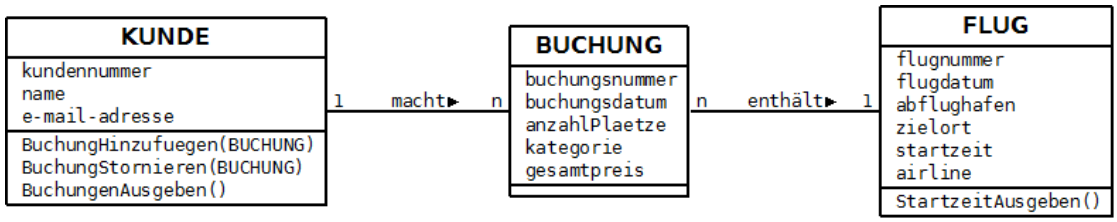
Informatik Abitur Bayern 2011 / II - Beispiellösung

Autor:
Weiler

1a



11

1b

```
public class KUNDE
{
    // Attribute deklarieren
    private int kundennummer;
    private String name;
    private String eMailAdresse;
    private BUCHUNG[] buchungen;
    private int anzahlBuchungen;

    //Konstruktor für Objekte der Klasse KUNDE
    public KUNDE(int kNr, String kName, String eMail)
    {
        // Attribute initialisieren
        kundennummer = kNr;
        name = kName;
        eMailAdresse = eMail;
        buchungen = new BUCHUNG[20];
        anzahlBuchungen = 0;
    }

    //Methoden
    public void BuchungHinzufuegen(BUCHUNG neueBuchung)
    {
        if(anzahlBuchungen < 20){
            buchungen[anzahlBuchungen] = neueBuchung;
            anzahlBuchungen = anzahlBuchungen + 1;
        }else{
            for(int i = 0; i < 19; i = i+1){
                buchungen[i] = buchungen[i+1];
            }
            buchungen[19] = neueBuchung;
        }
    }
}
```

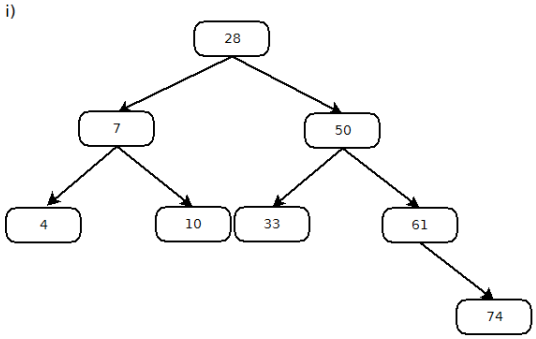
14

1c Die Verwendung eines Feldes für die Verwaltung der Flugbuchungen bringt beispielsweise folgende Nachteile mit sich: 2

- Durch die Länge 20 des Feldes ist die maximale Anzahl der abspeicherbaren Buchungen vorgegeben.
- Durch freie Plätze wird unnötig Speicherplatz verschwendet.
- Sind bereits alle Plätze des Feldes belegt, so müssen 19 Buchungen jeweils um eine Position nach vorne gerückt werden.
- Sollen die Feldelemente nach einem Merkmal sortiert werden, so kann der Aufwand beim Sortieren des Feldes recht hoch sein.

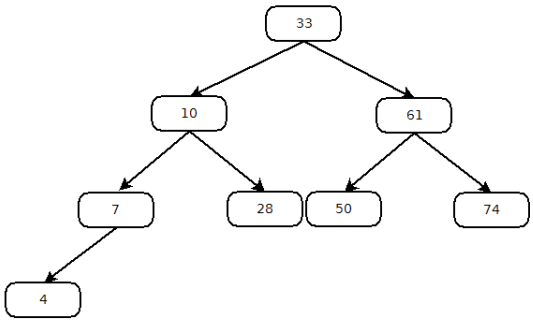
1d Mögliche Lösungen: 8

i)



oder

ii)

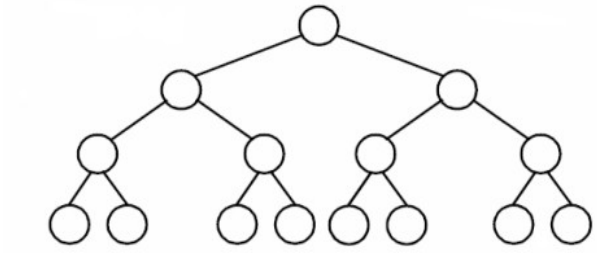


Da der gesuchte Baum die Höhe 4 hat, müssten im ungünstigsten Fall auch 4 Knoten (einschließlich der Wurzel) besucht werden.

Für die oben dargestellten Bäume ergeben sich folgende Besuchsfolgen des Postorder-Durchlaufs (Reihenfolge: linker Teilbaum → rechter Teilbaum → Wurzel):

i. 4 – 10 – 7 – 33 – 74 – 61 – 50 – 28
ii. 4 – 7 – 28 – 10 – 50 – 74 – 61 – 33

1e



1 Ebene: 1 Knoten 8

2 Ebenen: max. 1 + 2 = 3 Knoten

3 Ebenen: max. 1+ 2 + 4 = 7 Knoten

4 Ebenen: max. 1+ 2 + 4 + 8 = 15 Knoten

Allgemein: In einem Baum mit n Ebenen befinden sich höchstens $2^n - 1$ Knoten.

$2^n - 1 \geq 10\,000\,000$

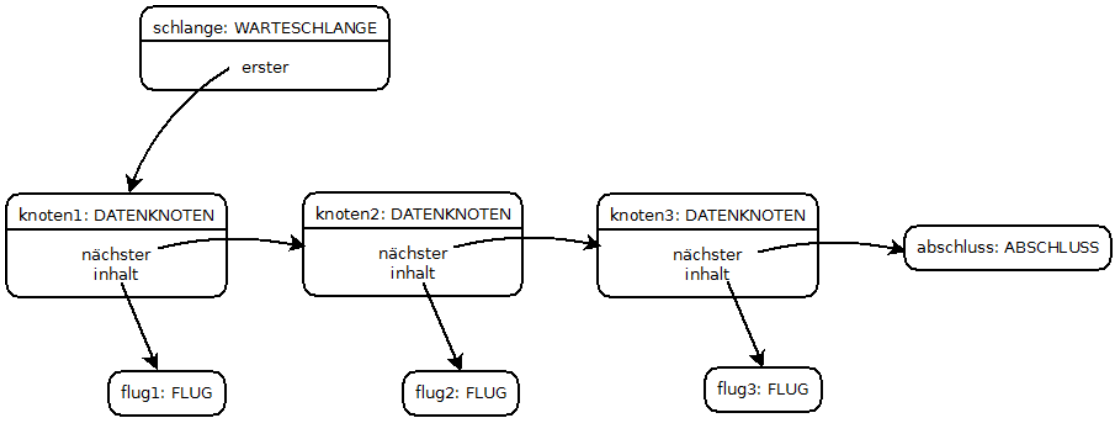
<=> $n * \ln 2 \geq \ln 10\,000\,001$

<=> $n \geq 23{,}2$

Im ungünstigsten Fall müssen 24 Knoten beim Zugriff auf einen bestimmten Kunden besucht werden.

Wird eine Datenmenge in Form einer verketteten Liste gespeichert, so müssen im ungünstigsten Verlauf einer Suche nach einem Datenelement alle Elemente der Liste abgefragt werden.
Würden also die 10 Millionen Kundendaten mit Hilfe einer verketteten Liste verwaltet werden, so müssten beim Zugriff auf einen bestimmten Kunden im ungünstigsten Fall alle 10 Millionen Knoten abgefragt werden. Dies wäre deutlich aufwendiger als die Suche in einem balancierten Binärbaum (24 << 10 000 000).

2a 6



2b Methode in der Klasse FLUG: 14

```
public int passagierzahlGeben(){
    return passagierzahl;
}
```

Methode in der Klasse LISTENELEMENT:

```
public abstract int restPassagierSummeGeben();
```

Methode in der Klasse DATENKNOTEN:

```
public int restPassagierSummeGeben(){
    return naechster.restPassagierSummeGeben() + inhalt.passagierzahlGeben();
}
```

Methode in der Klasse ABSCHLUSS:

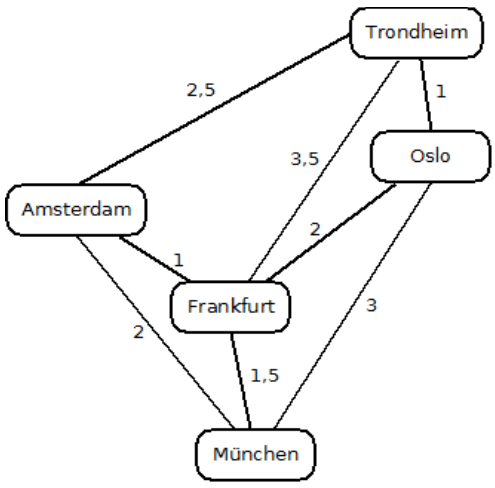
```
public int restPassagierSummeGeben(){
    return 0;
}
```

Methode in der Klasse WARTESCHLANGE:

```
public int passagierSummeGeben(){
    return erster.restPassagierSummeGeben();
}
```

2c Bei der Realisierung der Warteschlange wird das Entwurfsmuster **Kompositum** (composite pattern) verwendet. 3
Bei der Softwareentwicklung müssen häufig Standardsituationen umgesetzt werden. Zum Umgang mit diesen immer wiederkehrenden Aufgaben gibt es detailliert ausgearbeitete und gut beschriebene Lösungsschemata, die Entwurfsmuster. Greift man auf solche Lösungsansätze zurück, spart man Programmieraufwand und somit Kosten. Die Softwaremuster haben in der Regel eine möglichst lose Kopplung der beteiligten Klassen zum Ziel, wodurch die Wartung der Programme erleichtert und die Programmdokumentation übersichtlicher wird.

3a 8



	München	Frankfurt	Amsterdam	Oslo	Trondheim
München	0	1,5	2	3	-1
Frankfurt	1,5	0	1	2	3,5
Amsterdam	2	1	0	-1	2,5
Oslo	3	2	-1	0	1
Trondheim	-1	3,5	2,5	1	0

3b Mit dem Algorithmus **Tiefensuche** können alle Knoten eines zusammenhängenden Graphen ausgehend von einem Startknoten systematisch besucht werden. Der Name „Tiefensuche“ rührt daher, dass das Verfahren zunächst in die „Tiefe“ und erst dann in die „Breite“ des Graphen läuft. Man geht also vom jeweiligen Knoten erst zu einem noch nicht besuchten Nachbarknoten und setzt dort den Algorithmus rekursiv fort. 6

Herr Salesman besucht die Orte dann in folgender Reihenfolge (unter Berücksichtigung der Speicherreihenfolge):
Amsterdam – München – Frankfurt – Oslo – Trondheim

(Alternativ kann auch der Algorithmus **Breitensuche** verwendet werden.
Der Name „Breitensuche“ rührt daher, dass das Verfahren zunächst in die „Breite“ und dann erst in die „Tiefe“ des Graphen führt. Man geht von einem Knoten, der gerade besucht wird, zuerst zu allen Nachbarknoten, bevor deren Nachbarn besucht werden.

Herr Salesman besucht die Orte dann in folgender Reihenfolge (unter Berücksichtigung der Speicherreihenfolge):
Amsterdam – München – Frankfurt – Trondheim – Oslo)