Java-Einführungskurs

Informatik II für D-ITET
FS 2016, ETH Zürich

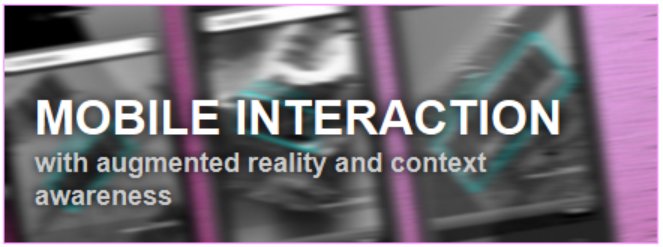
Hossein Shafagh
shafagh@inf.ethz.ch

Was haben wir heute vor?

- Vorbereitung auf die Übungen zu Informatik II
  - Vorstellung des Teams
  - Organisatorisches
- Theorie
  - Java-Technologie und Sprache
  - Unterschiede: C++ / Java
- Praxis: Übungsblatt 0
  - HelloWorld.java
  - Erste Schritte mit Eclipse
  - JUnit4 für automatisiertes Testen

institute for pervasive computing

RESEARCH GROUP FOR
Distributed Systems



Distributed Systems Group

Institute for Pervasive Computing

Hossein Shafagh
http://people.inf.ethz.ch/mshafagh

+

Hossein Shafagh
http://people.inf.ethz.ch/mshafagh

Übungsgruppen

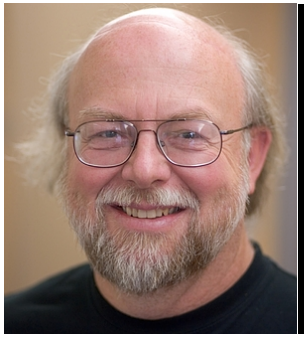
- Mi (13-14 Uhr) & Do (13-14 Uhr)
  - Überschneidung mit KA
- Tragt euch für Gruppen ein (258/286 bereits eingetragen)
- Ein paar Gruppen sind auf Englisch
  - Englisch können oder lernen wollen (gute Übung!)
- Skripte
- Testate
- Anwesenheit in den Übungsgruppen

Fragen & Interaktion!!!

What programming language should I use?


Java™

Warum Java?



"Java is C++ without the guns, knives, and clubs."

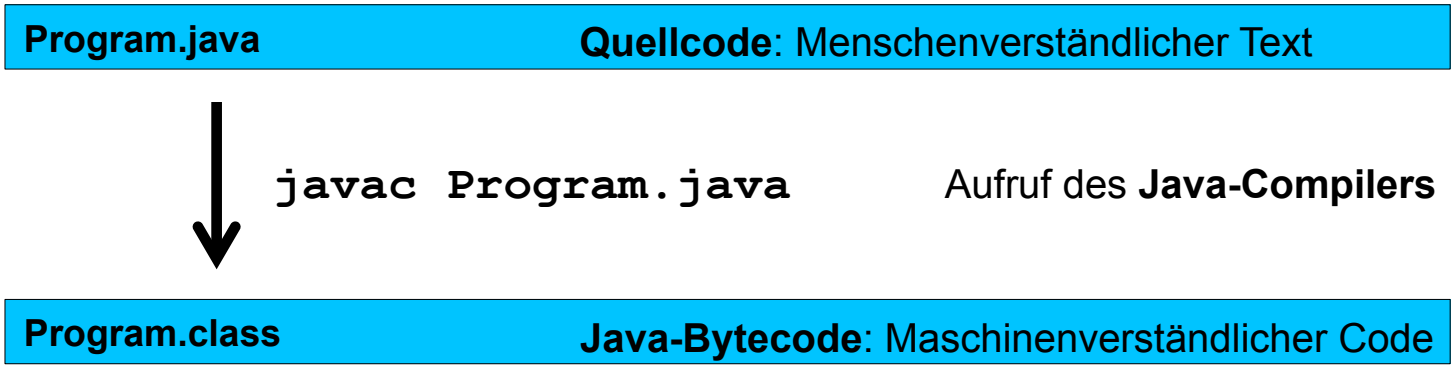
- James Gosling

- Objektorientiert
- „Einfacher" als C++
- Umfangreiches Ökosystem: Tools, Bibliotheken, ...
- Virtuelle Maschine: „Compile once – Run everywhere"

Werdegang eines Java-Programms

Program.java	**Quellcode**: Menschenverständlicher Text

Werdegang eines Java-Programms



Werdegang eines Java-Programms

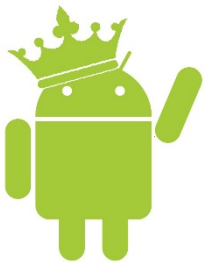
Program.java — **Quellcode**: Menschenverständlicher Text

↓ `javac Program.java` — Aufruf des **Java-Compilers**

Program.class — **Java-Bytecode**: Maschinenverständlicher Code

↓ `java Program` — Ausführen mittels **virtueller Maschine**

Plattformunabhängigkeit: Java-Bytecode ist ohne Änderung auf jeder Architektur lauffähig, auf welcher eine Laufzeitumgebung installiert ist.




Werdegang eines Java-Programms

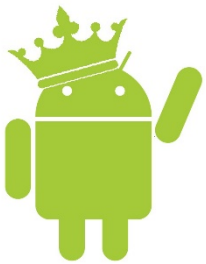
Program.java — **Quellcode**: Menschenverständlicher Text

↓ `javac Program.java` — Aufruf des **Java-Compilers**

Program.class — **Java-Bytecode**: Maschinenverständlicher Code

↓ `java Program` — Ausführen mittels **virtueller Maschine**

Plattformunabhängigkeit: Java-Bytecode ist ohne Änderung auf jeder Architektur lauffähig, auf welcher eine Laufzeitumgebung installiert ist.




Hello World!


```java
/**
 * Ein Programm
 */
public class Program {

	public static void main (String [] args) {
		System.out.println("Hello");
	}

}
```

Die Java-Technologie

- Java-Laufzeitumgebung (JRE):
  - Hauptbestandteil ist das Programm *java*
  - Java Virtual Machine (JVM)
  - Standardklassen und weitere Programmbibliotheken
  - JRE-Editionen: Java SE, Java ME, Java EE

JRE

- Java-Entwicklungswerkzeug:
  - Enthält die Programme *java*, *javac*, *javap*, ...
  - Enthält die JRE

JDK

Java-Sprache: Versionen

- JDK 1.0 (1996)
- JDK 1.1 (1997, z.B. Paketierung als .jar-Dateien)
- J2SE 1.2 (1998, z.B. Just-In-Time Compiler, Grafik)
- J2SE 1.3 (2000)
- J2SE 1.4 (2002, z.B. + Assertions und Server)
- Java 5.0 (2004, z.B. + Generics, Annotationen)
- Java 6.0 (2006)
- **Java 7.0** (2011, z.B. neue Filesystem-API, IPv6)
- Java 8.0 (2014)
- Java 9.0 (geplant für 2017, z.B. multi-GB heaps)

Java-Ökosystem

- Standardbibliothek
  - Datenstrukturen (List, Map, ...), Algorithmen (Sort, ...), Kryptographie, Kommunikation, graphische Benutzerschnittstellen

HelloMachine.java

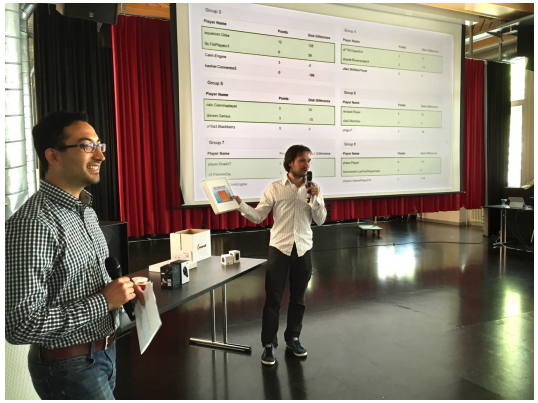
```
/**
 *  Demo: static void main,
 *        Argumente, Objekte, Objektmethoden, printout
 */
public class HelloMachine {

        public static void main(String [] args) {
                String userName = args[0];

                HelloMachine myHelloMachine = new HelloMachine();

                myHelloMachine.sayHello(userName);
        }

        private void sayHello(String name) {
                System.out.println("Hello, " + name + "!");
        }
}
```

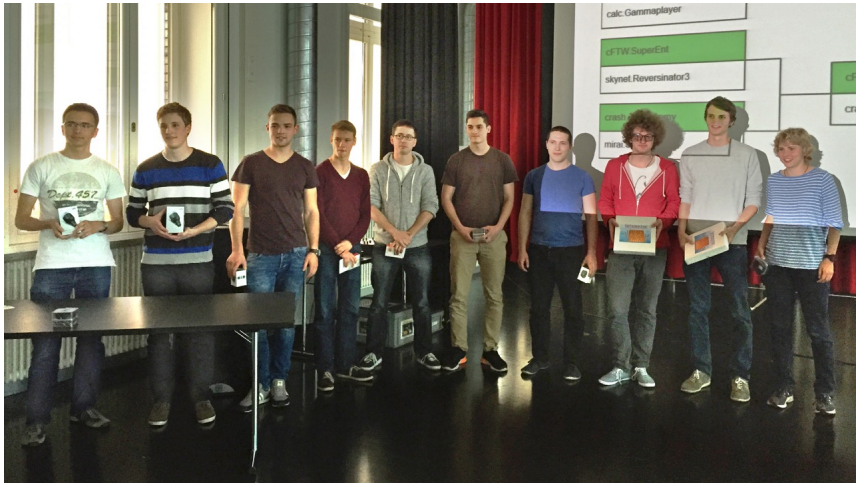
Beachten: Funktionssignaturen, static void main, Argumente, Objekte, Objektmethoden, printout

Java-Ökosystem

- Standardbibliothek
  - Datenstrukturen (List, Map, ...), Algorithmen (Sort, ...), Kryptographie, Kommunikation, graphische Benutzerschnittstellen
- Unzählige Zusätzliche Bibliotheken
  - Datenbanken, Web-Server, ...
  - Reversi (nur bei uns!)

Reversi-Turnier (letzte Vorlesungswoche)





Hossein Shafagh
http://people.inf.ethz.ch/mshafagh

Java-Ökosystem

- Standardbibliothek
  - Datenstrukturen (List, Map, ...), Algorithmen (Sort, ...), Kryptographie, Kommunikation, graphische Benutzerschnittstellen
- Unzählige Zusätzliche Bibliotheken
  - Datenbanken, Web-Server, ...
  - Reversi (nur bei uns!)
- Integrierte Entwicklungsumgebung (IDE)
  - Editor + Compiler + Debugger + Projektverwaltung + …
  - Beispiel: **Eclipse**, NetBeans, IntelliJIDEA, …

Java-Ökosystem

- Javadoc
  - Dokumentation durch *strukturierte* Kommentare

```
/**
 * Adds a value to at the beginning of a list.
 *
 * @param list the list to which the value is added
 * @param value the value which is added to the list
 * @return a new list with the new element first followed by the given list
 */
public static List add(List list, int value)
{
    return new List(value, list);
}
```

Interface Operation

```
public interface Operation
```

Simple calculator operation.

Version:
1.0
Author:
Me

Method Summary

void	`calculate(double operand)` Perform a single calculation.
double	`getResult()` Get the current result.

Method Detail

calculate

```
void calculate(double operand)
```

Perform a single calculation.

Parameters:
`operand` - the operand to use for calculation.

---

getResult

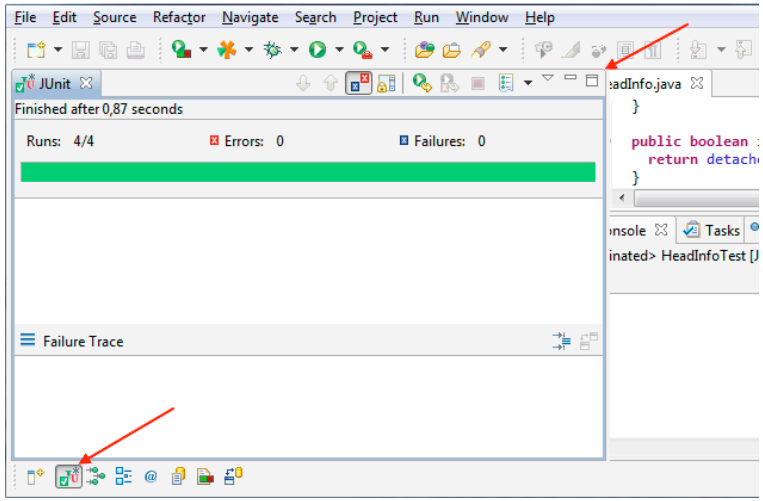
```
double getResult()
```

Get the current result.

Returns:
the current result. If no calculations were performed the result is undefined.

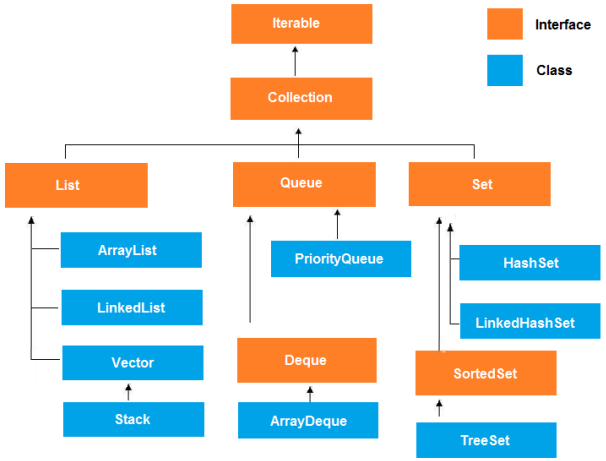
Java-Ökosystem

- Unit Testing (JUnit)
  - Bestandteil aller Übungen
  - Automatisiertes Testen des Codes
  - Generierung von Testberichten



Übersicht

- Java-Ökosystem
- Pakete, Organisation von Java-Code
- Zugriffsrechte
- Typen, Vererbung, Polymorphismus
- Fehlerbehandlung, Stack Traces



Java-Sprache: Pakete

- Klassen können Teil eines „Pakets“ sein
  - Definition in Hello.java: `package myPackage;`
  - Navigation per Punkt:
    `ch.ethz.itet.info2.myPackage.Hello`
- Nutzen
  - Vermeidung von Namenskollisionen
  - Kompakterer, einfacher lesbarer Code
- Pakethierarchie wird auf Verzeichnisbaum abgebildet
  `ch.ethz.itet.info2.myPackage.Hello`
  wird zu: `ch/ethz/itet/info2/myPackage/Hello.java`

Java-Sprache: Organisation

- Jede öffentliche Klasse steht in ihrer eigenen **gleichnamigen** *.java*-Datei

  Öffentliche Klasse: public class HelloMachine { … }
  Nicht-öffentliche Klasse: private class HelloHelper { … }

- Pro (öffentliche) Klasse generiert *javac* eine *.class*-Datei

Demo: Organisation, Pakete, Zugriff

- Public.java
  - Klasse „Public“ in Package „demo1“
  - public static void main() { … }
  - Benutzt Klasse ExtendedPublic

- ExtendedPublic.java
  - Öffentliche Klasse „ExtendedPublic“
  - + Private Klasse „Private“
  - In Paket „demo2“
  - public void foo() { … }

Java-Sprache: Bibliotheken

- Sammeln von Paketen in .jar-Dateien („java archives“)
- Zugriff auf Bibliothek
  - Bekanntmachen des Namens: `import …`
  - Namen aus dem eigenen Paket sind immer bekannt
- Standardbibliothek steht automatisch zur Verfügung und muss nicht importiert werden
  - Dokumentation: *http://docs.oracle.com/javase/7/docs/api/*

Java-Sprache: Zugriffsrechte

- `public`
- `protected`
- `package`
- `private`

- Keine `friends`

Java-Sprache: Primitive Typen

- Primitive Typen können auf dem Stack angelegt werden, ihre Instanzen sind **keine Objekte**!

  `boolean`

  `byte`, `short`, `int`, `long`

  `float`, `double`

  `char`

- Korrespondierende Klassen, z.B. `Integer`, `String`
  - Werden, wie alle Objekte, als Referenzen by Value übergeben (mehr dazu in Übung 3) und am Heap allokiert

Java-Sprache: Überall Objekte!

- Objekt: Instanz einer Klasse
- Zugriff ausschliesslich über Referenzen!
- Erzeugung mit `new`

```
ExtendedPublic eP = new ExtendedPublic();
```

- Entfernung durch Garbage Collector, **kein `delete`!**

Java-Sprache: Vererbung

- Java bietet keine Mehrfachvererbung!
  - Stattdessen: Schnittstellen (`interface`)
  - Eine Klasse kann mehrere Interfaces implementieren
  - Mehr dazu in Übung 6

- Polymorphismus: Funktionen sind grundsätzlich virtuell!
  - Wenn: `Porsche extends Car`
  - Dann: `Porsche.getSpeed()` überdeckt `Car.getSpeed()`

- Funktionen und Klassen können `abstract` sein
  - Abstrakte Klassen können nicht instanziiert werden

Java-Sprache: Fehler und Stack Traces

- Stack Traces ermöglichen das Zurückverfolgen von Fehlern zu ihrem Ursprung (+ Zeilennummern)

- Siehe Demo...


```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at StackTraceDemo.method2(StackTraceDemo.java:21)
        at StackTraceDemo.method1(StackTraceDemo.java:10)
        at StackTraceDemo.main(StackTraceDemo.java:5)
```

Java-Sprache: Fehlerbehandlung

- **Error:** "indicates serious problems that a reasonable application should not try to catch."
  - Beispiel: `OutOfMemoryError`
- **Exception:** "indicates conditions that a reasonable application might want to catch."
  - Beispiel: `FileNotFoundException`

- Abfangen mit `try`/`catch`, werfen mit `throw`
  - Nicht abgefangene Exceptions (und Errors) führen zum Programmabbruch!
  - Mehr dazu in Übung 1 und 2

Übungsblatt 0: Aufgabe 1

- `HelloWorld` mit Texteditor
- Ausführen auf der Kommandozeile

- `HelloWorld` in Eclipse
  - Falls schon installiert: Super!
  - Sonst: Links zu Dokumentation auf Vorlesungswebseite

Übungsblatt 0: Aufgabe 1 und Eclipse

- Import der Übungsdaten in Eclipse
  - Zip-File entpacken
  - Übungsordner in Eclipse-Workspace kopieren und entsprechend dem Gruppennamen/Projektnamen umbennenen (z.B.: *U00G01*)
  - File – New – Java Project – [Projektnamen Eingeben] – Next – Finish

- Neue Bibliothek einbinden
  - Rechtsklick auf Projekt – Properties – Java Build Path – „Libraries“

  - Einbinden von JARs: „Add (external) JARs“
  - Einbinden von JUnit4: „Add Library“ – Juint – Version 4

- Ausführen von HelloWorld

Übungsblatt 0: Aufgabe 2

- Erstes Java-Programm: Signum-Funktion

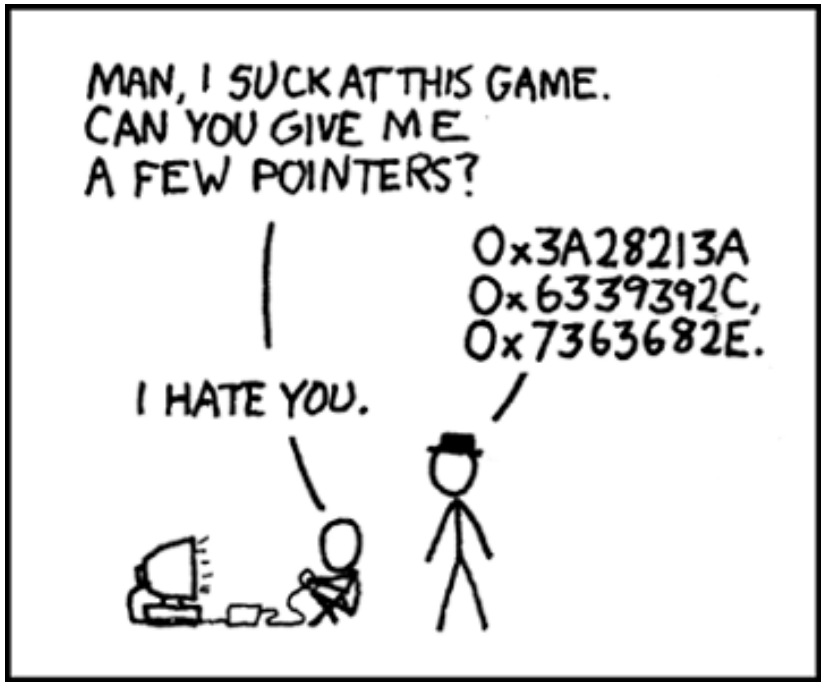
Übungsblatt 0: Aufgabe 3

- Automatisiertes Testen mit JUnit4
- … aus der Kommandozeile (wird nicht gezeigt…)
- … in Eclipse

Hossein Shafagh
http://people.inf.ethz.ch/mshafagh

Übungsblatt 0: Aufgabe 4

- Modellbildung


MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.

Fragen?