Methoden-Referenzen

Carsten Gips (HSBI)

Beispiel: Sortierung einer Liste

```java
List<Studi> sl = new ArrayList<Studi>();

// Anonyme innere Klasse
Collections.sort(sl, new Comparator<Studi>() {
    @Override public int compare(Studi o1, Studi o2) {
        return Studi.cmpCpsClass(o1, o2);
    }
});


// Lambda-Ausdruck
Collections.sort(sl, (o1, o2) -> Studi.cmpCpsClass(o1, o2));

// Methoden-Referenz
Collections.sort(sl, Studi::cmpCpsClass);
```

Hinweis: Klassen- vs. Instanz-Methoden

Methoden-Referenz 1: Referenz auf statische Methode

```java
public class Studi {
    public static int cmpCpsClass(Studi s1, Studi s2) {
        return s1.getCredits() - s2.getCredits();
    }

    public static void main(String... args) {
        List<Studi> sl = new ArrayList<Studi>();

        // Referenz auf statische Methode
        Collections.sort(sl, Studi::cmpCpsClass);

        // Entsprechender Lambda-Ausdruck
        Collections.sort(sl, (o1, o2) -> Studi.cmpCpsClass(o1, o2));
    }
}
```

Demo: methodreferences.DemoStaticMethodReference

Methoden-Referenz 2: Referenz auf Instanz-Methode (Objekt)

```java
public class Studi {
    public int cmpCpsInstance(Studi s1, Studi s2) {
        return s1.getCredits() - s2.getCredits();
    }

    public static void main(String... args) {
        List<Studi> sl = new ArrayList<Studi>();
        Studi holger = new Studi("Holger", 42);

        // Referenz auf Instanz-Methode eines Objekts
        Collections.sort(sl, holger::cmpCpsInstance);

        // Entsprechender Lambda-Ausdruck
        Collections.sort(sl, (o1, o2) -> holger.cmpCpsInstance(o1, o2));
    }
}
```

Demo: methodreferences.DemoInstanceMethodReferenceObject

Methoden-Referenz 3: Referenz auf Instanz-Methode (Typ)

```java
public class Studi {
    public int cmpCpsInstance(Studi studi) {
        return this.getCredits() - studi.getCredits();
    }

    public static void main(String... args) {
        List<Studi> sl = new ArrayList<Studi>();

        // Referenz auf Instanz-Methode eines Typs
        Collections.sort(sl, Studi::cmpCpsInstance);

        // Entsprechender Lambda-Ausdruck
        Collections.sort(sl, (o1, o2) -> o1.cmpCpsInstance(o2));
    }
}
```

Demo: methodreferences.DemoInstanceMethodReferenceType

Ausblick: Threads

```
public class ThreadStarter {
    public static void wuppie() { System.out.println("wuppie(): wuppie"); }
}


Thread t1 = new Thread(new Runnable() {
    public void run() {
        System.out.println("t1: wuppie");
    }
});

Thread t2 = new Thread(() -> System.out.println("t2: wuppie"));

Thread t3 = new Thread(ThreadStarter::wuppie);
```

Beispiel: methodreferences.ThreadStarter

Ausblick: Datenstrukturen als Streams

```
class X {
    public static boolean gtFour(int x) { return (x > 4) ? true : false; }
}

List<String> words = Arrays.asList("Java8", "Lambdas", "PM",
        "Dungeon", "libGDX", "Hello", "World", "Wuppie");

List<Integer> wordLengths = words.stream()
        .map(String::length)
        .filter(X::gtFour)
        .sorted()
        .collect(toList());
```

Beispiel: methodreferences.CollectionStreams

Wrap-Up

Seit Java8: **Methoden-Referenzen** statt anonymer Klassen (**funktionales Interface nötig**)

- Drei mögliche Formen:
  - Form 1: Referenz auf statische Methode: `ClassName::staticMethodName`
    (verwendet wie `(args) -> ClassName.staticMethodName(args)`)
  - Form 2: Referenz auf Instanz-Methode eines Objekts: `objectref::instanceMethodName`
    (verwendet wie `(args) -> objectref.instanceMethodName(args)`)
  - Form 3: Referenz auf Instanz-Methode eines Typs: `ClassName::instanceMethodName`
    (verwendet wie `(o1, args) -> o1.instanceMethodName(args)`)
- Im jeweiligen Kontext muss ein passendes funktionales Interface verwendet werden

LICENSE