Objektorientierte Softwareentwicklung

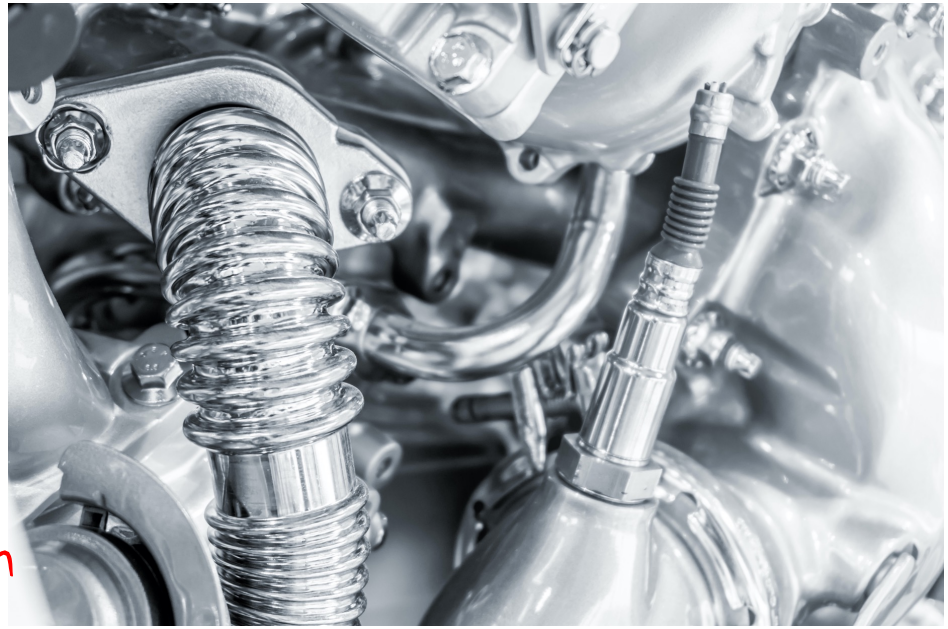
2 – Das Kapselungsprinzip

Kapselungsprinzip

- Englisch: Information Hiding – Verstecken von Informationen
- Warum sollen Programme Informationen verstecken?

Beispiel: Auto

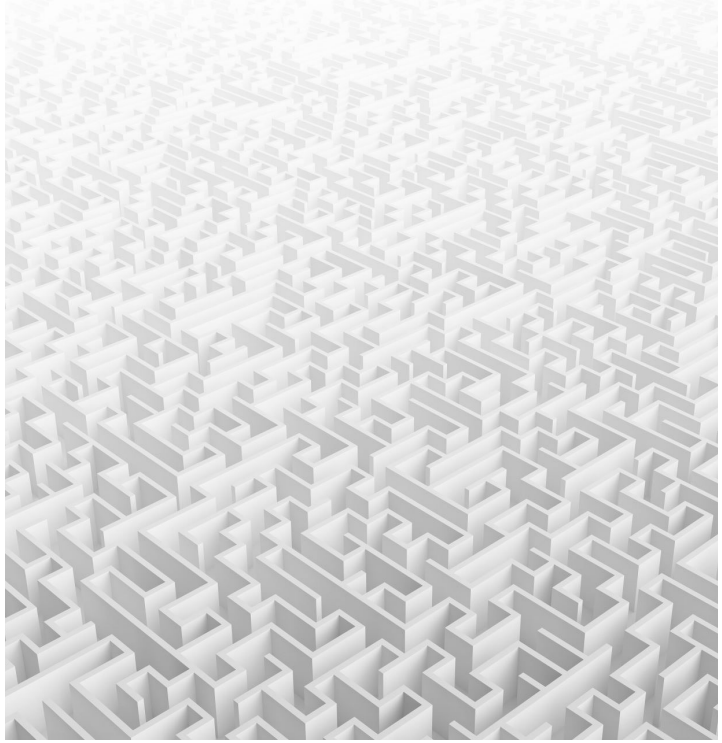
- Extrem komplexes System aus tausenden Bauteilen und Millionen Zeilen von Software
- Warum kann trotzdem (fast) jeder in ein paar Stunden Fahren lernen?

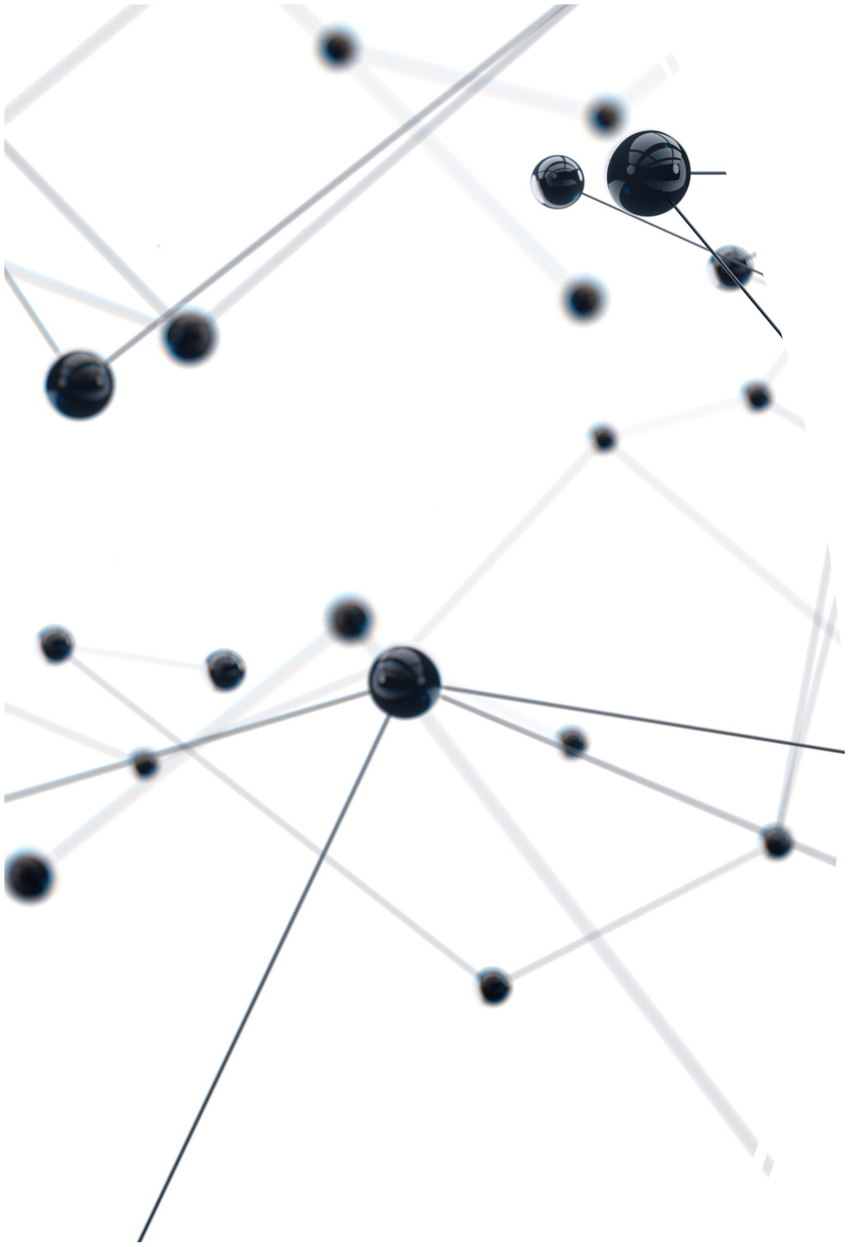


→ Komplexität hinter einfacher öffentlicher Schnittstelle verbergen (Gaspedal, Pedale, ...)

Motivation des Kapselungsprinzips

- Software ist extrem komplex
  - Große Projekte: mehrere Millionen Zeilen Quellcode
  - Bereits eine int-Variable hat über 4,2 Milliarden mögliche Zustände
  - Kombinatorische Explosion der möglichen Ausführungspfade durch Fallunterscheidungen und Schleifen
  - Kann von Menschen nicht mehr als Ganzes verstanden werden



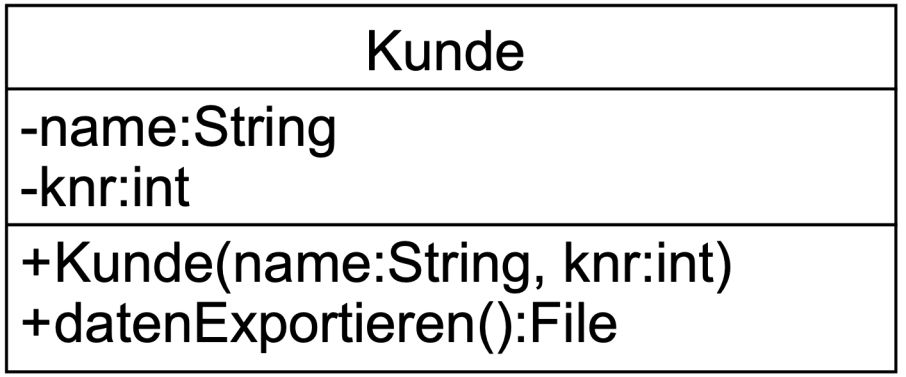


Lösung: Module mit Schnittstellen

- Software aus unabhängigen Modulen (Kapseln) aufbauen
- Module haben eine übersichtliche öffentliche Schnittstelle
- Implementierungsdetails sind verborgen
  - Müssen von Nutzern nicht verstanden werden
  - Können geändert werden, ohne den Rest des Systems zu beeinflussen

Umsetzung in Objektorientierung

- Kapsel = Objekt (Klasse)
- Daten verbergen
  - Alle Attribute immer **private**
- Öffentliche Schnittstelle über Methoden
- Falls Zugriff auf Attribute nötig
  - get-/set-Methoden



get-/set-Methoden

- Andere Namen: Zugriffsmethoden, Akzessormethoden
- **get-Methode**: öffentlicher **lesender** Zugriff auf privates Attribut
- **set-Methode**: öffentlicher **schreibender** Zug auf privates Attribut
- Vorteile
  - Zusätzliche Indirektion ermöglicht Flexibilität und Abstraktion (Kontrolle des Wertebereichs, Protokollieren, Austausch der Implementierung, etc.)
  - Lese-/Schreibzugriff kann getrennt ermöglicht werden
- Nachteile
  - Attribute sind eigentlich wieder global -> Abhängigkeitsprobleme
  - Verletzt OO-Grundprinzip: Daten und ihre Methoden in eine Klasse
  - get-/set-Methoden automatisch für alle Attribute: Anti-Pattern!
  - Ziel: möglichst ohne get-/set-Methoden auskommen (Ausnahme: DAO)

get-Methode

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

```
Kunde k1 = new Kunde("Pit", 23);
System.out.println(k1.getName());
```

→ Pit

Kunde
-name:String -knr:int
+Kunde(name:String, knr:int) +datenExportieren():File +getName():String +getKnr():int

set-Methode

```
public void setName (String name) {
    this.name = name;
}
```

```
Kunde k2 = new Kunde ("Anna", 42);
k2.setName ("Marianne");
System.out.println (k2.getName());
```

~> Marianne

Kunde
-name:String -knr:int
+Kunde(name:String, knr:int) +datenExportieren():File +getName():String +setName(name:String):void +getKnr():int +setKnr(knr:int):void