UNIVERSITÄT
LEIPZIG

Einführung in die Objekt-Orientierte Modellierung und Programmierung

Wintersemester 2025/2026

Dirk Zeckzer

Institut für Informatik

Teil XXI

Functional Interface

Functional Interfaces

Parameter von Methoden

- Bisher
  - Typ ist primitiver Datentyp
  - Typ ist Klasse
  - Typ ist Interface
- Neu: Methoden als Parameter von Methoden
- Diese Methoden haben
  - Keine, ein oder mehrere Parameter
  - Keine oder eine Rückgabe

Functional Interfaces: Beispiel

```
1     private static List<Integer> createListOfSquares(
2       final int n
3     ) {
4       final List<Integer> listOfSquares = new ArrayList<>();
5
6       for (int i = 1;
7            i <=  n;
8            ++i) {
9         listOfSquares.add(i * i);
10      }
11
12      return listOfSquares;
13    }
```

```
1     private static List<Integer> createListOfPowerToThree(
2       final int n
3     ) {
4       final List<Integer> listOfPowerToThree = new ArrayList<>();
5
6       for (int i = 1;
7            i <= n;
8            ++i) {
9         listOfPowerToThree.add(i * i * i);
10      }
11
12      return listOfPowerToThree;
13    }
```

Functional Interfaces: Beispiel

```
1     private static List<Integer> createListOfSquares(
2       final int n
3     ) {
4       final List<Integer> result = new ArrayList<>();
5
6       for (int i = 1;
7            i <=  n;
8            ++i) {
9         result.add(i * i);
10      }
11
12      return result;
13    }
```

```
1     private static List<Integer> createListOfPowerToThree(
2       final int n
3     ) {
4       final List<Integer> result = new ArrayList<>();
5
6       for (int i = 1;
7            i <= n;
8            ++i) {
9         result.add(i * i * i);
10      }
11
12      return result;
13    }
```

Functional Interfaces

- **Functional Interfaces**
  - Interface
  - Enhält nur eine Methode
  - Annotation: `@FunctionalInterface`
- Interface

  `java.util.function.Function<T,R>`
  - Verwendung als Typ
  - Erwartet eine Methode mit
    - einem Parameter vom Typ `T`
    - einem Return-Typ `R`
  - Methode des Interfaces:

    `R apply(T t)`

```
1    @FunctionalInterface
2    public interface Function<T,R> {
3      R apply(T t);
4    }
```

Functional Interfaces

- **Functional Interfaces**
  - Interface
  - Enhält nur eine Methode
  - Annotation: `@FunctionalInterface`
- Interface
  `java.util.function.Function<T,R>`
  - Verwendung als Typ
  - Erwartet eine Methode mit
    - einem Parameter vom Typ `T`
    - einem Return-Typ `R`
  - Methode des Interfaces:
    `R apply(T t)`
- Squares: `i * i`
  - Typ des Parameters: `Integer`
  - Typ der Rückgabe: `Integer`
- PowerOfThree: `i * i * i`
  - Typ des Parameters: `Integer`
  - Typ der Rückgabe: `Integer`
- `Function<Integer,Integer>`

Functional Interfaces: Beispiel

```java
public static List<Integer> createList(
 final int n,
 final Function<Integer,Integer> funktion
) {
 final List<Integer> result = new ArrayList<>();

 for (int i = 1;
      i <=  n;
      ++i) {
  Integer number = funktion.apply(i);
  result.add(number);
 }

 return result;
}
```

```java
public static Integer square(
 final Integer i
) {
 return i * i;
}

public static Integer powerToThree(
 final Integer i
) {
 return i * i * i;
}
```

```java
List<Integer> listOfSquares
 = createLists.createList(MAXIMAL_NUMBER,
                          CreateLists::square
                         );


List<Integer> listOfPowerToThree
 = createLists.createList(MAXIMAL_NUMBER,
                          CreateLists::powerToThree
                         );
```

Functional Interfaces: λ

```
1   private static List<Integer> createList(
2     final int n,
3     final Function<Integer,Integer> funktion
4   ) {
5     final List<Integer> result = new ArrayList<>();
6
7     for (int i = 1;
8          i <=  n;
9          ++i) {
10      Integer number = funktion.apply(i);
11      result.add(number);
12    }
13
14    return result;
15  }
```

```
1   final List<Integer> listOfSquares
2     = createList(MAXIMAL_NUMBER,
3                  (Integer i) -> {return i * i;}
4                 );
5
6   final List<Integer> listOfPowerToThree
7     = createList(MAXIMAL_NUMBER,
8                  (Integer i) -> {return i * i * i;}
9                 );
```

Functional Interfaces: λ

- λ-Ausdrücke werden verwendet, wenn die Funktionalität
  - nur einmal benötigt wird
  - einfach ist

Functional Interfaces: λ

- Methode

```
1  public static Integer square(
2   final Integer i
3  ) {
4   return i * i;
5  }
```

- λ-Ausdruck: Lange Variante

```
1  Function<Integer, Integer> funktion
2    = (Integer i) -> {return i * i;}
```

- λ-Ausdruck: Kurzform

```
1  Function<Integer, Integer> funktion
2    = i -> (i * i);
```