

MVC mit Processing

CS1016 Programmierung interaktiver Systeme

von Prof. Dr. Weigel



MVC mit Processing

1. **UML-Einführung**
2. Was ist das Problem mit dem Processing Sketch?
3. Was ist MVC? Welche Vorteile hat es?
4. MVC mit Processing
5. Interaktives Model-Testen mit der JShell

Unified **M**odelling **L**anguage

Regeln zur grafischen Darstellung und Dokumentation von Klassen

→ Spezialisiert auf Objektorientierte Programmierung

- + Vereinfachung im Vergleich zum Quellcode (keine unnötigen Details)
- + Schneller Überblick über
 - Relevante Klassen und Schnittstellen
 - Beziehungen zwischen Klassen, Schnittstellen, ...
- + Gut zur Kommunikation in Teams
 - Überblick in Meetings
 - Schnittstellenbeschreibung
- + Unabhängig von der Programmiersprache

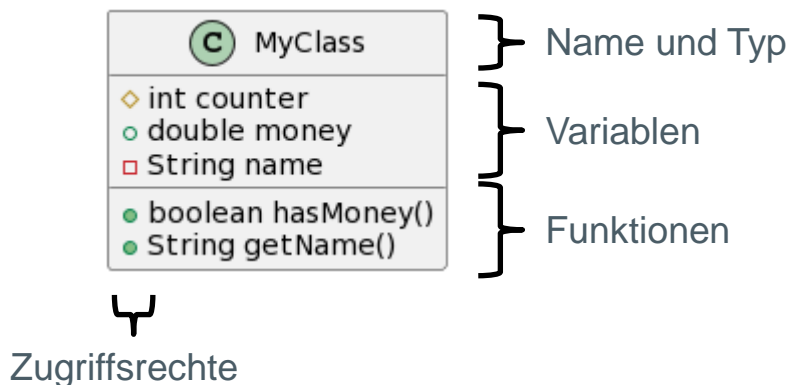
UML-Klassendiagramme: Stellt Klassen und deren Relationen dar

```

class MyClass {
    protected int counter;
    public double money;
    private String name;

    public boolean hasMoney() {
        return money > 0;
    }

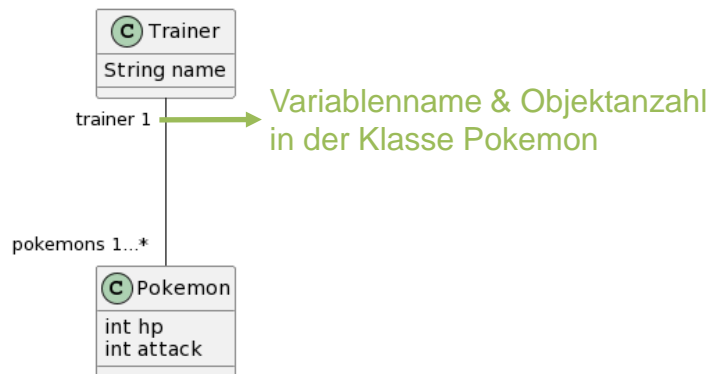
    public String getName() {
        return name;
    }
}
  
```



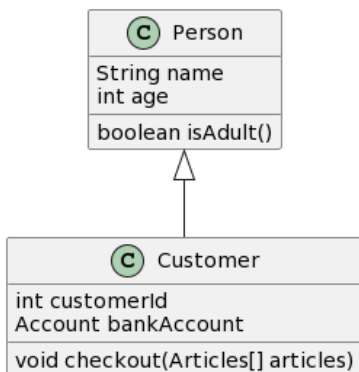
Pfeile und deren Bedeutung

Assoziation (Beziehung zwischen X und Y)

Falls nur eine Klasse die Andere kennt mit Pfeilspitze →

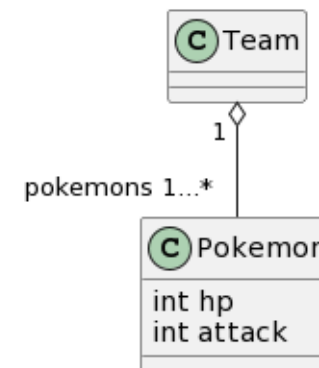
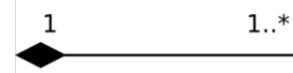


Vererbung (Customer extends Person)

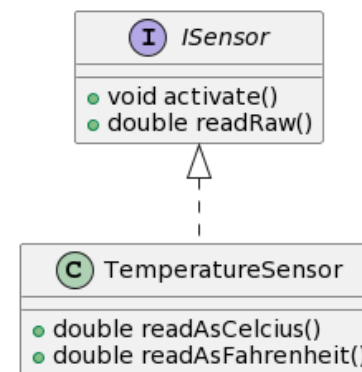


Aggregation (X besteht aus Y, bzw. Y ist Teil von X)

Falls Y nur mit X existieren kann:
Komposition

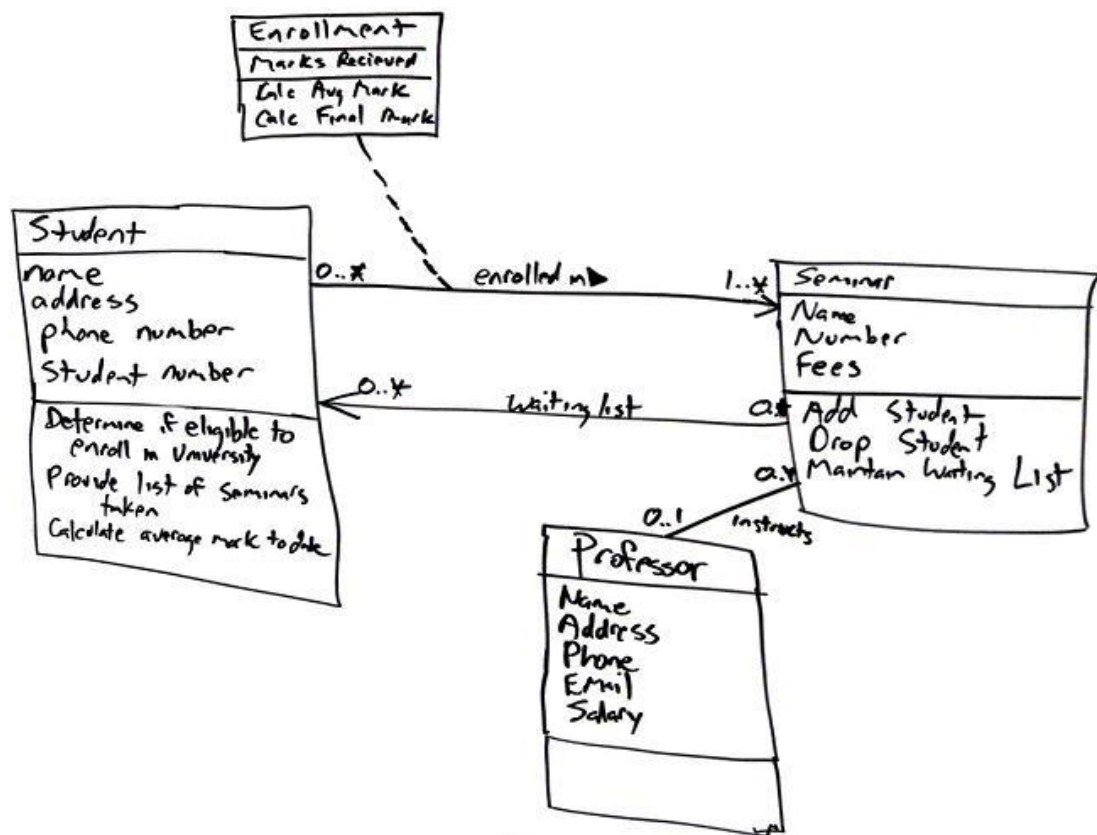


Implementierung (TemperatureSensor implements ISensor)

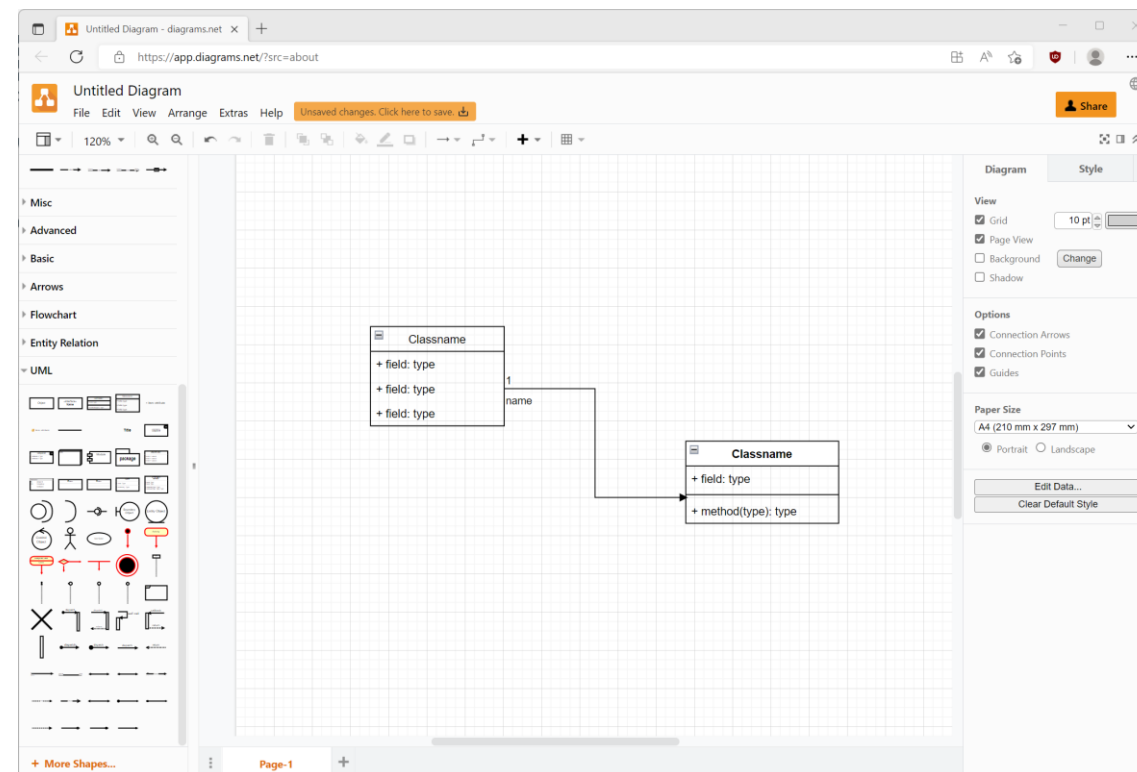


Wie erstelle ich ein UML Diagramm?

Papier (z.B. in einem Meeting oder der Klausur)



Tools (GUIs, Text und Quellcode-Generatoren)

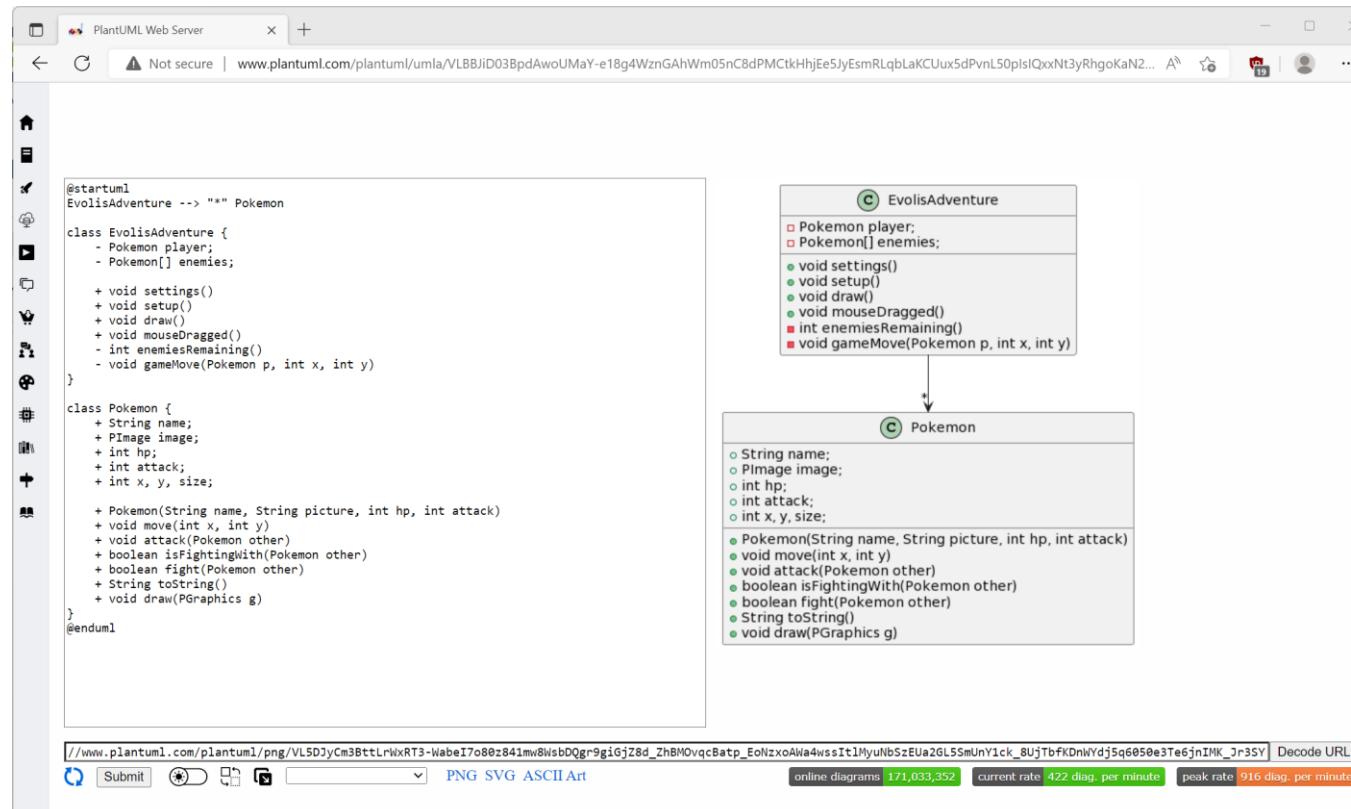


[Bildquelle: <http://agilemodeling.com/artifacts/classDiagram.htm>]

Beispiel: PlantUML

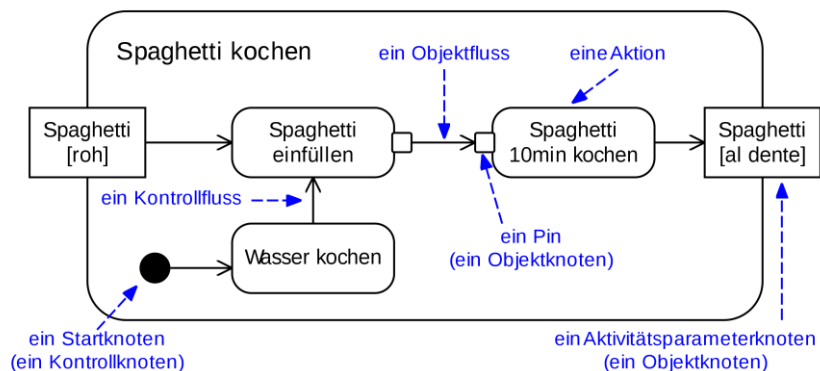
Erlaubt textuelle Beschreibung zum generieren von UML Diagrammen.

Online Generator: <http://www.plantuml.com/plantuml/uml/>

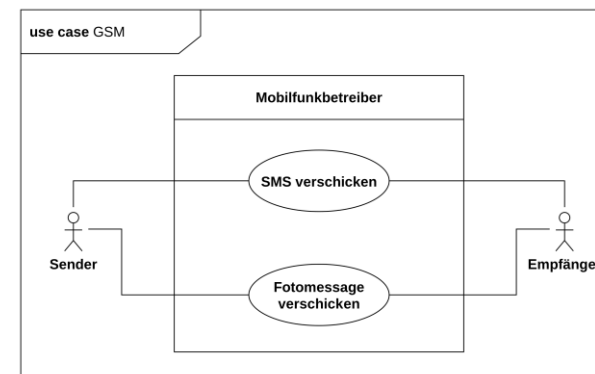


UML ist mehr als Klassendiagramme (Liste nicht vollständig)

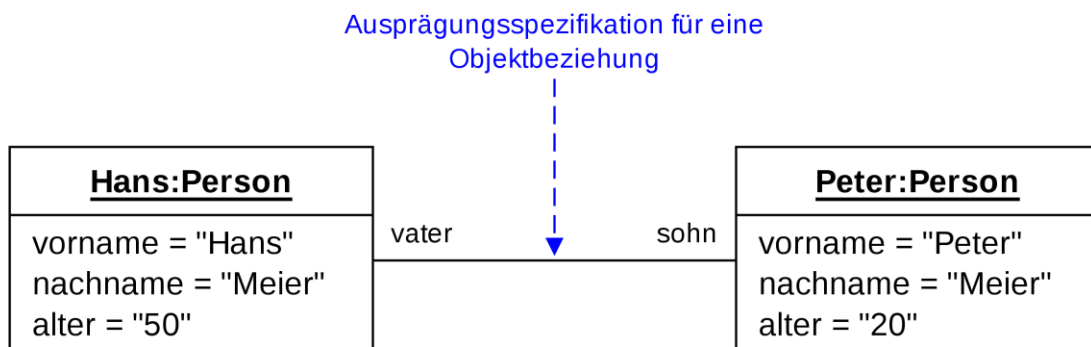
Aktivitätsdiagramm



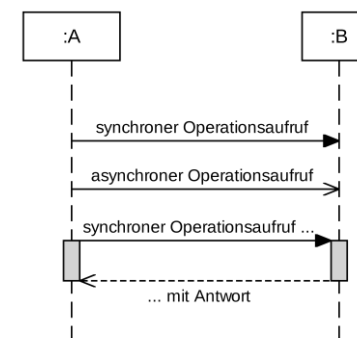
Anwendungsfalldiagramm



Objektdiagramm



Sequenzdiagramm



[Bilderquelle: Wikipedia.de]

MVC mit Processing

1. UML-Einführung
2. Was ist das Problem mit dem Processing Sketch?
3. Was ist MVC? Welche Vorteile hat es?
4. MVC mit Processing
5. Interaktives Model-Testen mit der JShell

Problem 1: setup() und draw() skalieren nicht!

Große Programme enthalten viele Zeilen Code

Beispiel: Firefox hatte 2020 etwa 21 Millionen LOC

Bearbeitet von vielen Entwicklern gleichzeitig

Kleine Module mit klar definierten Aufgaben sind:

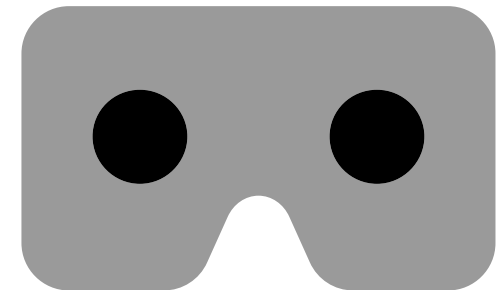
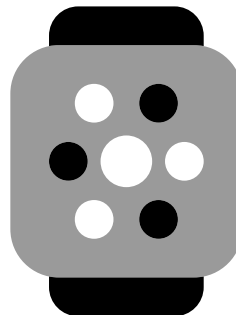
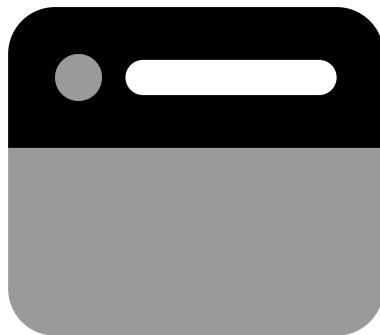
- leichter zu verstehen als große Methoden
- einfacher zu testen als ein Gesamtsystem

→ Wir müssen das Programm in logisch sinnvolle Klassen unterteilen

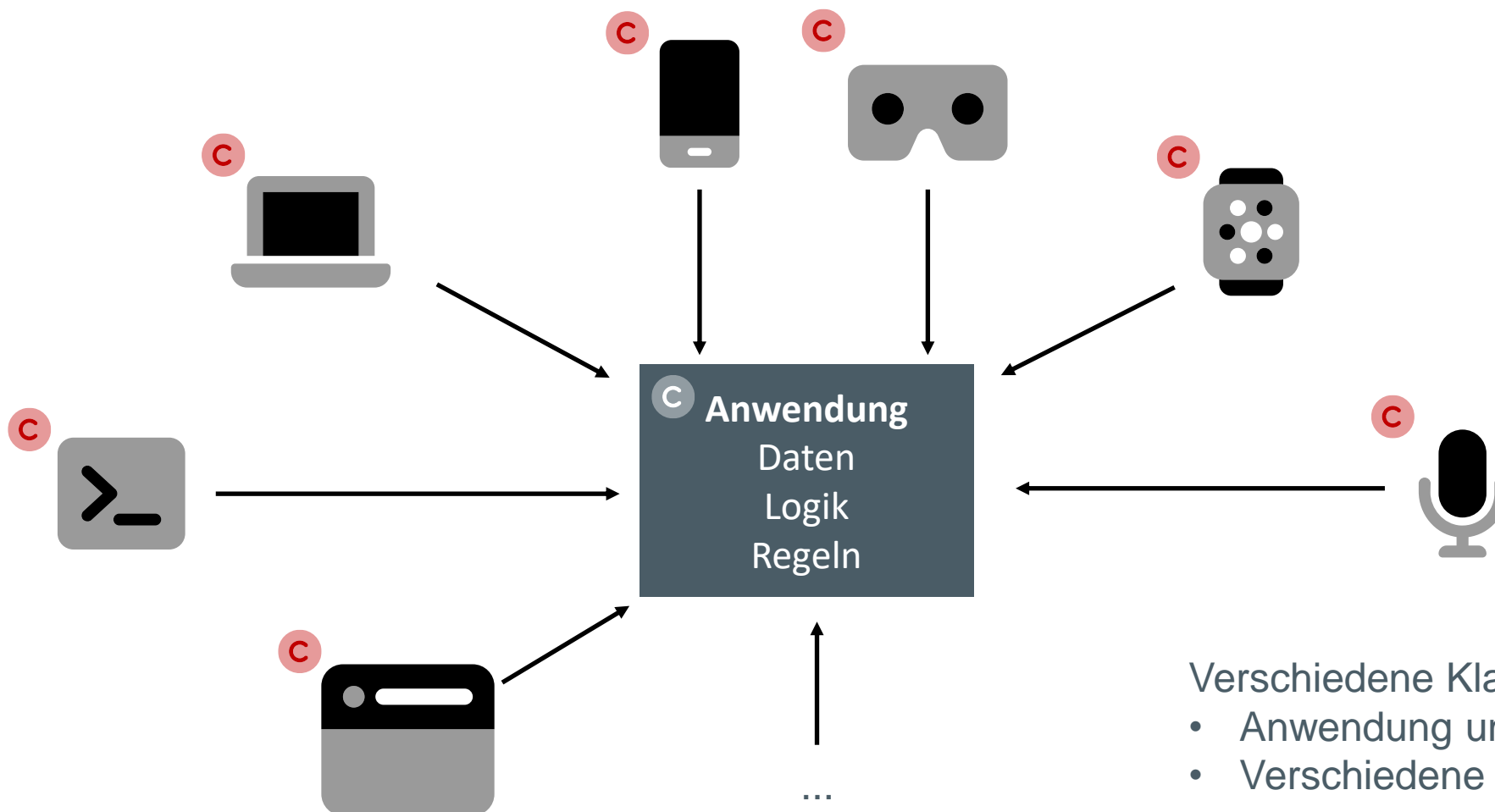


[Bildquelle: <https://images.immediate.co.uk/production/volatile/sites/30/2018/07/RedPepperAnchovySpaghetti-copy-1dec261.jpg?resize=960,872?quality=90&resize=556,505>]

Problem 2: Das Programm ist nur in Processing benutzbar



Lösungsansatz: Trennung von Benutzerschnittstelle und Anwendung



Verschiedene Klassen...

- Anwendung und Benutzerschnittstelle
- Verschiedene Benutzerschnittstellen

MVC mit Processing

1. UML-Einführung
2. Was ist das Problem mit dem Processing Sketch?
3. Was ist MVC? Welche Vorteile hat es?
4. MVC mit Processing
5. Interaktives Model-Testen mit der JShell

Model-View-Controller

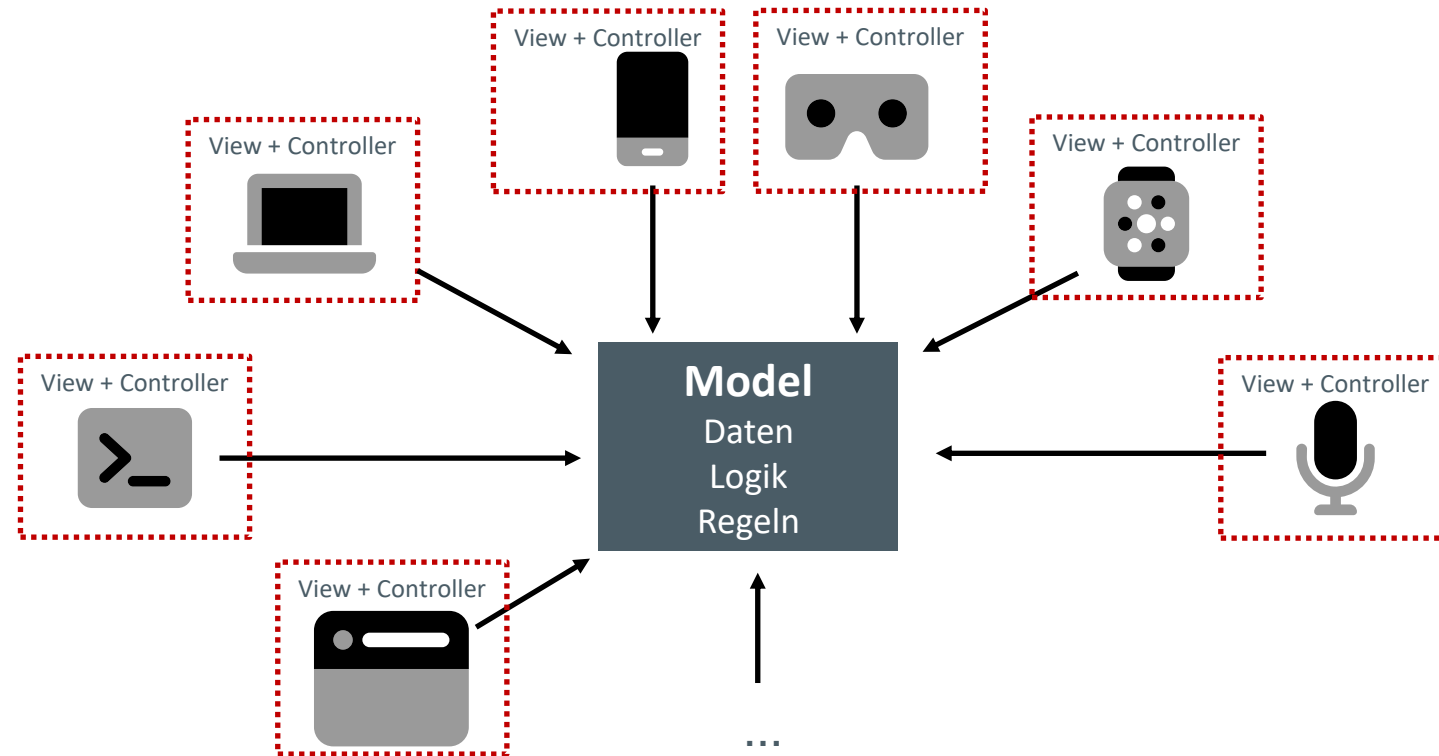
Konzept zur Trennung der Darstellung (View) vom Anwendungscode (Model)

Ursprünglich entwickelt für Smalltalk-79 bei Xerox PARC

Grundlage für sehr viele UI Frameworks

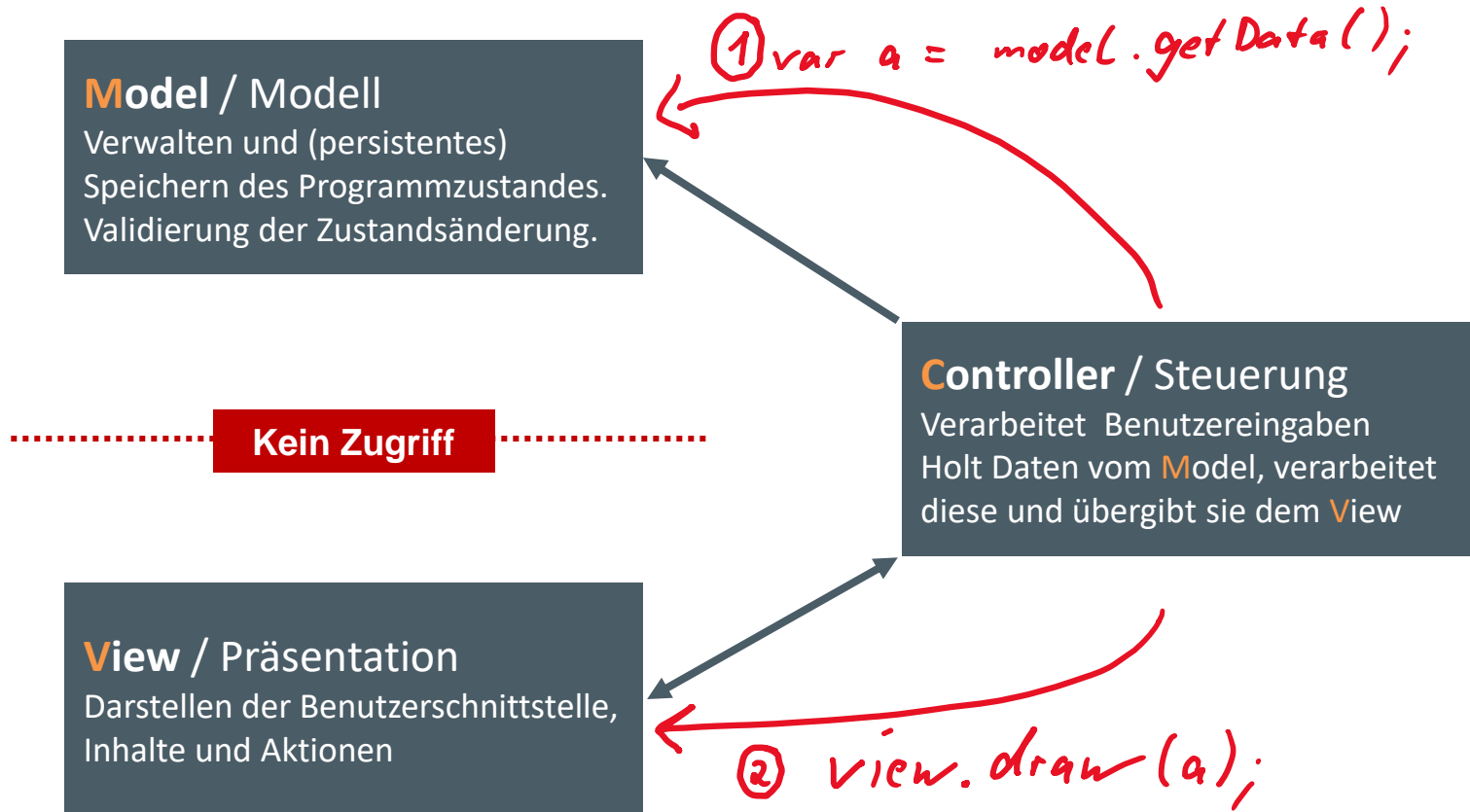
Weiterentwicklungen:

- hierarchical model-view-controller (HMVC)
- model-view-adapter (MVA)
- model-view-presenter (MVP)
- model-view-viewmodel (MVVM)



Grundidee:

Aufteilung der Aufgaben in verschiedene Klassen:



Model und View

- kennen einander nicht
- greifen nie aufeinander zu

Controller und Model

- Controller holt und ändert Daten vom Model
- Model kennt den Controller nicht

Controller und View

- Controller übergibt dem View die benötigten Daten
- Controller wählt aus **was** zu Präsentieren ist, View beinhaltet den Code **wie** es darzustellen ist
- Abhängig vom Framework:
View leitet Benutzereingaben an den Controller weiter

MVC-Beispiel als Klassendiagramm

View

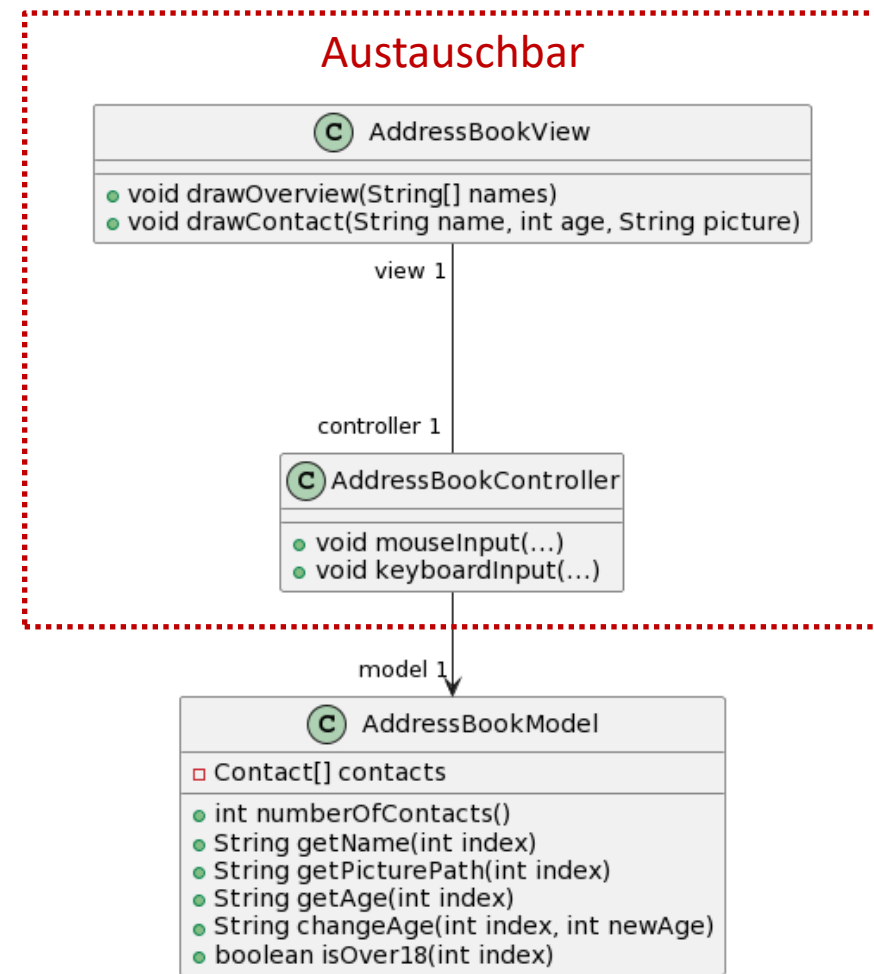
- drawOverview(): Stellt Liste mit allen Kontakten dar
- drawContact(): Zeichnet Details eines Kontaktes

Controller

- Holt Daten aus dem Modell (z.B. Name/Alter einer Person)
- Wählt aus was der View zeichnen soll (Overview oder Contact)

Model

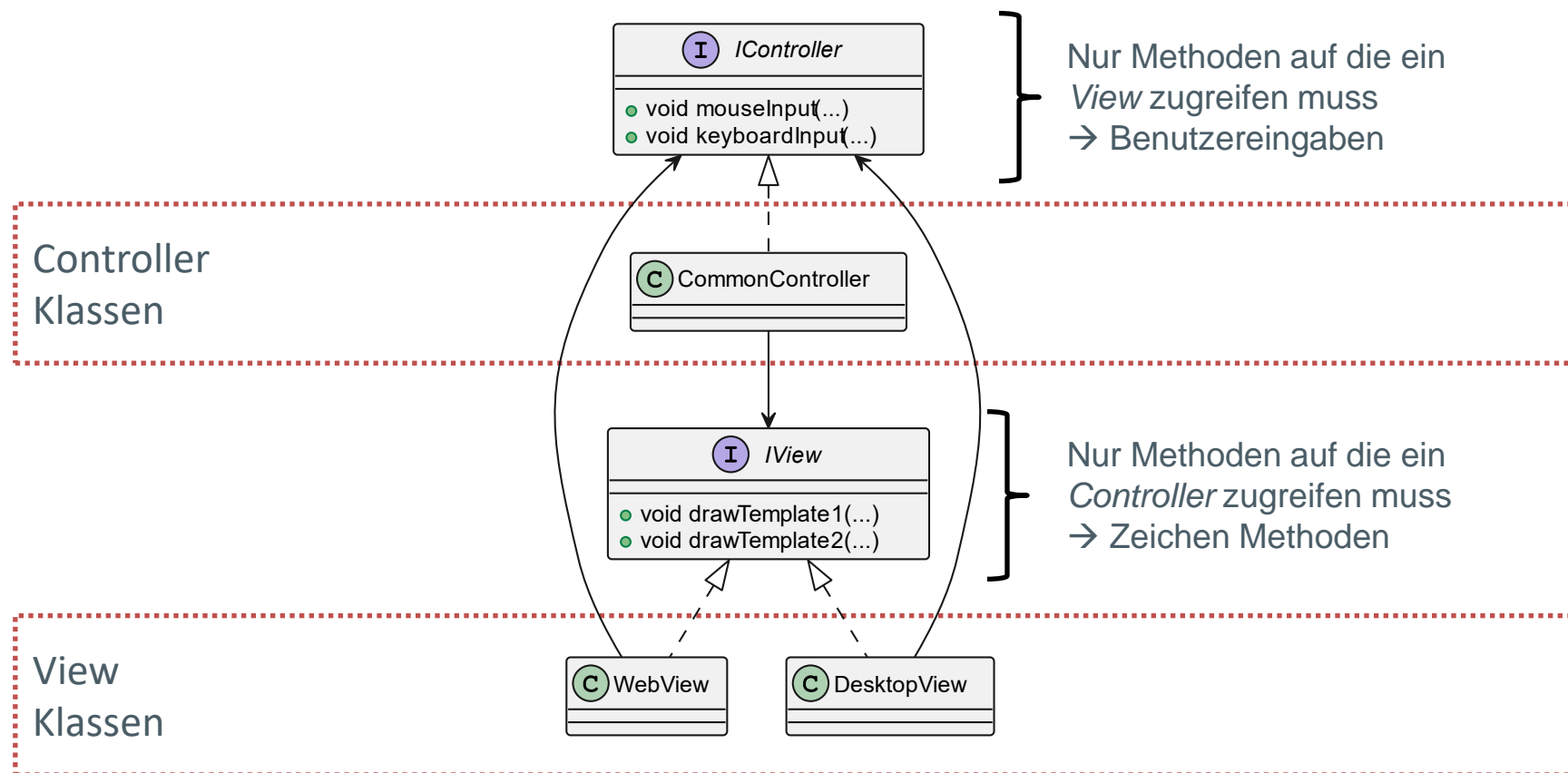
- Enthält Daten des Addressbuchs (contacts)
- Erlaubt Zugriff auf Daten (komplett oder teilweise)
- Validiert Änderungen der Daten (z.B. kein negatives Alter)



View und Controller Interfaces

Erlaubt das Wiederverwenden und Austauschen von View und Controller

Beispiel: Eine Web- und ein Desktop App nutzen den gleichen Controller



Probleme gelöst?

Kein Spaghetticode?

Der Code ist in drei Teile aufgetrennt, jede mit einer genau spezifizierten Aufgabe

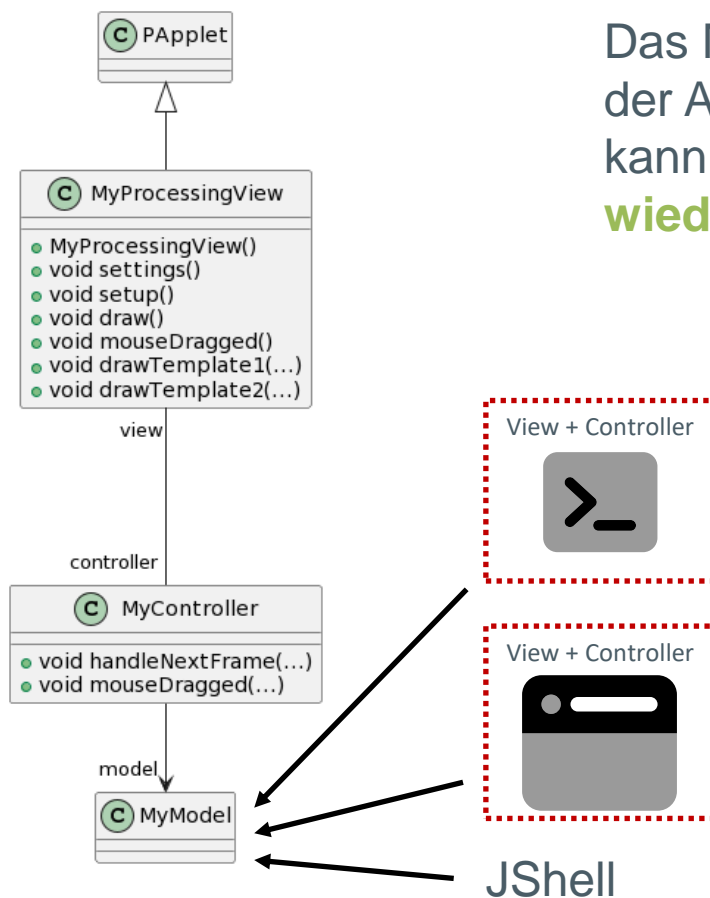
Erlaubt Aufgabenteilung im Team:

- Designer: View
- Entwickler: Model

Anmerkung: Model, View und Controller können auch aus mehreren Klassen bestehen

Mehr Benutzerschnittstellen?

Das Model – der komplexe Teil der Anwendung (Daten + Logik) – kann von anderen UIs **wiederverwendet** werden

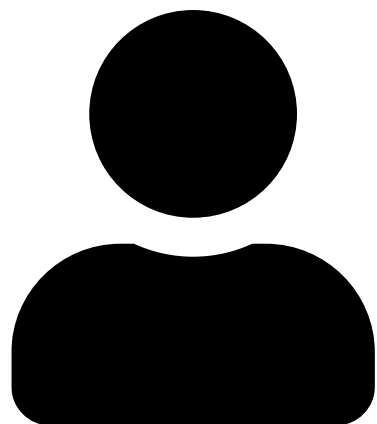




MVC mit Processing

1. UML-Einführung
2. Was ist das Problem mit dem Processing Sketch?
3. Was ist MVC? Welche Vorteile hat es?
4. **MVC mit Processing**
5. Interaktives Model-Testen mit der JShell

Wie sieht eine typische Interaktion aus?



Mensch

Benutzerschnittstelle

Maus, Keyboard, Touch,
Sprache, Gesten, ...

Eingabe

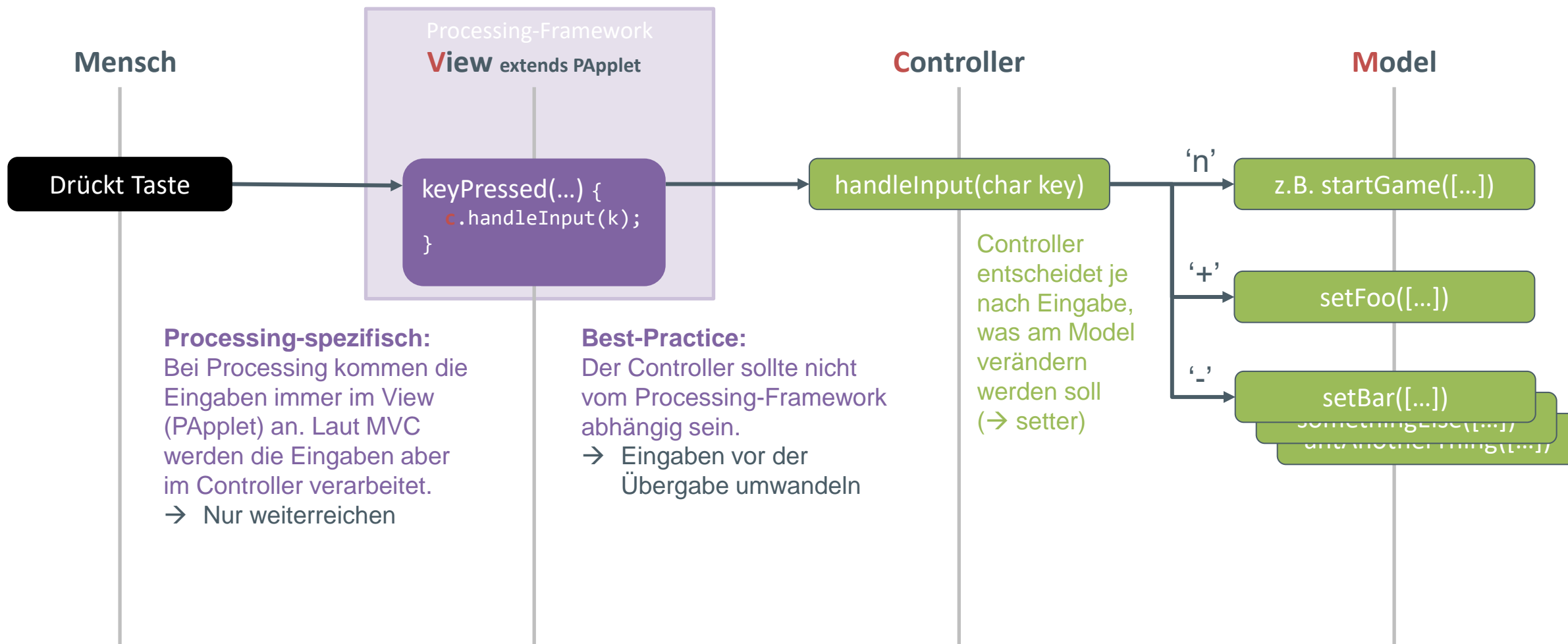
Ausgabe

Bild, Audio, Haptik, ...



Computer

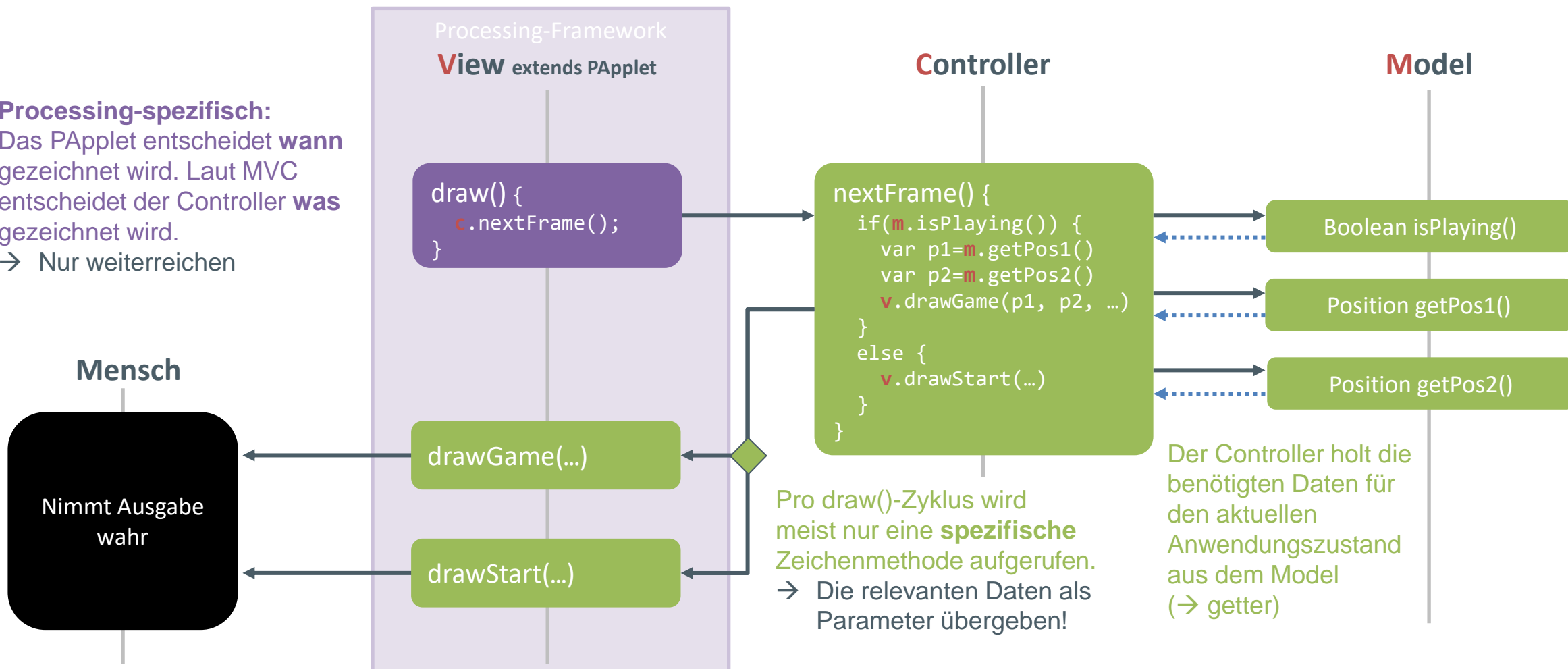
Eingaben mit Processing und MVC: Ein Beispiel



Legende: Lila-Boxen sind Processing-spezifisch; Grüne Boxen sind beim MVC immer gleich.

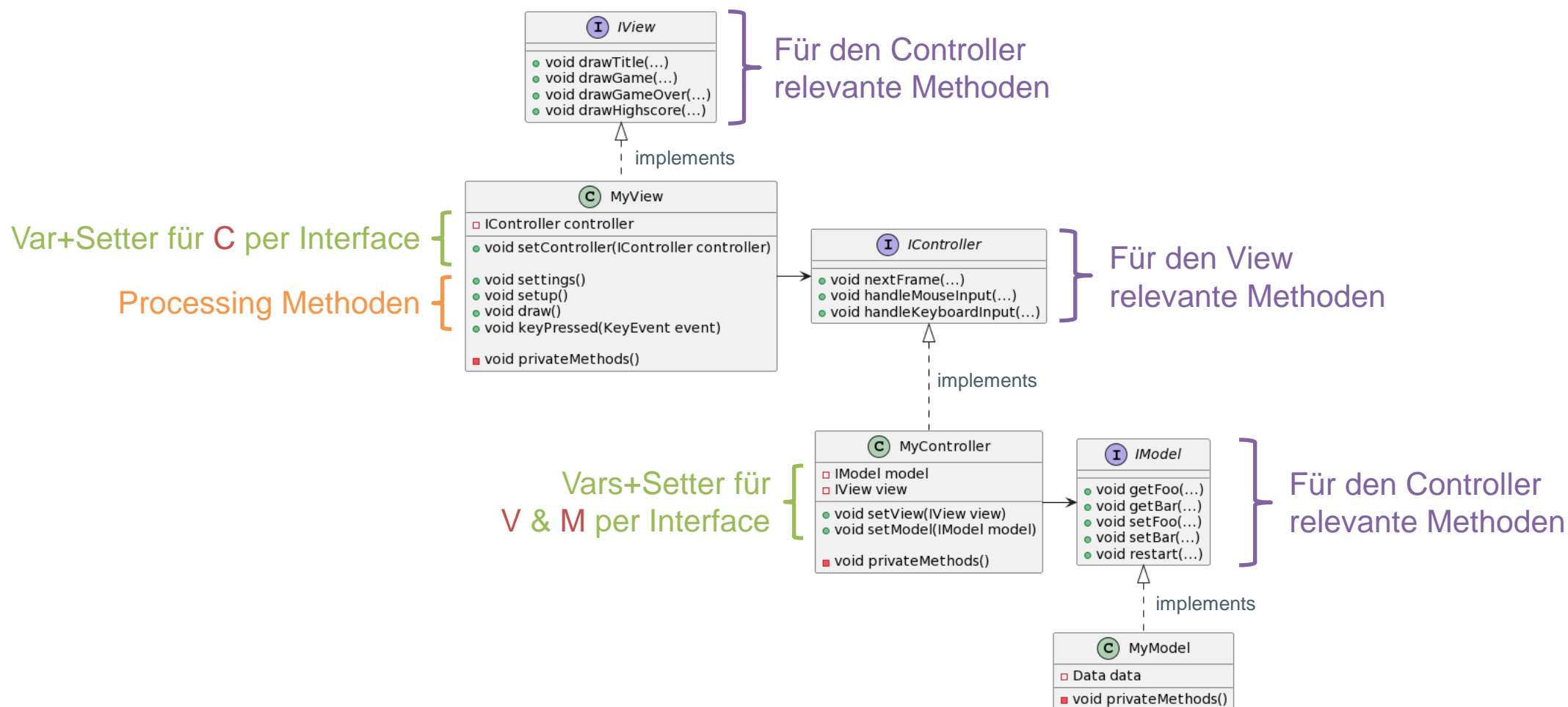
Ausgaben mit Processing und MVC: Ein Beispiel

Processing-spezifisch:
Das PApplet entscheidet **wann** gezeichnet wird. Laut MVC entscheidet der Controller **was** gezeichnet wird.
→ Nur weiterreichen



Legende: Lila-Boxen sind Processing-spezifisch; Grüne Boxen sind beim MVC immer gleich.

Typische **MVC** Klassen und Interfaces



Typische main() Methode für MVC mit Processing

```
public static void main(String[] args) {  
    var model = new MyModel(1000);  
    var controller = new MyController();  
    var view = new MyView();  
  
    // Connect M, V and C  
    controller.setModel(model);  
    controller.setView(view);  
    view.setController(controller);  
  
    // Starts the processing application  
    PApplet.runSketch(new String[]{"MyView"}, view);  
}
```

1. Erstellen von **Model**, **View**, **Controller** Objekten
2. Bekanntmachen der Objekte miteinander:
 - **Controller** kennt **View** & **Model** über Interfaces
 - **View** kennt nur den **Controller** über Interfaces
 - **Model** kennt niemanden*Internes abspeichern der Objekte als Variablen*
3. Starten eines Processing Sketches mit dem erstellten View-Objekt

Vorteil:

- Objekte sind leicht austauschbar
- MVC Konzeptes ist gut kontrollierbar

Zusammenfassung

	Model	Controller	View
Aufgaben	Anwendungslogik , Daten verwalten, Datenbankzugriffe, Netzwerkzugriffe, Spielstandverwaltung, ...	Verarbeitet Eingaben und passt das Model an. Entscheidet was der View darstellen soll, indem er eine Zeichenmethode auswählt und die notwendigen Daten als Parameter übergibt	Wie die Ausgabe der Benutzerschnittstelle dargestellt wird (z.B. Grafik, Audio oder Haptik)
Zugriff auf Processing	Nein	Nein	Ja
Kennt per Interface... Als Instanzvariable gespeichert Kein anderer Zugriff erlaubt (z.B. per Parameter oder static)	-	← Model und View →	← Controller
JShell Nutzung	Ja	Nicht notwendig	Nein



MVC mit Processing

1. UML-Einführung
2. Was ist das Problem mit dem Processing Sketch?
3. Was ist MVC? Welche Vorteile hat es?
4. MVC mit Processing
5. Interaktives Model-Testen mit der JShell

Was ist die JShell?

Ein Java REPL (**r**ead-**e**val-**p**rint **l**oop)

Kommandozeilenprogramm seit JDK 9

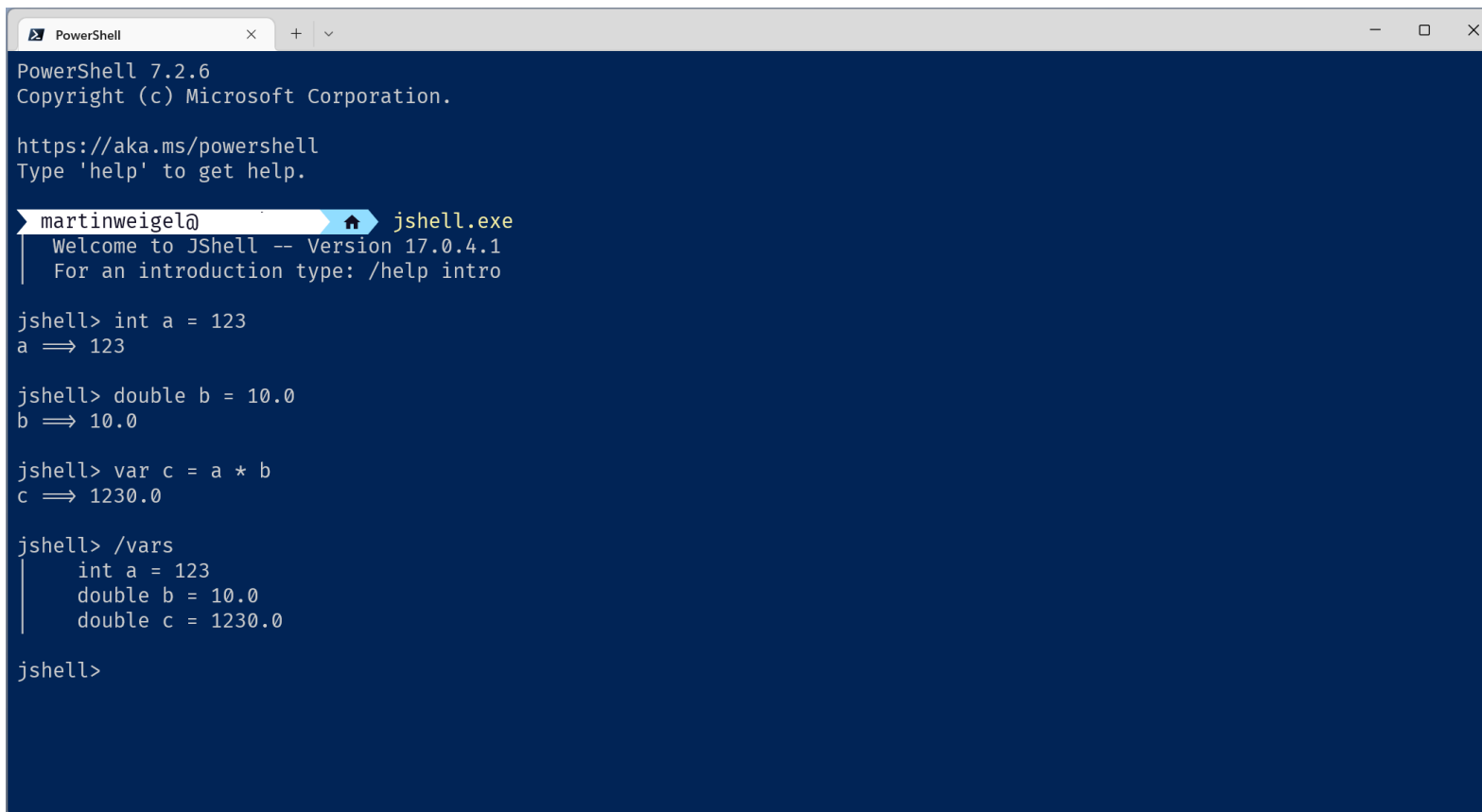
Häufig verwendet für:

- Interaktives Lernen von Java
- Testen von einzelnen Klassen

Im Kurs als zweite “Benutzerschnittstelle” verwendet

- Importieren und interaktives benutzen der Modell-Klasse
- Ähnlich zu einer Kommandozeilenanwendung
- Zum Prüfen, ob MVC eingehalten wurde (ohne Zusatzaufwand)

JShell Beispiel



```
PowerShell 7.2.6
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

martinweigel@ jshell.exe
Welcome to JShell -- Version 17.0.4.1
For an introduction type: /help intro

jshell> int a = 123
a => 123

jshell> double b = 10.0
b => 10.0

jshell> var c = a * b
c => 1230.0

jshell> /vars
int a = 123
double b = 10.0
double c = 1230.0

jshell>
```

Führt normalen Java-Code aus

Wichtige Kommandos:

/help

Zeigt Hilfe-Text

/vars

Zeigt Variablen und Inhalte

/open <file>

Inkludiert den Dateiinhalt

→ Abhängigkeiten beachten

/save <file>

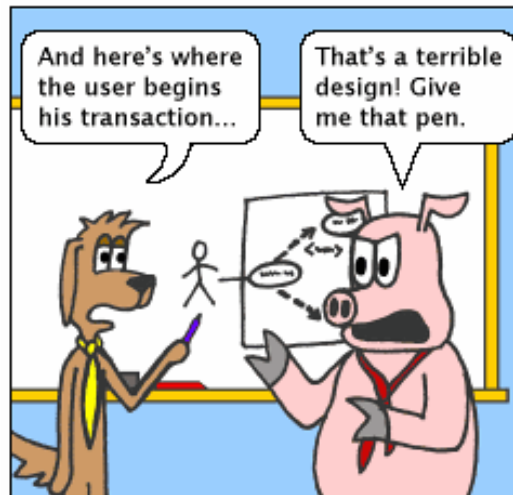
Speichert Quellcode

/exit

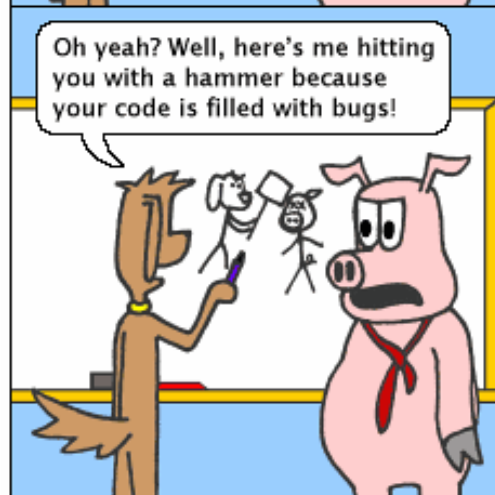
Beendet die JShell

Fragen?

Hackles



By Drake Emko & Jen Brodzik



<http://hackles.org>

Copyright © 2002 Drake Emko & Jen Brodzik

[Bildquelle: <http://hackles.org/cgi-bin/archives.pl?request=187>]