# Neues in Java 8

Sprach-Merkmale

# Lambda-Ausdrücke

- Ermöglichen Weitergabe von Code (d.h. „Verhalten“, „Logik“) an anderen Code
- … zur späteren Ausführung
- … in anderem Kontext.
- Konzept aus der Funktionalen Programmierung
- Lambda-Ausdrücke sind *functional interfaces* bzw. *single abstract method types (SAM)*

# Exkurs „Funktionalen Programmierung“

- Programmiersprachen-Paradigma, vgl. „Objektorientierte Programmierung“ oder „Strukturierte Programmierung“
- Die *Funktion* ist das zentrale Konstrukt der Programmiersprache („first class citicen“)
- Konzept der Funktion stark an Mathematik angelehnt (siehe auch „Abbildung“)

$$f(x) = y; \qquad x \rightarrow y;$$

# Exkurs „Funktionalen Programmierung“

- Merkmale einer Funktion
  - Eine Funktion ist eine Abbildung
  - Eine Funktion bildet Eingangswerte auf einen Ausgangswert ab.
  - Eine Funktion hat keine Seiteneffekte, sondern nur Eingabe- und Ausgabewerte
  - Eine Funktion hat keinen Zustand(!)

# Lambda-Ausdrücke

- Syntax: `(Parameterliste) -> {code}`
- Kurzformen der Syntax – dank Typ-Inferenz:
  - Wie bei Generics, außerdem..
  - Falls code ein (einziger) Ausdruck ist, können die geschweiften Klammern entfallen.
  - Falls code ein (einziger) Ausdruck ist, kann das Schlüsselwort return entfallen.
  - Falls nur ein Parameter existiert, kann die runde Klammer entfallen.
- Vereinfacht: *„Alles, was mit Inferenz erschlossen werden kann, kann entfallen!“*

# Methodenreferenzen

- Eine Methodenreferenz kann verweisen auf
  - Eine Methode
  - Einen Konstruktor
- Information über zu übergebende Parameter wird vom Compiler anhand der Signatur erschlossen(!)
  → Inferenz!

# Functional Interfaces

- Neues Sprach-Feature in Java 8
- Basistyp für Lambda-Ausdrücke
- Ist nicht von java.lang.Object abgeleitet(!)
- Alle Interfaces mit *einer einzigen abstrakten Methode* sind Functional Interfaces
  (Vgl. SAM type, Single Abstract Method type)
- Die optionale Annotation @FunctionalInterface erlaubt Prüfung durch den Compiler

# Single Abstract Method type (SAM)

- Interface mit genau einer *abstrakten* Methode
- Zusätzliche statische Methoden sind erlaubt
- Methoden mit default-Implementierung sind erlaubt
- Nicht neu in Java 8, es gibt viele ältere SAM-Typen: Runnable, Callable<T>, Comparator<T> etc. …
- Auch Methoden aus java.lang.Object dürfen deklariert werden – sie werden ja von jeder Klasse implementiert!

# Methoden-Implementierungen im Interface

- Default-Methoden
  - „default“-Schlüsselwort
  - Mehrere Default-Methoden in einem Interface sind möglich
- Statische Methoden
  - … ACHTUNG!
  - Konzept „Interface“ wird verwässert!
  - Abhängigkeiten?
    Schnittstelle wird schlechter wiederverwendbar!

# Reflection für Methoden-Parameter

- java.lang.reflect.Executable.getParameters() in
  - Method
  - Constructor
- Achtung! Parameter-Namen werden standardmäßig nicht im Byte-Code der .class-Dateien gespeichert
- Übersetzung mit Option `-parameters` ist erforderlich (siehe javac)

# Compact Profiles

- Java SE, komplette API:
  Beans, JNI, JAX-WS, Preferences, Accessibility, IDL, RMI-IIOP, CORBA, Print Service, Sound, Swing, Java 2D, AWT, Drag and Drop, Input Methods, Image I/O
- compact3
  Security[1], JMX, XML JAXP[2], Management, Instrumentation
- compact2
  JDBC, RMI, XML, JAXP

# Compact Profiles

- compact1
  Core (java.lang.*), Security, Serialization, Networking, Ref Objects Regular Expressions, Date and Time, Input/Output, Collections, Logging, Concurrency, Reflection, JAR, ZIP, Versioning, Internationalization, JNDI, Override Mechanism, Extension Mechanism, Scripting

# Neues in Java 8
# Sprach-Merkmale

Vielen Dank für Ihre Aufmerksamkeit!
Noch Fragen?

# Quellen

- API: https://docs.oracle.com/javase/8/docs/api/index.html
- Oracle Java 8 Release Notes

  http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html
-