Lambda Expressions

Shristi Technology Labs

Contents

- Overview of Functional Interface
- Example
- Lambda Expressions

Functional Interface

- An interface with only one abstract method - **Functional Interface**
- May or may not be annotated with **@FunctionalInterface**

Few Functional Interfaces

- Runnable
- Comparable
- Comparator
- Function<T,R>
- Predicate<T>
- Consumer<T>
- Supplier<T>

Functional Interfaces

The ***java.util.function*** package has 8 functional interfaces.

- Function
- BiFunction
- Consumer
- BiConsumer
- Supplier
- Predicate
- BiPredicate

Example

```
interface Calculator {
    void calculate(int x, int y);
}
interface ScientificCalc {
    void calcAngles(int x);
}
```

Lambda Expressions

- Is defined as a method without name, access specifier, return type.
- Make the code writing simple, short and readable.
- Short way to write the method in the same place where it will be used.
- Used with functional interfaces.

eg.

```
public int add(int x, int y){
  return x+y;
}
```

```
public void greet(){
  System.out.println("Hello World");
}
```

Using Lambda

```
(int  x, int y) -> {
      return x+y;
     }
```

```
() ->System.out.println("Hello world");
```

Syntax

```
(parameter)  ->   expression
(parameters)  ->   {statements};
()->expression
(arg1, arg2 . . .   )   ->    { body }
(type1 arg1, type2 arg2 . . .)   -> { body }
```

```
   ( int a, int b ) -> { return  a+b; }
   ()   -> System.out.println ( “Hello World”);
   (String S ) ->  { System.out.println (s) ; }
   (a,b)   ->  { return a+b } ;
```

```
//no argument - single statement, no curly braces
  () -> System.out.println ( “Hello World”);

//one argument - single statement with curly braces
  (message ) -> { System.out.println (message) ; }

//with two argument and types
  (int a,int b ) -> { return a*b; }

//two arguments - multiple statements
  (a,b) -> {
     int sum  = a+b;
     return sum;
  }
```

Lambda Expressions

Functional Interfaces

- Runnable
- Comparable
- Comparator
- Function<T,R>
- Predicate<T>
- Consumer<T>
- Supplier<T>

```
@FunctionalInterface
public interface BonusCalculator {

    public void  calcBonus(int x);
    public default void greetMessage(){
        System.out.println(“default method ");
     }
    public static void Check(){
         System.out.println("static method ");
      }
}
```

Example

```
interface Calculator{
    void calculate(int x, int y);
}
class Checker{
    public void findTotal(Calculator c,int x,int y){
        System.out.println("Numbers: "+x+ ", "+y);
        c.calculate(x, y);
    }
}
```

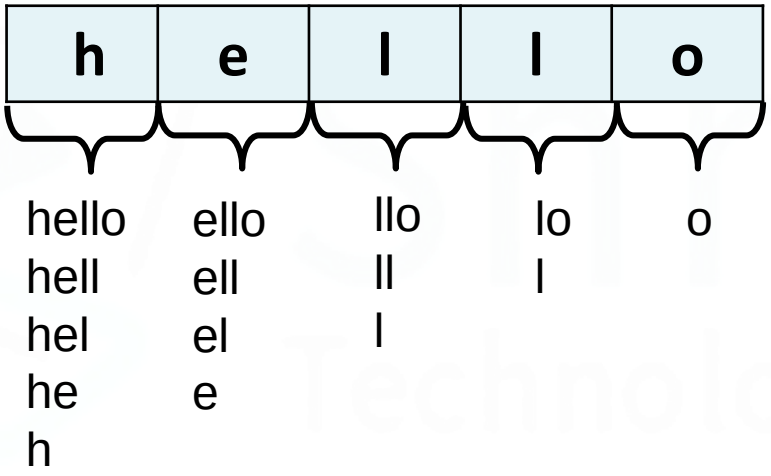
```
Checker check = new Checker();
check.findTotal((x,y)->{
    System.out.println("Sum "+(x+y));
}, 10, 20);
```

```
check.findTotal((x,y)->{
    System.out.println("Product "+(x*y));
}, 10, 20);
```

Contents

- Overview of Functional Interface
- Lambda Expressions

Find all substrings of a given string



Thank you