

# Übung 08: OOP

Abgabetermin: 15. 12. 2009

Name: \_\_\_\_\_

Matrikelnummer: \_\_\_\_\_

Gruppe: G1 (Prähofer)

G2 (Wolfinger)

G3 (Wolfinger)

G4 (Jahn)

| Aufgabe      | Punkte | gelöst                   | abzugeben schriftlich           | abzugeben elektronisch | Korr.                    | Pkte |
|--------------|--------|--------------------------|---------------------------------|------------------------|--------------------------|------|
| Aufgabe 08.1 | 12     | <input type="checkbox"/> | Java-Programm<br>Testergebnisse | Java-Programm          | <input type="checkbox"/> |      |
| Aufgabe 08.2 | 12     | <input type="checkbox"/> | Java-Programm<br>Testergebnisse | Java-Programm          | <input type="checkbox"/> |      |

a) Ich habe meine eigene Ausarbeitung von Aufgabe 07.2 verwendet.

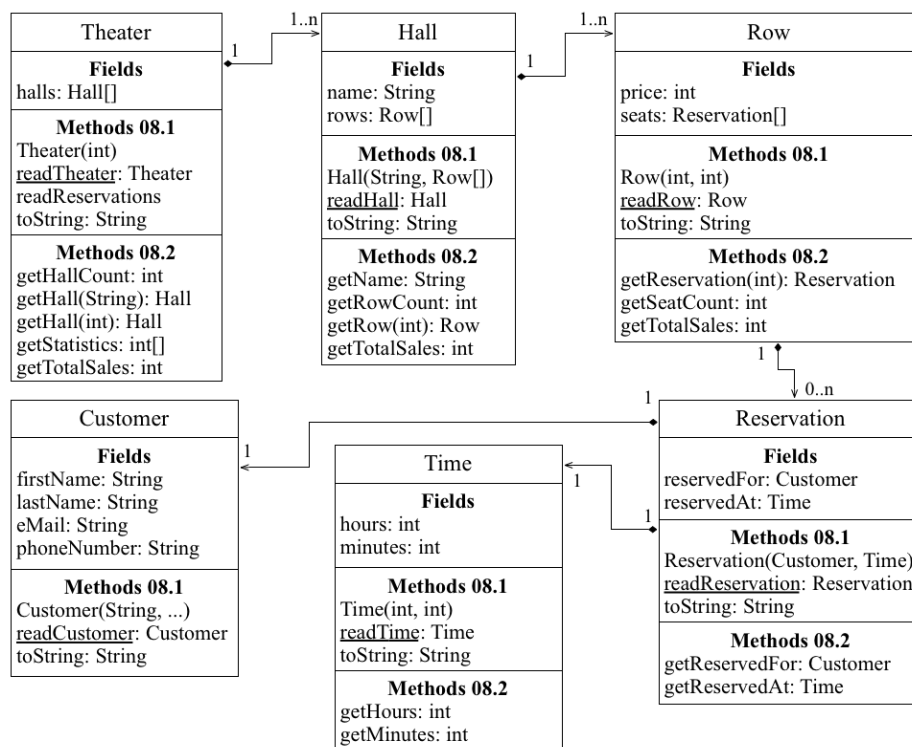
b) Ich habe die Musterlösung von Aufgabe 07.2 verwendet.

## Aufgabe 08.1: Reservieren von Kinoplätzen (Restrukturierung OOP)

Ihr Auftraggeber Peter Filmsky stellt bei der Abnahme verärgert fest, dass Ihr Reservierungs-Programm aus Aufgabe 07.2 nicht objektorientiert ist und fordert Sie auf nachzubessern. Nehmen Sie Ihre Lösung aus Übung 7 und machen Sie für 12 Übungspunkte daraus ein objektorientiertes Programm. Das Programm soll nach der Überarbeitung die gleichen Funktionen haben, aber objektorientiert strukturiert sein. Verteilen Sie die Klassenmethoden Ihrer *Theater*-Klasse in Aufgabe 07.2, als Objektmethoden in die Klassen *Theater*, *Hall*, *Row*, *Reservation*, *Time* und *Customer* machen. Implementieren Sie für jede Klasse:

- einen Konstruktor, zB initialisiert *Time(int hours, int minutes)* ein Objekt der Klasse *Time*
- eine statische *read*-Methode, die mit Hilfe der *In*-Klasse ein Objekt einliest; zB liest *readTime* ein *Time*-Objekt
- eine *toString*-Methode, die den Inhalt eines Objekts als Text liefert; zB liefert *toString* der Klasse *Time* eine Zeichenkette im Format *hh:mm* für ein *Time*-Objekt

Das Klassendiagramm zeigt die bereits aus Aufgabe 07.2 bekannten Felder und die Methoden die Sie in Aufgabe 08.1 implementieren sollen. Führen Sie bei Bedarf weitere Methoden ein.



Erstellen Sie für Ihr neues Hauptprogramm eine Klasse *TheaterManager* und implementieren Sie dort die *main*-Methode wie unten gezeigt. Benutzerschnittstelle und Anwendungslogik sollen getrennt sein. Machen Sie Konsolenausgaben nur in der Klasse *TheaterManager*. Erstellen Sie die auszugebenden Zeichenketten mit Hilfe der *toString*-Methoden. Die *toString*-Methode für ein *Theater* ruft *toString* für die *Hall*-Objekte auf, die *toString*-Methode für eine *Hall* ruft *toString* für die *Row*-Objekte auf.

```
public class TheaterManager {
    public static void main(String[] args) {
        In.open("halls.txt");
        Theater theater = Theater.readTheater();
        In.close();

        In.open("reservations.txt");
        theater.readReservations();
        In.close();

        Out.println(theater.toString());
    }
}
```

Peter Filmsky erwartet von Ihnen: Java-Programm, Testergebnisse

## Aufgabe 08.2: Kinoverwaltung

Filmsky ist begeistert. Er möchte das objektorientierte Programm an den Abendkassen seiner Filmsky-Kinos einzusetzen. Für weitere 12 Übungspunkte müssen Sie aber noch eine interaktive Benutzerschnittstelle programmieren. Das folgende Hauptmenu zeigt die Operationen. Das Hauptmenu soll nach jeder Operation wieder angezeigt werden:

```
***** Kinoverwaltung *****
>> aktiver Saal: Saal1
Saal wechseln ..... w
Saalbelegung anzeigen ..... a
Eintrittskarte drucken ..... t
Tagesumsaetze anzeigen ..... u
Reservierungsstatistik anzeigen.... s
Beenden ..... q
Welche Menuoption ?[a|w|t|u|s|q]:
```

Das Hauptmenu zeigt an welcher Saal aktiv ist. Die Funktion *Saal wechseln* zeigt eine numerierte Liste der Säle. Durch Eingabe der Nummer wechselt man zum gewünschten Saal. Das Programm soll Fehleingaben abfangen und solange fragen bis ein gültige Nummer eingegeben wird.

```
Welche Menuoption ?[a|w|t|u|s|q]: w
1: Saal1
2: Saal2
3: Saal3
Welcher Saal?[1|2|3]: 2

***** Kinoverwaltung *****
>> aktiver Saal: Saal2
...
```

Die Funktion *Saalbelegung* zeigt die bereits aus Aufgabe 07.2 bekannte Darstellung eines Saales. Für freie Plätze wird der Preis angezeigt, für reservierte Plätze wird \*\* angezeigt.

```
Welche Menuoption ?[a|w|t|u|s|q]: a

Saal2
-----01---02---03---04---05---06---07---08---09---10---11---12--
Reihe 1 | ** | ** | 6 | ** | 6 | 6 | 6 | 6 | 6 | 6 |
Reihe 2 | ** | 7 | 7 | ** | ** | 7 | 7 | 7 | 7 | 7 |
Reihe 3 | ** | 8 | 8 | ** | ** | ** | ** | ** | 8 | 8 | 8 | 8 |
Reihe 4 | ** | ** | ** | 8 | 8 | 8 | 8 | ** | 8 | 8 |
Reihe 5 | ** | ** | ** | 9 | 9 | ** | 9 | ** | ** | 9 | 9 | 9 |
```

Die Funktion *Eintrittskarte* gibt eine Eintrittskarte aus. Für jede Eintrittskarte wird Saal, Preis, Reihe und Sitz angezeigt. Für Eintrittskarten die reserviert sind, wird zusätzlich der Name des Kunden angezeigt.

```
Welche Menuoption ?[a|w|t|u|s|q]: t
Reihe: 3
Sitz: 4
-----
```

```
Saal2
Reihe: 5 Sitz: 9
reserviert fuer: Andrea Eckert
-----
```

Die Funktion *Tagesumsätze* zeigt die Reservierungsumsätze: für jede Sitzreihe des aktiven Saals, den Gesamtumsatz für den aktiven Saal und den Gesamtumsatz für alle Säle des Kinos.

```
Welche Menuoption ?[a|w|t|u|s|q]: u
```

```
Reihe 1:      EUR   18
Reihe 2:      EUR   21
Reihe 3:      EUR   48
Reihe 4:      EUR   32
Reihe 5:      EUR   54
-----
TOTAL (Hall)  EUR  173
TOTAL (Theater) EUR  577
```

Die Funktion *Reservierungsstatistik* errechnet ein Histogramm für die Reservierungen. Das Histogramm zeigt für jede Stunde des Tages die Anzahl der eingegangenen Reservierungen als Zahl und Säule. In der Säule steht ein \* für eine Reservierung.

```
Welche Menuoption ?[a|w|t|u|s|q]: s
```

```
00-01:      0
01-02:      0
02-03:      0
03-04:      0
04-05:      0
05-06:      0
06-07:      0
07-08:      0
08-09:      1 *
09-10:      1 *
10-11:      3 ***
11-12:     16 *****
12-13:      0
13-14:      3 ***
14-15:     19 *****
15-16:      2 **
16-17:     25 *****
17-18:      3 ***
18-19:      0
19-20:      1 *
20-21:      1 *
21-22:      0
22-23:      0
23-24:      1 *
-----
TOTAL:      76
```

Implementieren Sie folgende Methoden in der Klasse *Theater*. Die Methoden sind im Klassendiagramm mit *Methods 08.2* markiert (siehe Seite 1):

- *getHallCount* liefert die Anzahl der Säle des Kinos
- *getHall(String)* sucht den Saal zum gegebenen Namen
- *getHall(int)* liefert einen Saal zum gegebenen Index
- *getStatistics* liefert ein Array *int[24]* mit der Anzahl der Reservierungen pro Stunde

Implementieren Sie folgende Methoden in der Klasse *Hall*:

- *getName* liefert den Namen des Saales
- *getRowCount* liefert die Anzahl der Sitzreihen des Saales
- *getRow(int)* liefert die Sitzreihe für die gegebene Reihen-Nummer
- *getTotalSales* liefert den Tagesumsatz des Saales

Implementieren Sie folgende Methoden in der Klasse *Row*:

- *getReservation(int)* liefert die Reservierung für den gegebenen Sitzplatz oder *null* wenn keine Reservierung vorliegt
- *getSeatCount* liefert die Anzahl der Sitze der Reihe
- *getTotalSales* liefert den Tagesumsatz der Reihe

Implementieren Sie folgende Methoden in der Klasse *Reservation*:

- *getReservedFor* liefert den Kunden der diesen Platz reserviert hat
- *getReserverdAt* liefert den Reservierungszeitpunkt

Implementieren Sie folgende Methoden in der Klasse *Time*:

- *getHours* liefert die Stunden der Uhrzeit
- *getMinutes* liefert die Minuten der Uhrzeit

Implementieren Sie folgende Methoden in der Klasse *Customer*:

- *getFirstName* liefert den Vornamen des Kunden
- *getLastName* liefert den Nachnamen des Kunden

Sie haben Glück, der LVA-Leiter programmiert mit. Das Hauptmenu, die Benutzereingabe und die Funktion Saalwechsel sind in der Vorgabedatei bereits implementiert! Ergänzen Sie in der Klasse *TheaterManager* die fehlenden Funktionen:

- *printHall* gibt die Saalbelegung aus
- *printTicket* gibt eine Eintrittskarte aus
- *printSales* gibt die Tagesumsätze aus
- *printStatistics* gibt die Reservierungsstatistik aus

Peter Filmsky erwartet von Ihnen: Java-Programm, Testergebnisse