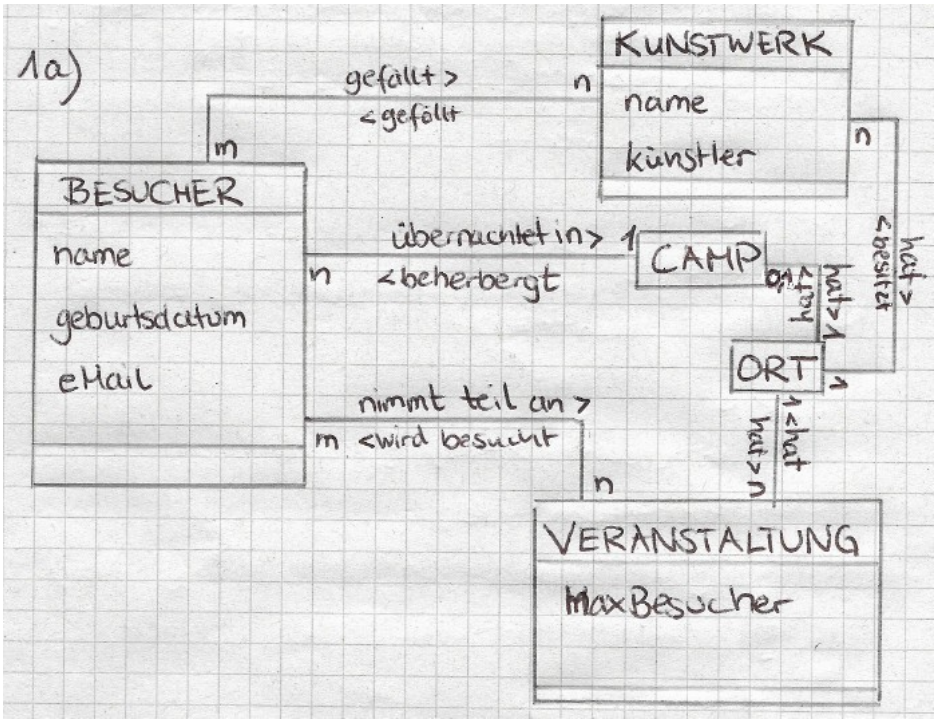
Informatik Abitur Bayern 2021/I - Beispiellösung

Autorinnen:
Anolick
Kornhas
Wiedemann

1a



7

b

- Anzahl der Campbesucher vorher bekannt
  -> Feldelementeanzahl muss nicht verändert werden
- Zugriff über Index schneller als rekursiver Zugriff bei Liste

2

c

```
public boolean nachbarplaetzeVerfuegbar(int n)
{
	int counter = 0;
	for (int i=0; i< besucher.length; i=i+1)
	{
		if (besucher[i] == null)
			counter = counter + 1 ;
		else
			counter = 0;
		if (counter >= n)
			return true;
	}
	return false;
}
```

7

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

2a In VERANSTALTUNGSLISTE:

18

```
public void sortiertEinfuegen(VERANSTALTUNG vNeu)
{
	anfang = anfang.sortiertEinfuegen(vNeu);
}

public int anzahlGeben(int tag, int zeitfenster)
{
	return anfang.anzahlGeben(tag, zeitfenster);
}
```

In LISTENELEMENT:

```
public abstract LISTENELEMENT sortiertEinfuegen(VERANSTALTUNG vNeu);
public abstract int anzahlGeben(int tag, int zeitfenster);
```

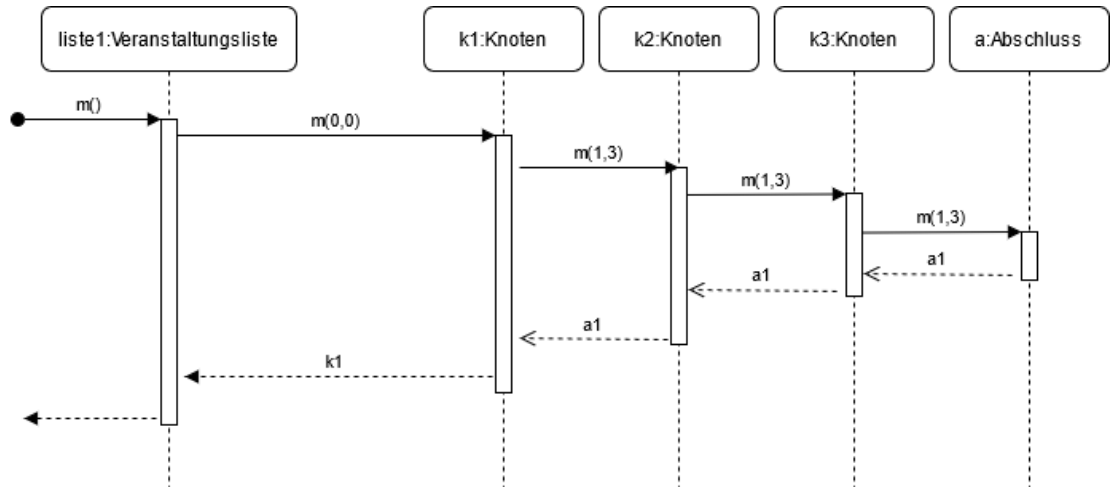
In KNOTEN:

```
public KNOTEN(VERANSTALTUNG vNeu, LISTENELEMENT lNeu)
{
            inhalt = vNeu;
            nachfolger = lNeu;
}

public LISTENELEMENT sortiertEinfuegen( VERANSTATUNG vNeu)
{
      if(inhalt.gibTag() > vNeu.gibTag()
            || (inhalt.gibTag() == vNeu.gibtTag()
            && inhalt.gibZeitfenster() > vNeu.gibZeitfenster()))
      {
            return new Knoten (vNeu, this);
      }
      else
      {
            nachfolger = nachfolger.sortiertEinfuegen(vNeu);
            return this;
      }
}

public int anzahlGeben(int tag, int zeitfenster)
{
      if (tag == inhalt.gibTag() && zeitfenster ==inhalt.gibZeitfenster())
      {
            return nachfolger.anzahlGeben(tag, zeitfenster) +1;
      }
      else
      {
            return nachfolger.anzahlGeben(tag, zeitfenster);
      }
}
```

In ABSCHLUSS:

```
public LISTENELEMENT sortiertEinfuegen(VERANSTALTUNG vNeu){
            return new KNOTEN(vNeu, this);
}

public int anzahlGeben(int tag, int zeitfenster){
            return 0;
}
```

b Löscht weiter hinten stehende, gleichzeitige Veranstaltungen 9



Quelle: S. Göritz

3a Suche nach Namen des Kunstwerks: 3

- geordneter Binärbaum bringt in der Regel Vorteile (maximal so viele Vergleiche, wie Baumebenen)

Suche nach Namen des Künstlers:

- geordneter Binärbaum müsste komplett durchsucht werden

b In KUNSTWERKBAUM: 8

```
public void ortAendern(String name, ORT ortNeu)
{
	wurzel.ortAendern(name, ortNeu);
}
```

In BAUMELEMENT:

```
public abstract void ortAendern(String name, ORT ortNeu);
```

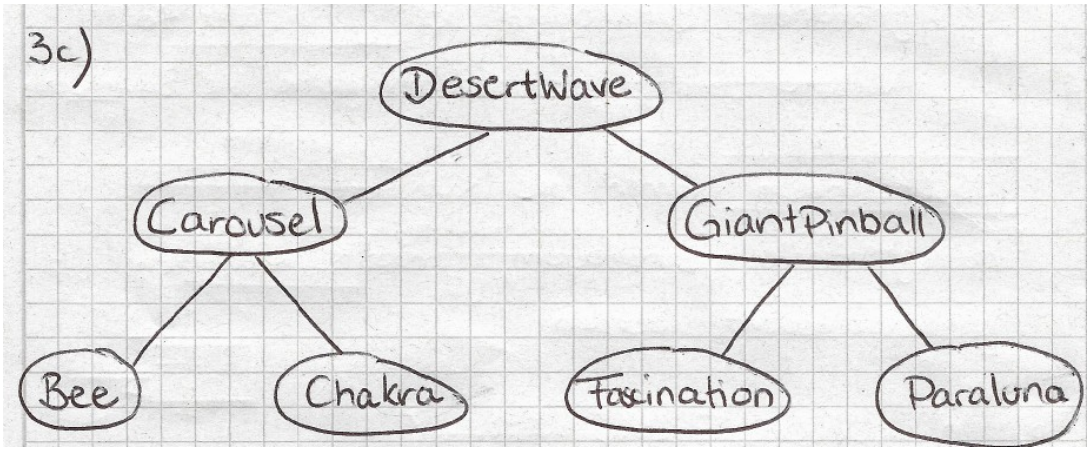
In KNOTEN:

```
public void ortAendern(String name, ORT ortNeu)
{
	int n = name.vergleicheMit(inhalt.nameGeben());
	if (n == 0)
	{
		inhalt.ortSetzen(ortNeu);
	}
	else if (n > 0)
	{
	nachfolgerRechts.ortAendern(name, ortNeu);
	}
	else
	{
	nachfolgerLinks.ortAendern(name, ortNeu);
	}
}
```

In ABSCHLUSS:

```
public void ortAendern(String name, ORT ortNeu){}
```

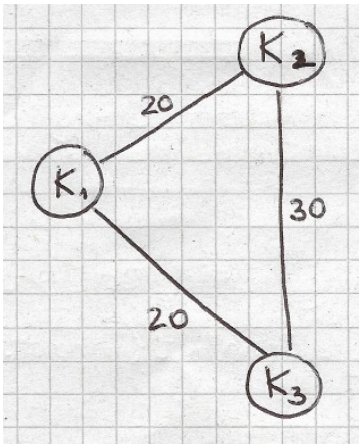
c

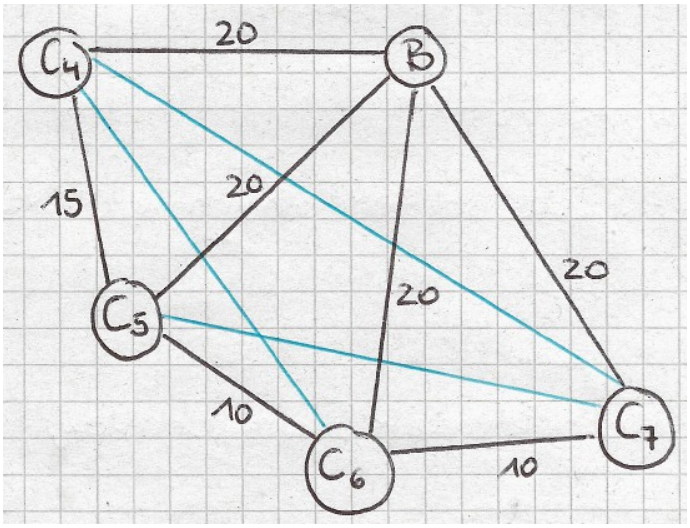


3

d Preorder behält die hierarchische Struktur bei, anders als Inorder 2

4a





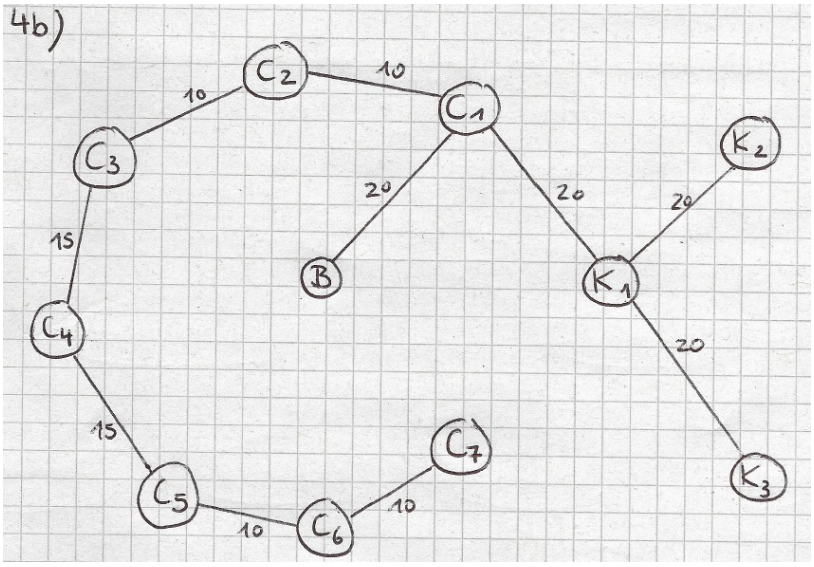
7

3 fehlende

- n ≙ Anzahl der Knoten
- Anzahl der Kanten eines vollständigen Graphen (Kleiner Gauß oder Summieren):
  - $\frac{n \cdot (n-1)}{2} = \frac{11 \cdot 10}{2} = 55$

→36 zusätzliche Wege

b



6

c `public ORT ortVorschlagen(int index)` 8

```
{
	KNOTEN[] nachbarn = new KNOTEN[anzahlKnoten];
	int platzZuweiser = 0;
	for (int i=0; i < anzahlKnoten; i++)
	{
		if (matrix[index][i] > 0)
		{
			nachbarn[platzZuweiser] = knoten[i];
			platzZuweiser++;
			}
		}
		return nachbarn[zufallszahl(platzZuweiser)].ortGeben();
	}
}
```