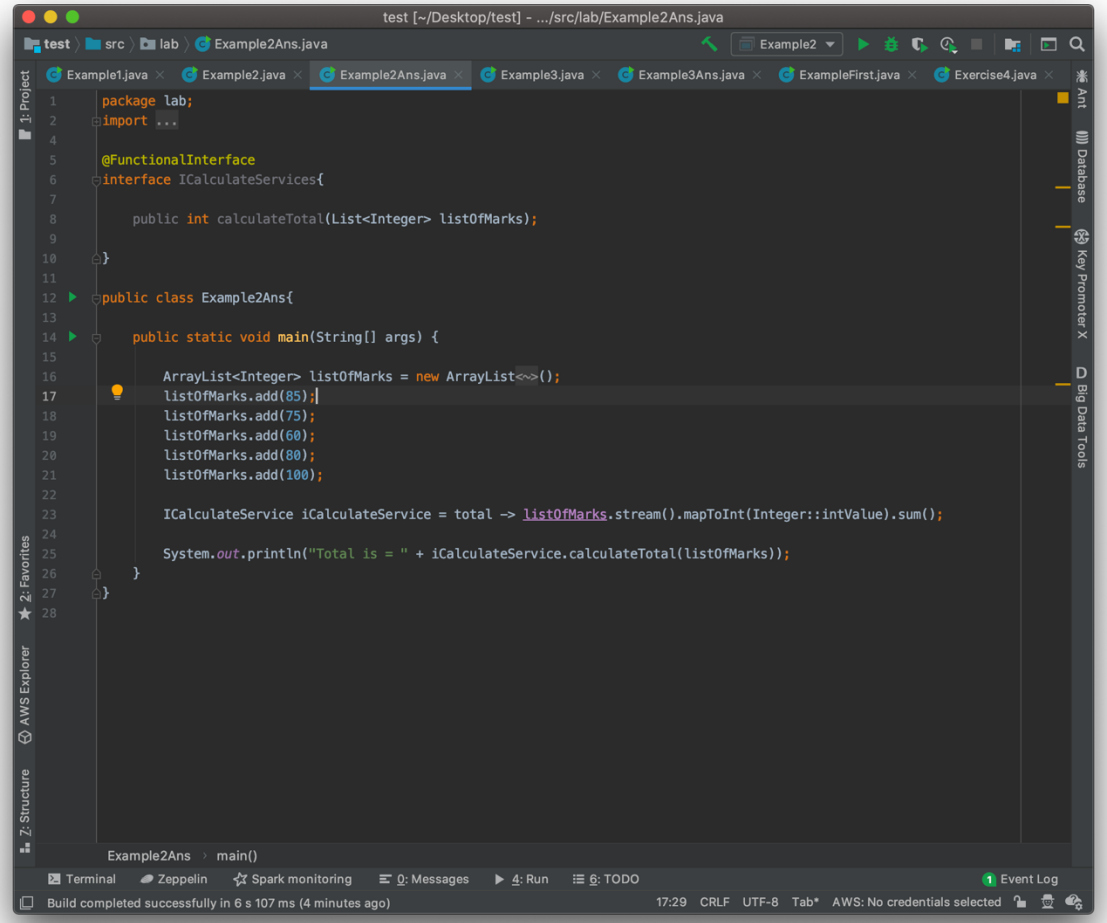
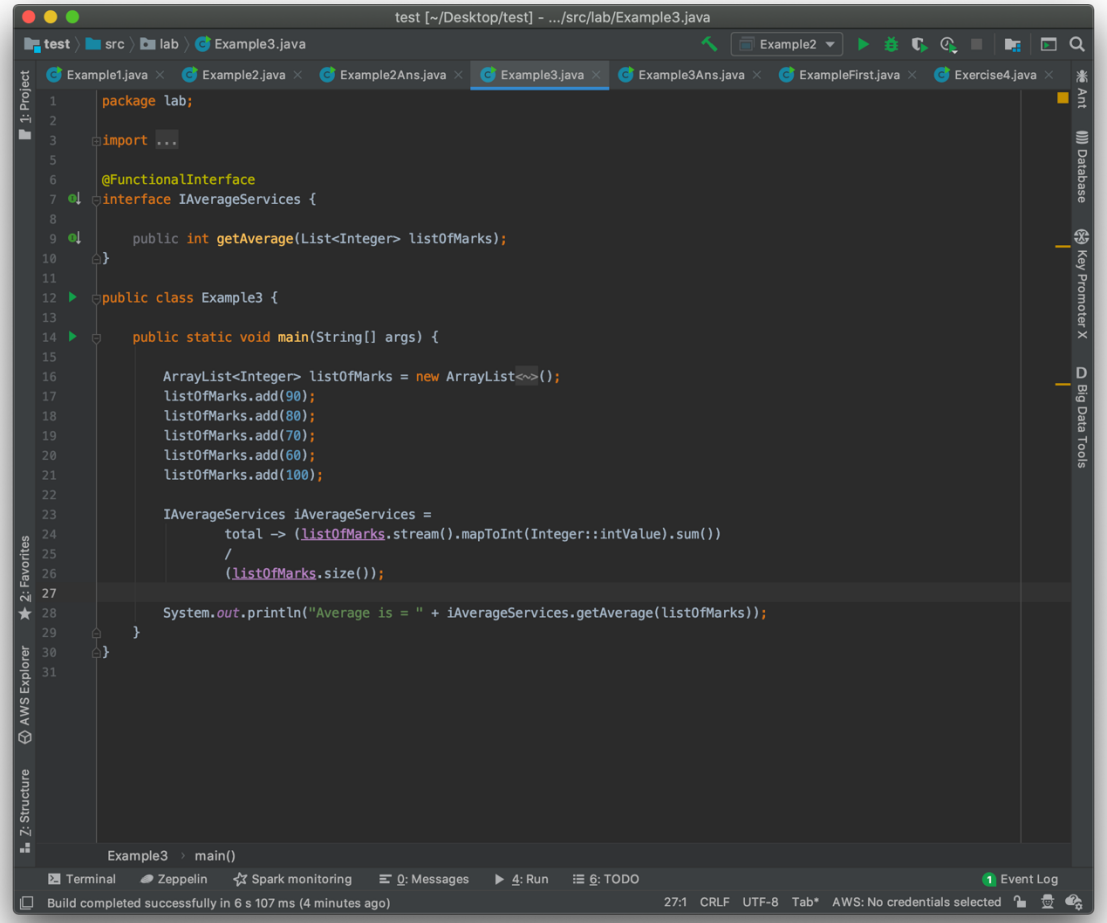
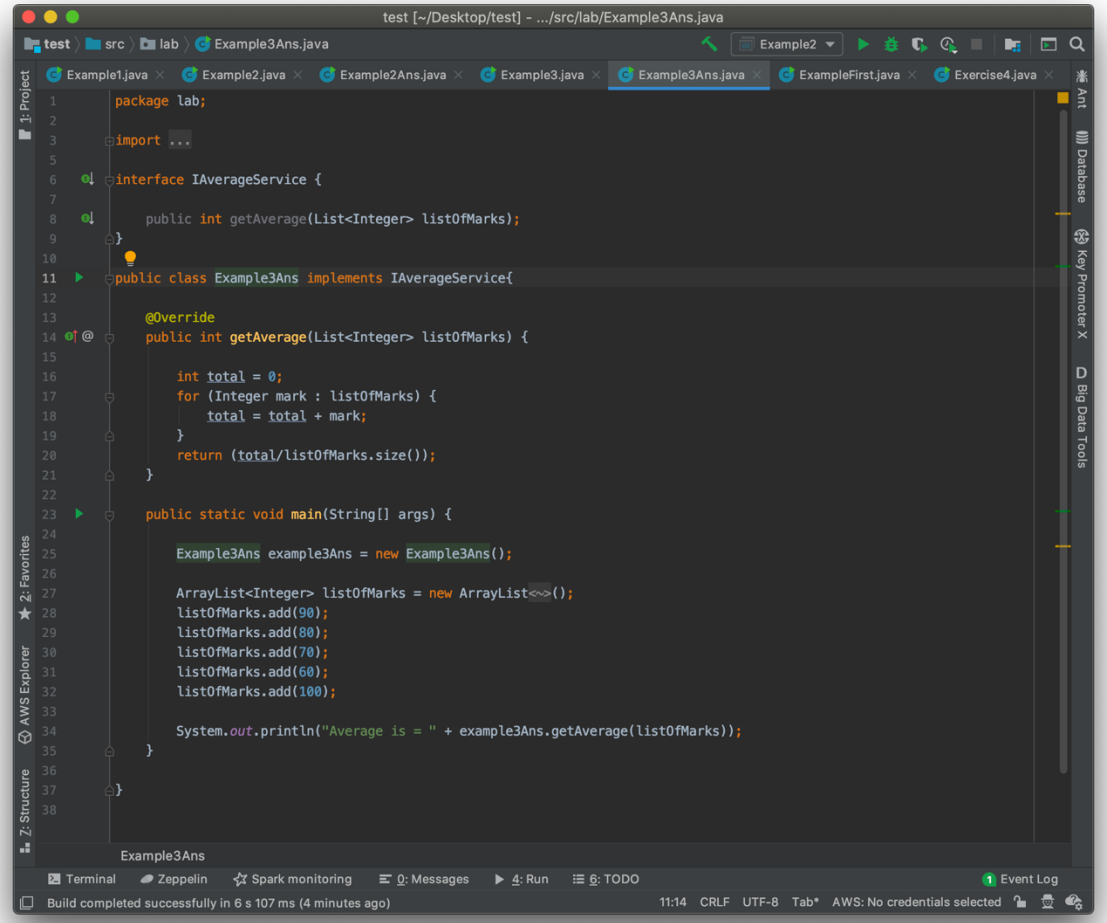
Lambda Expressions

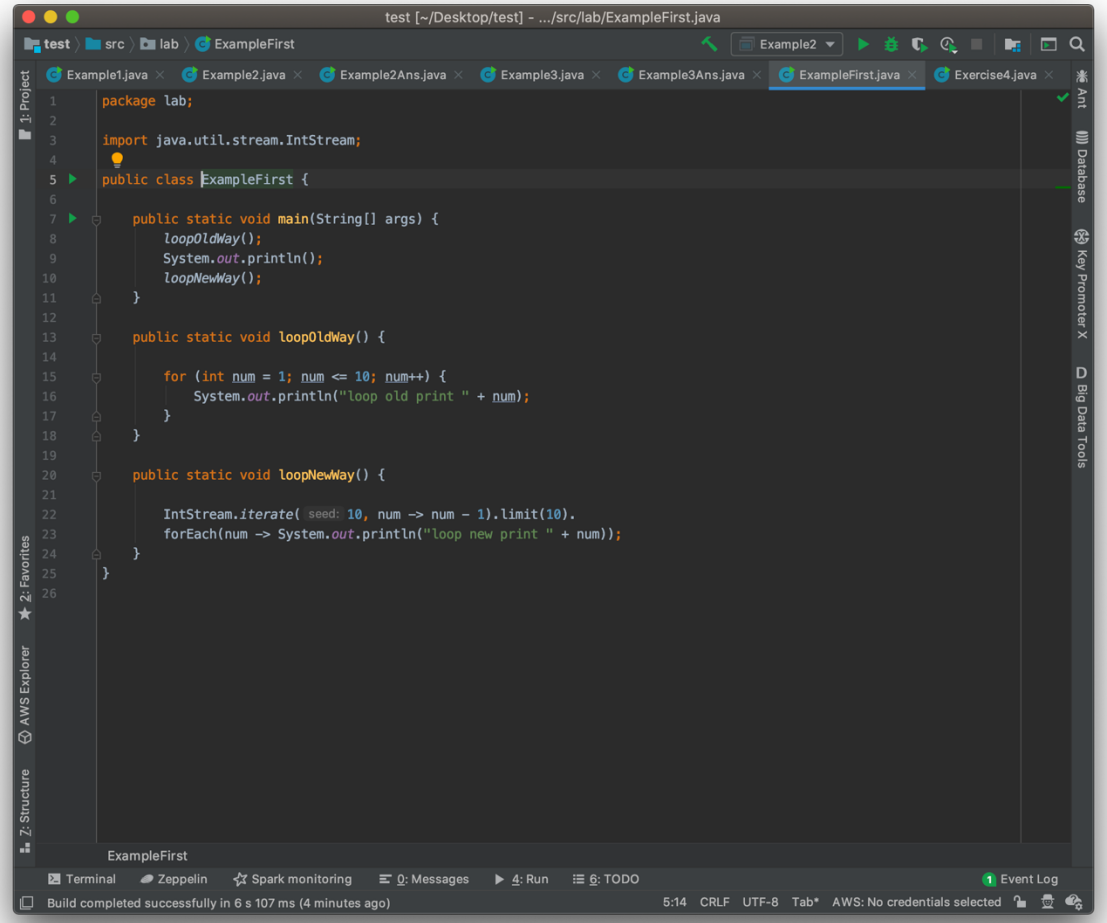



test [~/Desktop/test] - .../src/lab/Example2.java
test
src
lab
Example2.java
Example2
Example1.java
Example2.java
Example2Ans.java
Example3.java
Example3Ans.java
ExampleFirst.java
Exercise4.java
1: Project
2: Favorites
AWS Explorer
Z: Structure
Ant
Database
Key Promoter X
Big Data Tools
package lab;
import ...
interface ICalculateService{
public int calculateTotal(List<Integer> listOfMarks);
}
public class Example2 implements ICalculateService{
@Override
public int calculateTotal(List<Integer> listOfMarks) {
int total = 0;
for (Integer mark : listOfMarks) {
total = total + mark;
}
return total;
}
public static void main(String[] args) {
Example2 example2 = new Example2();
ArrayList<Integer> listOfMarks = new ArrayList<>();
listOfMarks.add(85);
listOfMarks.add(75);
listOfMarks.add(60);
listOfMarks.add(80);
listOfMarks.add(100);
int total = example2.calculateTotal(listOfMarks);
System.out.println("Total is = " + total);
}
}
Example2
Terminal
Zeppelin
Spark monitoring
0: Messages
4: Run
6: TODO
1 Event Log
Build completed successfully in 6 s 107 ms (5 minutes ago)
11:14
CRLF
UTF-8
Tab*
AWS: No credentials selected


test [~/Desktop/test] - .../src/lab/Example2Ans.java
test
src
lab
Example2Ans.java
Example2
Example1.java
Example2.java
Example2Ans.java
Example3.java
Example3Ans.java
ExampleFirst.java
Exercise4.java
1: Project
Ant
Database
Key Promoter X
Big Data Tools
2: Favorites
AWS Explorer
Z: Structure
package lab;
import ...
@FunctionalInterface
interface ICalculateServices{
public int calculateTotal(List<Integer> listOfMarks);
}
public class Example2Ans{
public static void main(String[] args) {
ArrayList<Integer> listOfMarks = new ArrayList<>();
listOfMarks.add(85);
listOfMarks.add(75);
listOfMarks.add(60);
listOfMarks.add(80);
listOfMarks.add(100);
ICalculateService iCalculateService = total -> listOfMarks.stream().mapToInt(Integer::intValue).sum();
System.out.println("Total is = " + iCalculateService.calculateTotal(listOfMarks));
}
}
Example2Ans
main()
Terminal
Zeppelin
Spark monitoring
0: Messages
4: Run
6: TODO
1 Event Log
Build completed successfully in 6 s 107 ms (4 minutes ago)
17:29
CRLF
UTF-8
Tab*
AWS: No credentials selected


test [~/Desktop/test] - .../src/lab/Example3.java
test
src
lab
Example3.java
Example2
Example1.java
Example2.java
Example2Ans.java
Example3.java
Example3Ans.java
ExampleFirst.java
Exercise4.java
1: Project
2: Favorites
AWS Explorer
Z: Structure
Ant
Database
Key Promoter X
Big Data Tools
package lab;
import ...
@FunctionalInterface
interface IAverageServices {
public int getAverage(List<Integer> listOfMarks);
}
public class Example3 {
public static void main(String[] args) {
ArrayList<Integer> listOfMarks = new ArrayList<>();
listOfMarks.add(90);
listOfMarks.add(80);
listOfMarks.add(70);
listOfMarks.add(60);
listOfMarks.add(100);
IAverageServices iAverageServices =
total -> (listOfMarks.stream().mapToInt(Integer::intValue).sum())
/
(listOfMarks.size());
System.out.println("Average is = " + iAverageServices.getAverage(listOfMarks));
}
}
Example3 › main()
Terminal
Zeppelin
Spark monitoring
0: Messages
4: Run
6: TODO
1 Event Log
Build completed successfully in 6 s 107 ms (4 minutes ago)
27:1
CRLF
UTF-8
Tab*
AWS: No credentials selected


test [~/Desktop/test] - .../src/lab/Example3Ans.java
test
src
lab
Example3Ans.java
Example2
Example1.java
Example2.java
Example2Ans.java
Example3.java
Example3Ans.java
ExampleFirst.java
Exercise4.java
1: Project
2: Favorites
AWS Explorer
Z: Structure
Ant
Database
Key Promoter X
Big Data Tools
package lab;
import ...
interface IAverageService {
public int getAverage(List<Integer> listOfMarks);
}
public class Example3Ans implements IAverageService{
@Override
public int getAverage(List<Integer> listOfMarks) {
int total = 0;
for (Integer mark : listOfMarks) {
total = total + mark;
}
return (total/listOfMarks.size());
}
public static void main(String[] args) {
Example3Ans example3Ans = new Example3Ans();
ArrayList<Integer> listOfMarks = new ArrayList<>();
listOfMarks.add(90);
listOfMarks.add(80);
listOfMarks.add(70);
listOfMarks.add(60);
listOfMarks.add(100);
System.out.println("Average is = " + example3Ans.getAverage(listOfMarks));
}
}
Example3Ans
Terminal
Zeppelin
Spark monitoring
0: Messages
4: Run
6: TODO
1 Event Log
Build completed successfully in 6 s 107 ms (4 minutes ago)
11:14
CRLF
UTF-8
Tab*
AWS: No credentials selected


test [~/Desktop/test] - .../src/lab/ExampleFirst.java
test
src
lab
ExampleFirst
Example2
Example1.java
Example2.java
Example2Ans.java
Example3.java
Example3Ans.java
ExampleFirst.java
Exercise4.java
1: Project
2: Favorites
AWS Explorer
Z: Structure
Ant
Database
Key Promoter X
Big Data Tools
package lab;
import java.util.stream.IntStream;
public class ExampleFirst {
public static void main(String[] args) {
loopOldWay();
System.out.println();
loopNewWay();
}
public static void loopOldWay() {
for (int num = 1; num <= 10; num++) {
System.out.println("loop old print " + num);
}
}
public static void loopNewWay() {
IntStream.iterate(seed: 10, num -> num - 1).limit(10).
forEach(num -> System.out.println("loop new print " + num));
}
}
ExampleFirst
Terminal
Zeppelin
Spark monitoring
0: Messages
4: Run
6: TODO
1 Event Log
Build completed successfully in 6 s 107 ms (4 minutes ago)
5:14
CRLF
UTF-8
Tab*
AWS: No credentials selected

test [~/Desktop/test] - .../src/lab/Exercise4.java

test > src > lab > Exercise4

Example2

Example1.java | Example2.java | Example2Ans.java | Example3.java | Example3Ans.java | ExampleFirst.java | Exercise4.java

```java
package lab;

import java.util.stream.IntStream;

public class Exercise4 {

    public static void main(String[] args) {

        System.out.println("Threads old approach");

        Runnable r1 = new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i <= 10; i++) {
                    System.out.println("Threads Old Way " + i);
                }
            }
        };
        new Thread(r1).start();

        System.out.println("Threads new approach");

        Runnable r2 = () -> {
            IntStream.iterate( seed: 0, i -> i + 1).limit(10).forEach(i -> {
                System.out.println("Runnable Threads new way 1 " + i);
            });
        };
        new Thread(r2).start();

        new Thread(() -> IntStream.iterate( seed: 0, i -> i + 1).limit(10).forEach(i -> {
            System.out.println("Threads new way 2 " + i);
        })).start();
    }
}
```

Exercise4

Terminal | Zeppelin | Spark monitoring | 0: Messages | 4: Run | 6: TODO | Event Log

Build completed successfully in 6 s 107 ms (4 minutes ago) 5:14 CRLF UTF-8 Tab* AWS: No credentials selected

Solution for Multiple Inheritance

```java
interface ITestable1{

    public default void readAnnotations(){
        System.out.println("Read Annotations 01");
    }
```

```java
interface ITestable2{

    public default void readAnnotations(){
        System.out.println("Read Annotations 02");
    }
}
```

```java
public class Annotate implements ITestable1, ITestable2{

    public static void main(String[] args) {
        Annotate diff = new Annotate();
        diff.readAnnotations();
    }

    @Override
    public void readAnnotations() {
        ITestable1.super.readAnnotations();
        ITestable2.super.readAnnotations();
    }
}
```

Expressions

Type	Syntax	Method Reference	Lambda expression
Reference to a static method	*Class::staticMethod*	*String::valueOf*	*s -> String.valueOf(s)*
Reference to an instance method of a particular object	*instance::instanceMethod*	*s:toString*	*() -> "string".toString()*
Reference to an instance method of an arbitrary object of a particular type	*Class:instanceMethod*	*String::toString*	*s -> s.toString()*
Reference to a constructor	*Class::new*	*String::new*	*() -> new String()*

Summary of Method Referenc

Method Reference Type	Syntax	Example
static	ClassName::StaticMethodName	String::valueOf
constructor	ClassName::new	ArrayList::new
specific object instance	objectReference::MethodName	x::toString
arbitrary object of a given type	ClassName::InstanceMethodName	Object::toString

Method References

Four Types of Method References

1) Method reference to an instance method of an object – object::instanceMethod
2) Method reference to a static method of a class – Class::staticMethod
3) Method reference to an instance method of an arbitrary object of a particular type – Class::instanceMethod
4) Method reference to a constructor – Class::new

1. Method Reference to an instance me t

```
package method.reference;

interface IReference {
    void display();
}

public class First {

    public void myMethod() {
        System.out.println("Instance Method");
    }

    public static void main(String[] args) {

        First first = new First();
        IReference iReference = first::myMethod;
        iReference.display();
    }
}
```

Problems | Console | Javadoc | Declaration

<terminated> First [Java Application] C:\Program Files\Java\jdk

Instance Method

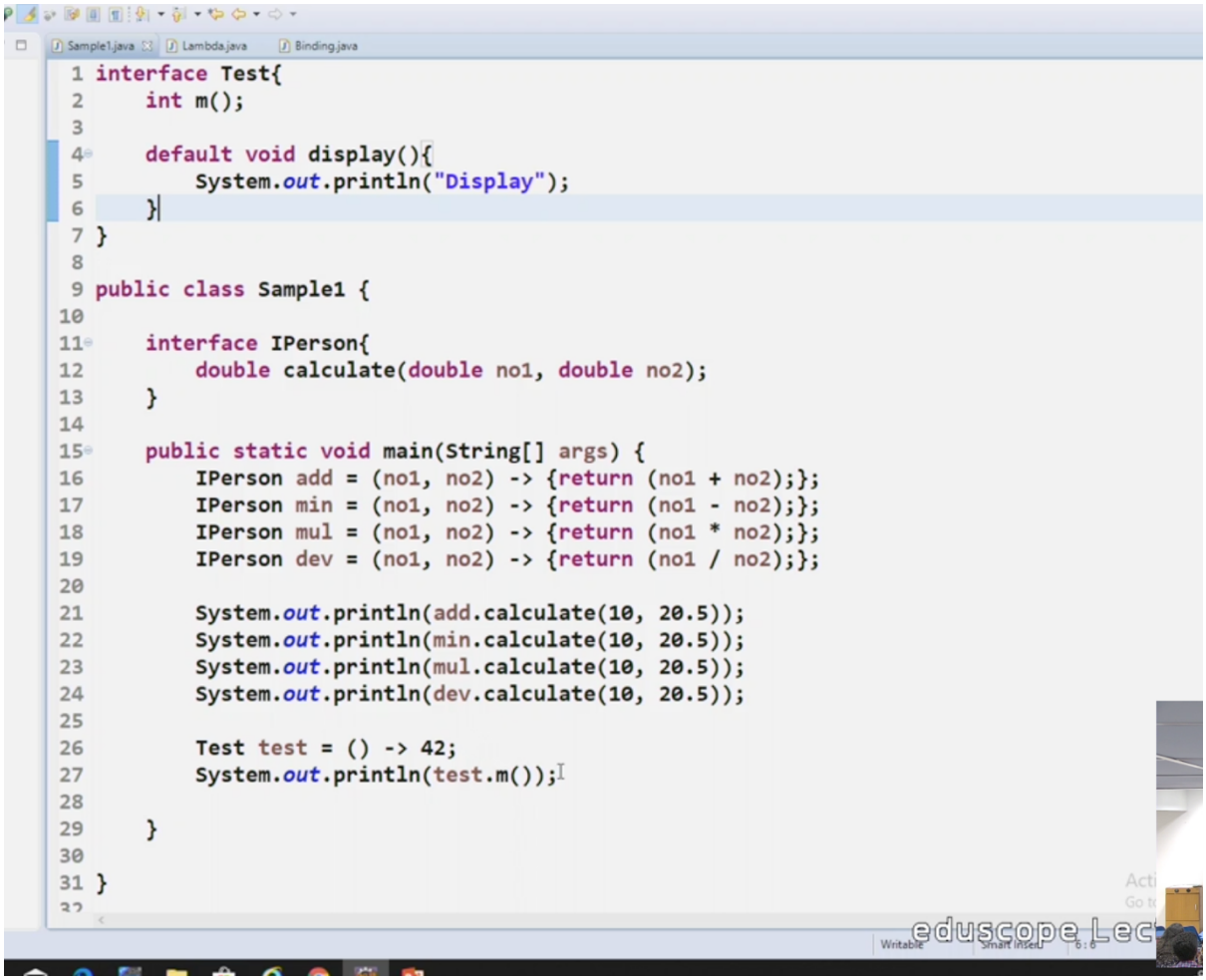
Sample1.java | Lambda.java | Binding.java

```
interface Test{
    int m();

    default void display(){
        System.out.println("Display");
    }

    static void staticMethod(){
        System.out.println("Static method");
    }
}

public class Sample1 {

    interface IPerson{
        double calculate(double no1, double no2);
    }

    public static void main(String[] args) {
        Test test = () -> 42;
        System.out.println(test.m());
        test.display();
        Test.staticMethod();



        IPerson add = (no1, no2) -> {return (no1 + no2);};
        IPerson min = (no1, no2) -> {return (no1 - no2);};
        IPerson mul = (no1, no2) -> {return (no1 * no2);};
        IPerson dev = (no1, no2) -> {return (no1 / no2);};

    }
```

<terminated> Sample

42
Display
Static method

eduscope Lec


Sample1.java
Lambda.java
Binding.java
interface Test{
int m();
default void display(){
System.out.println("Display");
}
}
public class Sample1 {
interface IPerson{
double calculate(double no1, double no2);
}
public static void main(String[] args) {
IPerson add = (no1, no2) -> {return (no1 + no2);};
IPerson min = (no1, no2) -> {return (no1 - no2);};
IPerson mul = (no1, no2) -> {return (no1 * no2);};
IPerson dev = (no1, no2) -> {return (no1 / no2);};
System.out.println(add.calculate(10, 20.5));
System.out.println(min.calculate(10, 20.5));
System.out.println(mul.calculate(10, 20.5));
System.out.println(dev.calculate(10, 20.5));
Test test = () -> 42;
System.out.println(test.m());
}
}
eduscope Lec

Lambda.java | MethodReference.java | DefaultMethods.java | Anonymous...

```java
interface MathOperation {
    int operation(int a, int b);
}

public class Lambda {

    public static void main(String[] args) {

        Lambda lambda = new Lambda();

        // with type declaration
        MathOperation addition = (int a, int b) -> a + b;

        // with out type declaration
        MathOperation subtraction = (a, b) -> a - b;

        // with return statement along with curly braces
        MathOperation multiplication = (int a, int b) -> {
            return a * b;
        };

        // without return statement and without curly braces
        MathOperation division = (int a, int b) -> a / b;

        System.out.println("10 + 5 = " + lambda.operate(10, 5, addition));
        System.out.println("10 - 5 = " + lambda.operate(10, 5, subtraction));
        System.out.println("10 x 5 = " + lambda.operate(10, 5, multiplication));
        System.out.println("10 / 5 = " + lambda.operate(10, 5, division));
    }

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

Problems | Console

<terminated> Lambda [Java App

```
10 + 5 = 15
10 - 5 = 5
10 x 5 = 50
10 / 5 = 2
```


Lambda Syntax

```java
(int x) -> x+1             // Single declared-type argument
(int x) -> { return x+1; } // same as above
(x) -> x+1                 // Single inferred-type argument, same as below
x -> x+1                   // Parenthesis optional for single inferred-type case

(String s) -> s.length()   // Single declared-type argument
(Thread t) -> { t.start(); } // Single declared-type argument
s -> s.length()              // Single inferred-type argument
t -> { t.start(); }          // Single inferred-type argument

(int x, int y) -> x+y      // Multiple declared-type parameters
(x,y) -> x+y               // Multiple inferred-type parameters
(x, final y) -> x+y        // Illegal: can't modify inferred-type parameters
(x, int y) -> x+y          // Illegal: can't mix inferred and declared types
```

Sample1.java

```java
interface IPerson{
    double calculate(double no1, double no2);
}

interface Test{
    int m();
}
public class Sample1 {

    public static void main(String[] args) {
        IPerson add = (no1, no2) -> {return (no1 + no2);};
        IPerson min = (no1, no2) -> {return (no1 - no2);};
        IPerson mul = (no1, no2) -> {return (no1 * no2);};
        IPerson dev = (no1, no2) -> {return (no1 / no2);};

        System.out.println(add.calculate(10, 20.5));
        System.out.println(min.calculate(10, 20.5));
        System.out.println(mul.calculate(10, 20.5));
        System.out.println(dev.calculate(10, 20.5));

        Test test = () -> 42;
        System.out.println(test.m());

    }

}
```

Sample1.java

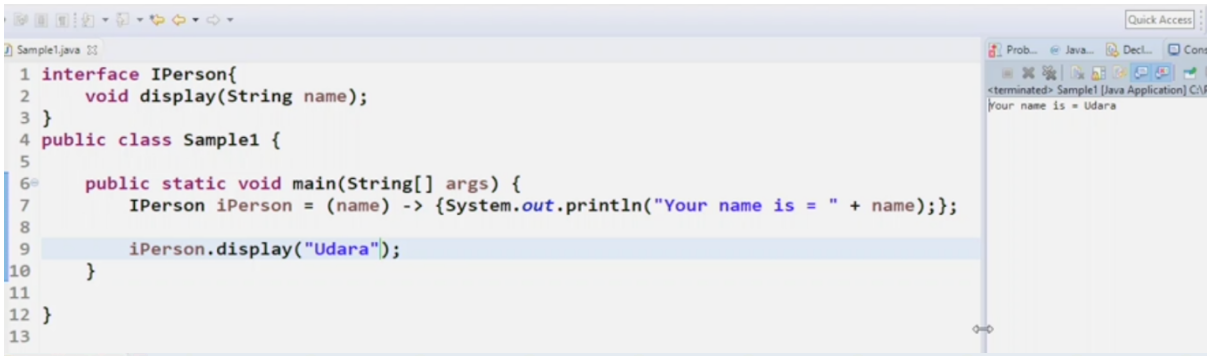
```java
interface IPerson{
    double calculate(double no1, double no2);
}
public class Sample1 {

    public static void main(String[] args) {
        IPerson add = (no1, no2) -> {return (no1 + no2);};
        IPerson min = (no1, no2) -> {return (no1 - no2);};
        IPerson mul = (no1, no2) -> {return (no1 * no2);};
        IPerson dev = (no1, no2) -> {return (no1 / no2);};

        System.out.println(add.calculate(10, 20.5));
        System.out.println(min.calculate(10, 20.5));
        System.out.println(mul.calculate(10, 20.5));
        System.out.println(dev.calculate(10, 20.5));
    }

}
```

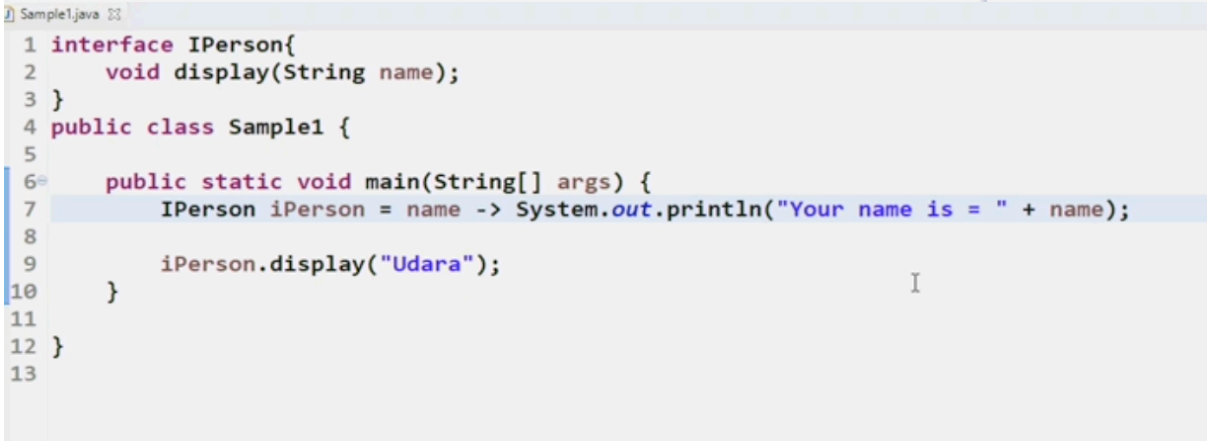
```
<terminated> Sample1 [Java Application]
30.5
-10.5
205.0
0.4878048780487805
```

Sample1.java

```java
interface IPerson{
    double calculate(double no1, double no2);
}
public class Sample1 {

    public static void main(String[] args) {
        IPerson iPerson = (no1, no2) -> {return (no1 + no2);};

        System.out.println(iPerson.calculate(10, 20.5));
    }

}
```

```
<terminated> Sample1 [Java Application]
30.5
```


Quick Access
Sample1.java
interface IPerson{
void display(String name);
}
public class Sample1 {
public static void main(String[] args) {
IPerson iPerson = (name) -> {System.out.println("Your name is = " + name);};
iPerson.display("Udara");
}
}
Your name is = Udara


Sample1.java
interface IPerson{
void display(String name);
}
public class Sample1 {
public static void main(String[] args) {
IPerson iPerson = name -> System.out.println("Your name is = " + name);
iPerson.display("Udara");
}
}