Expresii Lambda

Java 8

Expresii Lamda (-> este operatorul Lambda)

(parametri) -> expresie

(parametri) -> { instructiuni }

```
(int x, int y) -> x+y
```

```
double f(int x, int y) {
        return x+y;
}
```

```
(x) -> { System.out.print(x); }
```

```
void afis(Object x) {
     System.out.print(x);
}
```

Precizarea tipului argumentelor poate lipsi daca acesta poate fi dedus din comtext (inferat).
Parantezele pot lipsi daca este vorba de un singur parametru. Ex. a -> a+1

Interfețe funcționale

Sunt interțele care au **o singură metodă abstractă**

Exemple:

java.lang.Runnable - run()

java.util.Comparator - compare(T o1, T o2)

java.awt.event.ActionListener -

actionPerformed (ActionEvent e)

Exemplu

```
public class TestDiverse implements ActionListener{

    public static void main(String[] args) {
        //...
        Button b=new Button("Stop");
        b.addActionListener( this );
       //...
}
   @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
}
```

Expresii Lamda

```
Button b=new Button("Stop");
ActionListener  ob = (e) -> {System.exit(0); }  ;
b.addActionListener(ob);
```

sau

```
Button b=new Button("Stop");
b.addActionListener (  (e)->{System.exit(0);}  );
```

Expresii Lamda si interfete functionale

```
interface Numar {
    public int valoare(int n);
}
public class Exemplu {

    public static void main(String[] args) {
        Numar nr;
        nr = n -> n<0? 0:n ;
        System.out.println("N1=" + nr.valoare(-1));
        System.out.println("N2=" + nr.valoare(2));

        nr = (n)-> n*n ;
        System.out.println("N3=" + nr.valoare(3));
```

```
N1=0
N2=2
N3=9
```

Expresii Lamda si interfete functionale

```
new Thread (new Runnable()
{
    @Override
    public void run() {
        while(true) {
            asteapta(1000);
            System.out.print("*");
        }
    }
}
).start();
```

```
new Thread (  () -> {
        while(true) {
            asteapta(1000);
            System.out.print("*");
        }
    }
).start();
```

Expresii Lamda si interfete functionale

```
interface Functie {
      long  f ( int n);
}
//…
Functie factorial = (n) -> {
   long rez = 1;
   for(int i=2; i <= n; i++)
        rez *= i;
   return rez;
} ;

int m=7;
System.out.println( m + "!=" + factorial.f(m)  );
```

Expresii Lamda

() -> 9.5 public double f(){ return 9.5;}

(n) -> (n % 2)==0 public boolean g(int n) { return (n % 2)==0;}

Care expresii sunt corecte ?

() -> {}

() -> "Radu"

() -> Math.random() * 100

() -> {return "Maria";}

(Integer i) -> return "Alex" + i;

(String s) -> {" Student";}

Functie calcul1 = ()->1;

Functie calcul2 = x->x+2;

Functie calcul3 = x->3;

Functie calcul4 = (x)->"4";

```
interface Functie {
        long f ( int n);
}
```

Expresii Lamda

() -> 9.5	public double f(){ return 9.5;}
(n) -> (n % 2)==0	public boolean g(int n) { return (n % 2)==0;}

Care expresii sunt corecte ?

() -> {}

() -> "Radu"

() -> Math.random() * 100

() -> {return "Maria";}

(Integer i) -> return "Alex" + i;

(String s) -> {"Student";}

Functie calcul1 = ()->1;

Functie calcul2 = x->x+2;

Functie calcul3 = x->3;

Functie calcul4 = (x)->"4";

```
interface Functie {
        long f ( int n);
}
```

Expresii Lamda

() -> 9.5 public double f(){ return 9.5;}

(n) -> (n % 2)==0 public boolean g(int n) { return (n % 2)==0;}

Care expresii sunt corecte ?

() -> {}

() -> "Radu"

() -> Math.random() * 100

() -> {return "Maria";}

(Integer i) -> **{** return "Alex" + i; **}**

(String s) -> "Student" sau (String s) -> {**return** "Student";}

Functie calcul1 = (**v**)->1;

Functie calcul2 = x->x+2;

Functie calcul3 = x->3;

Functie calcul4 = (x)->**4;** //**"4";**

```
interface Functie {
        long f ( int n);
}
```

Referirea metodelor

numeClasa :: numeMetoda

Se aplică pentru obținerea unei referințe dacă metoda este

- statică
- de instanță

this :: numeMetoda

Referința către constructor

numeClasa :: **new**

Referirea metodelor

```
public interface FileFilter {
     public boolean accept(File pathname);
}
```

In clasa File:

```
public File[] listFiles(FileFilter filter)
   public boolean isDirectory()
```

Afisarea subdirectoarelor (Java 7)

```
File dir=new File("D:\\SD");
File[] subDir = dir.listFiles ( new FileFilter() {
            @Override
            public boolean accept(File f){
               return f.isDirectory();
            }
      }
);
for(File f: subDir)
         System.out.println(f);
```

Afisarea subdirectoarelor (Java 8)

```
File dir=new File("D:\\SD");
File[] subDir = dir.listFiles ( f ->  f . isDirectory () );
        for(File f: subDir)
            System.out.println(f);
```

Expresie Lambda

Afisarea subdirectoarelor (Java 8)

```
File dir=new File("D:\\SD");
File[] subDir = dir.listFiles (File :: isDirectory );
        for(File f: subDir)
            System.out.println(f);
```

Referirea metodelor

Expresiile Lambda

- O **expresie lambda** – o funcție anonimă
- Sintaxa expresiile lambda:

(arg) -> expresie sau **(arg)-> { instructiuni }**

- Expresiile lambda permit scrierea de cod concis
- O ***interfață functională*** este o interfață care declară o singură metodă abstractă
- Expresiile lambda sunt utilizate în contextul *interfețelor funcționale*
- *Referirea metodelor* o alternativa la Expresiile lambda care pot fi reutilizate

Bibliografie

- **Java 8 Lambdas**, by Richard Warburton, Published by O'Reilly Media, 2014
- ***Java 8 in Action,*** by Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft, 2014 Manning Publications
- **Building Java Programs**, Chapter 19, Pearson 2016