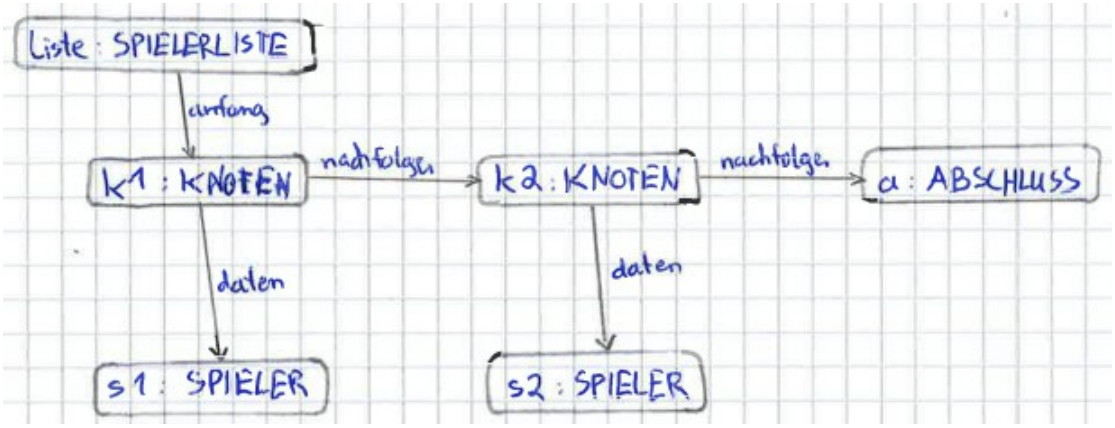
Informatik Abitur Bayern 2019 / I - Lösung

Autoren:
Dippon(1)
Neppl (2)
Rieger (3)

1a 5

```
public class SPIELER{
        private int xPos;
        private int yPos;
        private int punkte;
        private String nutzername;
        public SPIELER(String name){
               nutzername = name;
               punkte = 100;
               xPos = zufallszahlErzeugen(0,999)
               yPos = zufallszahlErzeugen(0,999)
        }
        public void punkteSetzen(int punkteNeu){
               punkte = punkteNeu;
        }
}
```

1b 4



1c 17

```
public class SPIELERLISTE{
…
        public void punktestandAendern(String benutzername, int wert){
               anfang.punktestandAendern(benutzername, wert);
        }

        public int maximum(){
               return anfang.maximum(0);
        }
}

public abstract class LISTENELEMENT{
…
        public abstract void punktestandAendern(String name, int wert);
        public abstract int maximum(int tmpMaximum);
}

public class KNOTEN ...{
…
        public void punktestandAendern(String benutzername, int wert){
               if(daten.nutzernameGeben().equals(benutzername)){
                      daten.punkteSetzen(daten.punkteGeben() + wert);
               }
               else{
                      nachfolger.punktestandAendern(benutzername, wert);
               }
        }
        public int maximum(int tmpMaximum){
               if(daten.punkteGeben() > tmpMaximum){
```

```
                    tmpMaximum = daten.punkteGeben();
                }
                return nachfolger.maximum(tmpMaximum);
        }
}

public class ABSCHLUSS...{
…
        public void punktestandAendern(String benutzername, int wert){
        }
        public int maximum(int tmpMaximum){
                return tmpMaximum;
        }
}
```

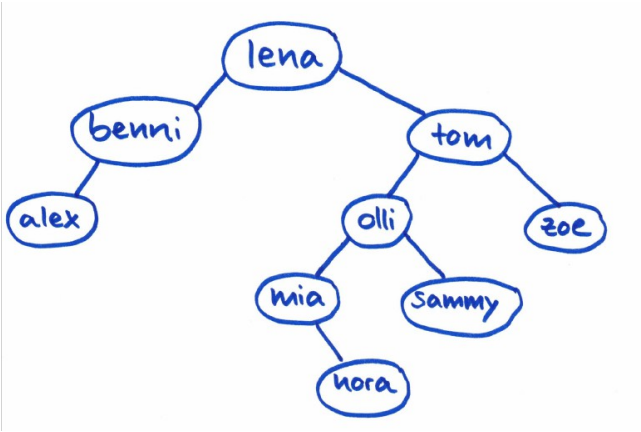
1d

FALLE
schadenpunkte komplexitätsstufe xpos ypos
…

5

Die Fallen können in einem Feld verwaltet werden, da ihre Anzahl immer gleich bleibt.

2a



3

2b Je weiter oben ein Knoten liegt, desto weniger Schritte werden bei der Suche nach ihm benötigt. Deswegen sollen in der untersten Ebene so wenig Knoten wie möglich enthalten sein.
Die Knoten mia, nora und olli sind als passende Wurzeln geeignet, da mit diesen die vorletzte Ebene vollständig besetzt ist und der linke bzw. rechte Teilbaum aus jeweils drei bis fünf Knoten besteht. Zusätzlich ist die unterste Ebene bei passender Einfügereihenfolge minimal gefüllt.

4

2c 2n-1 > 150 000

5

Die Höhe eines optimalen Baums mit 150 000 Knoten beträgt:
$2^n - 1 > 150000, woraus\ folgt: n > log_2(150000+1) \approx 17{,}2$. Der optimale Baum besteht aus 18 Ebenen. Im schlechtesten Fall sind 18 Vergleiche notwendig, welche insgesamt eine Zeit von $18 \cdot 5 \cdot 10^{-6} s = 9 \cdot 10^{-5} s = 90 \mu s$ in Anspruch nehmen. Dieser Zeitraum ist für den Nutzer während der Eingabe nicht wahrnehmbar.

2d

8

```
In der Klasse NUTZERBAUM:
public boolean istFrei(String nameNeu)
{
      return wurzel.istFrei(nameNeu);
}

In der Klasse BAUMELEMENT:
public abstract boolean istFrei(String nameNeu);

In der Klasse KNOTEN:
```

```
public boolean istFrei(String nameNeu)
{
     int x = benutzername.vergleichenMit(nameNeu);
     if (x<0){
         return reNf.istFrei(nameNeu);
     }
     else if(x>0){
         return liNf.istFrei(nameNeu);
     }else {
         return false;
     }
}

In der Klasse ABSCHLUSS:
public boolean istFrei(String nameNeu)
{
      return true;
}
```

2e Der Aufruf gibt eine Liste zurück, die alle Benutzernamen die mit „s“ beginnen in lexikographisch aufsteigender Reihenfolge enthält. 5

3a

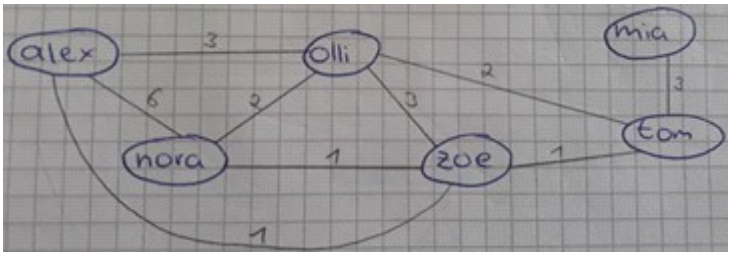
	Alex	Nora	Olli	Zoe	Mia	Tom
Alex		5	2			
Nora	5		1			
Olli	2	1		1		
Zoe			1			
Mia						3
Tom					3	

5

Der Graph hat folgende Eigenschaften:

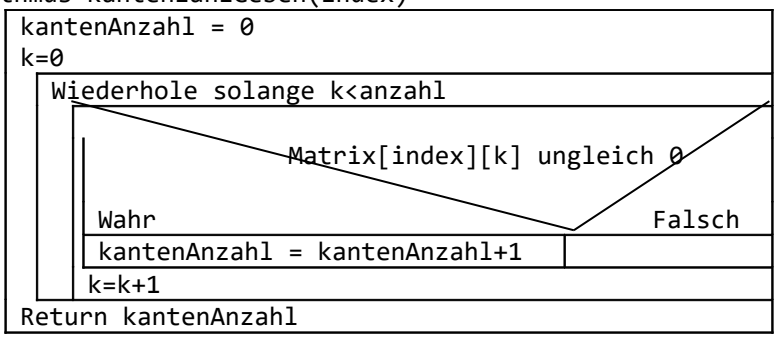
- Gewichtet *(Die Anzahl der Kooperationen)*
- Ungerichtet *(Wenn Alex mit Nora gearbeitet hat, dann auch Nora mit Alex)*
- Nicht zusammenhängend *(Mia und Tom haben nur miteinander, aber nie mit den anderen zusammengearbeitet)*

3b



4

3c Algorithmus KantenzahlGeben(index) 6



Es wird die Zeile in der Matrix betrachtet, die dem Knoten index entspricht. Alle Zellen in dieser Zeile werden betrachtet. Besteht eine Kante zwischen dem Knoten index und einem weiteren Knoten, so ist der Eintrag in der jeweiligen Zelle nicht leer und die Anzahl der Kanten wird um eins erhöht.

3d Algorithmus besterKooperatorGeben() 9

```
hoechsteAnzahl = 0        Die aktuell höchste Anzahl an Kanten
hoechsterIndex = 0        Index mit der höchsten Anzahl an Kanten
k = 0
wiederhole solange k<anzahl     Jeder Knoten wird betrachtet
      wenn hoechsteAnzahl < kantenAnzahlGeben(k) dann
            Der aktuelle Knoten hat mehr Kanten als alle bisherigen
            hoechsteAnzahl = kantenAnzahlGeben(k)
            hoechsterIndex = k
      endeWenn
      k = k+1
endewiederhole
return hoechsterIndex
```

Laut Algorithmus ist Olli am kooperativsten, da er mit 3 weiteren Spielern zusammengearbeitet hat. Alex hat nur mit 2 Spielern zusammengearbeitet, aber dafür häufiger. Er könnte daher als kooperativer als Olli angesehen werden.