

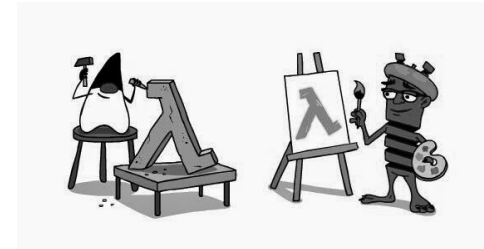
Programozás III

LAMBDA
KIFEJEZÉSEK

JDK 8-AS ÚJÍTÁSOK

Javasolt változtatás:

```
private void kiir(String cim) {  
    System.out.println(cim);  
    lista.stream().forEach((elem) -> {  
        System.out.println(elem);  
    });  
}
```



JDK 8-AS ÚJÍTÁSOK

```
33 private void kiir(String cim) {  
34     System.out.println(cim);  
35     for (String elem : lista) {  
36         System.out.println(elem);  
37     }  
38 }
```

Mi lehet a baj?

```
32 Can use functional operations  
33 -----  
34 (Alt-Enter shows hints) (String cim) {  
35     System.out.println(cim);  
36     for (String elem : lista) {  
37         System.out.println(elem);  
38     }  
39 }
```

LAMBDA KIFEJEZÉSEK

A lambda kifejezések olyan névtelen metódusok, amelyeket ott írunk meg, ahol ténylegesen használjuk.

```
PI.: private void gombBeallitas() {  
    JButton btn = new JButton("proba");  
    btn.addActionListener(  
        new ActionListener() {  
            @Override  
            public void actionPerformed(ActionEvent e) {  
                System.out.println("próba");  
            }  
        });  
    this.add(btn);  
    btn.setSize(100, 50);  
}
```

Anonymus inner class

LAMBDA KIFEJEZÉSEK

Ugyanez lambda kifejezéssel:

```
private void gombBeallitas() {  
    JButton btn = new JButton("proba");  
    btn.addActionListener(event -> System.out.println("próba"));  
    this.add(btn);  
    btn.setSize(100, 50);  
}
```

Vagyis a lambda kifejezéssel magát a viselkedést, tehát az implementációt adjuk át az addActionListener() metódusnak, nem pedig egy olyan objektumot amely megvalósítja a kívánt interfészt.

LAMBDA KIFEJEZÉSEK

Matematikai háttér (λ -kalkulus) – nagyon leegyszerűsítve:

Church féle λ -kalkulus – a kiszámíthatóságelmélet alapja, ma inkább a funkcionális programozásé.

1. „Névtelen” függvényekkel foglalkozik.

pl. $\text{sqsum}(x,y) = x*x + y*y$

helyett: $(x,y) \rightarrow x*x + y*y$;

Többváltozós függvény: egyváltozós függvényhívások egymásutánja.

LAMBDA KIFEJEZÉSEK

Funkcionális interfész:

Olyan interfész, amelynek csak egyetlen absztrakt metódusa van.

Pl. Runnable, Comparator, stb.
– a JDK 8-ban továbbiakat is bevezettek

A funkcionális interfészekhez lehet lambda kifejezéseket illeszteni.

(Funkcionális programozás)

LAMBDA KIFEJEZÉSEK

Szintaktika:

Argumentumlista nyíl a kifejezés törzse – pl.:
 $(int\ x,\ int\ y) \rightarrow x + y$

A törzs lehet:

– egyszerű kifejezés, pl.: $(int\ x,\ int\ y) \rightarrow x*x + y*y$

– blokk, pl.:
 $(String\ str) \rightarrow \{ System.out.println(„hello " + str); \}$

LAMBDA KIFEJEZÉSEK

Példa: egy metódusú interfész megvalósítása:

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

private void pelda1() {
    IntegerMath addition = (a,b)-> a+b;
    IntegerMath sub = (a,b)-> a-b;
    IntegerMath multi = (a,b) -> a*a +b*b;
    System.out.println("kivonás : "+ addition.operation(4, 4));
    System.out.println("összeadás :"+ sub.operation(10, 4));
    System.out.println("szorzás :"+ multi.operation(10, 4));
}
```

LAMBDA KIFEJEZÉSEK

Stream API

segítségével a Collection-ben lévő objektumokat stream-ként is feldolgozhatunk.

A Stream API előtti időkben ha egy listát akartunk bejárni, akkor külső iterációt kellett alkalmazni, ami terjengőssé tette a kódot, illetve nehezen párhuzamosítható.

A Stream API egy olyan megoldást nyújt, ahol nem kell külön for ciklusokat definiálnunk, az iteráció részleteit az API kezeli.

LAMBDA KIFEJEZÉSEK

Példa: rendezés

```
// hagyományos:
Collections.sort(lista, new RendezoOsztaly());

// lambda kifejezéssel
Collections.sort(lista, (String a1, String a2) -> a1.compareTo(a2));
```

Természetesen más típusú listával is hasonló.

LAMBDA KIFEJEZÉSEK

Példa:

```
// ciklussal
int meret = 5;
long darab = 0;
for (String str : lista) {
    if (str.length() == meret) {
        darab++;
    }
}
System.out.println("darabszám: " + darab);

// Stream API segítségével:
darab = lista.stream().filter(str -> str.length() == meret).count();
System.out.println("darabszám: " + darab);
```

LAMBDA KIFEJEZÉSEK

Példa:

```
String keresett = "gy";
System.out.println(keresett + "-t tartalmazó szavak:");
lista.stream().filter(a -> {
    return a.contains(keresett);
})
.forEach(a -> System.out.println(a));
```

OLVASNIVALÓ

Néhány link:

<http://www.bakaibalazs.hu/2014/12/java-se-8-lambda-kifejezesek.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>

<http://tutorials.jenkov.com/java/lambda-expressions.html>

http://www.tutorialspoint.com/java8/java8_lambda_expressions.htm

<http://java-latte.blogspot.hu/2014/02/functional-interface-and-lambda-in-java.html>

<http://java-latte.blogspot.hu/2015/07/lambda-expression-examples-and-functional-interface-in-java.html>

<http://java-latte.blogspot.hu/2014/02/lambda-examples-and-effectively-final.html>

<http://java-latte.blogspot.hu/2014/03/stream-lambda-in-java-8.html>