SEMESTERPROJEKT SE1

Gruppe 34:

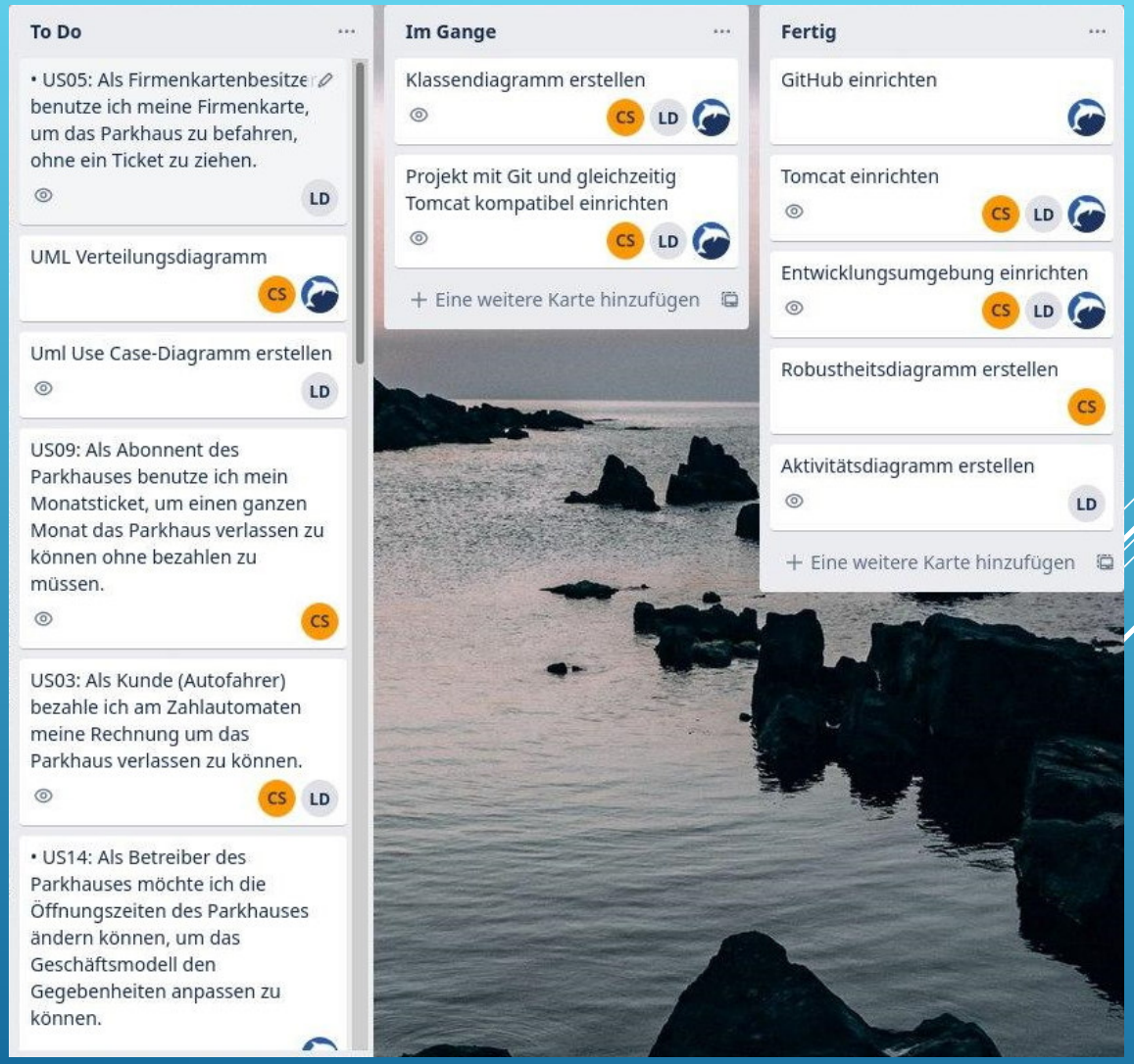
Lawin Daskin
Raphael Kuhn
Constanze Schirmacher

R

INHALTSVERZEICHNIS

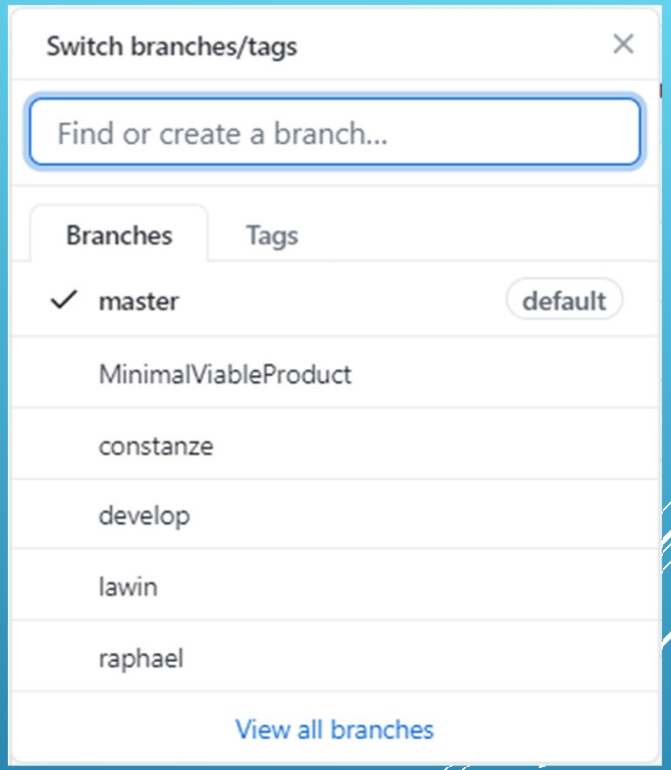
- Verwaltung von Aufgaben:
  - Wer macht was
  - Was ist noch zu tun
  - Prioritäten

KANBAN

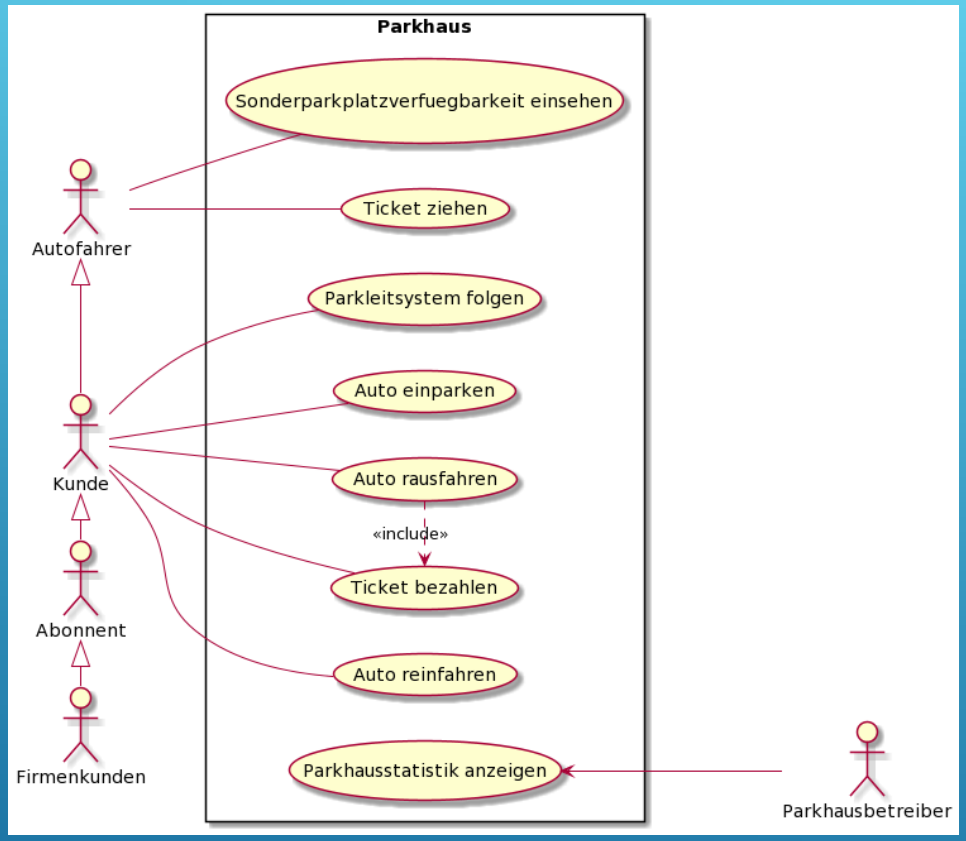


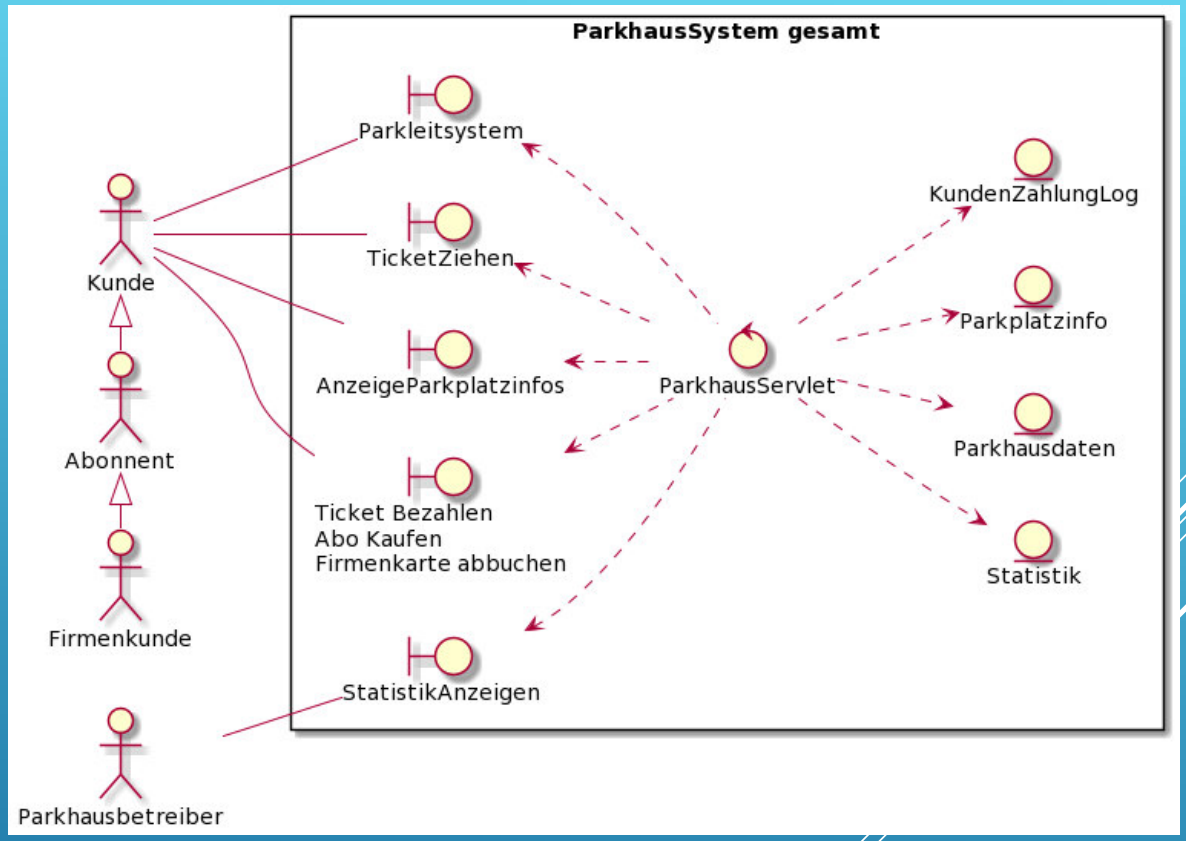
GIT

- Eigene und gemeinsame Branches
- MVP
- Gezielter Austausch neuer Features
- Helfen ohne Fehler/Probleme wegen unvollständigen Implementierungen
- Möglichkeit von Roll-backs in Problemfällen



R

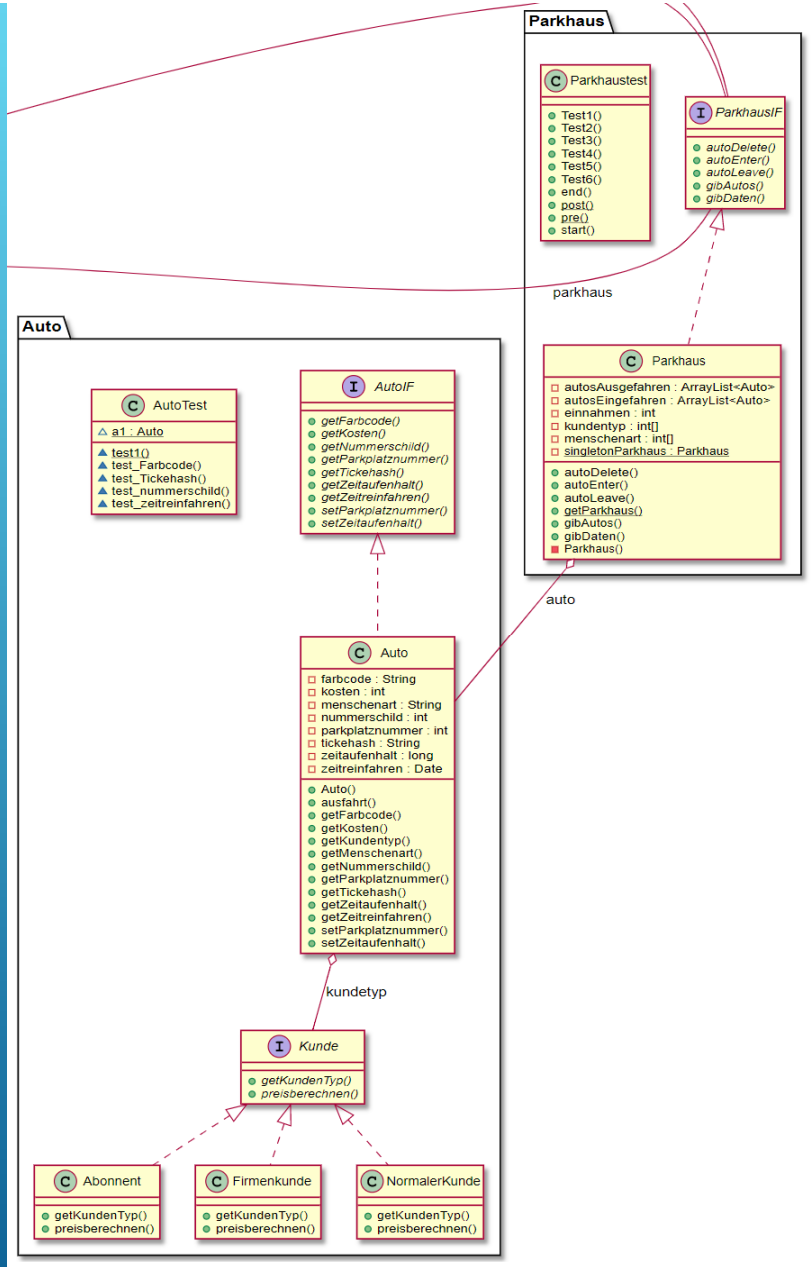




UML-DIAGRAMME

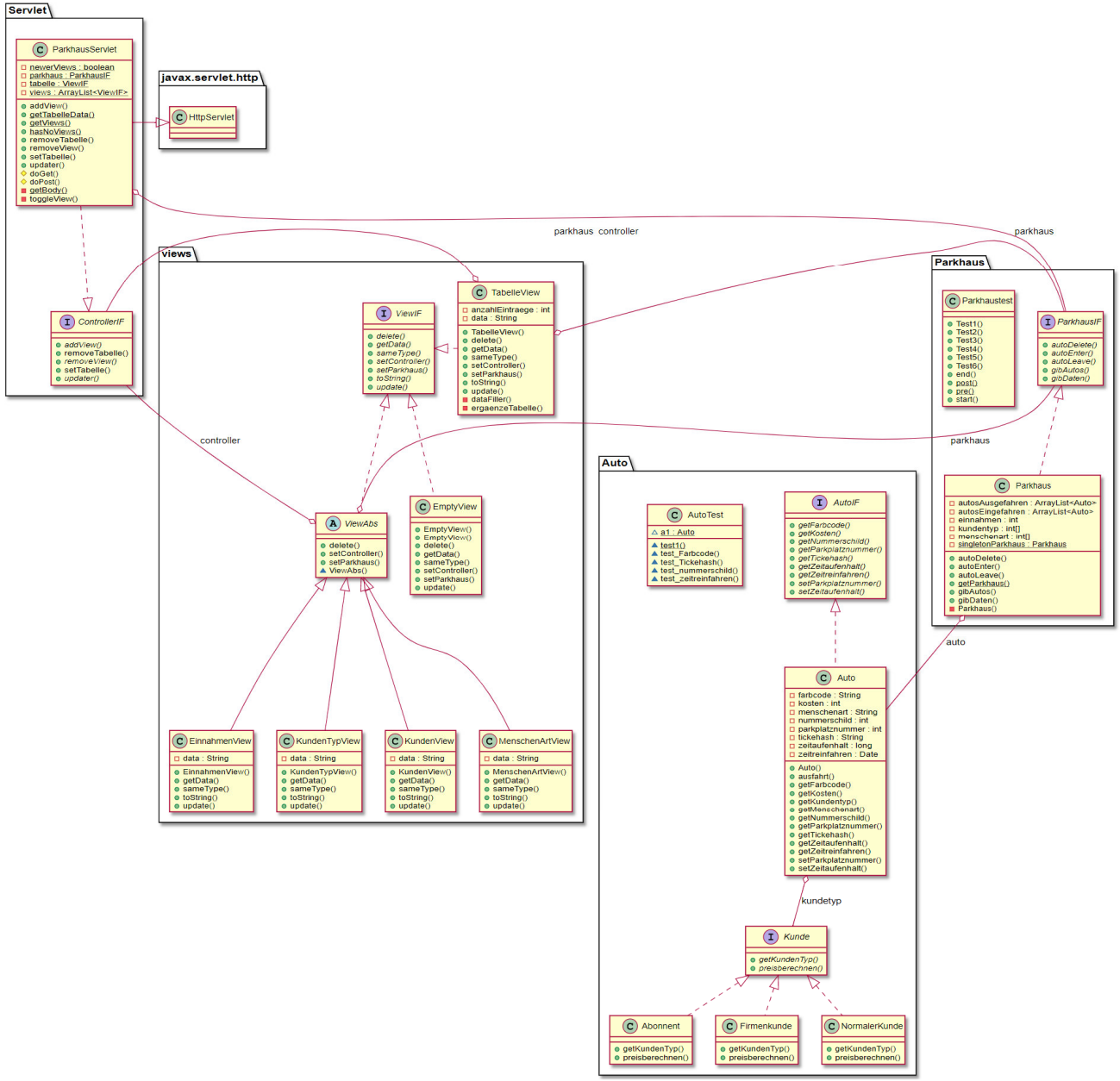
TESTEN

- Testen von aktiven Klassen
- Test driven Development



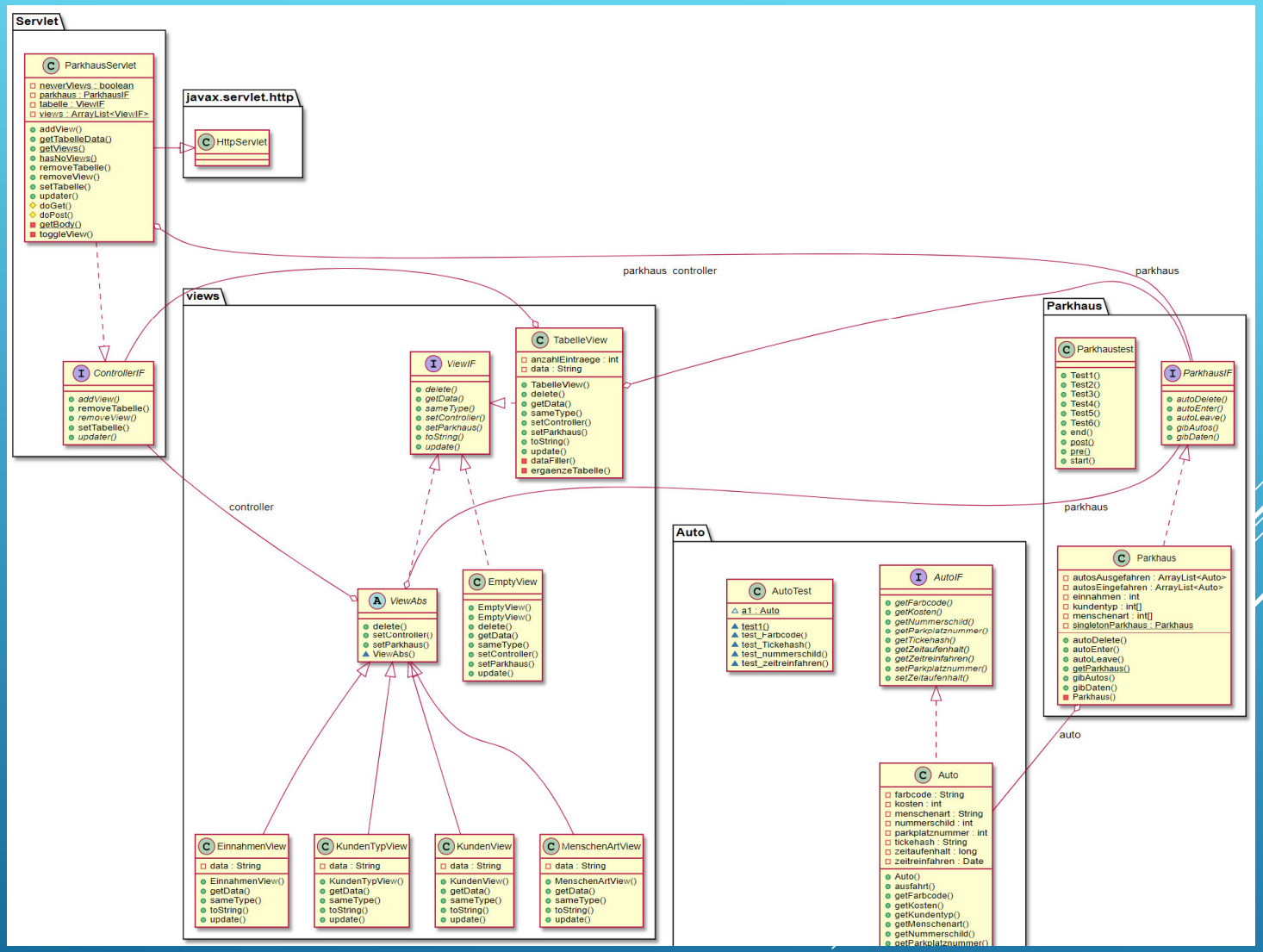
R

UML-KLASSEN-DIAGRAMM



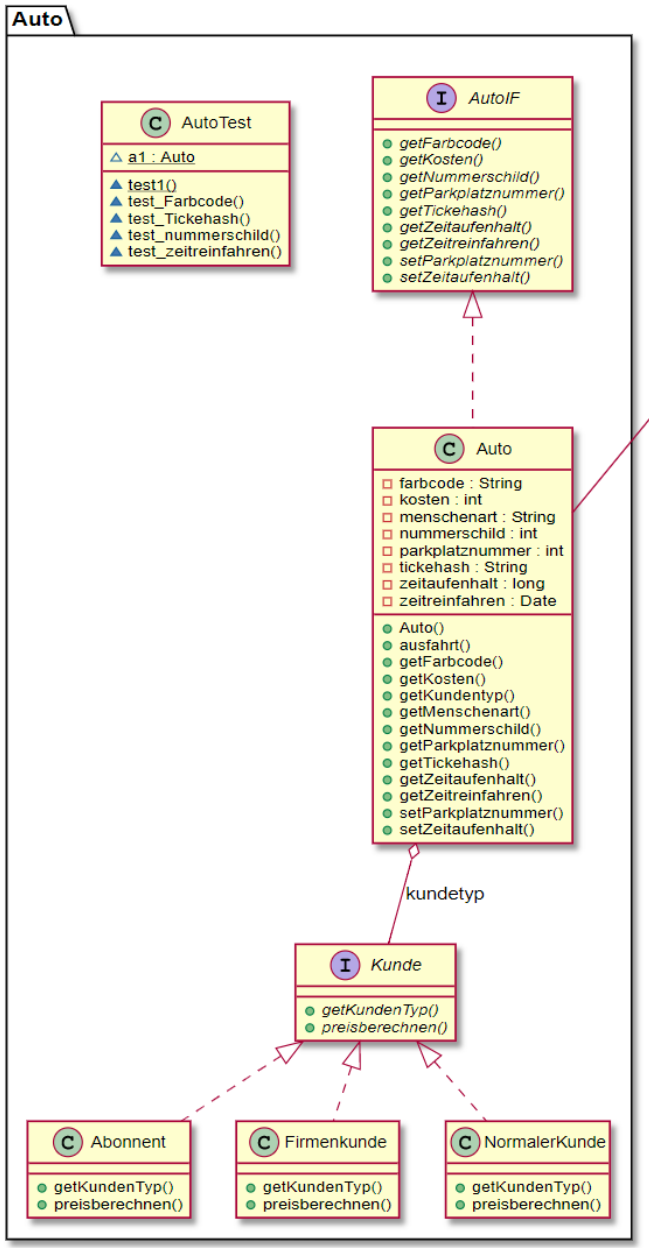
- Passive Variante:
  - Befehle werden von Controller verwaltet
  - Modell nur für Daten zuständig

MVC (OBSERVER)



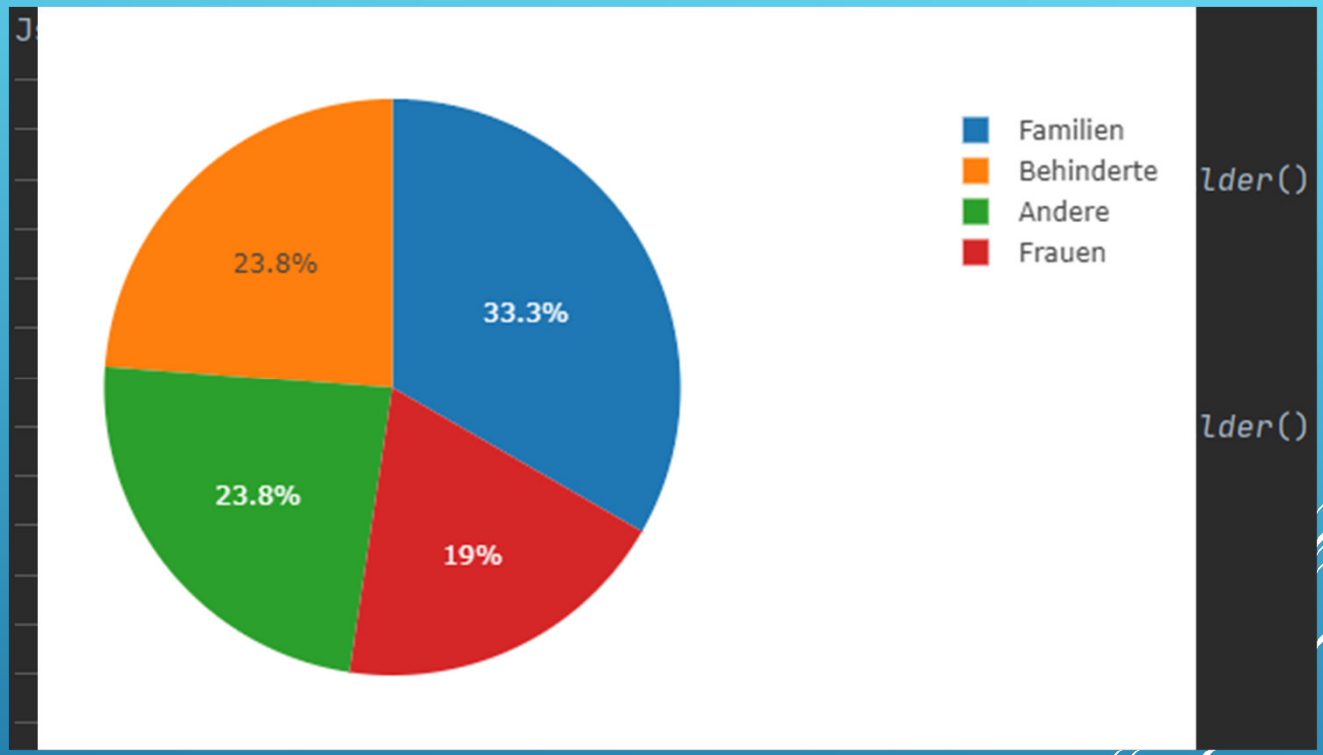
STRATEGY

- Semidynamisch
  - Kundenobjekte im Konstruktor erzeugt
  - Anschließend keine Änderungen



COMPOSITE

- JSON-Objekte
- Enthalten Daten für Graphen



SINGLETON & ITERATOR

- Eager Singleton
- Inklusive Iteratoren
  - Parkhaus.autoLeave()
  - ParkhausServlet.updater()

```
private final static Parkhaus singletonParkhaus = new Parkhaus();

private Parkhaus(){
    autosAusgefahren = new ArrayList<>();
    autosEingefahren = new ArrayList<>();
    kundentyp = new int[3];    // Abonennt, Firmenkunde, normal
    menschenart = new int[4];  // Frauen, Behinderte, Familie, andere
}

public static Parkhaus getParkhaus(){
    return singletonParkhaus;
}
```

```
for (Auto a : autosEingefahren){
    if (Integer.parseInt(altesAuto[1]) == a.getNummerschild() && altesAuto[6].equals(a.getFarbcode())){
        carLeaving = a;
    }
}
```

```
@Override
public void updater(){
    for (ViewIF e : views)
        e.update();
    tabelle.update();
}
```

STREAMS

- Parallele Streams

- Filter
- Map
- Reduce

```
float einnahmeAbonnenten = autos.parallelStream()
        .filter(Auto -> Auto.getKundentyp().equals("Abonnent"))
        .mapToInt(Auto -> Auto.getKosten())
        .reduce( identity: 0, (sum, Auto) -> sum + Auto)/100.0f;

float einnahmeFirmen = autos.parallelStream()
        .filter(Auto -> Auto.getKundentyp().equals("Firmenkunde"))
        .mapToInt(Auto -> Auto.getKosten())
        .reduce( identity: 0, (sum, Auto) -> sum + Auto)/100.0f;

float einnahmeNormal = autos.parallelStream()
        .filter(Auto -> Auto.getKundentyp().equals("NormalerKunde"))
        .mapToInt(Auto -> Auto.getKosten())
        .reduce( identity: 0, (sum, Auto) -> sum + Auto)/100.0f;
```

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT