



*Vorstellung des Semesterprojekts als
eine Corona-Ersatzprüfung für das Fach
Software Engineering I,
Sommersemester 2020*

Das beste digitale Parkhaus

*von Shada Maayouf
Matrikel nr.: 9030878*

*In Zusammenarbeit mit Max Höfer und Dennis Diegel
Fachbereich Informatik, Hochschule Bonn-Rhein-Sieg*

Inhalt der Präsentation



- Allgemeines: Vorstellung der App
- Die User Stories: Kleine Ziele stecken
- Kanban: Unsere Planung und Arbeit
- Der Quell-Code:
 - Klassen.
 - Angewendete Design Patterns.
 - Funktionale Programmierung.
- Demo



01.10.2020

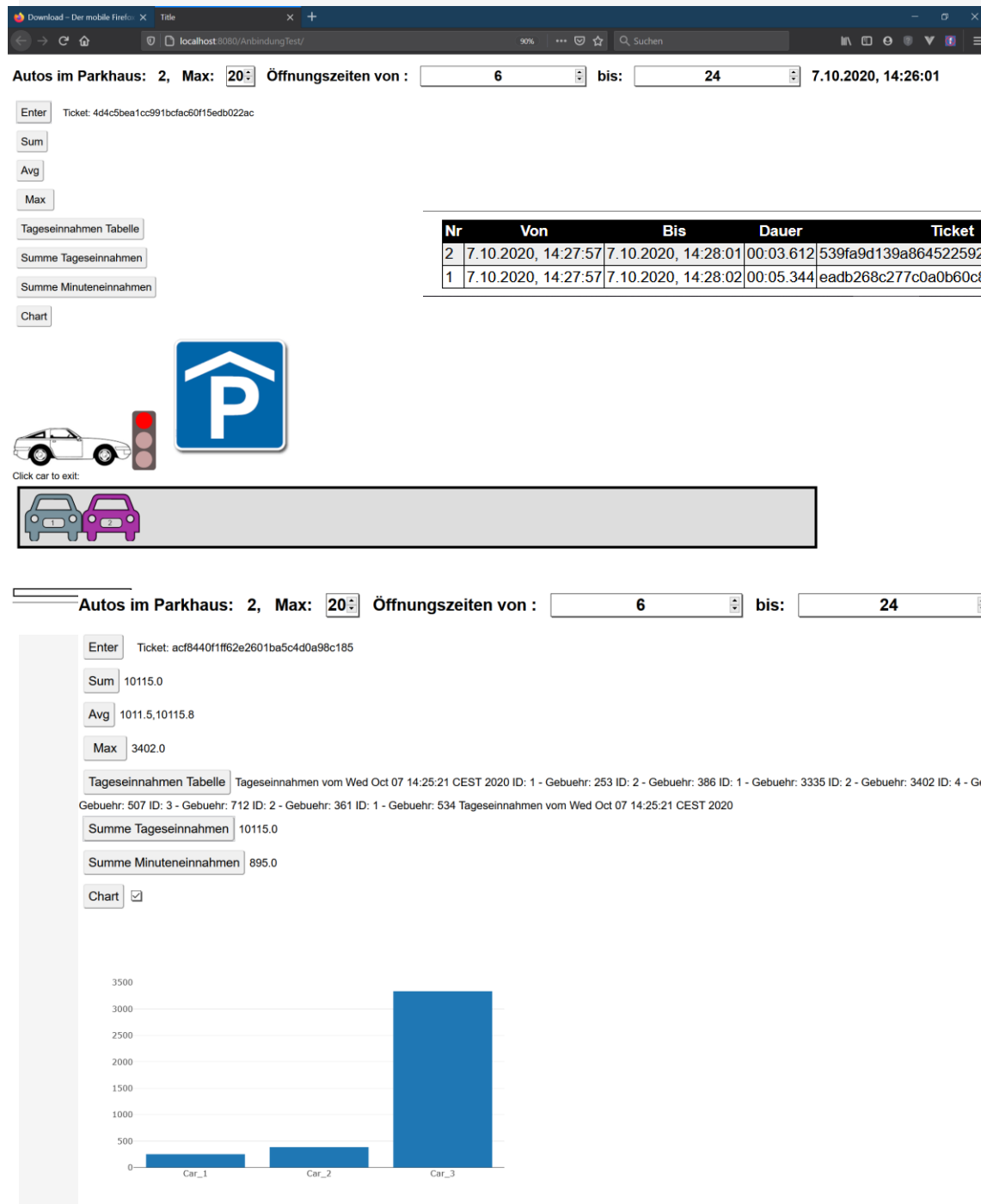
Shada Maayouf, 9030878

Vorstellung der App

- **Verwendete Version: parkhaus-6-o-2**
- **Hinzugefügte Funktionen:**
 - Summe und Mittelwert aller Einnahmen.
 - Den höchsten Einnahmewert.
 - Eine Tabelle der Tageseinnahmen.
 - Summe der Tages- sowie Minuteneinnahmen.
 - Ein Diagramm zur Abbildung.
- **Verwendete Version: parkhaus-6-o-2.**
- **Werkzeuge zur Realisierung:**
 - Java Servlets und Java Server Pages (JSP).
 - GitLab.
 - Junit.
 - Apache TomCat Server.

01.10.2020

Shada Maayouf, 9030878



The screenshot displays a web application for parking management. At the top, it shows the current status: "Autos im Parkhaus: 2, Max: 20, Öffnungszeiten von: 6 bis: 24, 7.10.2020, 14:26:01". Below this, there are input fields for "Enter", "Sum", "Avg", and "Max", each with a corresponding ticket ID. A table titled "Tageseinnahmen Tabelle" shows the sum of daily and minute revenues. A "Chart" button is also present.

The main table displays the following data:

Nr	Von	Bis	Dauer	Ticket	Preis
2	7.10.2020, 14:27:57	7.10.2020, 14:28:01	00:03.612	539fa9d139a8645225925ac4c06a4465	€ 3.61
1	7.10.2020, 14:27:57	7.10.2020, 14:28:02	00:05.344	eadb268c277c0a0b60c819fea8fd4ea5	€ 5.34

Below the table, there is a visual representation of the parking lot with a car icon and a "Click car to exit:" button. A bar chart at the bottom shows the revenue for three cars: Car_1 (approx. 300), Car_2 (approx. 400), and Car_3 (approx. 3400).

Die User Stories

- **Struktur:**
 - Insgesamt 30, aber nur 5 wurden implementiert.
 - Schätzung des Aufwands in Fibonacci Zahlen.
 - Auf die Qualitätskriterien von Bill Wake: INVEST geprüft.

01.10.2020

Shada Maayouf, 9030878

User Story	Autor	Story Points	Priorität
Betreiber (Parkhausbesitzer)			
USo1: Als Betreiber des Parkhauses benötige ich Listen und Summen über die eingenommenen Gebühren, um eine Übersicht zu bekommen.	Shada und Max	2	1
USo2: Als Parkhausbesitzer möchte ich Zahlungsvorgänge aufzeichnen, um meine Finanzen überblicken zu können.	Dennis (altes Team)	2	1
USo3: Als Parkhausbesitzer möchte ich einen Überblick über die durchschnittliche Parkdauer haben um eventuell meine Preisgestaltung anzupassen.	Dennis (altes Team)	1	1
USo4: Als Parkhausbesitzer möchte ich die Parkzeiten der Kunden verfolgen, um Dauerbelegung zu verhindern.	Dennis (altes Team)	1	1
Autofahrer (Parkhauskunde)			
USo5: Als Parkhauskunde möchte ich ein Ticket ziehen, um meine Einfahrtszeit dokumentiert zu haben und keine höhere Gebühr als gemäß meiner Nutzung bezahlen zu müssen.	Dennis (altes Team)	0	1
.....			

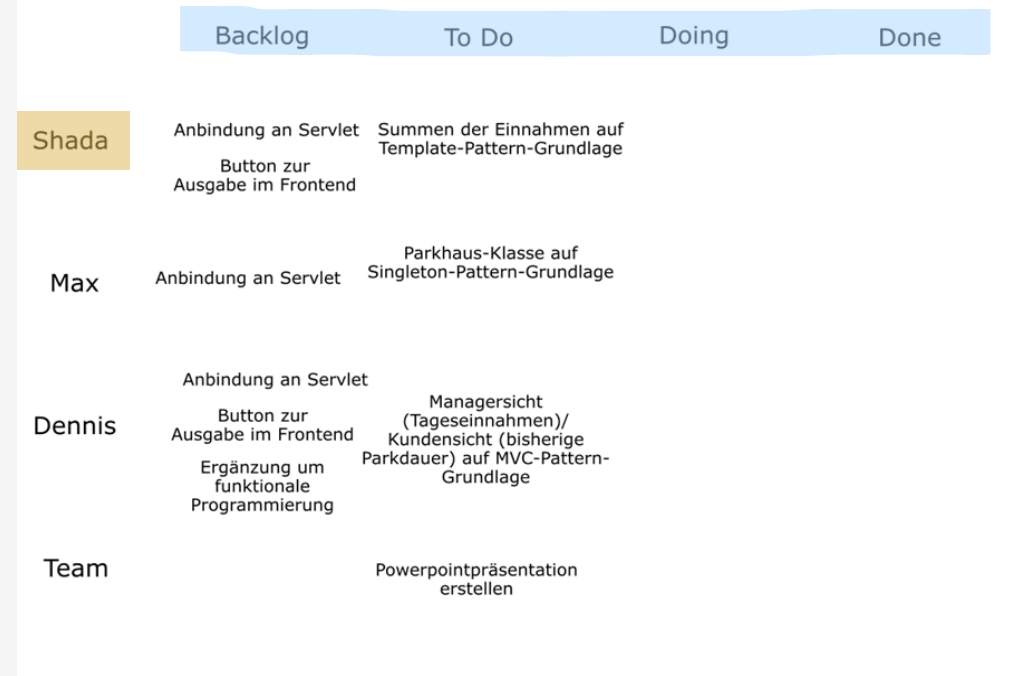
Eigenleistungen: Kanban-Board

- **5 Iterationen mit Feedbackschleifen.**
- **Verfolgung der Kanban-Prinzipien:**
 - Work in Progress.
 - Pull statt Push.
- **Meine Hauptaufgabe:** Berechnung der Kennzahlen auf Grundlage des Template-Design-Patterns.
 - Genaues Verständnis des Template Patterns: **0,5 PT**
 - Integrierung des Patterns in das Projekt: **1,5 PT**
 - Anbindung an weiteren Klassen insbesondere die Singleton Klasse: **1 PT**
 - Ausgabe über Button im Frontend: **0,5 PT**

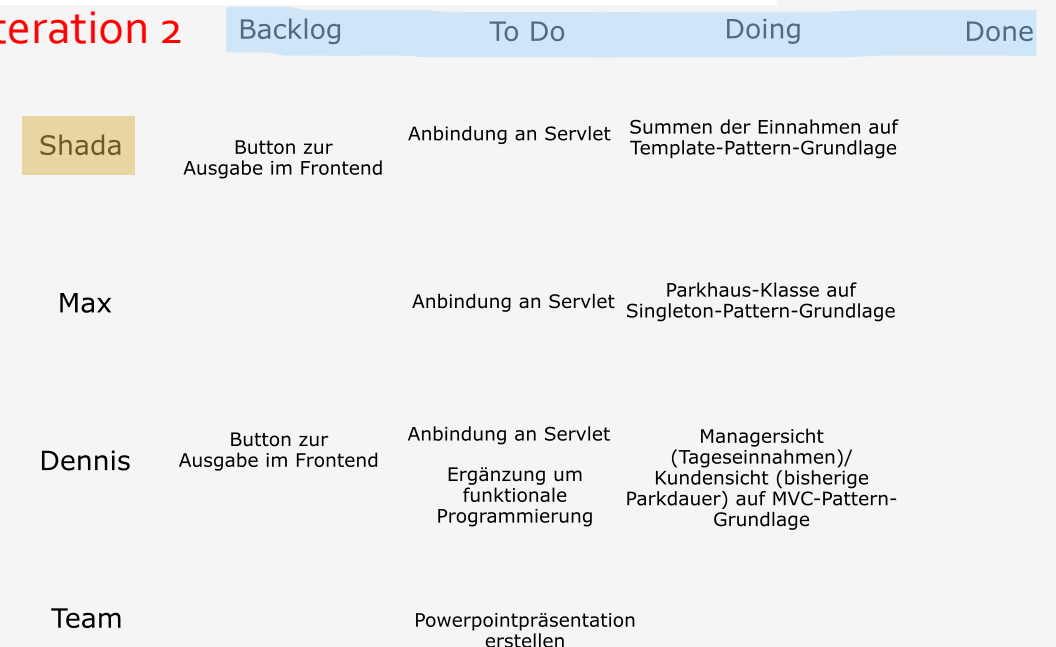
01.10.2020

Shada Maayouf, 9030878

Iteration 1



Iteration 2





Eigenleistungen: Kanban-Board

Iteration 4

	Backlog	To Do	Doing	Done
Shada			Anbindung an Servlet Button zur Ausgabe im Frontend	Summen der Einnahmen auf Template-Pattern-Grundlage
Max			Anbindung an Servlet	Parkhaus-Klasse auf Singleton-Pattern-Grundlage
Dennis			Anbindung an Servlet Button zur Ausgabe im Frontend	Managersicht (Tageseinnahmen)/ Kundensicht (bisherige Parkdauer) auf MVC-Pattern- Grundlage Ergänzung um funktionale Programmierung
Team		Powerpointpräsentation erstellen		

01.10.2020

Shada Maayouf, 9030878

Iteration 3

	Backlog	To Do	Doing	Done
Shada		Button zur Ausgabe im Frontend	Anbindung an Servlet	Summen der Einnahmen auf Template-Pattern-Grundlage
Max			Anbindung an Servlet	Parkhaus-Klasse auf Singleton-Pattern-Grundlage
Dennis		Button zur Ausgabe im Frontend	Anbindung an Servlet Ergänzung um funktionale Programmierung	Managersicht (Tageseinnahmen)/ Kundensicht (bisherige Parkdauer) auf MVC-Pattern- Grundlage
Team			Powerpointpräsentation erstellen	

Iteration 5

	Backlog	To Do	Doing	Done
Shada				Summen der Einnahmen auf Template-Pattern-Grundlage Anbindung an Servlet Button zur Ausgabe im Frontend
Max				Parkhaus-Klasse auf Singleton-Pattern-Grundlage Anbindung an Servlet
Dennis				Anbindung an Servlet Managersicht (Tageseinnahmen)/ Kundensicht (bisherige Parkdauer) auf MVC-Pattern- Grundlage Ergänzung um funktionale Programmierung
Team			Powerpointpräsentation erstellen	Button zur Ausgabe im Frontend

Singleton Pattern: Genau ein Parkhaus

• Servlets:

- Von der existiert eine Instanz zur Laufzeit.
- Lebt bis zur Beendigung des Servers.
- Requests kommunizieren direkt mit den Methoden des Servlets.

• Konsequenz:

- Alle Requests teilen die Instanzvariablen eines Servlets.

• Lösung: Request-Zugriffe synchronisieren.

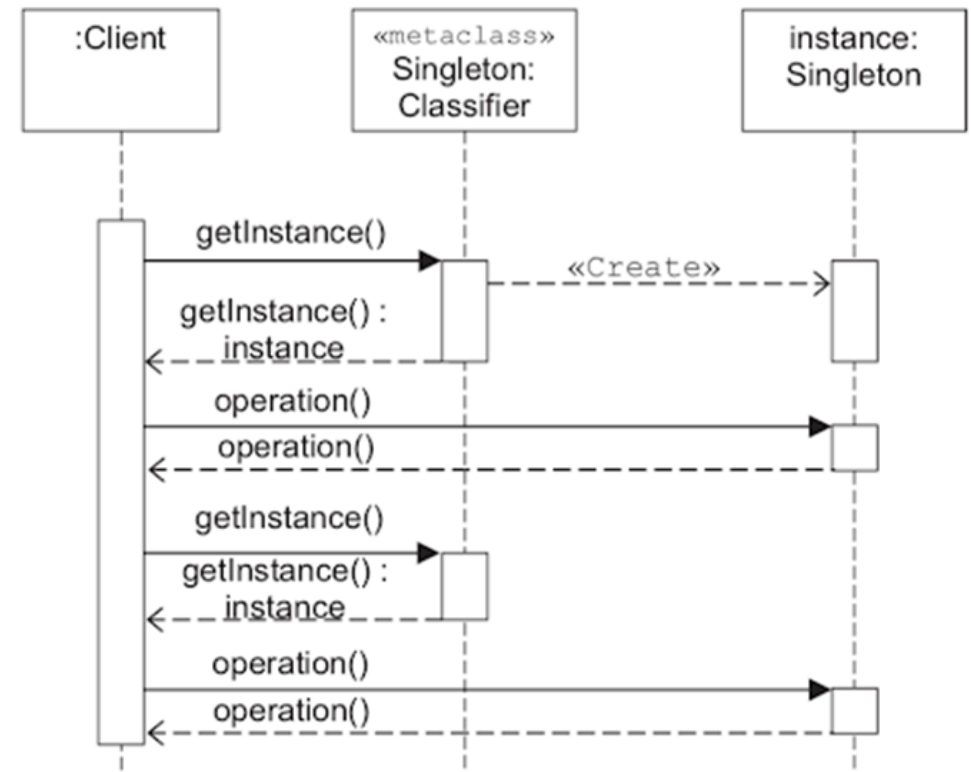
- Parkhaus-Klasse wird zur Singleton-Klasse.

01.10.2020

Shada Maayouf, 9030878



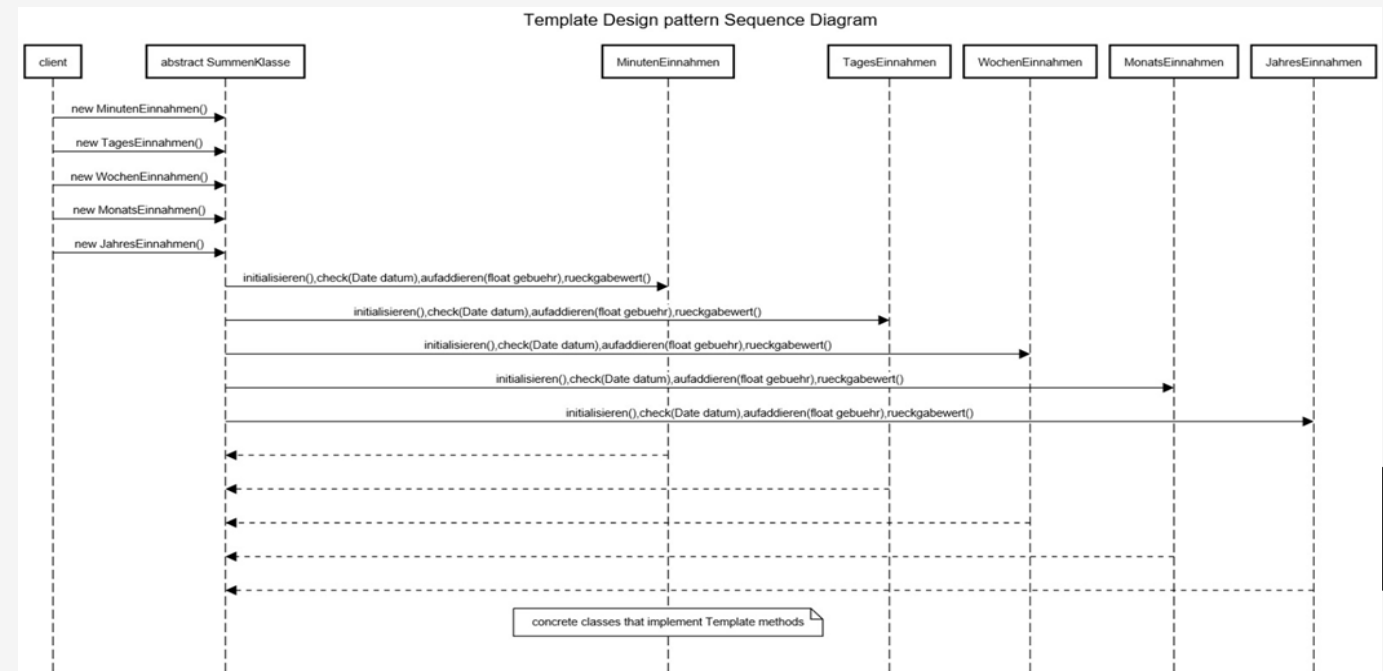
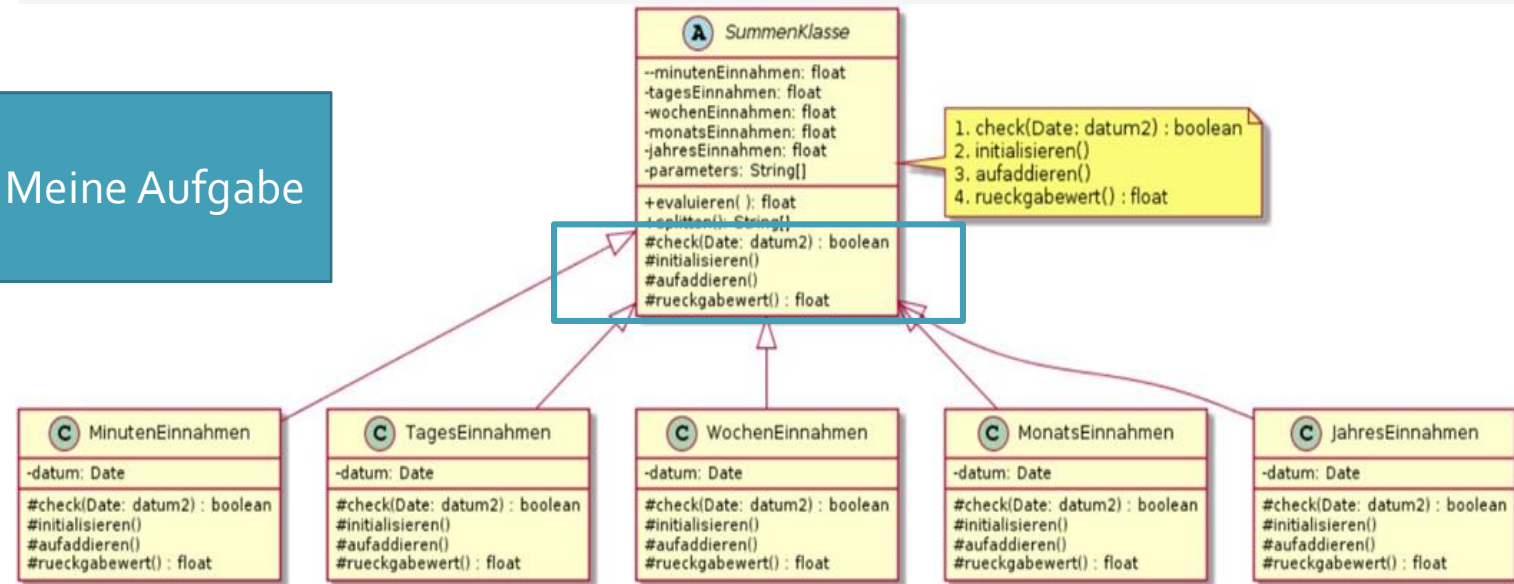
Aufgabe des
Teammitglieds
Max Höfer



Meine Aufgabe

Template Pattern: Kennzahlen

- **Anregung:**
 - Mehrere Funktionen der Anwendung basieren auf eine Aufsummierung der Kosten für die Parkgebühren.
 - Aufsummierung soll auf unterschiedliche Kriterien basieren.
- **Lösung: Template Design Pattern**
 - Umkehrung der Kontrolle (Dependency Inversion).
 - Überschreibung der einzelnen Schritte des Algorithmus ohne dessen Struktur zu ändern.



01.10.2020

Shada Maayouf, 9030878

Template Pattern: Kennzahlen

```
public abstract class SummenKlasse {

    float tagesEinnahmen, wochenEinnahmen, monatsEinnahmen,
    jahresEinnahmen, minutenEinnahmen;

    Float gebuehr;
    String[] parameters;
    Long unixStampLong;
    Long vergangeneZeit;

    public String[] splitten(String s){
        this.parameters = s.split( regex: "," );
        this.gebuehr = Float.parseFloat(parameters[4]);
        this.unixStampLong = Long.parseLong(parameters[2]);
        this.vergangeneZeit = Long.parseLong(parameters[3]);
        return parameters;
    }

    public SummenKlasse() {
    }
}
```

```
public float evaluieren()
{
    initialisieren();
    aufaddieren(this.gebuehr);
    return rueckgabewert();
}
```

//Template methods

```
public abstract boolean check(Date datum2);
public abstract void initialisieren();
public abstract void aufaddieren(float gebuehr);
public abstract float rueckgabewert();
```

```
public class TagesEinnahmen extends SummenKlasse {

    private static TagesEinnahmen singleton;
    Date datum;
    boolean neuerTag = true;

    static {
        singleton = new TagesEinnahmen();
    }

    private TagesEinnahmen() { super(); }

    public static TagesEinnahmen getInstance() { return singleton; }

    @Override
    public boolean check(Date datum2) {
        if(neuerTag){
            this.datum = datum2;
            neuerTag = false;
            return true;
        }
        Calendar c1 = Calendar.getInstance();
        Calendar c2 = Calendar.getInstance();
        c1.setTime(this.datum);
        c2.setTime(datum2);

        boolean sameDay = c1.get(Calendar.DAY_OF_YEAR) == c2.get(Calendar.DAY_OF_YEAR) &&
            c1.get(Calendar.YEAR) == c2.get(Calendar.YEAR);
        return sameDay;
    }

    @Override
    public void initialisieren() { super.tagesEinnahmen = 0; }

    @Override
    public void aufaddieren(float gebuehr) { super.tagesEinnahmen += gebuehr; }

    @Override
    public float rueckgabewert() { return this.tagesEinnahmen; }
}
```

Template Methode

```
public class MinutenEinnahmen extends SummenKlasse {
    Date datum;
    boolean neueMinute = true;

    private static MinutenEinnahmen singleton;

    static {
        singleton = new MinutenEinnahmen();
    }

    private MinutenEinnahmen() { super(); }

    public static MinutenEinnahmen getInstance() { return singleton; }

    @Override
    public boolean check(Date datum2) {
        // TODO Auto-generated method stub
        if(neueMinute){
            this.datum = datum2;
            neueMinute = false;
            return true;
        }
        Calendar c1 = Calendar.getInstance();
        Calendar c2 = Calendar.getInstance();
        c1.setTime(this.datum);
        c2.setTime(datum2);

        boolean sameMinute = c1.get(Calendar.MINUTE) == c2.get(Calendar.MINUTE) &&
            c1.get(Calendar.HOUR_OF_DAY) == c2.get(Calendar.HOUR_OF_DAY) &&
            c1.get(Calendar.DAY_OF_YEAR) == c2.get(Calendar.DAY_OF_YEAR) &&
            c1.get(Calendar.YEAR) == c2.get(Calendar.YEAR);
        System.out.println(c1.getTime()+" c1 "+c2.getTime()+" c2");
        //System.out.println(c1.get(Calendar.MINUTE)+"c1 minute "+c2.get(Calendar.
        return sameMinute;
    }

    @Override
    public void initialisieren() { super.minutenEinnahmen = 0.0f; }

    @Override
    public void aufaddieren(float gebuehr) { super.minutenEinnahmen += gebuehr; }

    @Override
    public float rueckgabewert() { return super.minutenEinnahmen; }
}
```

.10.2020

9030878

MVC-Pattern: persönliche Ansichten

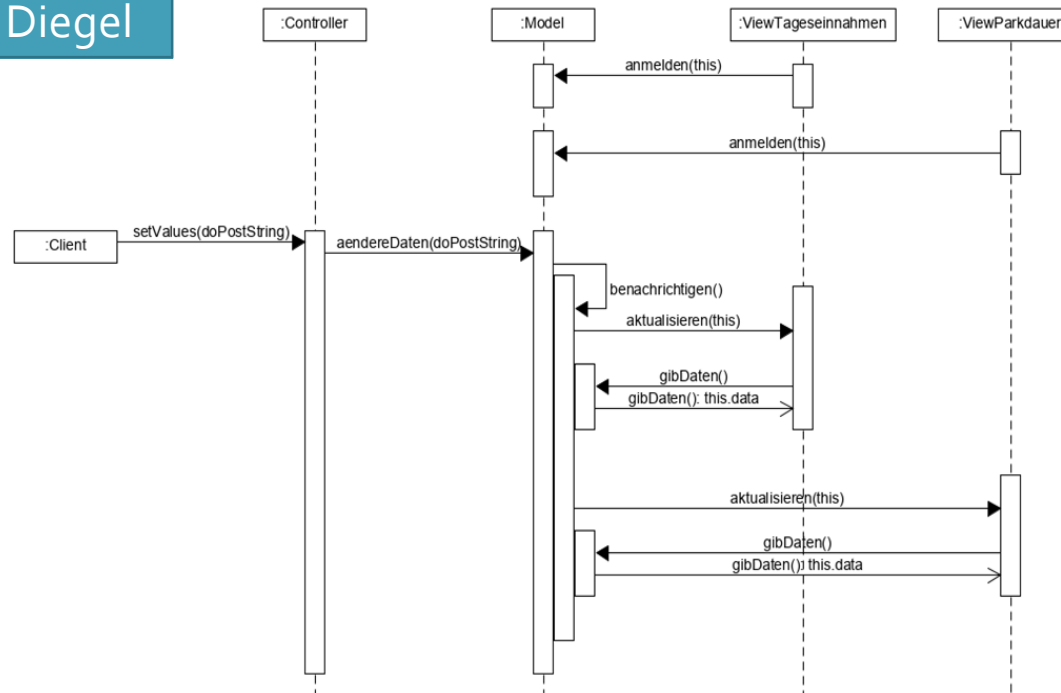
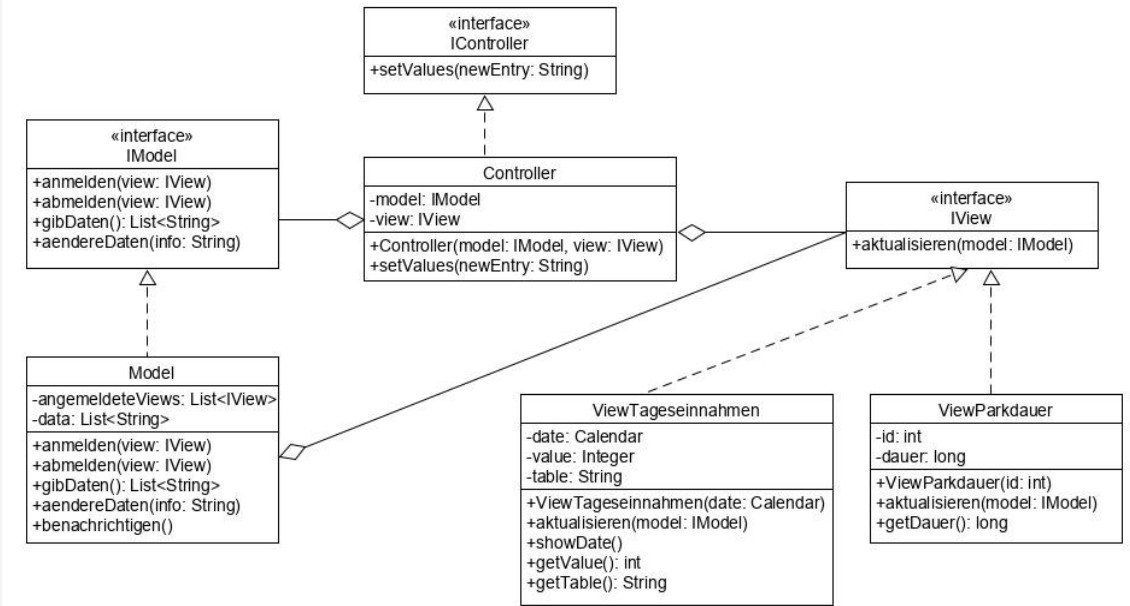
Aufgabe des
Teammitglieds
Dennis Diegel

Idee:

- Trennung von Anzeige (View), Daten (Model) und der Steuerung (Controller).
- Servlet (Controller) bearbeitet Requests.
- Servlet übergibt an eine Server Page (View).
- Mehrere Serverpages, ermöglichen mehrere personalisierte Anzeigen (Views).

01.10.2020

Shada Maayouf, 9030878



MVC-Pattern: funktionale Programmierung

- **Merkmale:**
 - Message Chaining.
 - Zustandslosigkeit.
- **Vorteile:**
 - Erlaubt Parallelisierung.
 - Laufzeit-Optimierung.
 - Zustandslosigkeit.
 - Kurzerer Code.

01.10.2020

Shada Maayouf, 9030878

Funktionale Programmierung

```
public void aktualisieren(IModel model) {
    List<String> newData = model.gibDaten();
    this.value = 0;
    this.table = "";
    this.table += "Tageseinnahmen vom " + this.date.getTime() + " <START DER LISTE>\n";
    this.table += newData.stream().stream()
        .map(string -> string.split(regex: ",")) //Aufteilen des Strings in Teilstrings
        .filter(array -> !" ".equals(array[4])) //filtern, ob Preis dabeisteht
        .filter(array -> {Calendar fromStamp = Calendar.getInstance(); //filtern, ob Daten vom gewünschten Tag
            fromStamp.setTimeInMillis(Long.parseLong(array[2]));
            return (this.date.get(Calendar.DAY_OF_YEAR) == fromStamp.get(Calendar.DAY_OF_YEAR)
                &&
                this.date.get(Calendar.YEAR) == fromStamp.get(Calendar.YEAR));}) Stream<String[]>
        .map(array -> "ID: " + array[1] + " - " + "Gebuehr: " + array[4] + "\n") Stream<String>
        .reduce(identity: "", String::concat);
    this.table += "Tageseinnahmen vom " + this.date.getTime() + " <ENDE DER LISTE>\n";

    System.out.println(this.table);

    this.value += newData.stream().stream()
        .map(string -> string.split(regex: ",")) //Aufteilen des Strings in Teilstrings
        .filter(array -> !" ".equals(array[4])) //filtern, ob Preis dabeisteht
        .filter(array -> {Calendar fromStamp = Calendar.getInstance(); //filtern, ob Daten vom gewünschten Tag
            fromStamp.setTimeInMillis(Long.parseLong(array[2]));
            return (this.date.get(Calendar.DAY_OF_YEAR) == fromStamp.get(Calendar.DAY_OF_YEAR)
                &&
                this.date.get(Calendar.YEAR) == fromStamp.get(Calendar.YEAR));}) Stream<String[]>
        .map(array -> Integer.parseInt(array[4])) Stream<Integer>
        .reduce(identity: 0, (x,y) -> x+y);
}
```

imperative Programmierung

```
@Override
public void aktualisieren(IModel model) {
    this.value = 0;
    this.table = "";
    this.table += "Tageseinnahmen vom " + this.date.getTime() + " <START DER LISTE>\n";
    System.out.println("Tageseinnahmen vom " + this.date.getTime() + " <START DER LISTE>");
    List<String> newData = model.gibDaten();
    for(String currentEntry : newData) {
        String[] params = currentEntry.split(regex: ",");
        String idString = params[1];
        Integer idInt = Integer.parseInt(idString);
        String priceString = params[4];
        if(!" ".equals(priceString)) {
            String unixStampString = params[2];
            Long unixStampLong = Long.parseLong(unixStampString);
            Calendar fromStamp = Calendar.getInstance();
            fromStamp.setTimeInMillis(unixStampLong);
            //System.out.println("fromEntry: " + fromStamp.getTime());
            if(
                this.date.get(Calendar.DAY_OF_YEAR) == fromStamp.get(Calendar.DAY_OF_YEAR) &&
                this.date.get(Calendar.YEAR) == fromStamp.get(Calendar.YEAR) ) {
                value += Integer.parseInt(priceString);
                this.table += "ID: " + idInt + " - " + "Gebuehr: " + Integer.parseInt(priceString) + "\n";
                System.out.println("ID: " + idInt + " - " + "Gebuehr: " + Integer.parseInt(priceString));
            }
        }
    }
    //this.showDate();
    this.table += "Tageseinnahmen vom " + this.date.getTime() + " <ENDE DER LISTE>\n";
    System.out.println("Tageseinnahmen vom " + this.date.getTime() + " <ENDE DER LISTE>");
}
```



Vielen Dank

01.10.2020

Shada Maayouf, 9030878