LPP - +3CFU (corso da 9 CFU)

I reference methods e introduzione agli stream

Viviana Bono

Referenze

Capitoli 3.6, 4 di **Java 8 in action**

API:
https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html

Zucchero sintattico: method reference (1)

Se nel body di una lambda c'è solo una chiamata di metodo, si può usare una sintassi più compatta, la method reference.

(lettura interessante: https://www.amitph.com/java-method-and-constructor-reference/)

```
// esempio con metodo statico:

// con lambda:
Function<String, Integer> stringToInteger = (String s) -> Integer.parseInt(s);
// con method reference:
Function<String, Integer> stringToInteger = Integer::parseInt;

// esempi con metodo chiamato su parametro:

List<String> str = Arrays.asList("a","b","A","B");
// con lambda:
str.sort((s1, s2) -> s1.compareToIgnoreCase(s2));
// con method reference:
str.sort(String::compareToIgnoreCase); // uso la classe del this 's1'
```

Zucchero sintattico: method reference (2)

```
// con lambda:
BiPredicate<List<String>, String> contains =
                (list, element) -> list.contains(element);
// con method reference:
BiPredicate<List<String>, String> contains =
                 List::contains; // uso la classe del this 'list'

// su espressione che si valuta a oggetto:
// con lambda:
// (args) -> expr.instanceMet(args)
// con method reference:
// expr::instanceMet
```

Zucchero sintattico: method reference (3)

```
// esempio con costruttore:
ClassName::new
// corrisponde a lambda:
// (lista_param_costruttore) -> new ClassName(lista_param_costruttore)

// come si usa? Factory! (SAS...) - Un esempio

class Employee{
  private String name;
  private int age;
  public Employee(String n, int a){
     name = n; age = a;
  }  ... }

// interfaccia funzionale, tipo della lambda empFactory sotto
public Interface EmployeeFactory{
 public abstract Employee getEmployee(String name, Integer age);
}

EmployeeFactory empFactory = Employee::new;
                  // corrisponde a: (name, age) -> new Employee(name, age)

Employee emp = empFactory.getEmployee("John Hammond", 25);
```

Introduzione agli stream di Java 8

Running example: classe Dish

```
public class Dish {
  private final String name;
  private final boolean vegetarian;
  private final int calories;
  private final Type type;
  public Dish(String name, boolean vegetarian, int calories, Type type) {
     this.name = name;
     ...
  }
  public String getName() {
    return name;
  }
  public boolean isVegetarian() {
    return vegetarian;
  }
  public int getCalories() {
    return calories;
  }
  public Type getType() {
    return type;
  }
```

```
@Override
public String toString() {
   return name;
}
public enum Type { MEAT, FISH, OTHER }
} // end class Dish
```

Running example: stream (1)

```
import java.util.stream.*;

// nel main
List<Dish> menu = Arrays.asList(
new Dish("pork", false, 800, Dish.Type.MEAT),
new Dish("beef", false, 700, Dish.Type.MEAT),
...
new Dish("salmon", false, 450, Dish.Type.FISH) );

import static java.util.stream.Collectors.toList;

List<String> threeHighCaloricDishName =
    menu.stream()
        .filter(d -> d.getCalories() > 300)
        .map(Dish::getName) // equivale a map(d -> d.getName())
        .limit(3)
        .collect(toList());
// stream() ottiene uno stream dalla lista
// filter() prende come argomento un Predicate<T>
// map() prende come argomento una Function<T, R>
// limit() prende i primi 3 elementi dello stream
// stream, filter, map, limit: intermediate operations, rest. uno stream
// collect: terminal operation
```

Running example: stream (2)

```
// gli stream si consumano, quindi si possono usare una sola volta:
List<String> title = Arrays.asList("Java8", "In", "Action");
Stream<string> s = title.stream();
s.forEach(System.out::println);
// equivale s.forEach(d -> System.out.println(d))
s.forEach(System.out::println);
// java.lang.IllegalStateException!!!

// aggiungiamo delle stampe al primo esempio 'threeHighCaloricDishName:
List<String> threeHighCaloricDishName =
    menu.stream()
        .filter(d -> { System.out.println("filter"+d.getName();)
                       return.getCalories() > 300;
                     }
                )
        .map(d -> { System.out.println("filter"+d.getName();)
                    return d.getName();
                  }
             )
        .limit(3)
        .collect(toList());

// COSA STAMPA? A ogni passo viene eseguita una filter + una map
// Tecnica di ottimizzazione detta "loop fusion"
```