Einführung in die Functionale Programmierung

Was ist Funktionale Programmierung?

= Programmieren mit mathematischen Funktionen (mit Ausdrücken und Funktionen, ohne Zustand und Daten zu verändern)

Aus was bestehen pure Funktionale Programme?

- Werte und Typen
- Funktionsdefinitionen
- Funktionsanwendungs-Ausdrücke

```
def add(x: Int, y: Int): Int = x + y

val result = add(3, 5)  // Function Application Expression
println(result)  // 8
```

- High-Order Funktionen = Funktion, die eine andere Funktion als Parameter nimmt oder eine Funktion zurückgibt.

```
def applyFunction(f: Int => Int, x: Int): Int = f(x)

val square: Int => Int = x => x * x
val result = applyFunction(square, 5)  // square(5) -> 25

println(result)  // 25
```

```
def multiplyBy(factor: Int): Int => Int = x => x * factor

val double = multiplyBy(2)  // Gibt eine Funktion zurück, die mit 2
multipliziert
val result = double(10)  // 10 * 2 = 20

println(result)  // 20
```

// compose -> rechts nach links

- Funktionskomposition = Verkettung von Funktionen

```
val addOne: Int => Int = x => x + 1
val square: Int => Int = x => x * x
```

```
val addOneThenSquare = square.compose(addOne)  // Zuerst addOne, dann square
val result = addOneThenSquare(4)  // (4 + 1)² = 25

println(result)  // 25
```

```
// andThen -> links nach rechts
val squareThenAddOne = square.andThen(addOne)  // Zuerst square, dann addOne
val result2 = squareThenAddOne(4)  // 4² + 1 = 17

println(result2)  // 17
```

```
// Mischung
val multiplyByTwo: Int => Int = x => x * 2

val composedFunction = addOne.compose(square).andThen(multiplyByTwo)
val result3 = composedFunction(3)  // ((3²) + 1) * 2 = 20

println(result3)  // 20
```

Was unterscheidet Funktionale Programmierung von Imperativer Programmierung und Objektorientierter Programmierung?

- kein Konzept eines Wertespeichers -> keine zentrale Speicherstelle, die Werte speichert/verändert
- Keine Zeiger oder Referenzen
- Keine Zuweisungen und Speicheränderungen -> Variablen sind unveränderlich
- Keine Seiteneffekte -> Funktionen geben immer das gleiche Ergebnis zurück für die gleichen Eingaben
- Keine Anweisungen, keine Anweisungssequenzen
- Nur Ausdrücke -> alles ist ein Ausdruck, der einen Wert zurückgibt

Welche Vorteile haben pure Funktionale Programme?

- Parallele Ausführung
- Lazy Evaluation
- Einfache Testbarkeit
- Komposition von Funktionen
- Memoization - bereits berechnete Ergebnisse werden gespeichert und wiederverwendet