Informatik II (D-ITET)

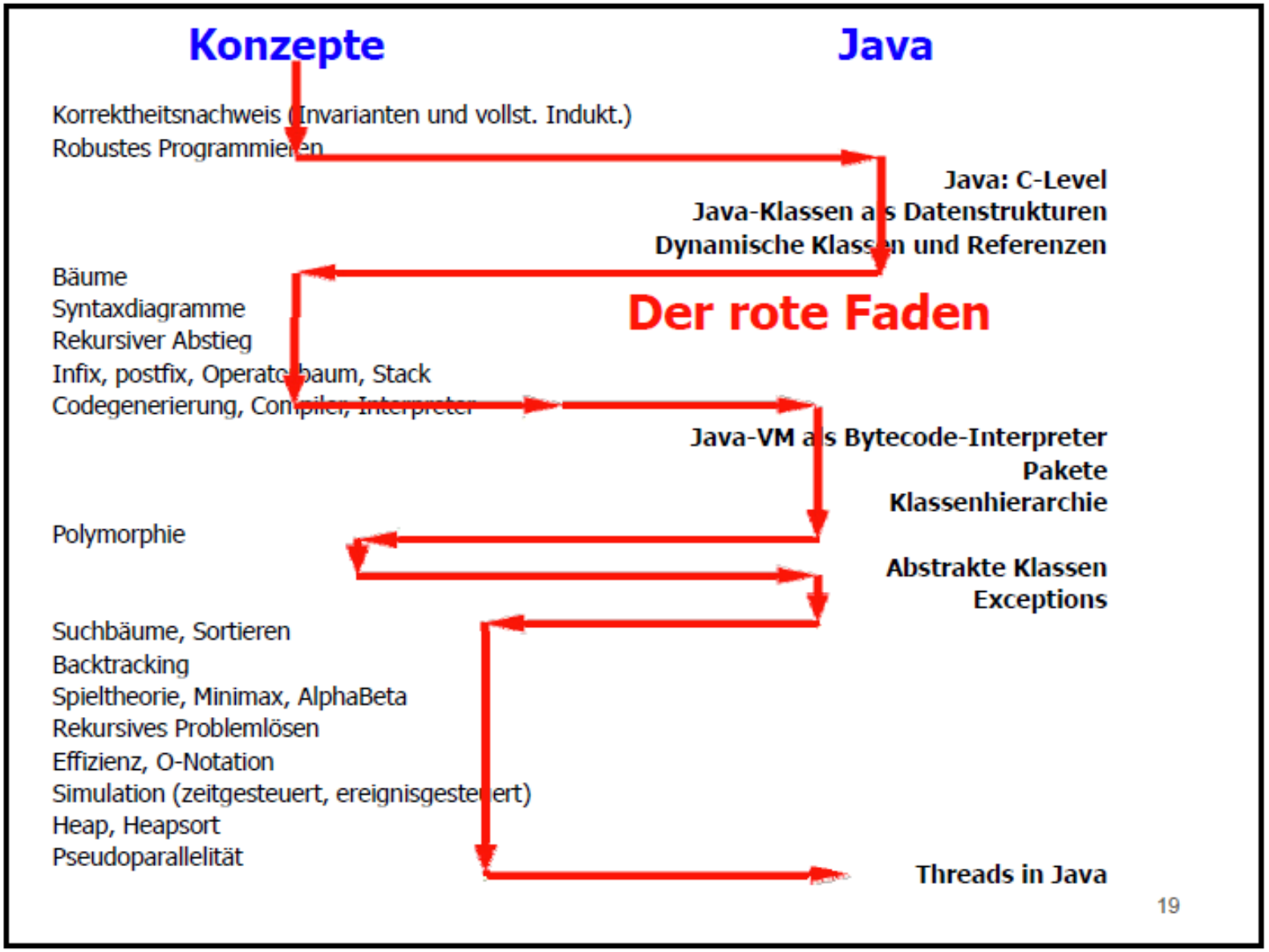
Übungsstunde 1

simon.mayer@inf.ethz.ch

Distributed Systems Group, ETH Zürich

Ablauf

- Ueberblick + Administratives
- Besprechung der Vorlesung
- Einfuehrungsthemen: Eclipse, JUnit, Serienabgabe
- Uebungsbezogenethemen: Exceptions, Javadoc,...


Konzepte
Java
Korrektheitsnachweis (Invarianten und vollst. Indukt.)
Robustes Programmieren
Java: C-Level
Java-Klassen als Datenstrukturen
Dynamische Klassen und Referenzen
Bäume
Syntaxdiagramme
Rekursiver Abstieg
Infix, postfix, Operatorbaum, Stack
Codegenerierung, Compiler, Interpreter
Der rote Faden
Java-VM als Bytecode-Interpreter
Pakete
Klassenhierarchie
Polymorphie
Abstrakte Klassen
Exceptions
Suchbäume, Sortieren
Backtracking
Spieltheorie, Minimax, AlphaBeta
Rekursives Problemlösen
Effizienz, O-Notation
Simulation (zeitgesteuert, ereignisgesteuert)
Heap, Heapsort
Pseudoparallelität
Threads in Java
19

This semester's menu...

- (Weitere) Grundlagen der Programmierung
  - Objektorientierung; Modellbildung, Formalisierung, Abstraktion
- Algorithmen (anhand von Java)
  - Suchen, Rekursion, Backtracking
  - Komplexitaet
- Datenstrukturen
  - (Spiel)Baeume, Heaps, Listen
- Weiteres: Simulation, Testen/Debugging, Parallelitaet

Administratives

- Abgabe der Übungen (**zu zweit!**)
  - So viel wie moeglich per email...
  - [v.a. theoretische Übungen handschriftlich...]
  - Spaeter mehr zur Uebungsabgabe

- Fragen zum Stoff/Vorlesung/Übungen
  - Folien online: http://people.inf.ethz.ch/mayersi/
  - Fragen: Am besten per email an mich...

Administratives

- Prüfung
  - Sowohl theoretisches Wissen (z.B. Komplexitaet von Algorithmen)
  - ...als auch «praktische» Programmierung (i.e., mit Papier und Kuli)

Testat

- Woechentliche Aufgabenblaetter
  - Insgesamt mindestens **75%** der Aufgaben bearbeitet
  - ...und mindestens **50%** der Punkte erreicht
- Punkte = Punkte der Uebungen 1 bis 11
- Uebung 12: Bonus!
- Keine verspaeteten Abgaben

Plagiate

- Guttenberg
- Schlechte Pruefungsvorbereitung...
- Graubereich
- 0 Punkte fuer Aufgabenblatt

2er Teams

- Festlegen und so...

Abgabemodalitaet

- Mittwoch!

- Programme per Email (unten mehr)
- Ansonsten: Briefkasten bei uns (CNB H Stock)

Betreute Rechnerzeiten

- Mittwoch, 1pm – 2pm
- Wieviele haben grundsaetzlich an sowas Interesse?

Ablauf

- Ueberblick + Administratives
- Besprechung der Vorlesung
- Einfuehrungsthemen: Eclipse, JUnit, Serienabgabe
- Uebungsbezogenethemen: Exceptions, Javadoc,...

Thema: Java Kurzrepetition

```
package u0a1;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Java: „Removed from C/C++“

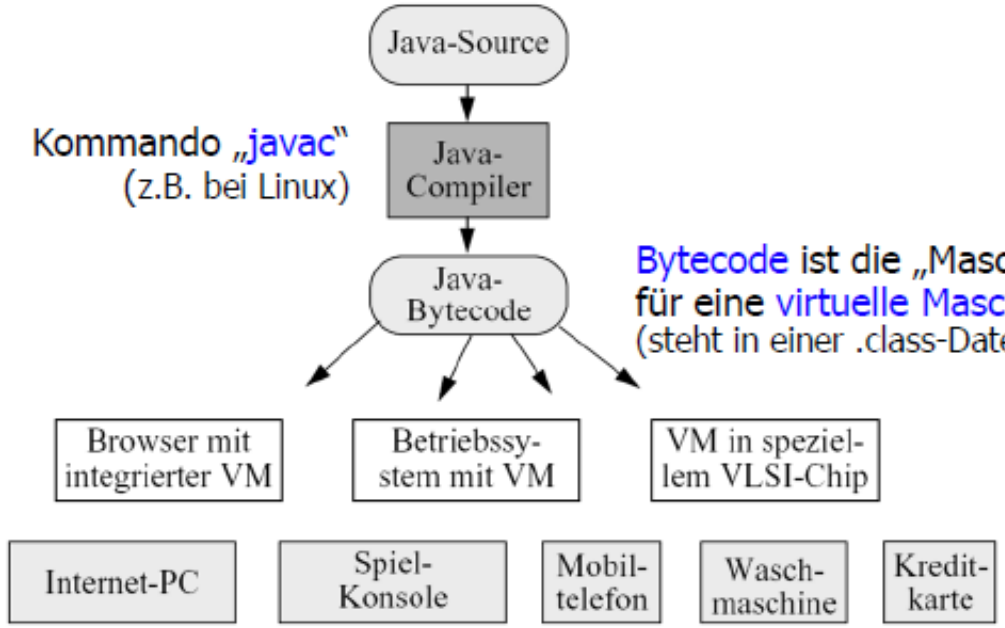
La perfection est atteinte non quand il ne reste rien à ajouter, mais quand il ne reste rien à enlever.

Antoine de Saint-Exupéry

- Templates
- Strukturen, union (statt dessen: Klassen / Objekte)
- Zeigerarithmetik, malloc (aber: Arrays und new)
- Funktionen (statt dessen: Methoden)
- Mehrfachvererbung (statt dessen: Interfaces)
- Präprozessor: TYPEDEF, ...
- #DEFINE und const (statt dessen: final)
- Überladen von Operatoren
- goto (statt dessen: break/continue, Exceptions)
- using namespace, #include, .h-Dateien (aber: Pakete)
- Destruktoren, free, delete (aber: Garbage-Collector; finalize)
- Implizite Typkonvertierung
- Friends (aber: „friendly access“ innerhalb von Paketen)
- sizeof x (statt dessen: x.length)

Plattformunabhängigkeit durch Bytecode-Interpretation

Ein Java-Programm läuft (prinzipiell) auf allen gängigen Computern und Betriebssystemen (PC, Server, Mobiltelefonen, Linux, Windows...)



Besprechung der Vorlesung

- Algorithmen
  - «altaegyptische Multiplikation» als Beispiel

	a	*wenn (b = 1);*
*a * b =*	*2a * b/2*	*wenn (b gerade);*
	*a + (2a * (b-1)/2)*	*sonst;*

  - Problem wird auf kleinere «Version» desselben Problems reduziert
    - Rekursion (Erinnerung an Info I?)

Besprechung der Vorlesung

- Java Programmbau und Ausfuehrung
  - Datei *Mult.java* mit einem Java-Programm als Inhalt...

1. **Kompilieren**: Sourcecode → Java Bytecode

   Befehl: `javac Mult.java`

   Generiert Datei «Mult.class», die den Bytecode enthaelt

2. **Aufrufen**:

   Befehl: `java Mult.class`

System.out

```
int count = 0;
while (System.in.read() != -1) count++;
System.out.println("Eingabe hat " +
   count + " Zeichen.");
```

Konkatenation von strings

System.out: Standard-Ausgabestrom

- print gibt das übergebene Argument (auf einem Display) aus
- println erzeugt zusätzlich danach noch einen Zeilenumbruch („newline")
- es können u.a. int, float, string, boolean... ausgegeben werden

Ablauf

- Ueberblick + Administratives
- Besprechung der Vorlesung
- Einfuehrungsthemen: Eclipse, JUnit, Serienabgabe
- Uebungsbezogenethemen: Exceptions, Javadoc,...

Thema: Eclipse IDE

- Integrated Development Environment
- Derzeitige Version: Helios
- Eine Menge an hilfreichen Features, z.B. Code completion,...
- Demo!
  - Neues Projekt + externen Quellen
  - Programmausfuehrung + Run Configurations

Thema: JUnit 4

Bibliothek zum automatisierten Verifizieren und Validieren von Software.

1. Wird mit Eclipse mitgeliefert (als Plugin)

2. Einbinden in Eclipse Projekt:

   Rechtsklick auf Projekt -> Build Path -> Configure Build Path
   Add Libraries...
   «JUnit»
   «Junit 4»

Thema: Uebungsabgabe

1. Archiv herunterladen und entpacken
2. Eclipse: Neues Java Projekt (Link additional sources)
3. Bearbeiten/Aufgaben loesen
4. Packen/Zippen + Abschicken

Ablauf

- Ueberblick + Administratives
- Besprechung der Vorlesung
- Einfuehrungsthemen: Eclipse, JUnit, Serienabgabe
- Uebungsbezogenethemen: Exceptions, Javadoc,...

Übung 0

- Aufgabe 1: HelloWorld.java

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

- Aufgabe 2: Signum.java + Main.java

```
public static void main(String[] args) {
  System.out.println("signum(-3) = " + Signum.signum(-3));
  System.out.println("signum(0) = " + Signum.signum(0));
  System.out.println("signum(7) = " + Signum.signum(7));
}
```

- Aufgabe 3: Automatisiertes Testen
- Aufgabe 4: Und ein bisschen «Theorie» und Modellbildung mit Wein

Übung 1

- 1. Terminierung & Korrektheit (altaegyptische Multiplikation)
- 2. Rekursion, Aufwandabschaetzung fuer Algorithmen
- 3. Exceptions, Unit-Testing, Dokumentieren von Programmen

Übung 1

- 1. Terminierung & Korrektheit (altaegyptische Multiplikation)
  - Induktionsbeweis
    - Verankerung (Beweis, dass korrekt fuer Base Case)
    - Schritt (Erweiterung des Beweises fuer Schritt i auf Schritt i+1

  a) Funktioniert Induktion auch ueber die Variable a?

  b) Terminiert der Algorithmus? Beweisen!

  c) Veraenderter Algorithmus! Base Case ist nun nicht mehr (b = 1) sondern (b = 0)! Neuer Beweis...

Übung 1

- 2. Aufwandabschaetzung fuer Algorithmen
- Wieviele Methodenaufrufe erzeugen...

```
gerade(int x)
verdopple(int x)
halbiere(int x)

f(int a, int b)          noch keine Rekursion!

f(int a, int b)          nun mit Rekursion!
```

hierfuer kann das Resultat aus b) verwendet werden...

Übung 1

- Ueberpruefen von Benutzereingaben mittels Exceptions
- Exceptions
  - An exception is an event that occurs during the execution of a program that **disrupts the normal flow** of instructions.
  - Exceptions koennen geworfen (throw) und abgefangen (catch) werden.
  - Demo!

Tutorial: http://download.oracle.com/javase/tutorial/essential/exceptions/

Übung 1

- Unit-Testing
  - Automatisiertes Verfizieren und Validieren von Code
  - JUnit4
  - Demo!

Übung 1

- Dokumentieren mittels Javadoc
  - Generieren von Javadoc fuer gesamtes Projekt:

```
Project -> Generate Javadoc
```

Gutes Tutorial:
http://homepages.fh-giessen.de/~hg7132/javaprog/uebungen/javadoc_tutorial.html

Informatik II (D-ITET)

Übungsstunde 1

simon.mayer@inf.ethz.ch

Distributed Systems Group, ETH Zürich