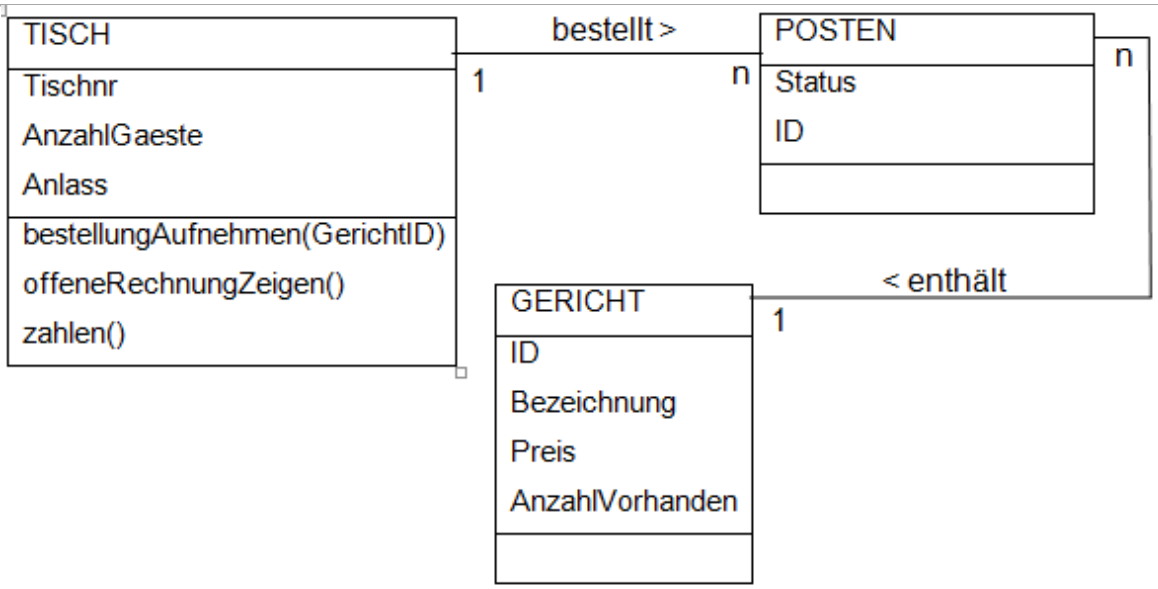
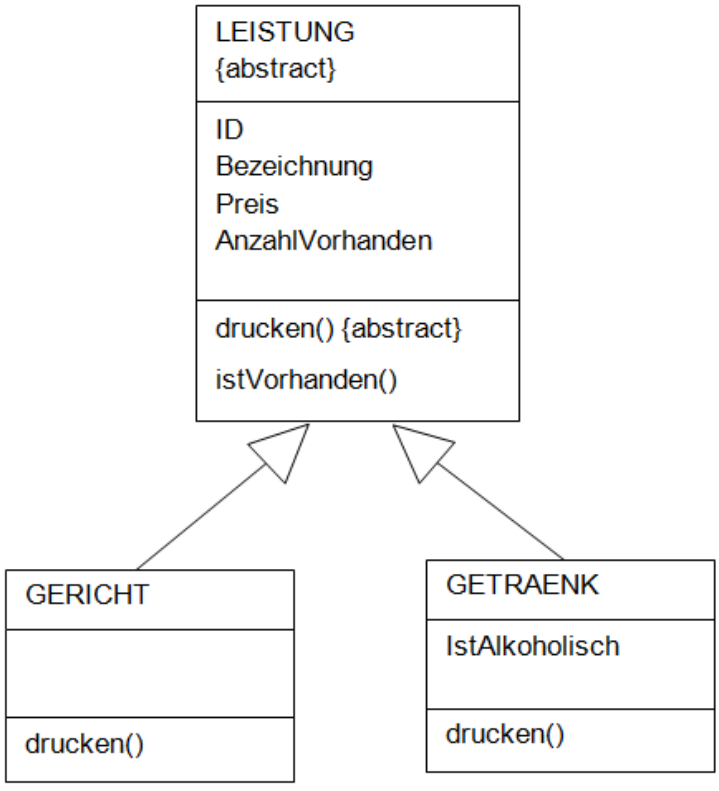
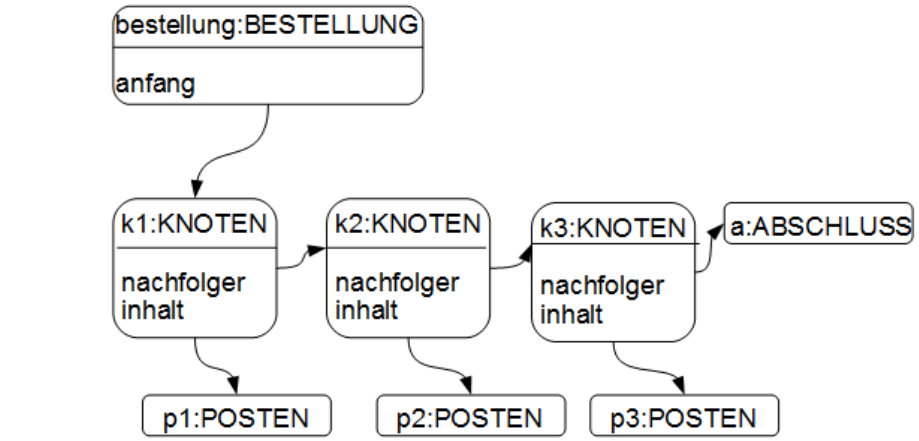
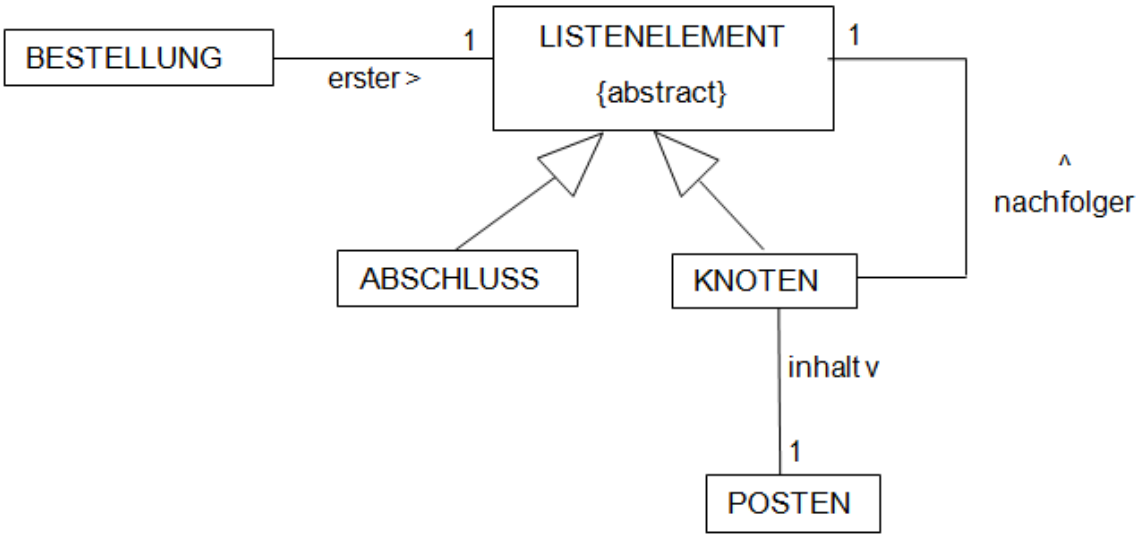
	Informatik Abitur Bayern 2011 / I - Beispiellösung	**Autor** **Rank**
1a)		10
b)	 Hinweis: Die Methode drucken() muss nicht als abstrakt definiert werden. Sie könnte auch in den Unterklassen geeignet überschrieben werden, wenn es nötig ist.	5

c)	Methoden der Klasse LEISTUNG:
	`public boolean istVorhanden() {`
	`return (anzahlVorhanden > 0);`
	`}`
	
	`abstract void drucken();`
	
	Methoden der Klasse GETRAENK:
	`public void drucken() {`
	`// das Attribut istAlkoholisch ist vom Datentyp boolean`
	`if (istAlkoholisch) {`
	`System.out.print ("A ");`
	`}`
	`System.out.println("Bezeichnung: " + Bezeichnung + "`
	`Preis: "`
	`+ Preis + " EUR");`
	
	`}`
	
	Methoden der Klasse GERICHT:
	`public void drucken() {`
	`System.out.println("Bezeichnung: " + Bezeichnung + " Preis: "`
	`+ Preis + " EUR");`
	`}`
		13

d)	 	11

e)	In der Klasse BESTELLUNG:
	`public double rechnungsbetragGeben() {`
	`  return erste.rechnungsbetragGeben();`
	`  }`
	
	In der Klasse LISTENELEMENT:
	`public abstract double rechnungsbetragGeben();`
	
	In der Klasse KNOTEN:
	`public double rechnungsbetragGeben() {`
	`  double preis = 0;`
	`  if (inhalt.istAusgeliefert()){`
	`  preis = inhalt.preisGeben();`
		13

```
    }
    return nachfolger.rechnungsbetragGeben() + preis;

}
```

In der Klasse ABSCHLUSS:

```
public double rechnungsbetragGeben() {
    return 0;
}
```

f) SELECT Bezeichnung, Preis
FROM speisen
WHERE AnzahlVorhanden > 10 — 3

g) 1. Das Programm verbindet sich mit der Datenbank.
2. Das Programm übergibt die Anfrage der Datenbank.
3. Das Abfrageergebnis wird zeilenweise ausgelesen.
4. Die Verbindung wird beendet. — 4

2a) Der Graph ist zusammenhängend, ungerichtet und gewichtet. — 6

Zugehörige Adjazenzmatrix, wobei die Städtenamen mit deren Anfangsbuchstaben abgekürzt werden:

	B	C	G	M	S	T
B	0	-1	12	9	-1	-1
C	-1	0	3	-1	3	6
G	12	3	0	-1	-1	8
M	9	-1	-1	0	-1	2
S	-1	3	-1	-1	0	8
T	-1	6	8	2	8	0

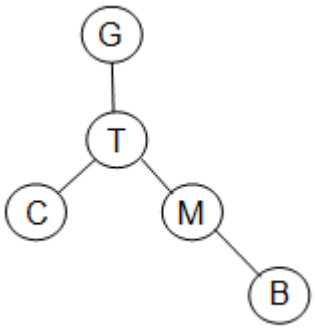
b) Es wird zusätzlich ein Feld besucht vom Typ boolean[] benötigt. Hierin wird gespeichert, welche Städte schon abgearbeitet wurden. Wurde eine Stadt besucht, so ist der zugehörige Wert true, ansonsten false.

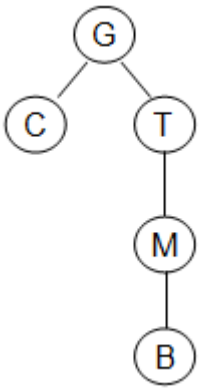
```
void tiefensuche(int startKnotenNummer){
      int startNummer = startKnotenNummer;
      for(int i=0; i< anzahlKnoten; i=i+1)
      {
          besucht[i] = false;
      }

      besuchen(startNummer);
```

```
}


void besuchen(int knotenNummer){
//abfragen ob der Knoten schon besucht wurde
      if(!besucht[knotenNummer]) {
          //aktiven Knoten auf besucht setzen
          besucht[knotenNummer] = true;


          //in der Matrix die Zeile des aktiven Knotens nach Kanten
          durchforsten
          for(int abzweigNummer = 0;abzweigNummer < anzahlKnoten;
          abzweigNummer = abzweigNummer +1){


                //es gibt eine Kante und deren Zielknoten ist noch
                nicht besucht
                if((matrix[knotenNummer][abzweigNummer] > 0) && (!
                besucht[abzweigNummer]){
                     //in die Tiefe gehen und den Abzweigknoten
                     besuchen
                      besuchen(abzweigNummer);
                 }
          }


          //Der aktive Knoten mit der knotenNummer ist fertig
          bearbeitet und wird in der Konsole ausgegeben
          drucken(knotenNummer);
      }
}
```

c) Der Graph ist kein Baum, da immer noch ein Zyklus existiert. — 4

Entferne die Kante C-G:



	Alternativ: Entferne die Kante C-T:  	
		80