Informatik Abitur Bayern 2016 / I - Beispiellösung

Autoren:
Sadlo/Herr

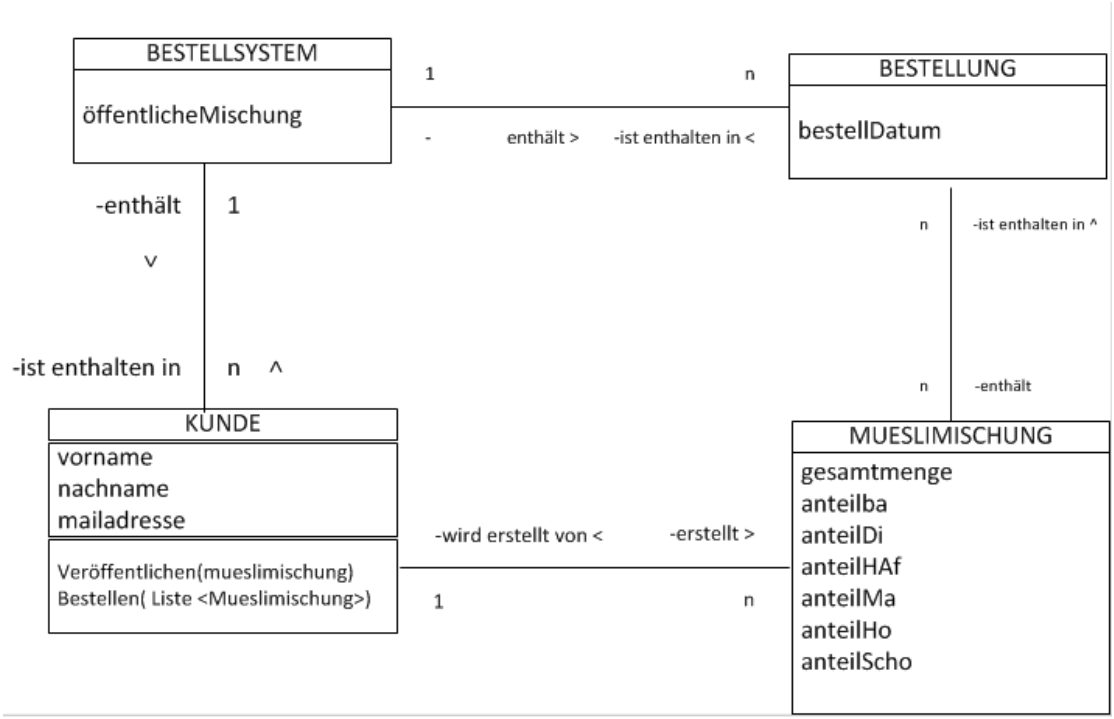
1 Phasen bei der Durchführung eines Softwareprojekts. 4

z.B.: Das Wasserfallmodell

- Anforderungsanalyse: Festlegung der Anforderungen an das System. Ergebnis ist eine Anforderungsbeschreibung → dient als "Vertrag" zwischen Anwender/Entwickler
- Entwurf: Modellierung des Systems
- Implementierung: Codierung des Entwurfs in einer Programmiersprache
- Test und Integration: Test der einzelnen Komponenten, Einbau, Systemtest
- Einsatz & Wartung: Fehlerbeseitigung nach Inbetriebnahme

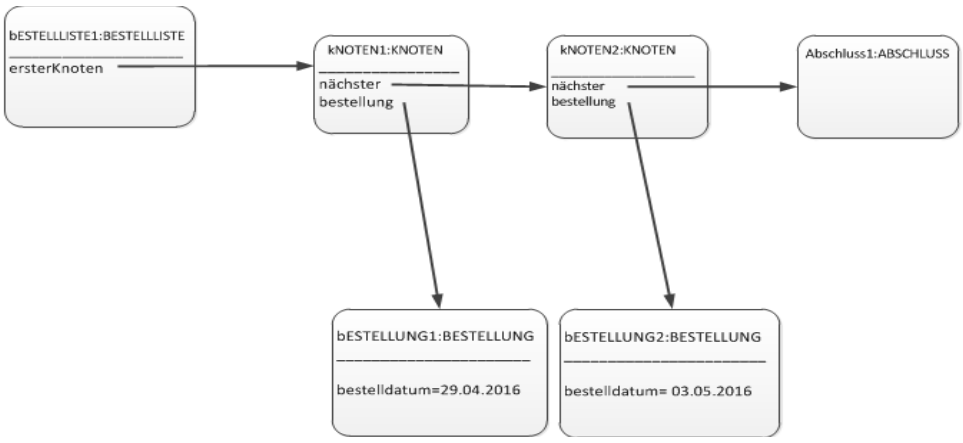
Hinweis: Da die Wartung nicht mehr zur eigentlichen Erstellung gehört, muss sie daher nicht aufgeführt sein.

2 9



3a Die Entwickler hätten auch ein Feld verwenden können,jedoch muss ein Feld von Beginn an mit einer festen großen Feldgröße initialisiert werden um alle Bestellungen aufnehmen zu können (hoher Speicherbedarf). Des weiteren entstehen Kosten für das Umkopieren der Elemente in ein größeres Feld, falls der Platz nicht reicht, oder fertige Bestellungen entfernt werden müssen. Einfach verkettete Listen besitzen eine flexible Länge und unterstützen sehr einfach die Operationen zum Einfügen und Entfernen;deshalb sind sie im Szenario einer Listedem Feld vorzuziehen. 3

3b 4



3c **Klasse BESTELLISTE:** 8

```
public int umsatzBerechnen(){
       return erster.gesamtPreisBerechnen();
}
```

Klasse LISTENELEMENT:

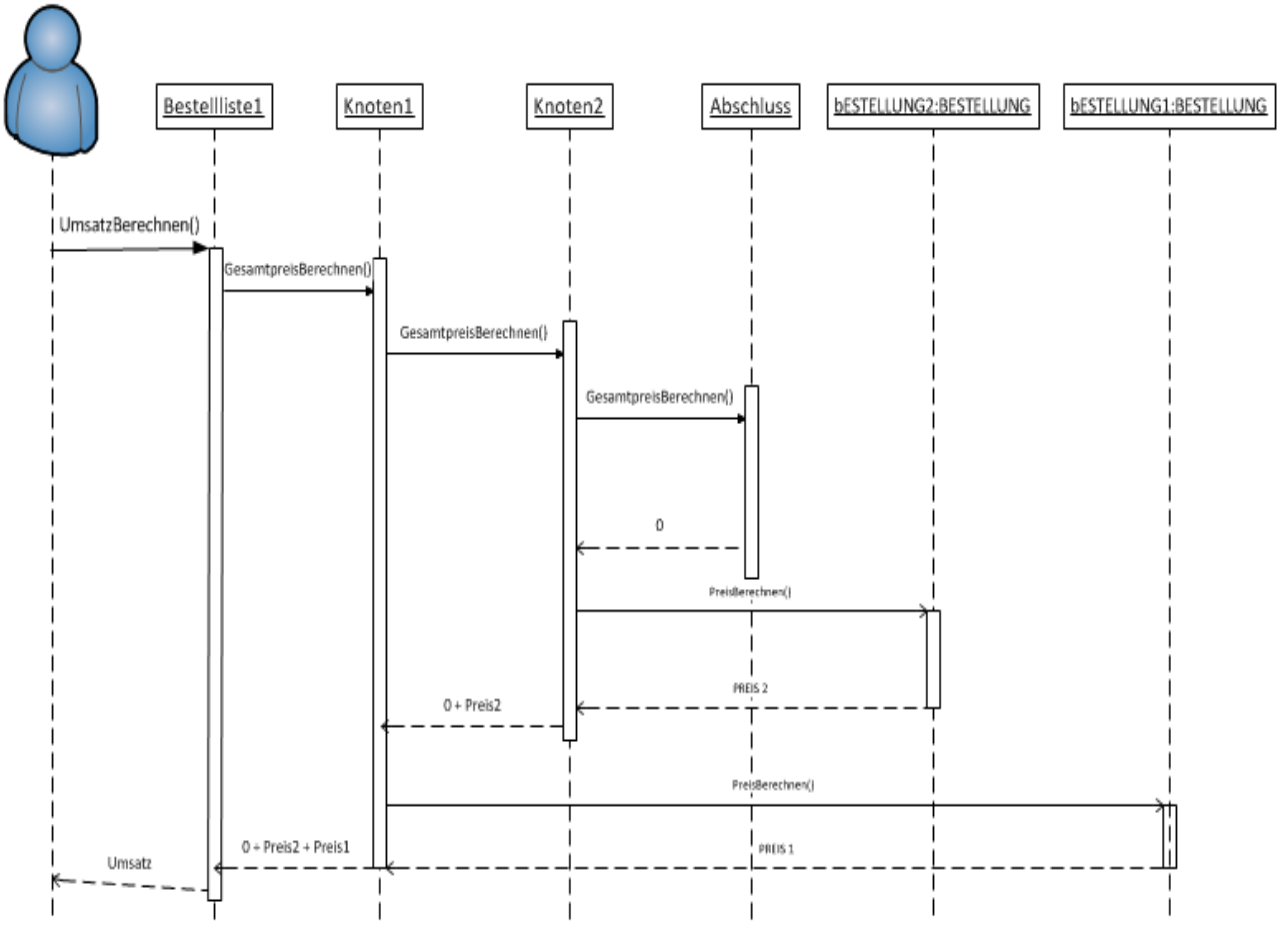
```
public abstract int gesamtPreisBerechnen();
```

Klasse KNOTEN:

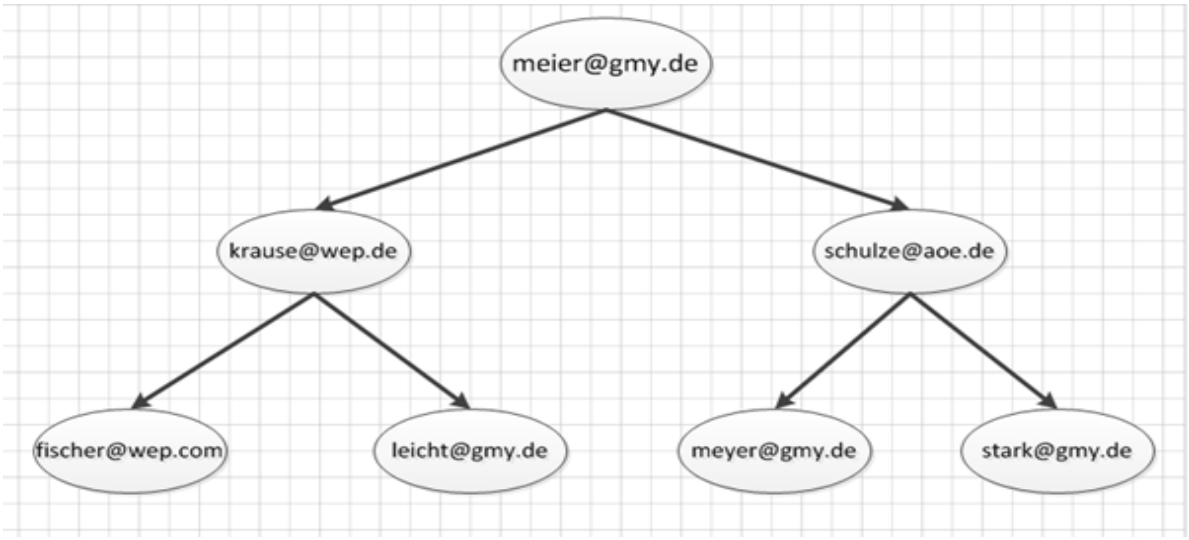
```
public int gesamtPreisBerechnen()      {
 return nächster.gesamtPreisBerechnen()+this.inhalt.preisBerechnen();
}
```

Klasse ABSCHLUSS:

```
public int gesamtPreisBerechnen( )     {
 return 0;
}
```



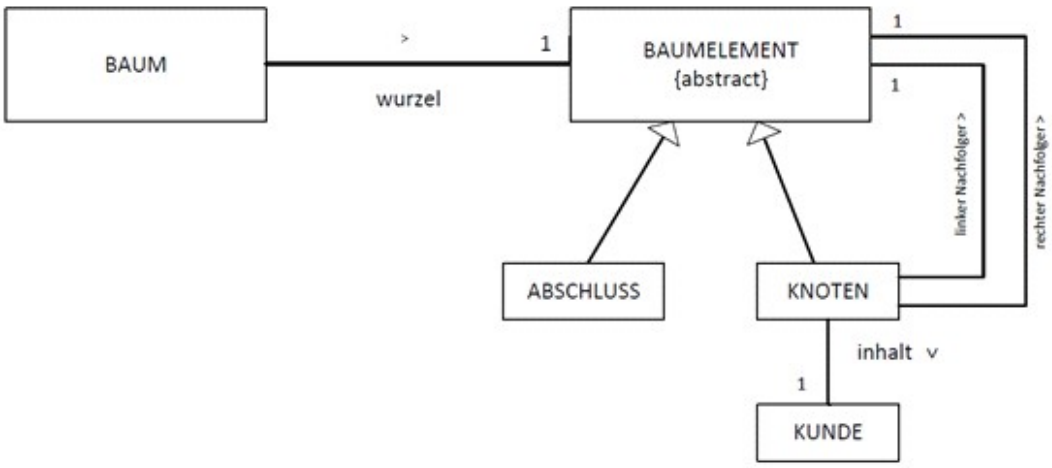
4a



Mögliche Einfügereihenfolge:
meier, krause, schulze, fischer, leicht, meyer, stark

4b Ein Baum mit n Ebenen kann $2^n - 1$ Elemente aufnehmen. Jeder Vergleich halbiert die Anzahl der möglichen Elemente. Man benötigt somit $\log_2 (10.000) = 13{,}28$ Vergleiche, also maximal 14 Vergleiche. 4

4c 8



Methoden:

Klasse BAUM:

```
public String KundenAdressenAusgeben()   {
    return wurzel. KundenAdressenAusgeben()
}
```

Klasse BAUMELEMENT:

```
public abstract String KundenAdressenAusgeben();
```

Klasse KNOTEN:

```
public String KundenAdressenAusgeben()   {
  return linkerNachbar.KundenAdressenAusgeben() +
   this.inhalt.emailAdresseAusgeben() +
   rechterNachbar.KundenAdressenAusgeben();
}
```

Klasse ABSCHLUSS:

```
public String KundenAdressenAusgeben()   {
   return "";
}
```

5a Der Graph ist: 6
- nicht zusammenhängend
- ungerichtet
(-gewichtet)

Die Adjazenzmatrix sieht wie folgt aus:

	Bananen	Dinkelflocken	Haferflocken	Mandeln	Rosinen	Schokolade
Bananen	0	0	0	0	0	0
Dinkelflocken	0	0	1	0	0	0
Haferflocken	0	1	0	3	1	2
Mandeln	0	0	3	0	1	2
Rosinen	0	0	1	1	0	0
Schokolade	0	0	2	2	0	0

5b Deklaration der Attribute, Implementierung des Konstruktors der Klasse MUESLIMATRIX 6

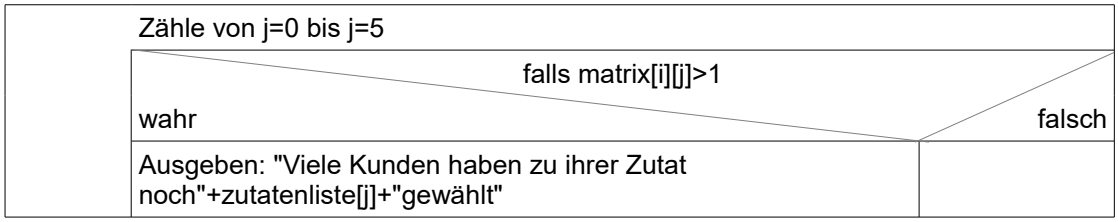
```
public class MUESLIMATRIX{
      private String[] zutatenliste;
      private int[][] matrix;

      public MUESLIMATRIX(){
            zutatenliste = new String[6];
            zutatenliste[0]= "Bananen";
            zutatenliste[1]= "Dinkelflocken";
            zutatenliste[2]= "Haferflocken";
            zutatenliste[3]= "Mandeln";
            zutatenliste[4]= "Rosinen";
            zutatenliste[5]= "Schokolade";

            int[][] matrix = new int[6][6];
            for(i=0;i<6;i++){
                  for(j=0;j<6;j++){
                        matrix[i][j]=0;
                  }
              }
      }
}
```

Hinweis: Üblicherweise wird die Adjazenzmatrix mit -1 (keine Kante) vorbelegt, Diagonale mit 0. Im speziellen Fall liegt ein vollständiger Graph vor, da prinzipiell jede Zutat mit jeder anderen kombiniert werden kann. Deshalb kann die Adjazenzmatrix in diesem Fall durchgängig mit 0 vorbelegt werden.

5c Methode *dazuEmpfehlen(i)* als Struktogramm:



5d Alle erreichbaren Knoten lassen sich z.B.mit der Tiefensuche genau einmal ausgeben. Hierfür muss die Information, ob ein Knoten bereits besucht wurde, in einem Attribut (`besucht`) gespeichert werden. Als Datentyp bietet sich ein **`boolean`**-Feld an. 8

```
public void Tiefensuche(String start){
      if(KnotennummerGeben(start)>-1){
            for(i=0;i<anzahlKnoten;i++){
                  besucht[i]=false;
            }
            Besuchen(KnotennummerGeben(start));
      }
}

public void Besuchen(int knotenNummer){
      if(!besucht[knotenNummer]){
            besucht[knotenNummer]=true;
            System.out.print(knoten[knotenNummer].BezeichnungGeben());
            for(i=0;i<anzahlKnoten;i++){
                  if(matrix[knotenNummer][i]>0){
                        Besuchen(i);
                  }
            }
      }
}
```

5e 3

```
SELECT bezeichner
FROM zutaten
WHERE menge<verbrauch;
```

5f Zunächst einmal muss die Verbindung zur Datenbank hergestellt werden (1). Anschließend muss die Datenbankabfrage (Vgl.5e) übermittelt werden (2). Wenn die Ergebnistabelle ausgewertet wurde (3), wird die Verbindung wieder geschlossen (4). 2