LPP - +3CFU (corso da 9 CFU)

Introduzione alle lambda espressioni di Java 8

Viviana Bono

Per rendere il codice...

- ... più modulare
- ... più compatto
- ... più leggibile

Note

Capitolo 2 di **Java 8 in Action**

Codice: `https://github.com/java8/Java8InAction/tree/master/src/main/java/lambdasinaction`

Cominciamo da una lista di oggetti

```
// main

List<Apple> someApples =
            filter...(inventory, ...);
```

`inventory` è una `List<Apple>`. La classe `Apple` ha stato = "color, weight", e metodi accessori.

Filtro 1

```
public static List<Apple>
            filterGreenApples(List<Apple> inventory){
    List<Apple> result = new ArrayList<>();
    for(Apple apple: inventory)
          if("green".equals(apple.getColor()))
              result.add(apple);
     return result;
}
```

Filtro 2 por color

```
public static List<Apple>
          filterApplesByColor(List<Apple> inventory,
                                          String color){
      List<Apple> result = new ArrayList<>();
      for(Apple apple: inventory)
           if(apple.getColor().equals(color))
              result.add(apple);
      return result;
}
```

Filtro 3 param. peso

```
public static List<Apple>
  filterApplesByWeight(List<Apple> inventory,
                                   int weight){
 List<Apple> result = new ArrayList<>();
  for(Apple apple: inventory)
       if(apple.getWeight() > weight)
            result.add(apple);
  return result;
}
```

Filtro 4

```
public static List<Apple>
  filter(List<Apple> inventory,
         ApplePredicate p){
    List<Apple> result =
          new ArrayList<>();
    for(Apple apple : inventory)
        if(p.test(apple))
          result.add(apple);
    return result;
}
```

```
interface ApplePredicate{
      public boolean test(Apple a);
}

class AppleRedAndHeavyPredicate
 implements ApplePredicate{
   public boolean test(Apple apple){
     return "red".equals(apple.getColor())
                 && apple.getWeight() > 150;
     }
}

// main
List<Apple> redAndHeavyApples =
  filter(inventory,
         new AppleRedAndHeavyPredicate());
```

Design pattern STRATEGY (lo vedrete a SAS)

Filtro 5

Con classe anonima

```
List<Apple> redApples = filter(inventory,
      new ApplePredicate() {
           pubblic boolean test(Apple apple) {
              return "red".equals(apple.getColor());
           }
      });
```

Sintassi delle lambda espressioni in Java 8

```
(parameters) -> expression
```

oppure:

```
(parameters) -> { statements; }
```

L'identità (tra int) in Java 8

$\lambda x : \texttt{int}.x$ diventa:

```
(x:int) -> x
```

oppure:

```
(x:int) -> {return x;}
```

(a volte si può omettere il tipo dei parametri, se inferibile dal contesto)

Qual è il tipo dell'identità?

$\lambda x : \texttt{int}.x : \texttt{int} \to \texttt{int}$

```
I identita = (x:int) -> x;
```

Non viene introdotto un nuovo costruttore di tipo, si usano le interfacce funzionali:

```
interface I{
  int id(int x);
}
```

Uno e un solo metodo astratto, che moralmente è un tipo funzionale, *il cui body è implementato dal body della lambda espressione* (può contenere metodi statici e metodi di default). Vediamo un semplice esempio di uso:

```
System.out.println(identita.id(7)); // stampa 7
```

Filtro 6

Con lambda espressione

```
List<Apple> redApples = filter(inventory,
 (Apple apple) -> "red".equals(apple.getColor()));
```

Qual è il tipo della lambda espressione?

Filtro 6 (continua)

```
interface Predicate<T>{
     public boolean test(T t);
}

public static List<T> fiter(List<T> list,
                     Predicate<T> p) {
    List<T> result = new ArrayList<>();
    for (T e: list)
       if (p.test(e)) result.add(e);
    return result;
}

List<Apple> redApples =
     filter(inventory,
              (Apple apple) -> "red".equals(apple.getColor()));
```

ma anche:

```
List<String> evenNumbers =
     filter(numbers, (Integer i) -> i % 2 == 0);
```

Esempio 1 - sorting con un Comparator

```
 // java.util.Comparator
public interface Comparator<T> {
      public int compare(T o1, T o2);
}

// classe anonima
inventory.sort(
     new Comparator<Apple>() {
            public int compare(Apple a1, Apple a2){
                return a1.getWeight().compareTo(a2.getWeight());
});

// lambda espressione
inventory.sort(
(Apple a1, Apple a2) -> a1.getWeight().compareTo(a2.getWeight())
```

(i dettagli del sort non sono importanti)

Esempio 2 - esecuzione di un blocco di codice con Runnable

```
public interface Runnable{ public void run();
}

// classe anonima
Thread t = new Thread(new Runnable() { public void run(){
System.out.println("Hello world"); }
});

// lambda espressione
Thread t = new Thread(() -> System.out.println("Hello world"));
```