GUI

Das Projekt braucht
ein(e?) GUI

GUI

GUI ???

- Graphical
- User
- Interface

GUI

- Unsere Projekte hatten eine GUI !
- BlueJ musste dieses bereitstellen, wenn das Konzept „objects first“ tragfähig sein soll.
- Allerdings ist es eine GUI für die Entwicklung selbst, nicht für die entwickelte Anwendung.

GUI

Warum?

- realistische Anwendungsentwicklung
- Behandlung des MVC - Konzeptes

GUI

MVC – Konzept

Benutzerschnittstellen bestehen aus den drei Klassen

- Model
- View
- Controller

MVC – Konzept

Ziel:

Kapselung des eigentlichen Modells

und

Entkoppeln von [der Art] seiner Darstellung

MVC – Konzept

Aufgaben der drei Objekte

a) Model

- Enthält das eigentliche Fachwissen
- Kennt seine Views (mehrere möglich!), die über das Beobachtermuster angekoppelt sind
- Benachrichtigt seine Views bei Änderung seiner Daten (update)

MVC – Konzept

Aufgaben der drei Objekte

b) View

- Darstellung auf dem Bildschirm
- Stellt sicher, dass sein Zustand dem Zustand des Modells entspricht
- Meldet die Benutzeraktionen (z.B. von Eingabefeldern, Buttons, ...)

MVC – Konzept

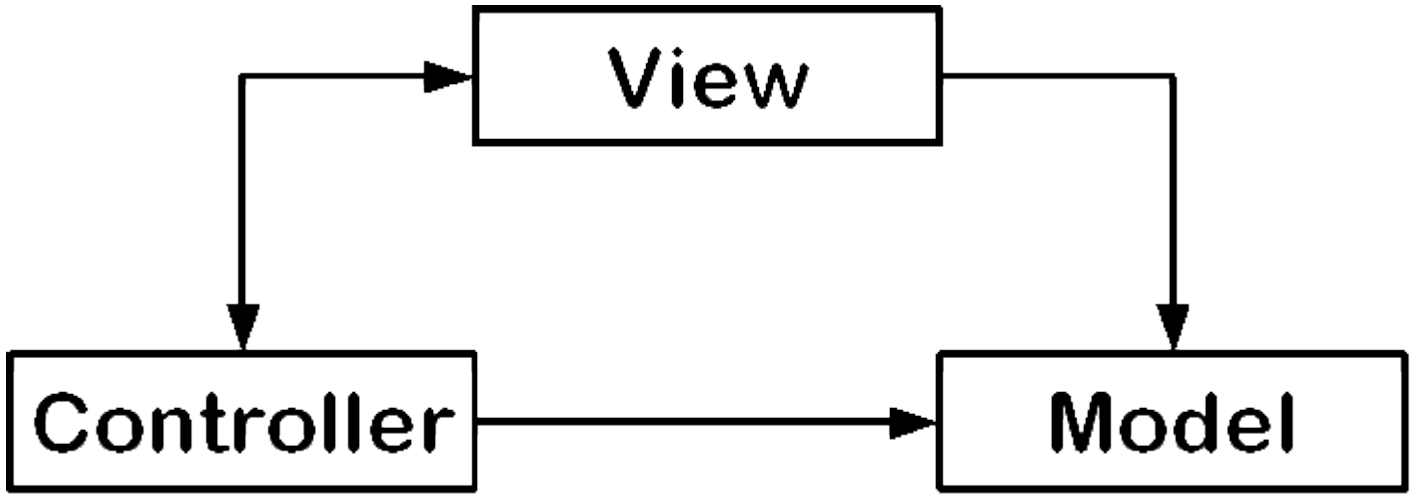
Aufgaben der drei Objekte

c) Controller

- Sichert den konsistenten Zustand des Systems
- Verarbeitet die Benutzeraktionen

MVC – Konzept

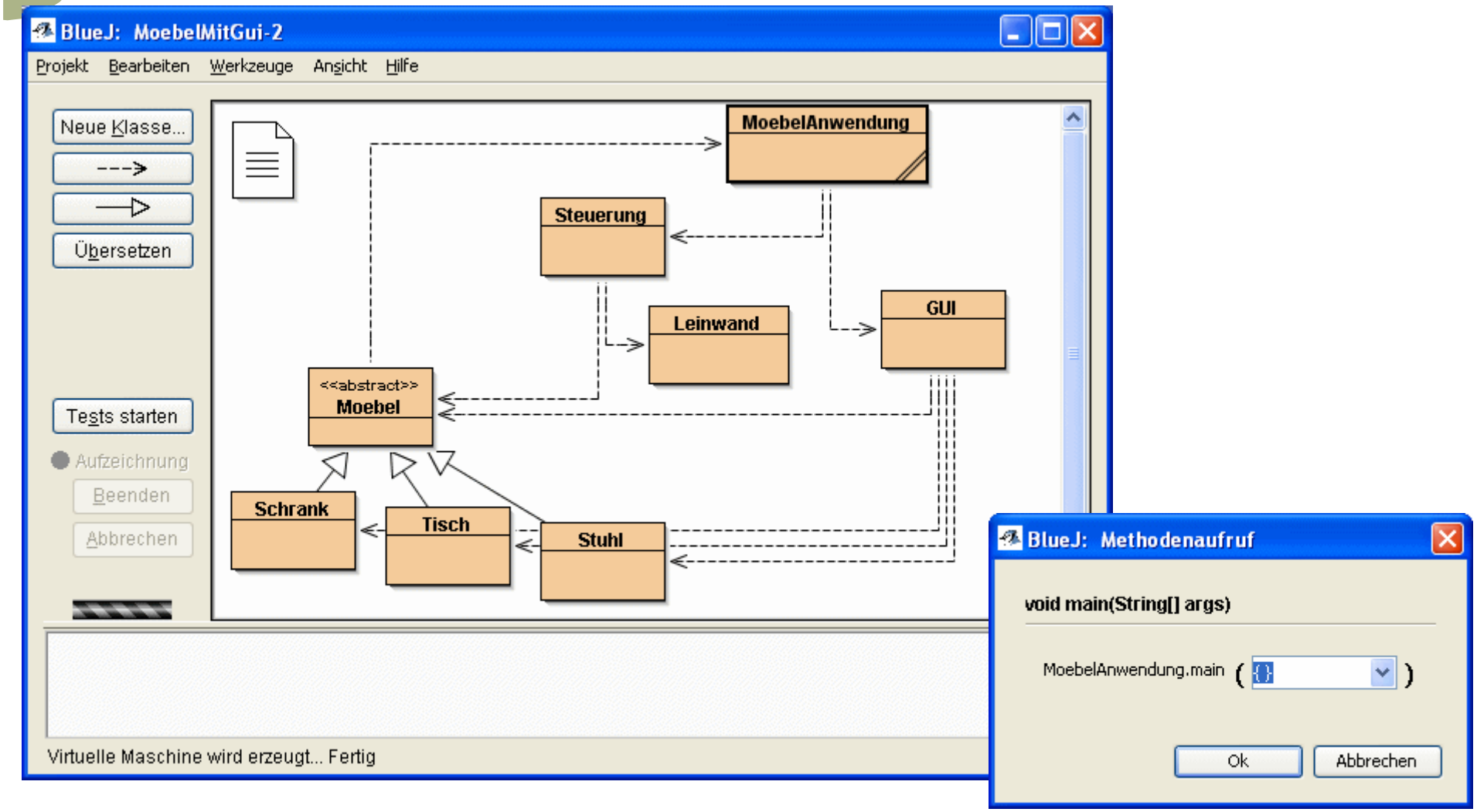
Grafische Darstellung der Zusammenhänge



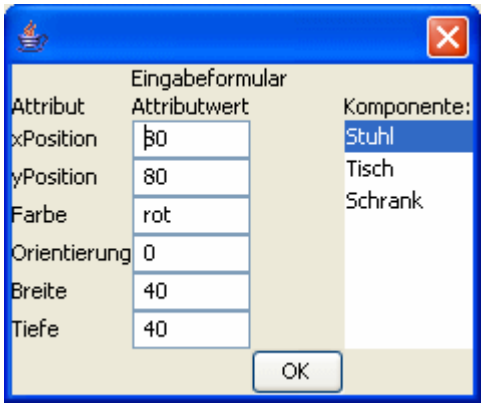
MVC – Konzept

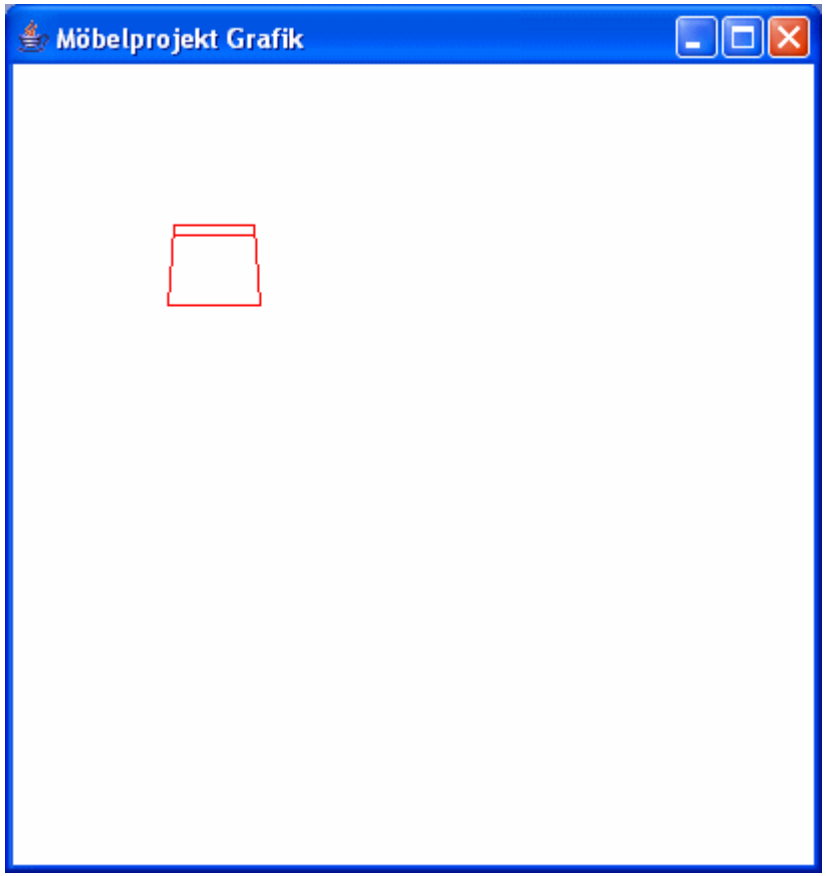
- Die Grafik legt nahe, dass das Modell selbst weder seinen Controller noch seine View – Komponente kennt (siehe aber Seite 7)
- Jede View – Komponente kennt ihren Controller und das Modell
- Jeder Controller kennt seine View – Komponente und das Modell

GUI


BlueJ: MoebelMitGui-2
Projekt Bearbeiten Werkzeuge Ansicht Hilfe
Neue Klasse...
Übersetzen
Tests starten
Aufzeichnung
Beenden
Abbrechen
MoebelAnwendung
Steuerung
Leinwand
GUI
<<abstract>>
Moebel
Schrank
Tisch
Stuhl
Virtuelle Maschine wird erzeugt... Fertig
BlueJ: Methodenaufruf
void main(String[] args)
MoebelAnwendung.main ({})
Ok
Abbrechen

GUI


Eingabeformular
Attribut
Attributwert
Komponente:
xPosition
80
yPosition
80
Farbe
rot
Orientierung
0
Breite
40
Tiefe
40
Stuhl
Tisch
Schrank
OK


Möbelprojekt Grafik

GUI

Analyse der Beispiellösung

- Ist es eine Lösung?
- Welche Mängel hat sie noch?
- Die Schülerinnen und Schüler sollten alternative Konzepte angeben können
- Die Entwürfe sollten aus Anwendersicht analysiert werden und zu einem Anforderungskatalog führen

GUI

Beispiele

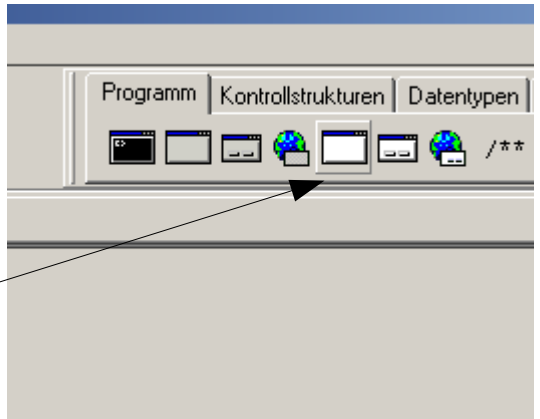
- Was ist das Model ?
  - die Möbelklassen allein, so wie sie jetzt sind, sicher nicht, da zum Möblierungs-system weitere Aspekte als nur ihre grafische Darstellung gehören werden
- Was ist der Controller ?
  - mehrere Views mit Listenern
  -Ablaufsteuerung
- Welche Views werden benötigt ?

Aufgabe zur GUI

- Hier keine Antworten, aber eine kleine weitere Übung
- Werkzeug ist der JAVA – Editor von Gerhard Röhner

http://www.bildung.hessen.de/abereich/inform/skii/material/java/editor.htm

- Starten Sie ihn!
- Erzeugen Sie einen JFrame

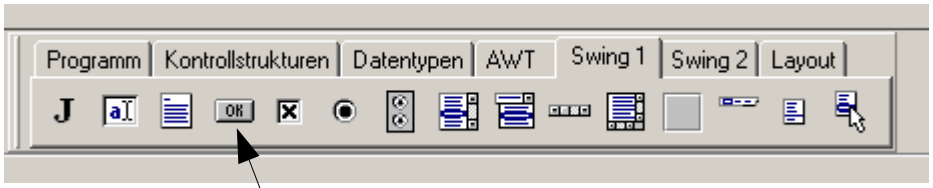


Aufgabe zur GUI

- Legen Sie einen Ordner für das Projekt an
- Speichern Sie die Datei unter dem Namen Gui.java
- Im Gegensatz zu BlueJ haben wir hier eine drag and drop – Oberfläche
- Das Übersetzen erzeugt aber schon eine funktionierende Anwendung!
- Allerdings macht sie noch nichts, man kann sie aber ordnungsgemäß schließen

Aufgabe zur GUI

- Wählen Sie die Lasche Swing1 aus



- Fügen Sie einen JButton ein und beobachten Sie die Veränderungen im Quelltext

Aufgabe zur GUI

```
jButton1.setBounds(24, 16, 81, 41);
jButton1.setText("jButton1");
cp.add(jButton1);
jButton1.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    jButton1ActionPerformed(evt);
  }
});
```

contentpane cp,
die Inhaltsfläche

Aufgabe zur GUI

```
jButton1.setBounds(24, 16, 81, 41);
jButton1.setText("jButton1");
cp.add(jButton1);
jButton1.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    jButton1ActionPerformed(evt);
  }
});
```

Listener registriert die Mausereignisse

Aufgabe zur GUI

```
jButton1.setBounds(24, 16, 81, 41);
jButton1.setText("jButton1");
cp.add(jButton1);
jButton1.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    jButton1ActionPerformed(evt);
  }
});
```

Erzeugt Objekt zu anonymer Klasse

ActionListener ist ein interface actionPerformed ist seine einzige zu implementierende Methode

Aufgabe zur GUI

```
jButton1.setBounds(24, 16, 81, 41);
jButton1.setText("jButton1");
cp.add(jButton1);
jButton1.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    jButton1ActionPerformed(evt);
  }
});
```

ähnlich Command - Entwurfsmuster

Aufgabe zur GUI

- die Methode macht hier also nichts selbst
- sie ruft dafür eine Ereignismethode auf (die hier natürlich auch noch leer ist) :

```
jButton1ActionPerformed(evt);
```

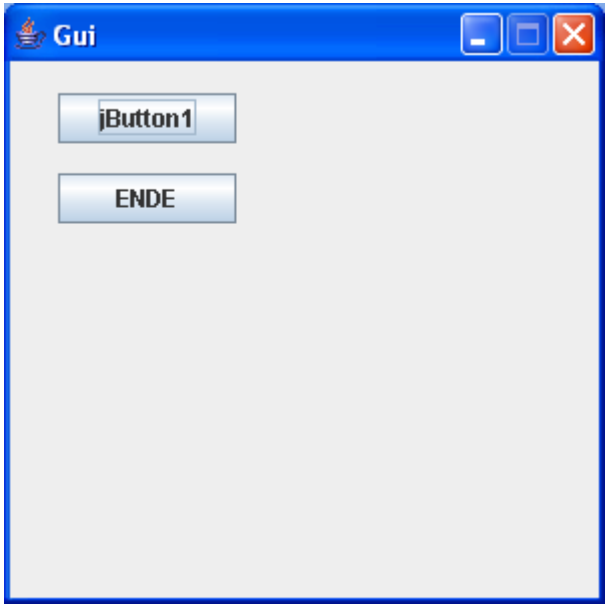
- die finden wir etwas weiter unten im Text
- Welche Vorteile hat solch ein Entwurf?

Aufgabe zur GUI

- Welche Vorteile hat solch ein Entwurf?
- Einer ist sicherlich die Verbesserung der Übersichtlichkeit durch Trennung der Definition der Komponente und der Behandlung ihrer Ereignisse
- Ein zweiter Vorteil ergibt sich (siehe auch das Command – Entwurfsmuster) durch die mögliche Wiederverwendung dieser Methode von anderer Stelle in Gui

Aufgabe zur GUI

- Fügen Sie einen weiteren Button ein und beschriften Sie ihn mit ENDE



- Die Gui sieht nun so aus:
- Der ENDE – Button soll das Fenster ebenso schließen wie das Fenster-Kreuzchen

Aufgabe zur GUI

- „ebenso“ bedeutet aber, dass wir den zugehörigen Programmtext schon haben
- er sollte nur einmal auftauchen!
- wir holen ihn von dort, wo er schon steht, beim windowClosingEvent

```
public void beendenAktion(AWTEvent evt)
{ System.exit(0); }
```

- beendet nun in gleicher Weise beim Aufruf aus beiden Ereignismethoden, ggf. auch noch von einem Menü aus

Aufgabe zur GUI

- Ein Menü sollte diese Möglichkeit auch anbieten, üblicherweise das Datei – Menü
- mit drag and drop bekomme ich die Menüleiste nicht integriert
- also fügen wir sie per Texteinfügung ein

Aufgabe zur GUI

- Deklaration

```
private JMenuBar jMenuBar;
private JMenu menueDatei;
private JMenuItem menueEintragEnde;
```

- Definition

```
jMenuBar = new JMenuBar();
setJMenuBar(jMenuBar);
menueDatei = new JMenu("Datei");
jMenuBar.add( menueDatei );
JMenuItem jMenueEintragEnde = new JMenuItem("Ende");
menueDatei.add(jMenueEintragEnde);
```

Aufgabe zur GUI

- und Aktion

```
jMenueEintragEnde.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            beendenAktion(evt);
        }
});
```

Aufgabe zur GUI

- Das versehentliche Schließen des Fensters können wir nun zentral mit einem Dialog „verhindern“:

```
if (JOptionPane.showConfirmDialog(this,
                "wirklich beenden ?",
                "Schließen?",
                JOptionPane.OK_CANCEL_OPTION)
    ==JOptionPane.OK_OPTION){
 System.exit(0);
dispose();
 }
```

... allerdings ...

Aufgabe zur GUI

- ... allerdings
müssen wir vorher noch im Kopf die Standardaktion für das Schließen des Fensters überschreiben:

```
setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
```