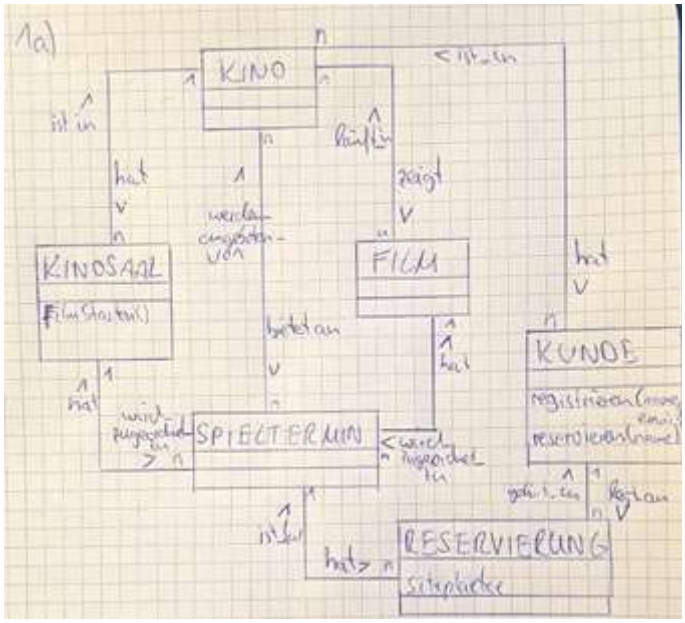
Informatik Abitur Bayern 2020 / I - Lösung

Autor:
Hermann

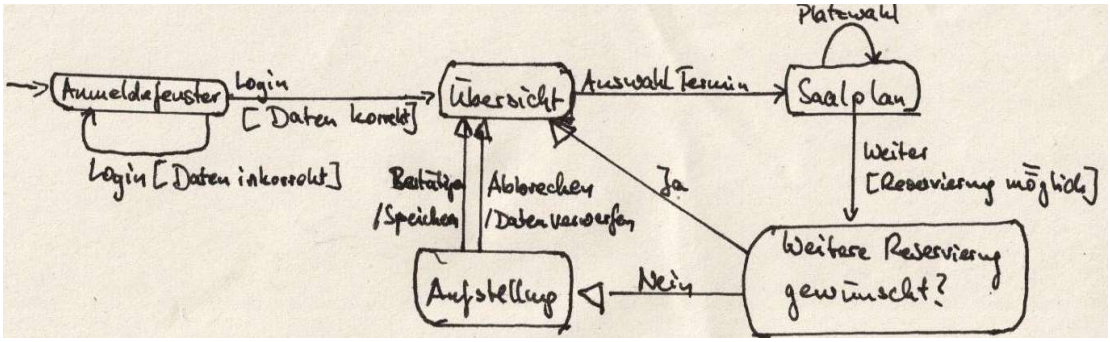
11

1a



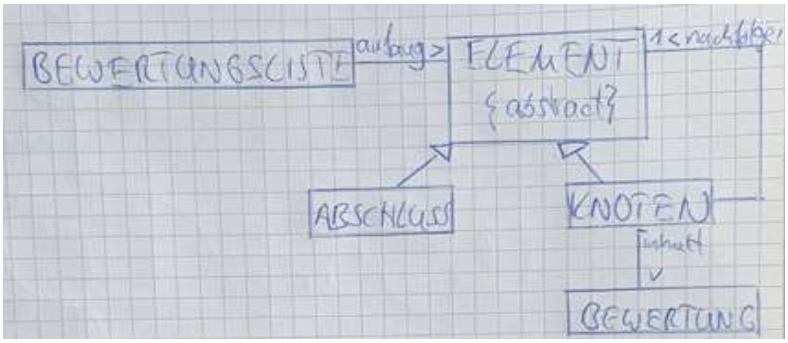
b Für die Klasse Saal ist ein Feld sinnvoll, da die Anzahl der Säle gleich bleibt. Für die Klasse Spieltermin ist eine einfach verkettete Liste sinnvoller, da sich die Anzahl der Spieltermine ändern kann und damit eine dynamische Datenstruktur nötig ist. 2

c 7



2a Das zugrunde liegende Konzept heißt Trennung von Struktur und Daten. Das hat zum einen den Vorteil, dass Struktur und Daten unabhängig voneinander ersetzt werden können. Zum anderen können Struktur und Daten leichter gewartet werden. 3

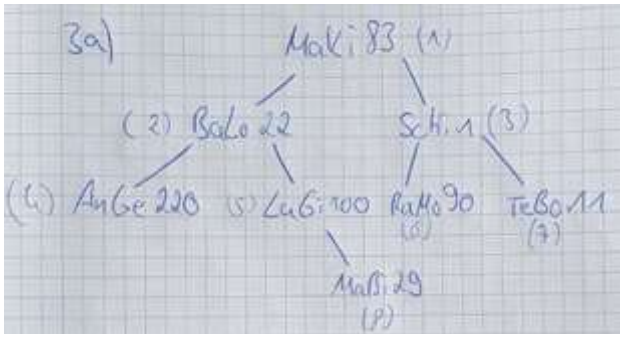
b Ein großer Vorteil dieses Entwurfsmusters ist es, dass bei der Umsetzung nicht überprüft werden muss, ob gerade der letzte Knoten vorliegt oder nicht. Ein weiterer Vorteil ist die Wiederverwendbarkeit dieser bewährten Softwarelösung. 5



c
```
public class BEWERTUNGSLISTE{

        ELEMENT anfang;

        public void einfuegen(BEWERTUNG bewertungNeu){
                        anfang = new KNOTEN(anfang, bewertungNeu);
              }

              public void ausgeben(int sterneMin, int sterneMax){
                        anfang.ausgeben(sterneMin, sterneMax);
              }

              public double durchschnittGeben(String filmtitel){
                        return anfang.summeGeben(filmtitel)/anzahlGeben(filmtitel);
              }

}

abstract class ELEMENT{
              public abstract void ausgeben(int sterneMin, int sterneMax);
              public abstract double summeGeben(String Filmtitel);
}

public class ABSCHLUSS extends ELEMENT{
              public void ausgeben(int sterneMin, int sterneMax){
              }

              public double summeGeben(String filmtitel){
                        return 0;
        }
               }

public class KNOTEN extends ELEMENT{
               ELEMENT nachfolger;
               BEWERTUNG daten;

               KNOTEN(ELEMENT element, BEWERTUNG bewertung){
                         nachfolger = element;
                         daten = bewertung;
               }

               public void ausgeben(int sterneMin, int sterneMax){
                        if(daten.anzahlSterneGeben() <= sterneMax &&
                        daten.anzahlSterneGeben() >= sterneMin){
                                          daten.ausgeben();
                         }
                         nachfolger.ausgeben(sterneMin, sterneMax);
               }

               public double summeGeben(String filmtitel){
                         if(daten.gehoertZuFilm(filmtitel)){
                         return daten.anzahlSterneGeben() +
                         nachfolger.summeGeben(filmtitel);
                                        } else{
                                               return nachfolger.summeGeben(filmtitel);
                                        }
                                }
```

20

3a



4

b Maximale Anzahl an Knoten in einem Binärbaum mit n Ebenen: $2^n - 1$ 4

$T(9{,}295) = 2^9 - 1 - 295 = 216$

$T(n,k)\colon 2^n - k - 1$

c Abhängig davon, ob die Kundennummer größer oder kleiner als die des aktuellen Knotens ist, wird die Methode einfügen auf dem rechten oder linken Nachfolger des aktuellen Knotens rekursiv ausgeführt. Ist der Abschluss erreicht, so erzeugt dieser einen neuen Knoten, um den neuen Kunden einzufügen, und gibt ihn zurück. 5

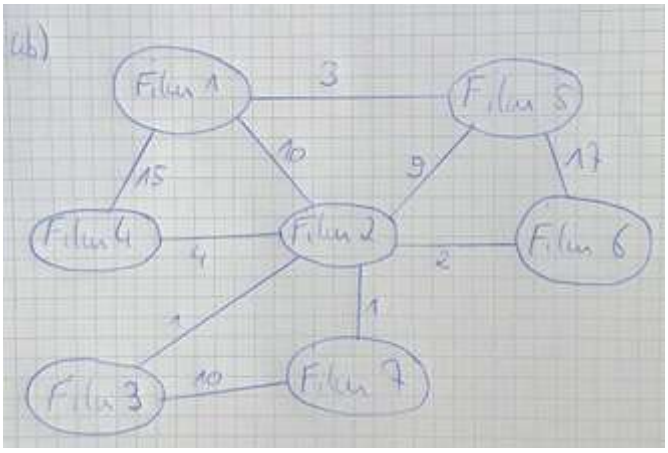
4a 5

	Film 1	Film 2	Film 3	Film 4	Film 5	Film 6	Film 7
Film 1	-1	10	0	15	0	0	0
Film 2	10	-1	0	4	6	2	0
Film 3	0	0	-1	0	0	0	9
Film 4	15	4	0	-1	0	0	0
Film 5	0	6	0	0	-1	17	0
Film 6	0	2	0	0	17	-1	0
Film 7	0	0	9	0	0	0	-1

Für die Einträge, bei denen keine Verbindung im Diagramm existiert, wurde der Wert 0 gewählt, da die Filme dann nicht kombiniert wurden. Die regulären Einträge sind dabei stets natürliche Zahlen. Für die Einträge auf der Diagonalen wurde der Wert -1 gewählt, um die Einträge von dem Rest zu unterscheiden.

Auch andere Begründungen und Werte sind insbesondere auf der Diagonalen denkbar (so werden „Kultfilme“ gerne mehrfach angeschaut.

b 4



c 8

```
public KNOTEN besteEmpfehlungSuchen(i){
                int besterWert = 0;
                int besterKnoten = 0;


        for(int j=0; j<anzahlKnoten; j++){
                        if(matrix[i][j] > besterWert){
                                        besterWert = matrix[i][j];
                                        besterKnoten = j;
                        }
        }

        if(besterWert > 0){
                        return knoten[besterKnoten];
        }
}
```

d 1. Kunde könnte Film schon gesehen haben 2

2. Kunde könnte Film i nicht gefallen haben

(3. Film ist gar nicht mehr auf dem Programm)

80