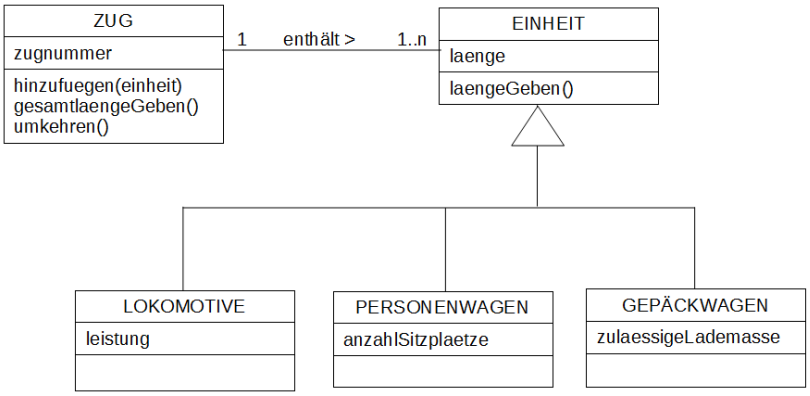
Informatik Abitur Bayern 2016 / II - Beispiellösung

Autor:
Großmann/
Reinold

1a

4



1b JAVA:

12

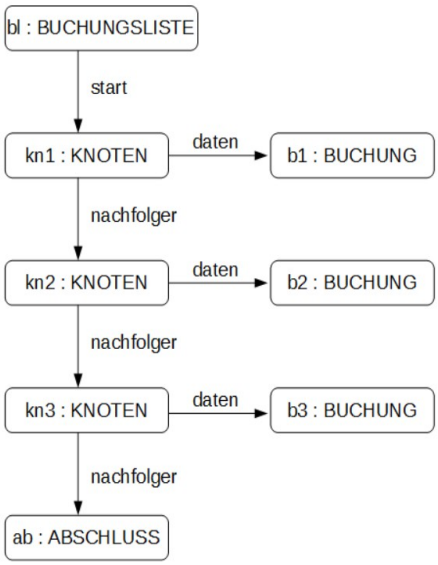
```
public class ZUG
{
    private EINHEIT[] einheiten;
    private int anzEinheiten;
    private int zugnummer;

    public ZUG(int zugNr, EINHEIT loko)
    {
        this.einheiten[] = new EINHEIT[20];
        this.einheiten[0] = loko;
        this.zugnummer = zugNr;
        this.anzEinheiten = 1;
    }

    public void hinzufuegen(EINHEIT einheit)
    {
        if(this.anzEinheiten < 20)
        {
            this.einheiten[anzEinheiten] = einheit;
            this.anzEinheiten++;
        }
    }

    public void umkehren()
    {
        EINHEIT[] e = new EINHEIT[20];
        e[0] = this.einheiten[0];
        for(int i = 1; i<anzEinheiten; i++)
        {
            e[i] = this.einheiten[anzEinheiten-i];
        }
        this.einheiten = e;
    }
}
```

2a 5



2b JAVA: 12

Klasse BUCHUNGSLISTE

```
public int anzahlReisendeGeben(int bahnhofsnummer)
{
    return start.anzPersonenGeben(bahnhofsnummer);
}
```

Klasse LISTENELEMENT

```
public abstract int anzPersonenGeben(int bahnhofsnummer);
```

Klasse KNOTEN

```
public int anzPersonenGeben(int bfNr)
{
    return buchung.personenzahlGeben(bfNr) +
              nachfolger.anzPersonenGeben(bfNr);
}
```

Klasse BUCHUNG

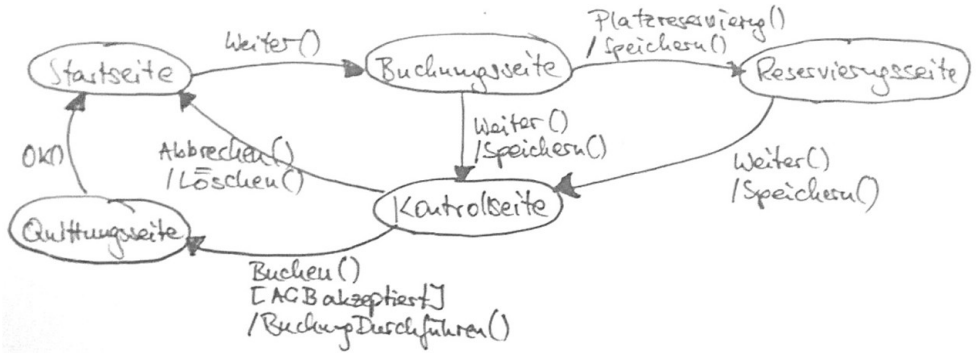
```
public int personenzahlGeben(int bahnhofsnummer)
{
    if((this.bahnhof1 <= bahnhofsnummer) && (this.bahnhof2 >
bahnhofsnummer))
    {
        return this.personenzahl;
    }
    return 0; /* else kann entfallen, da nach dem Return-Statement im
wahr-Fall die Arbeit der Methode beendet wird.*/
}
```

Klasse ABSCHLUSS

```
public int gibPersonenzahl(int bahnhofsnummer)
{
    return 0;
}
```

2c Da eine Liste einfach erweitert werden kann, macht es Sinn, die Wagen so zu verwalten, da sich deren Zahl ja ständig ändert. Die Anzahl der Sitzplätze in einem Wagen ist jedoch fest, weshalb sich die Datenstruktur Feld hierfür eignet. 2

2d



8

3a

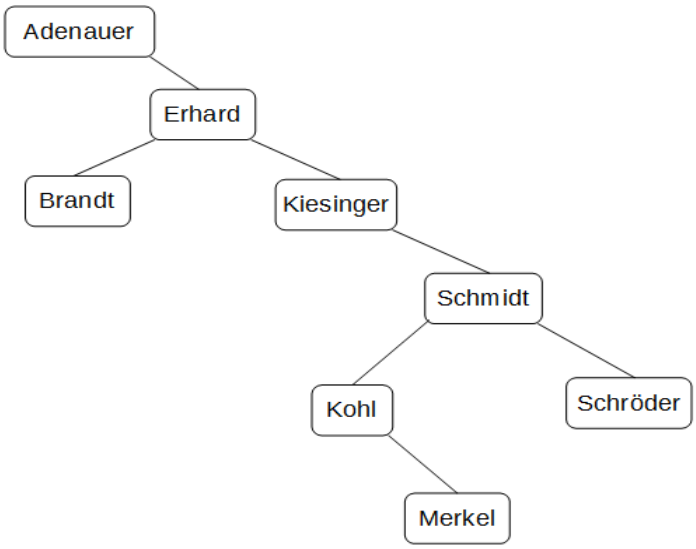
```
SELECT      kundennummer
FROM        kunden
WHERE       vorname = 'Herbert' AND nachname = 'Huber';
```

3

3b Die Kombination aus Vorname und Nachname ist nicht eindeutig, weshalb durch dieses Verfahren die Kundennummer an eine Person mit gleichem Namen ausgegeben werden könnte.
Ein weiteres Problem ist, dass nicht geprüft wird, ob es sich wirklich um diese Person handelt, oder ob diese nur angibt, die entsprechende Person zu sein.
Eine Verbesserung könnte durch das gängige Verfahren über E-Mail geschehen. Beispielsweise könnte an einen Kunden eine E-Mail mit der Kundennummer geschickt werden. Dazu muss bei der Registrierung eines neuen Kunden die E-Mail-Adresse gespeichert werden. So ist sichergestellt, dass nur eine Person mit Zugriff auf entsprechendes E-Mail-Konto die Information der Kundennummer erhält.

3

3c



4

3d Ungünstigster Fall: 2 Mio. Ebenen
Günstigster Fall: $2^n - 1 >= 2\,000\,000 \Rightarrow n >= \log_2(2\,000\,000 + 1) = 20{,}93\ldots \Rightarrow 21$ Ebenen
In einem Binärbaum müssen, um ein Blatt zu erreichen, maximal so viele Vergleiche wie Ebenen durchgeführt werden. In diesem Beispiel wären das im ungünstigsten Fall 2 Mio.
Im ungünstigsten Fall entartet ein geordneter Binärbaum zu einer einfach verketteten Liste, bei der die Suche nur am ersten Element begonnen werden kann. Ein geordneter Binärbaum ist nur dann optimal, wenn er ausbalanciert ist.

6

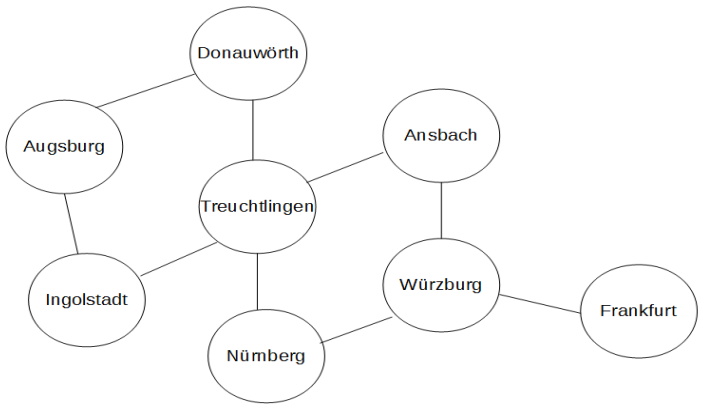
4a Da keine Kosten, Entfernungen o. Ä. angegeben sind, kann der Graph nur unbewertet erstellt werden. Des Weiteren sind bei allen Strecken beide Richtungen angegeben, weshalb der Graph ungerichtet ist.

2

4b

	Ansbach	Augsburg	Donauwörth	Frankfurt	Ingolstadt	Nürnberg	Treuchtlingen	Würzburg
Ansbach							1	1
Augsburg			1		1			
Donauwörth		1					1	
Frankfurt								1
Ingolstadt		1					1	
Nürnberg							1	1
Treuchtlingen	1		1		1	1		
Würzburg	1			1		1		

6



4c Wird die Route nach diesem Baum geplant, so werden zwar alle Knoten erreicht. Allerdings werden viele Strecken mehrfach befahren.Für eine Rundreise ist dies suboptimal. 3

4d JAVA: 0

```
public class STRECKENGRAPH
{
    private String[] bahnhoefe;
    private boolean[][] verbindungen;
    private int anzBahnhoefe;

    public STRECKENGRAPH()
    {
        this.bahnhoefe[] = new String[100];
        this.verbindungen[][] = new boolean[100][100];
        this.anzBahnhoefe = 0;
    }

    public void bahnhofHinzufuegen(String name)
    {
        this.bahnhoefe[this.anzBahnhoefe] = name;
        this.anzBahnhoefe++;
    }

    public void streckeHinzufügen(String name1, String name2)
    {
        int i1 = bahnhofsnummerGeben(name1);
        int i2 = bahnhofsnummerGeben(name2);
        this.verbindungen[i1][i2] = true;
        this.verbindungen[i2][i1] = true;
    }
}
```