hochschule mannheim
Fakultät für Informatik

Programmierung 2

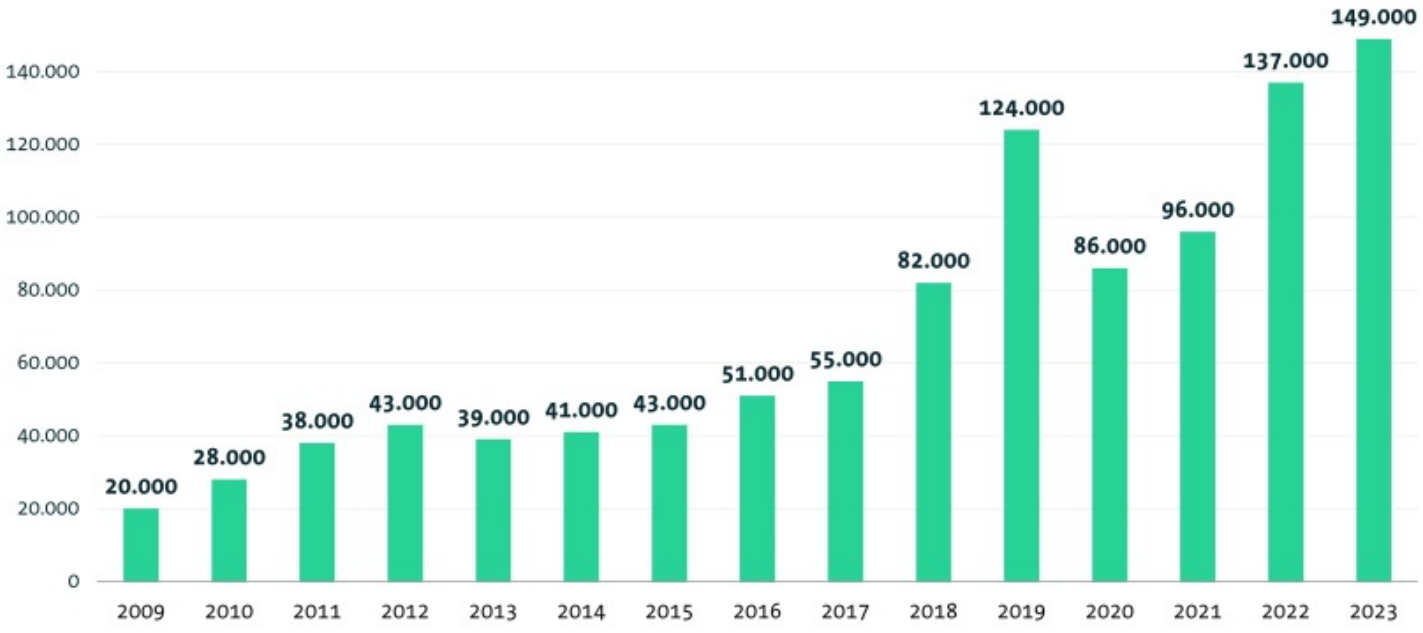
Für alle Studierenden
OHNE Studienleistung!

Herzlich willkommen!

Prof. Dr. Oliver Hummel
Wintersemester 2024/25

Unternehmen fehlen aktuell 149.000 IT-Fachkräfte

Anzahl zu besetzender IT-Stellen in der Gesamtwirtschaft



Basis: Unternehmen ab 3 Beschäftigten in Deutschland (n=853) | Datenerhebung: jeweils im September | Quelle: Bitkom Research 2023

bitkom

Was erwartet Sie heute?

1. Gegenseitiges Kennenlernen & Erwartungsabgleich
2. Zusammenarbeit und Veranstaltungsorganisation
3. Wiederholung der Grundlagen der Objektorientierung

Zu meiner Person

Oliver Hummel… seit März 2017 Professor für Big Data

- *davor als Entwicklungsleiter der IQSER GmbH für die Entwicklung einer semantischen Middleware verantwortlich*

Leiter des Gründungszentrums der Hochschule

- https://launchtomars.de

MACH dein Ding!

Weitere Stationen

- Vertretungsprofessor am KIT
- Juniorprofessur an der Uni Mannheim
- Consulting für Perot Systems
- Promotion an der Uni Mannheim
- Informatikstudium an der TU Kaiserslautern

Zu meiner Person II

→ Gute Mischung aus akademischer & praktischer Erfahrung

- Projekte mit Lufthansa, Henkel, Audi, EADS, der Bundesmarine u.a.
- zahlreiche nationale und internationale Veröffentlichungen
- umfangreiche Lehrerfahrung und Hochschuldidaktik-Ausbildung
- gute Vernetzung in der Startup-Community

Kontaktmöglichkeiten

Prof. Dr. Oliver Hummel

- E-Mail: o.hummel@hs-mannheim.de
- Büro: Gebäude A, Raum 007a
- Sprechstunde: nach Vereinbarung
- Homepage: http://bit.ly/hum-hsma

Achtung, Moodle-Nachrichten werden nicht gelesen!

Stichwort: Problemlösungskompetenz

→ *Wiederkäuen allein reicht nicht!*

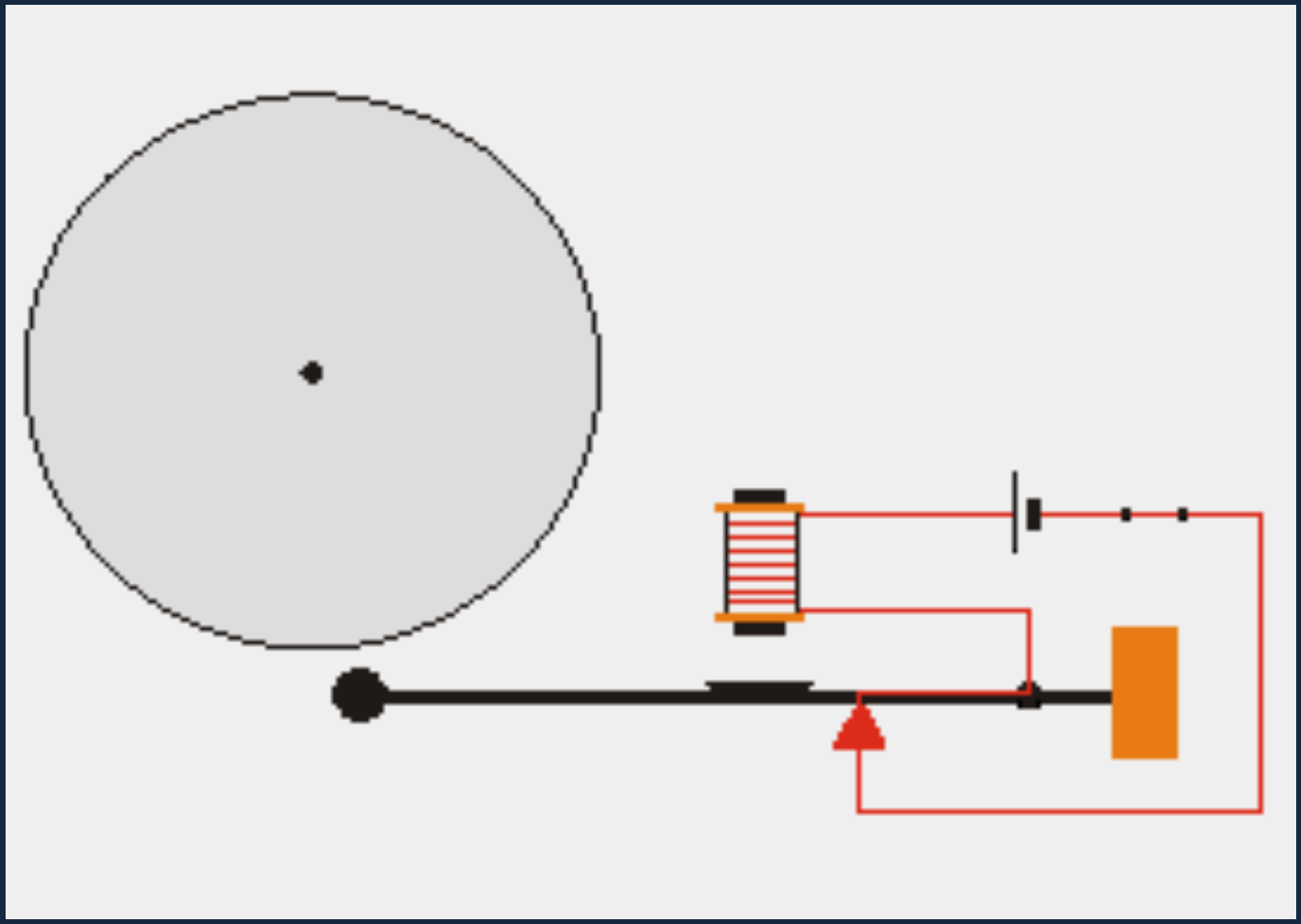
- weder später im Beruf
- noch in Ihrem Studium

Informatik ist ein anspruchsvolles Fach, das es erfordert, erlerntes Wissen gezielt zur Lösung von bis dato unbekannten Problemen einzusetzen

- und: die eine „1+1=2“-Lösung gibt es oft nicht

→ Sie erlernen hier nur die Grundlagen, denken Sie bitte (wie immer im Leben) selbstständig & kritisch (weiter)

Damals im Physikunterricht



Voraussetzungen

Absolut **wichtige Grundlagen** aus PR1 sind:

- Syntax laut Cheat Sheet: https://services.informatik.hs-mannheim.de/~hummel/java-cheat-sheet.pdf
- Nutzung einer IDE (Eclipse)

- folgende Konzepte verwenden können:
  - Eingabe – Verarbeitung – Ausgabe
  - Unterprogramme
  - Datentypen
  - Fallunterscheidungen & Schleifen
  - Arrays und Listen
  - static und andere Modifier (public, private, final ...)

→ *ich erwarte, dass Sie evtl. **Lücken ASAP selbstständig aufarbeiten**!*

Ferner wichtig ist ein **Grundverständnis der Objektorientierung**

- https://www.youtube.com/watch?v=DxD-aQIar4I

Vorlesungsorga

Vorlesung (2 Doppelstunden)

- zur Vermittlung neuer Inhalte, mit kleineren Übungsaufgaben
  - Montag 13:40 – 15:10 Uhr
  - Mittwoch 9:45 – 11:15 Uhr

Labor-Übungen (2 Doppelstunden)

- zur Wiederholung, Festigung und Diskussion der Vorlesungsinhalte
  - sowie zum Bearbeiten und Testieren der Studienleistungen
- regelmäßig dienstags 13:40 – 16:50 Uhr in A008 und A010
  - *gemeinsam mit Michael Köhler und stud. Tutoren*

Arbeitsaufwand

Programmierung 2

- 8 SWS (4 SWS Vorlesung + 4 SWS Übung)
- 10 ECTS ≡ 10 * 30 h = 300 h Aufwand im Semester
  - ***davon 2/3 in der Vorlesungszeit → 200 h***
  - *1/3 zur Klausurvorbereitung und Nachbereitung des Semesters*

- Das bedeutet bei 15 Vorlesungswochen
  - 8 WS Vorlesung/Übung pro Woche = 120 WS Vorlesung
    ~90 h Zeitstunden Präsenz

D.h., es bleiben noch 110 Stunden
(→ pro Woche ca. 7,5) für:

- Vor- und Nachbereitung
- Übungsbearbeitung
- Literaturstudium
- **eigenständiges Üben**
- etc.

~ 1/3 Ihrer Wochenarbeitszeit von 40 Stunden!

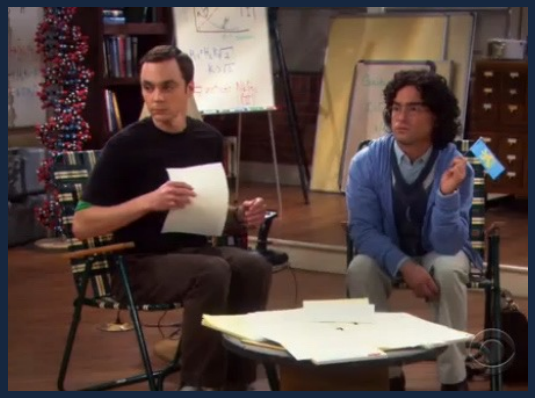
Unser „Roommate Agreement“

Mein Ziel ist es, eine angenehme Lernatmosphäre für alle zu schaffen

Ich biete dazu:

- eine gut vorbereite Veranstaltung auf dem Stand von Wissenschaft und Technik
- einen pünktlichen Beginn und ein pünktliches Ende
- den Einsatz moderner Lehrmethoden und Lernen ohne Angst
- Raum für Rückfragen und Diskussionen
- Verschnaufpausen für ggf. nötige Diskussionen untereinander
- eine Sprechstunde und die Möglichkeit Fragen per Mail zu stellen
- die verwendeten Materialien zum Download

Roommate Agreement → Ihr Teil



www.youtube.com/watch?v=N0qDy0T5WXM

Bitte unterstützen Sie in Ihrem eigenen Interesse meinen Einsatz

- *in aller Kürze: durch Ruhe, Ihre Aufmerksamkeit und Mitarbeit*

Im Detail erwarte ich mindestens:

- Ruhe
  - *jeder hat einmal einen schlechten Tag, nur dann verhalten Sie sich bitte ruhig*
  - *stellen Sie elektronische Geräte auf lautlos & nutzen Sie sie nur "upon request" bzw. vorlesungsbezogen*
- Mitarbeit
  - *Nutzen Sie die gemeinsame Zeit möglichst sinnvoll und denken und arbeiten Sie mit*

Vorlesungsmaterialien & Übungsaufgaben

... finden Sie auf Moodle: http://moodle.hs-mannheim.de

→ *Alle Kurse → Fakultät für Informatik → Prof. Oliver Hummel → PR2 IB SS24*

Die Folien sind im Wesentlichen Gedankenstütze für mich!

- sie sind KEIN ausformuliertes Skript
  - und auch kein Buchersatz
- Tafelbeispiele und mündliche Erklärungen u.ä. fehlen

- *→ daher bitte anwesend sein & eigene Notizen anfertigen!*

- Literatur: https://bigdataengineering.blogspot.com/2019/09/buchtipps-fur-informatik-studierende.html
  - weitere Hinweise ggf. in den Folien direkt

Zusammenfassend: Fördern und Fordern

jedoch: Google (Gemini) is your friend!

Fördern

- Wichtigster Hinweis: es gibt (fast) keine dummen Fragen
  - Sie sind hier, um etwas zu lernen
  - Wir Betreuer sind hier, um Ihnen etwas beizubringen
    - auch wir haben “mal angefangen“
- Ich möchte Sie inspirieren und motivieren, Ihre eigenen Ideen zu verfolgen und umzusetzen

Zusammenfassend: Fördern und Fordern

Fordern

- Programmiersprachen erlernt man wie echte Sprachen nur durch „Sprechen“
  - → ***also durch Programmieren!***
- Regeln und Termine sind einzuhalten
- keine Ausreden bei den Pflichtübungen

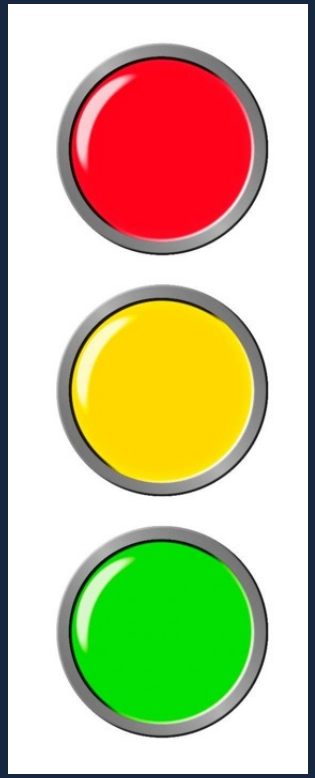
→ Fangen Sie direkt mit dem Lernen und Üben an und bleiben Sie am Ball!

- *Sie erinnern sich an die 13,5 h in der Woche?*

Studienleistung

Wird mit grün (ab ca. 80%), gelb (ab 50%) oder rot (unter 50%) bewertet

Die Studienleistung ist beim 2. Rot nicht bestanden



Geplante Testate (ohne Gewähr)

- Qualifier (Teil 2: 08.10.), *Teil 1: 30.09. 17 Uhr*
- Schriftliches Testat (29.10.)
- Heimaufgabe 1 (19.11.)
- Schriftliches Testat (03.12.)
- Heimaufgabe 2 (07.01.)

Bei einer evtl. Erkrankung reichen Sie bitte innerhalb von 2 Tagen eine ärztliche Krankschreibung bei mir ein

- *es wird am Ende des Semesters einen schriftlichen Nachtermin geben*

Denken Sie über Ihre Prioritäten nach!

https://youtu.be/L1v8kbYm58c?t=60

1,0

Lern- und Kompetenzziele laut Modulkatalog

Nach Besuch der Veranstaltung sind die Studierenden in der Lage:

- verschiedene Konzepte in Java zu beurteilen
- alle wichtigen **Konzepte** von Java **anzuwenden**
- **nichttriviale objektorientierte Programme** in Java zu entwickeln
- in kleinen Teams zu arbeiten und ihre Arbeit vorzustellen
- den Aufwand für Algorithmen abzuschätzen
- unterschiedliche Algorithmen und dynamische Datenstrukturen in Hinblick auf ihre Anwendung zu beurteilen und zu implementieren

Wegweiser PR2-Inhalte

- Objektorientierung / Klasse Object (Wiederholung)
- Polymorphismus (Vertiefung)
- JUnit und Software-Qualität
- Innere Klassen und Lambdas
- Input/Output (Grundlagen)
- Exception Handling
- Grundlagen der Nebenläufigkeit (Threads)
- Generics (Grundlagen)
- Collection Framework
- Wichtige Datenstrukturen
  - Hash(maps), Bäume, Tries, Graphen
- Sortieren

Wdh.: Objektorientierung: Gemeinsames Vokabular

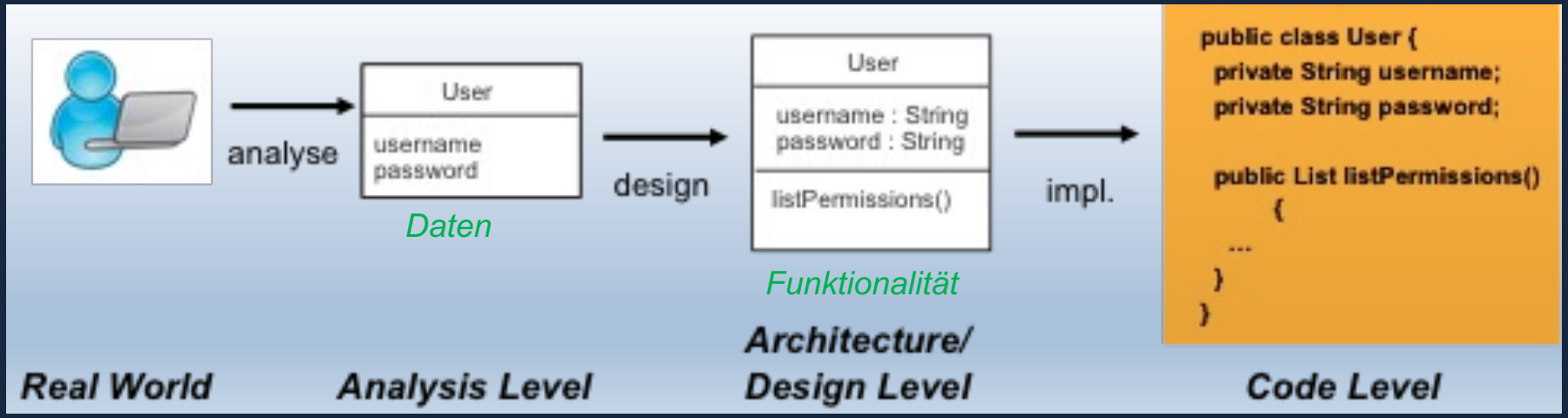
Die Idee der **Objektorientierung** ist, die **reale Welt** im Softwaresystem „nachzubauen“

- wollen wir bspw. Software für eine Bank entwickeln, sollen die **Objekte der realen Welt auch in Software** erscheinen
  - *Bank, Bankkunde, Bankkonto, Aktie, Darlehensvertrag...*
  - **schwierig** bleibt jedoch oft die Zuordnung von **Abläufen** in der realen Welt, also bspw. wohin mit den Geschäftsprozessen

→ die Hoffnung ist, dass sich so Analysten, Architekten, Entwickler und Tester (also alle sog. Stakeholder) besser im Projekt zurecht finden

Gemeinsames Vokabular über alle Entwicklungsstufen hinweg

- ausgehend von den Datenstrukturen



Daten mit Struktur

Die objektorientierte Modellierung erfolgt ausgehend von den Daten, die Objekte charakterisieren

- Klassen geben die Struktur für ihre Objekte vor

- *→ skizzieren Sie eine Java-Klasse für das gezeigte Beispiel*

In einer Datenbank könnte das vorherige Bild etwa wie folgt gespeichert werden

Tabelle Student (vgl. DM)

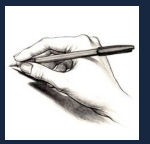
Name `String`	**Vorname** `String`	**Matrikelnummer** `int`	**Studiengang** `String`
Mayer	Klaus	123456	UIB
Müller	Hans	234567	IMB
Fischer	Katja	224561	IMB
Vogel	Christina	324561	IB
Schneider	Claudia	134569	CSB

Daten und zugehörige Funktionen

Der nächste Schritt ist das **Hinzufügen von Funktionalität**, also Methoden

- Daten und Funktionalität können natürlich auch iterativ hinzugefügt werden

Bspw. könnten wir der Klasse noch ihr Startsemester hinzufügen und auf dieser Basis das Fachsemester berechnen



→ Eine wichtige **Heuristik** ist, dass **Methoden möglichst nur auf Attribute ihrer Klasse** zugreifen sollen

- sog. *Information Expert Pattern* (vgl. Larman)

Merksätze zur Objektorientierung

1. Objektorientierung heißt, dass Daten und zugehörige Funktionalität in einer Klasse zusammengefasst werden

2. Software-Objekte sind wie Sachbearbeiter, die gemeinsam eine Firma (-> das Programm) am Laufen halten

Üben!

Bearbeiten Sie zur Übung folgende kleinere Aufgabenstellungen:

1. Kontext **Bank**: Implementieren Sie ein Bankkonto mit einem Inhaber(namen), einer Kontonummer sowie einem Kontostand. Ergänzen Sie dann Methoden zum Ein- und Auszahlen von Geldbeträgen.

2. Kontext **Autovermietung**: implementieren Sie eine PKW-Klasse, mit passenden Attributen und der Funktionalität um gefahrene Kilometer einzugeben, die auf den Kilometerstand addiert werden

3. Kontext **Weinhandlung**: Implementieren Sie eine Klasse für einen Wein, die die in einer Flasche enthaltene Alkoholmenge berechnen kann

4. Kontext **Lauf-App**: Implementieren Sie eine Klasse für einen Lauf, die an Hand der gespeicherten Strecke und der dafür benötigten Zeit die durchschnittliche Zeit pro Kilometer berechnet

Blitz-Abfrage OO-Grundlagen

Sind Ihnen folgende Punkte geläufig?

1. Geheimnisprinzip
2. Punktnotation für Methoden und Attribute von Objekten
3. Nutzen und Nutzung der toString-Methode
4. wie sich Objekte vergleichen lassen (equals-Methode)
5. Unterschied von Stack und Heap und Zusammenhang mit Objekten
6. Konstruktoren
7. Überladen von Methoden (und Konstruktoren)
8. this
9. static
10. Wrapper-Klassen für primitive Datentypen und Autoboxing

Zusammenfassend: Grundlegendes Entwurfsvorgehen

Aus Gründen der Änderbarkeit und Testbarkeit von Programmen empfiehlt es sich folgende (erste) Heuristiken zu befolgen:

1. Beginnen Sie mit der Datenmodellierung
   - der Klassen und ihrer Attribute → *Substantive aus der realen Welt*
2. Fügen Sie die Geschäftslogik dort hinzu, wo die entsprechenden Daten sind
   - z.B. Methoden zur Bearbeitung von Geschäftsvorfällen → *Verben aus der realen Welt*
3. Befolgen Sie das Geheimnisprinzip
   - Geschäftsobjekte sollten innerhalb des Systems verbleiben
   - Halten Sie die Schnittstelle des Systems (die public-Methoden) so klein wie möglich
4. Trennen Sie Ein- und Ausgabe (UI) von der Geschäftslogik (Architektur!)
   - die UI liegt „außerhalb“ des Systems und nutzt dessen Schnittstelle
   - achten Sie auf Testbarkeit (insbes. keine Nutzeringaben in Geschäftsmethoden!)

Wichtige Entwurfsprinzipien

Separation of Concerns

- klare Aufgabenteilung

Single Responsibility Principle

- pro Paket/ Klasse/Methode nur eine Aufgabe

Geheimnisprinzip

- nur die Dinge nach außen geben, die wirklich notwendig sind

DRY: Don‘t Repeat Yourself

- Keine Code-Duplizierung

YAGNI: You ain‘t gonna need it

- Features erst einbauen, wenn sie wirklich benötigt werden

Single Responsibility Principle (SRP)

Methoden in der objektorientierten Programmierung sind oft sehr kurz

- das ist durchaus gewünscht
- über die Vor- und Nachteile streiten die Gelehrten...

→ *Methoden sollen immer nur eine Aufgabe (Verantwortlichkeit) ausfüllen*

- *und vor allem auch niemals Kommandos und Abfragen vermischen*
- sog. Command-Query Separation

```
public class Die {
   private int faceValue;
   ...
   public void roll() {
      faceValue = (int)(Math.random() * 6 + 1);
   }
   public int getFaceValue() {
      return faceValue;
   }
}
```

```
public class Missile {
   private String name;
   ...
   public String getName() {
      launch();
      return name;
   }
}
```

Das sollten Sie können/üben...

1. Den Zusammenhang zwischen Software-Objekten und Objekten in der realen Welt erklären
2. Einfache Klassen mit Attributen und Methoden schreiben
3. Den Unterschied zwischen einer prozeduralen und objektorientierten Implementierung erklären

Gibt es noch Fragen?