Fachhochschule Rosenheim
University of Applied Sciences

Programmieren 3

Martin Binder, FH Rosenheim, © 2015

Organisation

- Voraussetzung
  - Programmieren 1 und 2
- Ablauf: 2 Stunden Vorlesung, 2 Stunden Übung pro Woche
- Prüfung: Klausur, 90 Minuten,
  Hilfsmittel: Ein Fachbuch freier Wahl mit ISBN Nummer oder ein RZ-Handbuch, jeweils ohne persönliche Notizen
- Die Folien sind kein vollständiges Skript!
  Zur Prüfung sind zusätzliche persönliche Mitschriften und Literaturstudien sowie Teilnahme an den Übungen notwendig.

Inhalte von Programmieren 1 und Programmieren 2

Prg1: Grundlagen der Programmierung in C (prozedurale Programmierung)

- Grundelemente
  (Variablen, Ausdrücke, Operatoren, Ablaufstrukturen, Funktionen, Parameterübergabe, Ein- Ausgabe, Dateiverarbeitung)
- Datenstrukturen
  (Aufzählungstypen, Felder, Strukturen, Bitfelder, Zeiger, dynamische Strukturen, Listen)
- Strukturierte Programmentwicklung

Prg2: OOP in Java (Objektorientierte Programmierung)

- Klassen und Objekte
- Interfaces und Vererbung
- Behälter und Iteratoren
- Fehler und Ausnahmen
- Programmierregeln und Konventionen

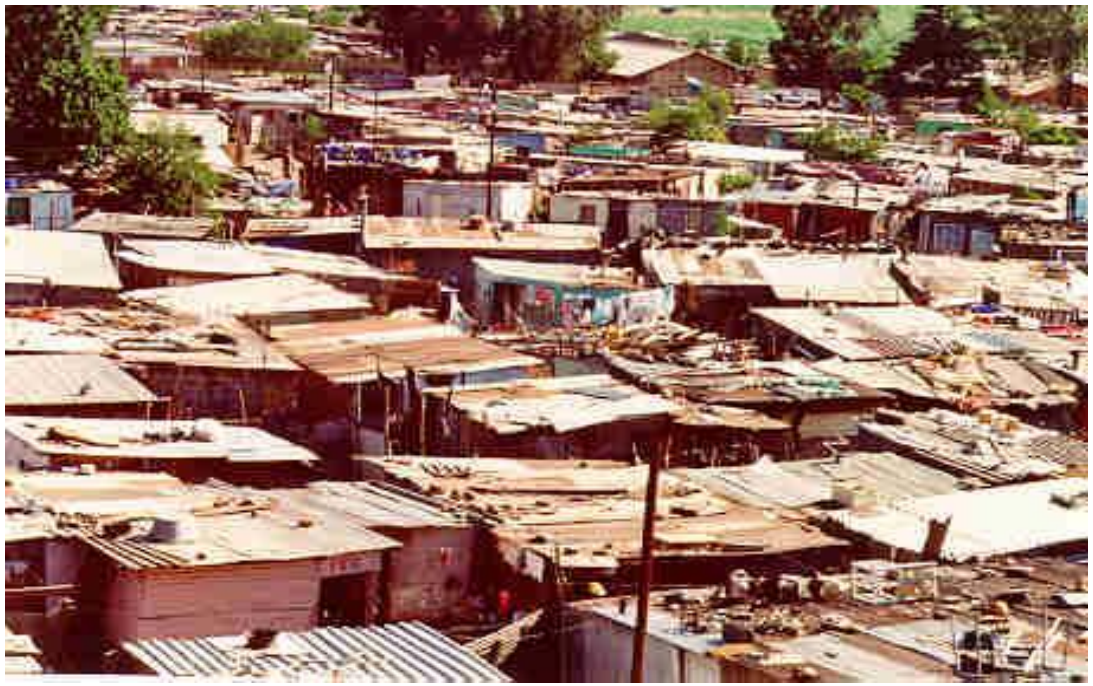
Motivation für Programmieren 3: Programmieren von großen Systemen

- **typische Probleme im Programmcode großer Systeme**
  - Unverständlichkeit (Kommentierung, Bezeichner, Code-Layout)
  - Redundanz
  - Überfrachtung (z.B. lange Methoden/Funktionen)
  - Vergesse Details (z.B. *equals* und *hashCode*)
  - Ungeeignete Datenstrukturen (z.B. *List* statt *Set*)

Typische Probleme im Programmcode großer Systeme

- Ungeeignete/Falsche Algorithmen (z.B. Bubblesort statt Quicksort).
- Unklare Verträge zwischen Methode und Benutzer (Übergabe von *null,* Reihenfolge von Aufrufen)
- Wachsende physikalische Abhängigkeiten (*import*) (Seiteneffekte, Tests, Compilezeit)
- Wachsende Abhängigkeiten von Basistechnologien (Applikationsserver, Datenbank, GUI)
- Konzeptionelles Chaos (Fehlerbehandlung, Logging/Tracing, Nebenläufigkeit)
- Wachsende Komplexität des Codes

So sehen große SW-Systeme oft aus



- Wachstum nicht geplant und unkontrolliert
- viele verschiedene „Architekten“
- jedes Haus ist anders, viele sind instabil
- Infrastruktur ist ungenügend oder fehlt.

Ziele von Programmieren 3

- Auf die Details kommt es an! (equals, hashCode, toString, compareTo).
- Richtige Verwendung vorhandener Algorithmen (z.B. Sortieren, Anagramm)
- Wahl der richtigen Datenstrukturen (List, Set, Map, Intervalle, Index, fachliche Datentypen)
- Verwendung vorhandener Mikro-Designs (Entwurfsmuster).
- Entwerfen mit Komponenten und Schnittstellen
- Fortgeschrittene Programmierkonzepte (Threads, Generics, Funktionale Programmierung, Annotationen, Ausnahmebehandlung auf Komponentenebene)

Inhalt Programmieren 3

- Kapitel 1: Generics
- Kapitel 2: Klassen und Schnittstellen
- Kapitel 3: Entwurfsmuster
- Kapitel 4: Komponenten und Schnittstellen
- Kapitel 5: Anwendungsprogrammierung
- Kapitel 6: Fehler und Ausnahmen
- Kapitel 7: Threads
- Kapitel 8: Annotationen
- Kapitel 9: Antipatterns (Gruselkabinett)

Literatur (1)

- Joshua Bloch: Effektiv Java programmieren, Addison Wesley (2008)
- Robert C. Martin: Clean Code, mitp (2009)
- Andrew Hunt, David Thomas: Der pragmatische Programmierer, Hanser Verlag (2003)
- Johannes Siedersleben: Moderne Softwarearchitektur, dpunkt.verlag (2004)
- Martin Fowler: Refactoring, Addison Wesley (1999)
- Liquori, R., Liquori P. und Schulten L.: Java kurz und gut. O‘Reilly (2008)
- Johannes Nowak: Fortgeschrittene Programmierung mit Java 5, dpunkt.verlag (2004)
- Louis, D., Müller P.: Das Java 6-Codebook, Addison-Wesley (2008)
- Schiedermeier, R, Köhler, K.: Das Java-Praktikum, dpunkt.verlag (2008)

Literatur (2)

- Freeman, Freeman, Sierra: Entwurfsmuster von Kopf bis Fuß, O'Reilly (2005)
- Martin Fowler: Patterns für Enterprise Application-Architekturen, mitp Verlag (2003)
- William J. Brown et al.: AntiPatterns, mitp Verlag (2007)
- Erich Gamma et al.: Entwurfsmuster, Addison-Wesley (1994)
- UML 2 glasklar, Jeckle, Rupp, Hahn, Zengler, Queins, dpunkt.verlag (2007)

- http://java.sun.com insbesondere http://java.sun.com/docs/books/
- Ullenboom, Ch.: Java ist auch eine Insel www.galileocomputing.de/openbook/javainsel/ (2011)
- Guido Krüger:
  Handbuch der Java-Programmierung, Addison Wesley, http://www.javabuch.de (2015)

SEU / IDE

Software-Entwicklungsumgebung(SEU)/ Integrated Development Environment (IDE), z.B.

- Eclipse: http://www.eclipse.org/
- Netbeans: http://netbeans.org/
- weitere, z.B. IntelliJ, …

Die SEU / IDE bietet vielseitige Unterstützung:

- Code-Vervollständigung und Code-Generierung (z.B. durch stubs oder import)
- Formatierung des Quellcodes
- Generierung von Kommentarzeilen (JavaDoc)
- DEBUG
- Tests mit JUnit
- systematische Änderungen durch "Refactor"

Farbcode

Blau = Schnittstelle (Interface)

Gelb = Klasse

Grün = Anwendung, Testtreiber