Informatik Bayern 2021/II - Beispiellösung

Autor*innen
Anolick,
Kornhas,
Wiedemann

1a Bei einem Feld wird die Anzahl der Plätze zu Beginn fest definiert, während eine Liste dynamisch erweitert werden kann. Da spontan neue Termine hinzugefügt werden sollen, eignet sich die Liste besser. 2

b Die Methode gibt in der Klasse TERMIN falsch zurück, in der Unterklasse TRAININGSTERMIN wird die Methode überschrieben und gibt immer wahr zurück. Die Klasse WETTKAMPFTERMIN erbt die Methode von ihrer Oberklasse und gibt ebenfalls falsch zurück. 3

c 5

```
public float anwesenheitsquoteBerechnen()
{
	int anzahlAnwesend = 0;
	for (int i = 0; i < 8; i++)
	{
		if (anwesenheit[i])
		{
			anzahlAnwesend ++;
		}
	}
	return anzahlAnwesend/ 8.0f;
}
```

d in TERMINLISTE: 13

```
public int anzahlTrainingsGeben(int spielerID)
{
	return anfang.anzahlTrainingsGeben(spielerID):
}

public void trainingsbeteiligungAusgeben()
{
	anfang.trainingsbeteiligungAusgeben();
}
```

in LISTENELEMENT:

```
public abstract int anzahlTrainingsGeben(int spielerID);
public abstract void traingsbeteiligungAusgeben();
```

in KNOTEN:

```
public int anzahlTrainingsGeben(int spielerID)
{
	boolean[] anwesend = daten.anwesenheitGeben();
	if (daten.istTraining() && anwesend[spielerID])
	{
		return nachfolger.anzahlTrainingsGeben(spielerID)+1;
	}
	else
	{
		return nachfolger.anzahlTrainingsGeben(spielerID);
	}
}

public void trainingsbeteiligungAusgeben()
{
```

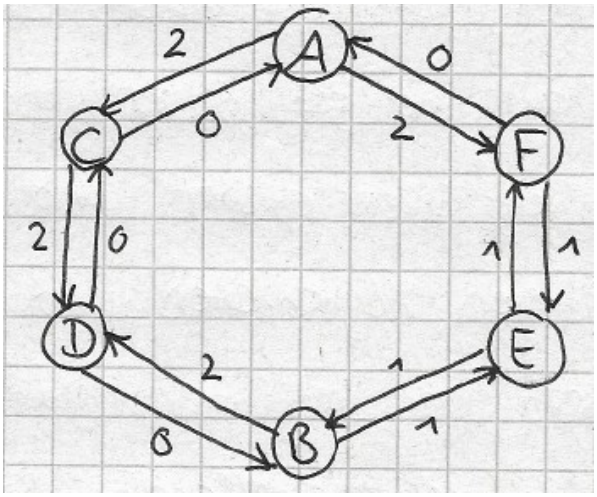
```
        if (inhalt.istTraining())
        {
                System.out.println("Training "+inhalt.datumGeben()+" " +
                        daten.anwesenheitsquoteBerechnen*100+"%");
        }
        nachfolger.trainingsbeteiligungAusgeben();
}
```

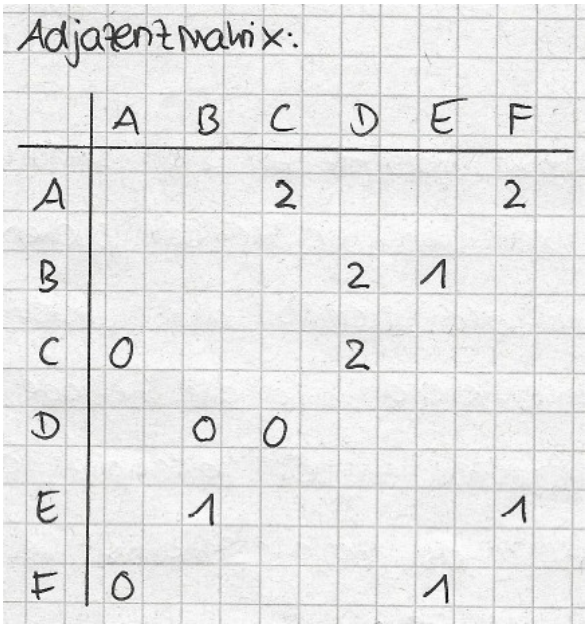
in ABSCHLUSS:

```
public int anzahlTrainingsGeben(int SpielerID)
{
        return 0;
}

public void trainingsbeteiligungAusgeben() { }
```

2a Spielpaarungen: A vs. B, C vs. E, D vs. F 8





b 5

```
public void punktzahlenAktualisieren()
{
        for(int i = 0; i < anzahl; i = i + 1)
        int punkte = 0;
```

```
        for (int j = 0; j < anzahl ; j = j + 1)
        {
              if (matrix[i][j] > 0)
                    punkte = punkte + matrix[i][j];
        }
        teilnehmer[i].punktzahlSetzen(punkte);
  }
```

Hinweis: Da die Sequenzen in den bedingten Wiederholungen jeweils einzeilig sind, kann die Klammerung entfallen.

c Die Methode ermittelt den noch freien Spieler mit der höchsten Punktzahl und gibt dessen Index (spielerID) zurück. Falls kein freier Spieler existiert, wird -1 zurückgegeben. Bei mehreren Möglichkeiten wird der kleinste Index zurückgegeben. Der Wert des Attributs nochFrei wird anschließend bei dem betreffenden Spieler auf false gesetzt. 8

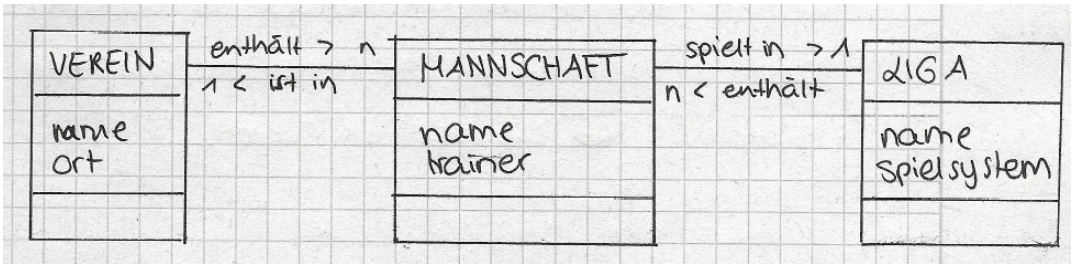
Zunächst wird das Attribut nochFrei bei allen Spielern auf wahr gesetzt. Dann wird die Methode *suchen()* einmal aufgerufen und die SpielerID des besten Spieler ermittelt. Anschließend wird die Methode nochmal solange aufgerufen, bis ein passender Partner gefunden wird – also in der Adjazenzmatrix bei den beiden Indizes -1 steht.

d 10

```
public int zusatzpunkteGeben(int index)
{
      for (int i = 0; i < anzahl; i++) {
            teilnehmer[i].besuchtSetzen(false);
      }
      return suche(index);
}

public int suche(index)
{
      int zusatzpunkte = 0;
      teilnehmer[index].besuchtSetzen(true);
      for (int i = 0; i < anzahl; i++) {
            if (teilnehmer[i].besuchtGeben() == false)
                  if (matrix[index][i] == 2)
                        zusatzpunkte = suche(i) +1;
      }
      return zusatzpunkte;
}
```

3a 7



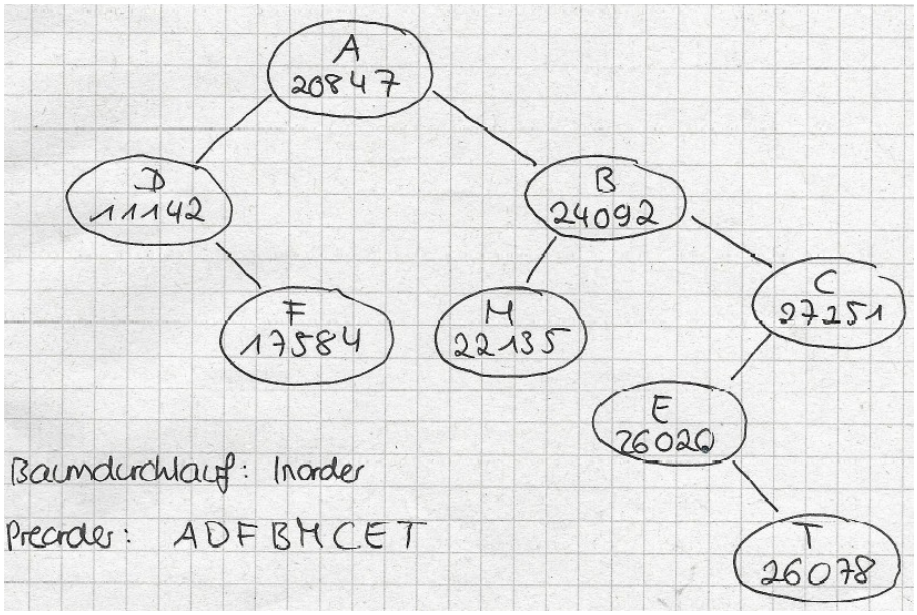
Weitere Klassen:

- MITGLIED spielt in einer Mannschaft (Name, Geburtstag, …)
- WETTKAMPF zur Verwaltung von Spielen (Datum, Ort, …)

b Unterteilung in verschiedene Mannschaften (z.B. Kindermannschaft, Jugendmannschaft, Erwachsenen-Mannschaft, …) als Unterklassen und einer Oberklasse MANNSCHAFT. 4

4a
```
SELECT DRang, Name, Geschlecht
FROM Spieler
WHERE Verein = “TTV Bounceout“
ORDER BY DRang;
```
3

b 7



Baumdurchlauf: Inorder liefert numerische Sortierung
Preorder: ADFBMCET

c Da der Baum nicht nach Vereinen sortiert ist, muss für die Vereinsrangliste trotzdem der komplette Baum durchlaufen werden. Der Vorteil des geordneten Binärbaums kommt hier nicht zum tragen. 2

d Maximale Höhe: 540.000 (Liste) 3
Minimale Höhe: Aufrunden($\log_2(540.000+1)$) ≈ Aufrunden(19.04) = 20