



Bern University of Applied Sciences • Engineering and Information Technology

Scala

Stefan Heinemann

08. November 2011

Enkitung
Mallo Witt
Oppktorinstiert
Funktioniah Programminung
Typinforunz

Inhalt

Inhalt

Einleitung

Hallo Welt

Objektorientiert

Funktionale Programmierung

Einleitung

- 1. OO geht so weit, dass es keine primitiven gibt
- 2. Funktional erklären!
- 3. Elegantere Problemlösung

- objektorientiert und funktional
- statisch typisiert
- ▶ Bytecode → JVM
- ► Integration mit Java
- Kompilierbar
- Scriptsprache





1. Variablen können nur einen bestimmen Wert annehmen

- objektorientiert und funktional
- statisch typisiert
- ▶ Bytecode → JVM
- ► Integration mit Java
- Kompilierbar
- Scriptsprache





- objektorientiert und funktional
- statisch typisiert
- ightharpoonup Bytecode ightarrow JVM
- ▶ Integration mit Java
- Kompilierbar
- Scriptsprache

Einleitung statisch typisiert Bytecode → JVM Integration mit Java

- objektorientiert und funktional
- statisch typisiert
- ightharpoonup Bytecode ightarrow JVM
- ▶ Integration mit Java
- Kompilierbar
- Scriptsprache

Einleitung

Einleitung

statisch typisiert
 Bytecode → JVM
 Integration mit Java

- objektorientiert und funktional
- statisch typisiert
- $\qquad \qquad \textbf{Bytecode} \to \textbf{JVM}$
- ▶ Integration mit Java
- Kompilierbar
- Scriptsprache

Einleitung

objektorientiert und funktional

statisch typisiert

Bytecode -- JVM

Integration mit Java

Kompilierbar

Scriptsprache

- objektorientiert und funktional
- statisch typisiert
- $\blacktriangleright \ \mathsf{Bytecode} \to \mathsf{JVM}$
- ▶ Integration mit Java
- Kompilierbar
- Scriptsprache



- 1. Obligatorisches Hallo Welt
- 2. Unvermeidlich, Code zu zeigen
- 3. object = Singleton-Objekt, keine statischen Sachen
- 4. Singleton wird beim ersten Aufruf erzeugt
- 5. Syntax ähnlich wie Java
- 6. Datentypen werden hinter die Deklaration gestellt
- 7. Main-Methode wie in Java

Hallo Welt

```
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello World!")
  }
}
```



- 1. Da Objektorientierte Programmierung bekannt, nicht interessant das ganze zu erklären, darum ein Beispiel
- 2. Erklären was wo ist
- 3. Abstract classes

Klassen

```
class foo(var x:Int, y:Int) extends bar(y) {
    if (y > 0) {
      throw new Exception("Invalid Argument");
    var member:String = "Ich bin drin!"
    override def toString(): String =
       "foo says hello"
10
    private def doSomething() {
      println("something stupid?")
13
14
```



- ► Keine Interfaces
- ► Keine Mehrfachvererbung



► Keine Interfaces

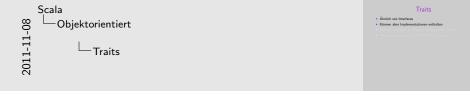
Traits

► Keine Mehrfachvererbung





- ► Ähnlich wie Interfaces
- ► Können aber Implementationen enthalten
- ▶ Können mehrfach an Klassen weitervererbt werden
- ▶ Weniger problematisch als Mehrfachvererbung



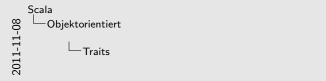
- 1. Ähnlich zu abstrakte Klassen
- 2. Abstrakte Klassen können nicht mehrfach vererbt werden

- ► Ähnlich wie Interfaces
- ► Können aber Implementationen enthalten
- ► Können mehrfach an Klassen weitervererbt werden
- ► Weniger problematisch als Mehrfachvererbung



- Können mehrfach an Klassen weitervererbt werden

- ► Ähnlich wie Interfaces
- ► Können aber Implementationen enthalten
- ► Können mehrfach an Klassen weitervererbt werden
- ► Weniger problematisch als Mehrfachvererbung



- Können mehrfach an Klassen weitervererbt werder
- Weniger problematisch als Mehrfachvererbung

- ► Ähnlich wie Interfaces
- ► Können aber Implementationen enthalten
- ▶ Können mehrfach an Klassen weitervererbt werden
- ▶ Weniger problematisch als Mehrfachvererbung





- ► Funktionen sind auch Objekte
- ► Speichern in Variablen
- ▶ Übergeben an Funktionen
- ▶ Rückgabewerte
- Callbacks
- ► Anonym möglich



Funktionsobjekte

Funktionen sind auch Objekte

Speichern in Variablen

- ► Funktionen sind auch Objekte
- ► Speichern in Variablen
- ▶ Übergeben an Funktionen
- ▶ Rückgabewerte
- Callbacks
- ► Anonym möglich



Funktionsobjekte

- Funktionen sind auch Objekte

- Spichern in Variablen

- Übergeben an Funktionen

- ► Funktionen sind auch Objekte
- ► Speichern in Variablen
- ▶ Übergeben an Funktionen
- ▶ Rückgabewerte
- Callbacks
- Anonym möglich

Funktionsobjekte
Funktionen sind auch Objekte
Speichern in Variablen
Übergeben an Funktionen
Rückgabewerte

- ► Funktionen sind auch Objekte
- ► Speichern in Variablen
- ▶ Übergeben an Funktionen
- Rückgabewerte
- Callbacks
- Anonym möglich

- 1. in Java ist dies nicht möglich
- 2. Callbacks mit anonymen klassen

- ► Funktionen sind auch Objekte
- ► Speichern in Variablen
- ▶ Übergeben an Funktionen
- ▶ Rückgabewerte
- Callbacks
- Anonym möglich

Scala

Funktionale Programmierung

Funktionsobjekte

Funktionsobjekte
unktionen sind auch Objekte
peichern in Variablen
bergeben an Funktionen

Anonym möglich

- ► Funktionen sind auch Objekte
- ► Speichern in Variablen
- ▶ Übergeben an Funktionen
- Rückgabewerte
- Callbacks
- ► Anonym möglich

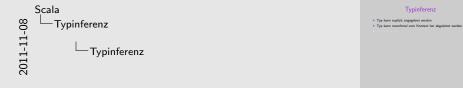


Pattern Matching





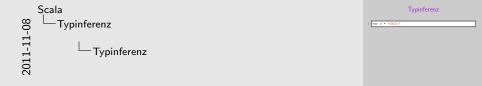
- ► Typ kann explizit angegeben werden
- ▶ Typ kann manchmal vom Kontext her abgeleitet werden
- Kürzerer Code



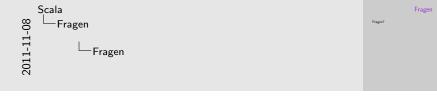
- ► Typ kann explizit angegeben werden
- ▶ Typ kann manchmal vom Kontext her abgeleitet werden
- Kürzerer Code



- **Typinferenz**
- ► Typ kann explizit angegeben werden
- ► Typ kann manchmal vom Kontext her abgeleitet werden
- ► Kürzerer Code



```
var x = "hallo"
```



Fragen

Fragen?