Organisatorisches und GUI-Programmierung

PROG 2: Einführung in die Programmierung für Wirtschaftsinformatiker

---

Steffen Helke
Technische Universität Berlin
Fachgebiet Softwaretechnik

8. April 2013

Übersicht

- Organisatorisches
- Inhalte dieser Lehrveranstaltung
- Beginn mit Teil 1 von PROG 2:

  Entwicklung grafischer Oberfläche

Wer führt diese Veranstaltung durch?

Vorlesung

- Steffen Helke, Fachgebiet Software Qualitätsmanagement

Übungsleiter

- Marcus Mews, Fachgebiet Softwaretechnik

Tutorien

- Tobias Brandt
- Amir Czwink
- Sadik Hasanovic
- Saskia Heidebring
- Dorian Laqua

Wer beantwortet Fragen zur Vorlesung?

- **Ansprechpartner**
  Steffen Helke
- **Email-Adresse**
  steffen.helke@tu-berlin.de
- **Sprechstunde**
  Mittwoch 11-12 Uhr
- **Büro**
  TEL 1203

Wer beantwortet Fragen zu den Übungen?

- **Ansprechpartner**
  Marcus Mews
- **Email-Adresse**
  m.mews@tu-berlin.de
- **Sprechstunde**
  Donnerstag 12-13 Uhr
- **Büro**
  TEL 1005

- **Tutorensprechstunden**
  Wird noch bekanntgegeben!
- **Tutorenraum**
  TEL 1012

Was ist das für eine Veranstaltung?

Art

- Pflichtveranstaltung im Bachleorstudium der Wirtschaftsinformatik
- Bestandteil des Moduls Programmieren II für Wirtschaftsinformatiker

Durchführung

- Vorlesung (VL) + Tutorien (TU)

Aufwand

- 4 SWS / 6 LP

Voraussetzungen

- Inhalte des Moduls Programmieren für Wirtschaftsinformatiker 1 (PROG 1)

Welche Prüfungsleistungen sind abzulegen?

Übungen

- Erfolgreiche Bearbeitung von 11 Übungsblättern
- Binäre Bewertung
- Gruppen von jeweils 2 Studierenden

Klausur

- Klausurtermin: 19.7.2013
- Nachklausurtermin: 30.9.2013

Randbedingung

- Voraussetzung für die Klausuranmeldung ist die erfolgreiche Bearbeitung der Übungsaufgaben

Organisatorisches zu den Übungsblättern

Ausgabe

- .. ist immer am Montag, 18:00 abends
- Das erste Übungsblatt kommt am 15.4.2013!

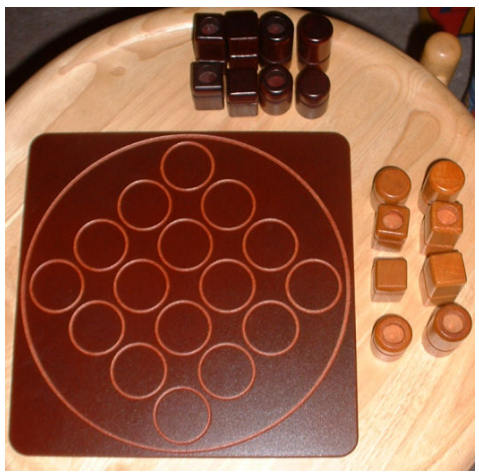
Abgabe

- .. ist immer am Montag, 8:00 morgens
- Spätere Abgaben können nicht gewertet werden!

Aufgabenstellung

- Fast alle Übungsblätter beschäftigen sich mit der Entwicklung des Brettspiels Quarto!

  http://de.wikipedia.org/wiki/Quarto!



Wo melde ich mich an und finde Infos?

PROG 2 - Homepage

https://www.isis.tu-berlin.de/course/view.php?id=7894

- Regelmäßig Aufrufen
  Bekanntgabe von Raumänderungen, Krankheit usw.
- Diverse Unterlagen
  Literatur, Übungsblätter, Folien usw.

Anmeldung zu den Tutorien

https://www.moseskonto.tu-berlin.de/moseskonto/

- Anmeldeschluss ist am
  10.04.2013, Mitternacht
- Nachzügler bitte direkt an den Übungsleiter wenden
  m.mews@tu-berlin.de

Vorlesungs- und Übungstermine

	Raum	Wochentag	Uhrzeit	Dozent
VL	H 1058	Montag	14-16	Steffen Helke
T1	MAR 0.015	Dienstag	10-12	Amir Czwink
T2	FH 303	Dienstag	10-12	Saskia Heidebring
T3	FH 303	Dienstag	12-14	Saskia Heidebring
T4	FH 316	Dienstag	12-14	Dorian Laqua
T5	TEL 1011	Dienstag	12-14	Marcus Mews
T6	FH 312	Dienstag	14-16	Sadik Hasanovic
T7	FH 301	Dienstag	14-16	Tobias Brandt

Zusätzlich bietet unser Tutor Sedat Koca ein Fachmentorium an: Freitag, 16-18 Uhr im TEL 109!

Inhalte der Vorlesung

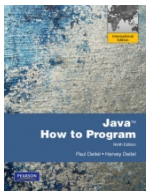
- Entwicklung grafischer Oberflächen
- Fehlerbehandlung mit Exceptions
- Ein- und Ausgabe mit Dateien
- Datenformate mit XML
- Nebenläufiges Programmieren mit Threads
- Protokolle wie TCP / UDP, Kommunikation mit Sockets
- Verteilte Anwendungen in Netzwerken (Client/Server)
- Aufruf entfernter Methoden (RMI)
- Unit-Test und andere Techniken zur Qualitätsicherung

Literatur

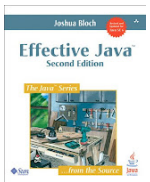
P.Pepper: *Programmieren lernen: Eine grundlegende Einführung mit Java*. Springer-Verlag, 2007.

C.Ullenboom: *Java ist auch eine Insel: Das umfassende Handbuch*. Galileo Computing, 2010.

G.Krüger, T.Stark: *Handbuch der Java-Programmierung*. Addison-Wesley, 2009.



H.M.Deitel, P.J.Deitel: *Java How to Program*. Pearson Education, 2011.



J.Bloch: *Effective Java*. Addison-Wesley, 2008.

Teil 1 der Vorlesung PROG 2

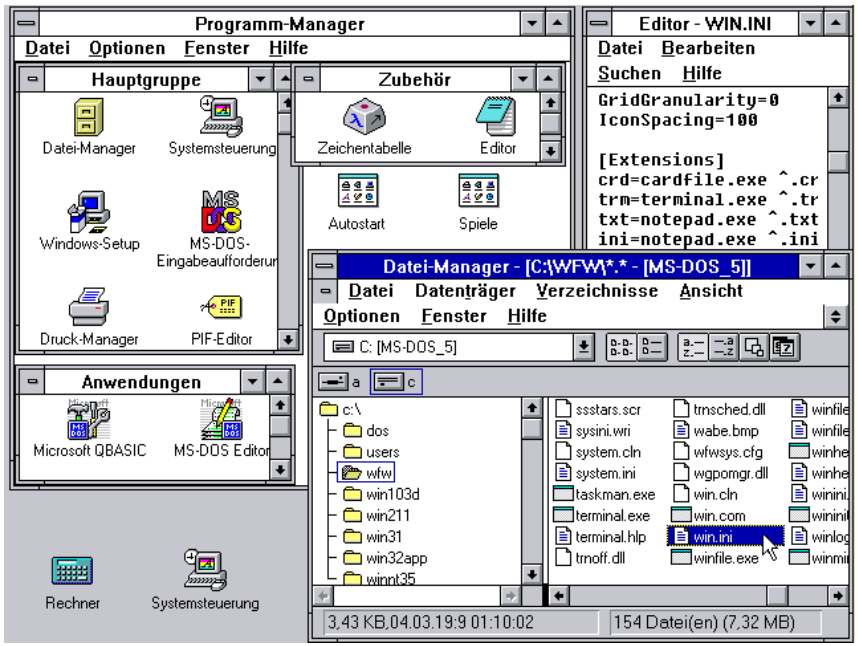
Entwicklung grafischer Schnittstellen

Graphical User Interface

Quelle: *Inhalt & Gestaltung nach Vorlesungsfolien von Peter Pepper und Odej Kao, TU Berlin*
Methodische- und Praktische Grundlagen der Informatik 4 (MPGI 4), WS 2010/11 bzw. WS 2011/12

Warum grafische Oberflächen?

- Mensch-Maschine-Interaktion *ohne* grafische Benutzeroberflächen heute undenkbar (*unverkaufbar*)
- Standardoberfläche sind (noch immer) Fenster
- Interne Strukturierung der Fenster durch Komponenten



Beispiel Windows 3.1 (1992), Quelle: http://de.wikipedia.org/wiki/Grafische_Benutzeroberfläche

Apple-Lisa von Macintosh (1983)

Apple Lisa (1983), Quelle: http://de.wikipedia.org/wiki/Apple_Lisa

LISA: Local Integrated Software Architecture

Seid wann gibt es grafische Oberflächen?

1981 Entwicklung erster Bitmap-Terminals in Palo Alto im Xerox-Labor

1983 Apple-Lisa von Macintosh

1984 X-Windows-System für Unix (MIT, IBM und Digital Equipment Corporation)

1985 Windows 1.03 von Microsoft

1986 Commodore 64 (GEOS) und Atari

1987 Amiga 500

1992 Durchbruch mit Windows 3.1 von Microsoft

Anmerkung: Es gibt eine europäische Norm (EN ISO 9241) zur Festlegung von Anforderungen an grafische Benutzerschnittstellen.

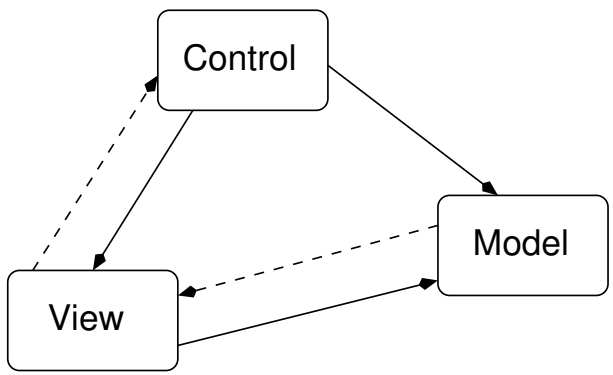
Grundlegende Konzepte von GUIs

Aspekte

1. Grafische (und ästhetische) Gestaltung der Oberflächen
2. Interaktion zwischen Benutzer, GUI und Programm

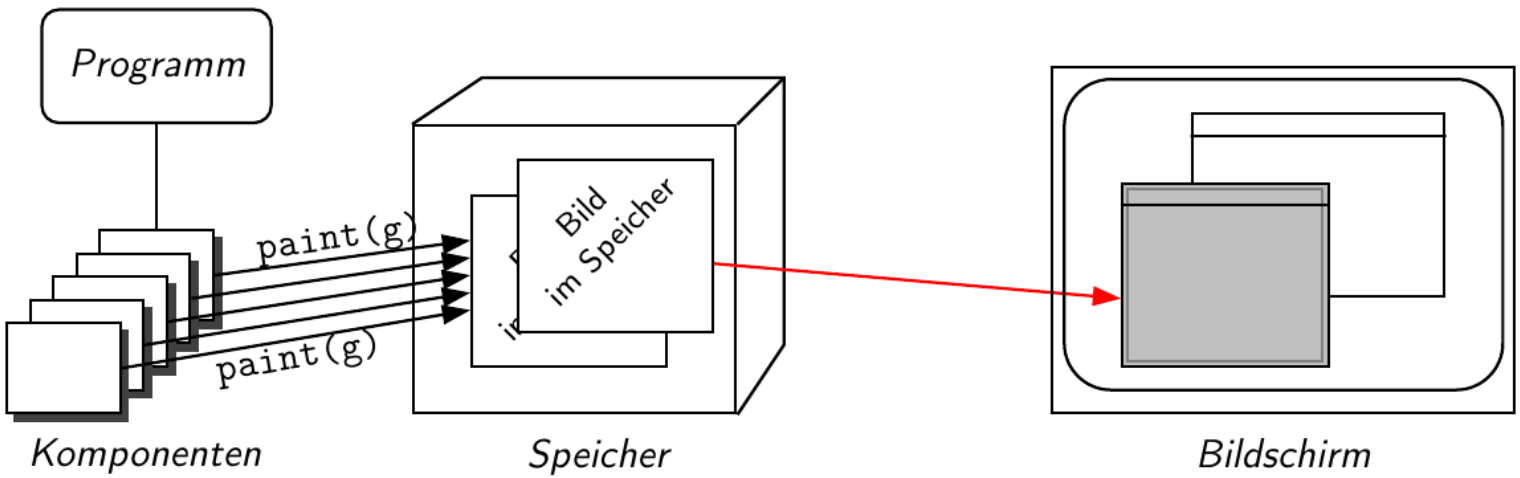
Architektur-Pattern: Model-View-Control (MVC)

- **Model:** Datenmodell
- **View:** Präsentation
- **Control:** Programmsteuerung



Wie bekommen wir ein GUI-Fenster?

1. Erzeugung von Objekten innerhalb eines Programms zur Darstellung eines GUI-Fensters
2. Generierung der binären Repräsentation des Fensters im Speicher (Matrix von Farbwerten für Pixel)
3. Auslesen des Speichers und Anzeige auf dem Bildschirm durch das Grafiksystem der Maschine



Quelle: Abbildung aus *Vorlesungsfolien Peter Pepper, TU Berlin, MPGI 4, WS 2011/12*

⇒ Technische Umsetzung mit Hilfe von Software-Bibliotheken

Software-Bibliotheken zur GUI-Entwicklung

Was sollte unterstützt werden?

- Grundelemente zum Zeichnen
  $\Rightarrow$ Linien, Polygone, Farben, Schriftarten, ...
- Steuerelemente als fertige GUI-Komponenten
  $\Rightarrow$ Fenster, Schaltflächen, Textfelder, ...
- Komponenten zur Behandlung von Ereignissen
  $\Rightarrow$ Eingaben mit Maus, Tastatur, ...

Angebote in Java (Auswahl)

- AWT aus JFC – Abstract Window Toolkit (1996, Sun)
- Swing aus JFC – Java Foundation Classes (1998, Sun)
- SWT – Standard Widget Toolkit aus Eclipse (2001, IBM)

Softwaretechnische Herausforderungen

Technische Ziele

- Umgang mit der starken Abhängigkeit der Grafik-Software von der plattformspezifischen Hardware
- Garantieren des Java-Grundsatzes: *write once run everywhere*
- Effiziente Ausführung durch spezifische Beschleuniger
- Sicherstellen von Transparenz und Schlankheit der zu entwickelnden GUI-Bibliotheken
- Unterstützung fortgeschrittener Entwicklungswünsche

Lösungsansätze

- Schwergewichtige Komponenten (AWT, SWT)
- Leichtgewichtige Komponenten (Swing)

Abstract Window Toolkit (AWT)

Einordnung

- seit 1996 Bestandteil der Java Foundation Classes (JFC)
- Verwendung nativer (schwergewichtiger) Komponenten des Betriebssystems (außerhalb des Java-Speicherbereichs)
- Layout-Probleme beim Wechseln der Plattform möglich

Funktionalitäten

- vergleichsweise wenig und teilweise etwas veraltet
- Primitivoperationen zum Zeichnen (Linien, Farbe, Text, ...)
- Steuerungskomponenten zur Abfrage von Eingabeereignissen
- Dialogelemente zur Kommunikation mit dem Benutzer
- Darstellung/Manipulation von Bitmaps, Ausgabe von Sound

$\Rightarrow$ böse Zungen sagen auch *Annyoing Window Toolkit*

Swing aus JFC (Java Foundation Classes)

Einordnung

- Vorstellung 1997 auf der Konferenz *JavaOne* mit Swing-Musik
- seit 1998 Teil der JFC, baut teilweise auf AWT auf
- Verwendung leichtgewichtiger Komponenten unabhängig vom Betriebssystems (gerendert in Java)
- nicht thread-safe

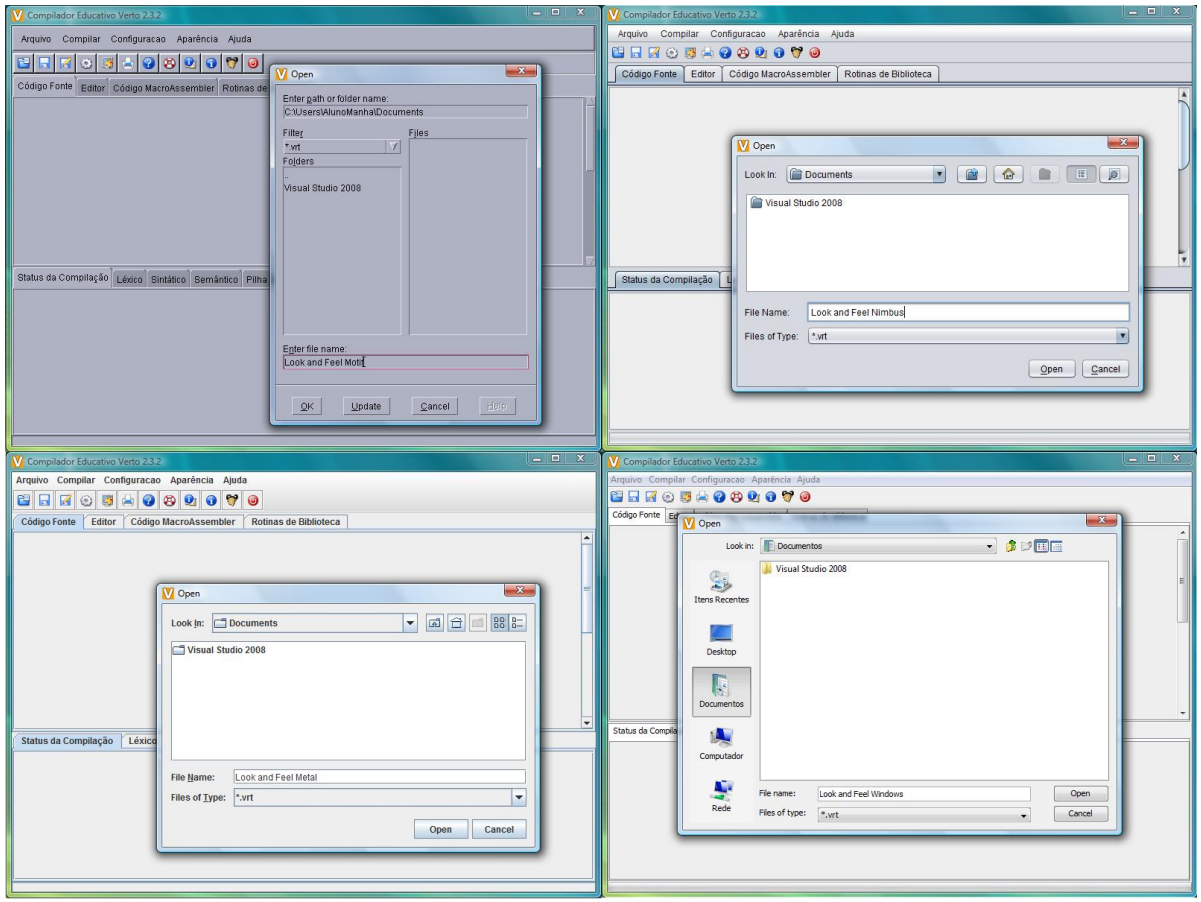
Funktionalitäten

- anpassbares Erscheinungsbild und Verhalten (look and feel)
- mehr Komponenten als AWT: Tabellen, Bäume, ...
- GUI-Steuerung mit Tastenkombinationen möglich
- automatisches Double Buffering, unterstützt MVC-Pattern
- Drag & Drop, Tool-Tips, Multiple Document Interface

Was bedeutet Look and Feel?

- Swing bietet diverse Motive für Erscheinungsbild & Verhalten

  - Motiv
  - Metal
  - Ocean
  - Nimbus
  - Linux-Gtk
  - Windows-Spez.
  - Mac-Spez.
  - ...



Quelle: http://pt.wikipedia.org/wiki/Look_and_Feel, Aufruf 15.4.2012, Urheber Ricardo Ferreira de Oliveira

Standard Widget Toolkit (SWT)

Einordnung

- Entwicklung von IBM 2001 für Eclipse
- schwergewichtige, native Komponenten
- Effizienzprobleme auf manchen Nicht-Windows-Plattformen
- SWT-Bibliotheken oft nicht standardmäßig verfügbar
- Threading-Modell ähnlich zu Swing, d.h. nicht thread-safe!

Funktionalitäten

- Basiskomponenten ähnlich zu Swing
- spezielle Bibliothek JFace, um Basiskomponenten zu komplexeren GUI-Elementen zusammenzufassen
- Actions zur Entkopplung von GUI-Event und Aktion
- Komplexere GUI-Elemente: Wizards, Dialoge, ...