Lambda expressions

Agenda

1 Lambda expressions

2 Examples for using Lambda expressions

Lambda expressions

Lambda expressions

- **Lambda expressions** means a block of code that you can pass around so it can be executed later, once or multiple times.
- It uses a new operator: **->**

Points to be noted:

- The body of a lambda expression can contain **zero, one or more statements**.
- When there is a **single statement,** curly brackets are **not mandatory** and the return type of the anonymous function is same as that of the body expression.
- If there are more than one statement, then those must be enclosed in curly brackets (**a code block**) and the return type of the anonymous function is same as the type of the value returned within the code block, or void if nothing is returned.

Lambda expressions

Syntax:
lambda operator -> body

Example 1: `(int x,int y)-> x+y;`

The above example adds the values of x and y and returns the same

Example 2: `(String s) -> {System.out.println(s);}`

Here the String s passed gets printed

Examples for using Lambda expressions

Example1 – without Lambda Expression

```
interface iface1 {
        int add(int x, int y);
}
class class1 implements iface1 {
        public int add(int x, int y) {
            return x+y;
        }
}
public class BeforeJava8 {
    public static void main(String[ ] args) {
        iface1 i1 = new class1();        //continued..
```

Example1 – without Lambda Expression

```
        int ans = i1.add(10, 20);
        System.out.print("i1.add(10, 20)= ");
        System.out.println(ans);
        ans = i1.add(100, 200);
        System.out.print("i1.add(100, 200)= ");
        System.out.println(ans);
    }
}
```

OUTPUT

```
i1.add(10, 20)= 30
i1.add(100, 200)= 300
```

Example2 - using Lambda expressions

```
interface iface2 {
        int add(int x, int y);
}
public class AfterJava8Lambda {
    public static void main(String[] args) {
        iface2 i1 = (x,y)->( x+y);
        int ans =i1.add(10, 20);
        System.out.println(" i1.add(10, 20), ans="+ans);
        ans =i1.add(100, 200);
        System.out.println(" i1.add(100, 200), ans="+ans);
    }
}
```

OUTPUT

```
i1.add(10, 20)= 30
i1.add(100, 200)= 300
```

Example3 – Lambda Expression with ArrayList

```
import java.util.ArrayList;

public class ArrayListLambdaTest {
          public static void main(String args[ ]) {
                    //Creating an ArrayList
                    ArrayList<Integer> list1 = new ArrayList<Integer>();
                    list1.add(1); list1.add(2);
                    list1.add(3); list1.add(4);
                    //Using lambda expression to print all elements
                    System.out.println(" printing all elements: ");
                    list1.forEach(n -> System.out.print(n + "  "));         //continued..
```

Example3 – Lambda Expression with ArrayList (Continued)

```
        //Using lambda expression to print even numbers
        System.out.println("\n printing even numbers: ");
        list1.forEach(
            n -> {
                if (n%2 == 0)
                    System.out.print(n + "  ");
               }
        );
    }
}
```

OUTPUT

```
printing all elements:
1  2  3  4
printing even numbers:
2 4
```

wipro

Thank you