Informatik Abitur Bayern 2018 / I - Lösung

Autoren:
Nuber

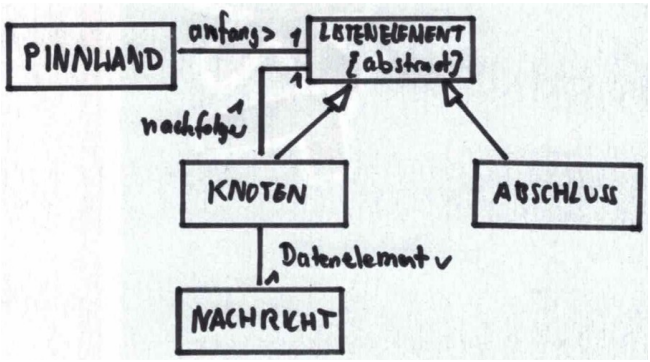
1a Phasen bei der Durchführung eines Softwareprojekts. 5

z.B.: Das Wasserfallmodell

- Anforderungsanalyse: Festlegung der Anforderungen an das System. Ergebnis ist eine Anforderungsbeschreibung → dient als "Vertrag" zwischen Anwender/Entwickler
- Entwurf: Modellierung des Systems
- Implementierung: Codierung des Entwurfs in einer Programmiersprache
- Test und Integration: Test der einzelnen Komponenten, Einbau, Systemtest
- Einsatz & Wartung: Fehlerbeseitigung nach Inbetriebnahme

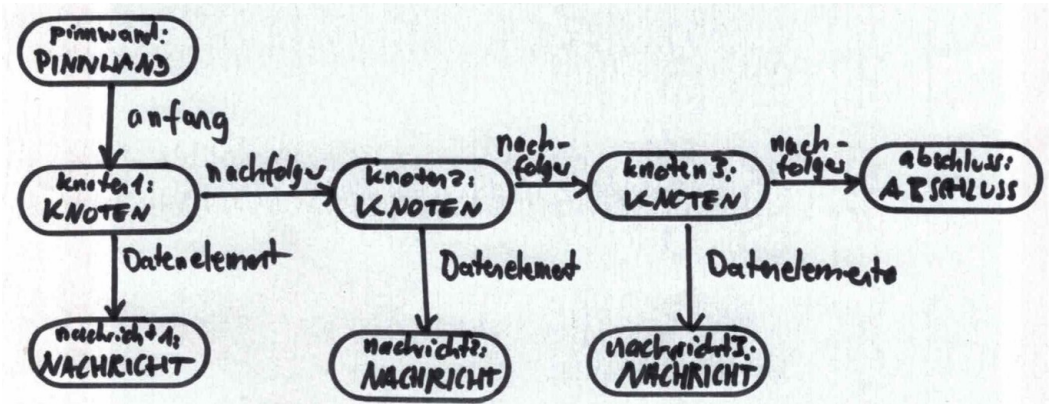
Hinweis: Da die Wartung nicht mehr zur eigentlichen Erstellung gehört, muss sie daher nicht aufgeführt sein. Auch andere Modelle als das Wasserfallmodell können vorgestellt werden.

1b 5



1c damit der Speicher dynamisch ist (z.B. Größe der Nachrichtenmenge, Löschen einer Nachricht) und der benötigte Speicherplatz möglichst gering gehalten wird 2

1d 4



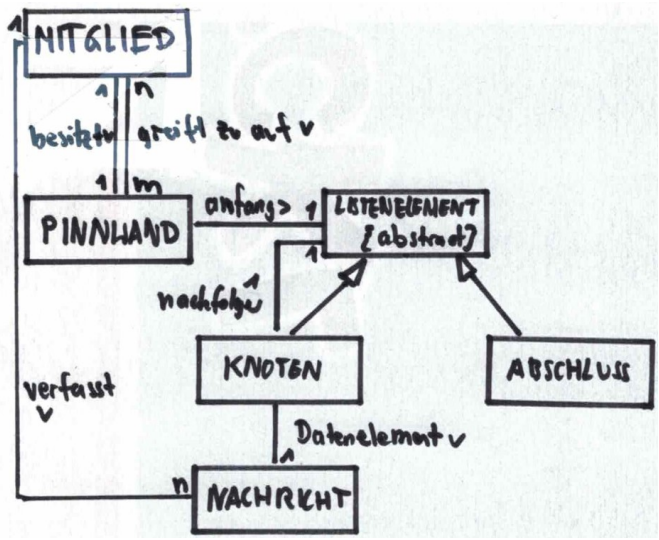
1e 8

```
class PINNWAND{
   private LISTENELEMENT anfang;
   int anzahlNachrichtenGeben(){
       return anfang.anzahlNachrichtenGeben();
   }
}

abstract class LISTENELEMENT{
   abstract int anzahlNachrichtenGeben();
}

class KNOTEN extends LISTENELEMENT{
   private LISTENELEMENT nachfolger;
   int  anzahlNachrichtenGeben(){
       return 1 + nachfolger. anzahlNachrichtenGeben();
   }
}
```

```
class ABSCHLUSS extends LISTENELEMENT{
   int  anzahlNachrichtenGeben(){
       return 0;
   }
}
```

Hinweis: Bei dieser Implementierung ist explizit eine Implementierung inklusive Klassenkopf gefordert.

1f



4

2a Wenn die Knoten mit aufsteigenden Mitgliedsnummern in den Baum eingefügt, so entsteht die Struktur einer einfach verketteten Liste (entarteter Baum). Dabei gehen die Vorteile des ausbalancierten Baums verloren, z. B. die schnelle Suche.

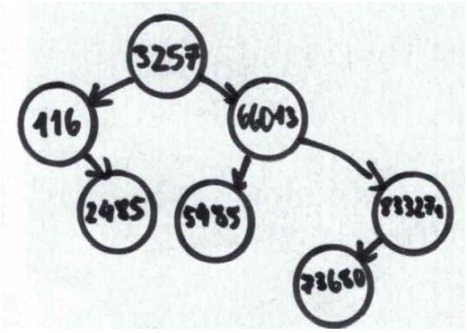
2

2b

```
int nummer = zufallszahlBestimmen()
wiederhole solange Suchen(nummer)!=null
   nummer = zufallszahlBestimmen()
```

4

2c



3

2d 2486; 116; 5485; 73680; 833271; 66013; 3257

3

Postorder: Linker Teilbaum – rechter Teilbaum - Wurzel

2e $\log_2(250000+1) \approx 17{,}9$, also minimal 18 Ebenen.

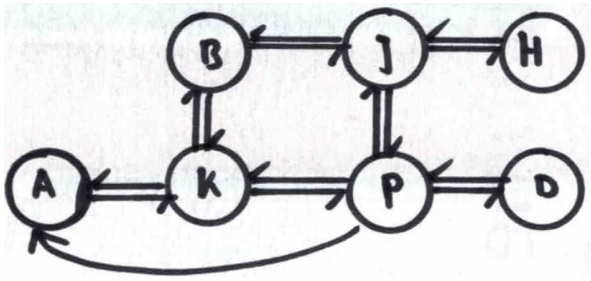
3

Eine analytische Berechnung ist hier nicht erforderlich. Alternativ: $2^n - 1 \geq 250000$ *mit* $2^{18} = 262144 > 250001$ *und* $2^{17} = 131072 < 250000$ *, also werden 18 Ebenen benötigt.*

2f
```
class BAUM{
   private BAUMELEMENT wurzel;
   void mitgliederAusgeben(String vorname, String nachname){
        wurzel.mitgliederAusgeben(vorname, nachname);
   }
}

abstract class BAUMELEMENT{
   abstract void mitgliederAusgeben(String vorname, String nachname);
}

class KNOTEN extends BAUMELEMENT{
   private BAUMELEMENT linkerNachfolger;
   private BAUMELEMENT rechterNachfolger;
   private MITGLIED daten;

   void  mitgliederAusgeben(String vorname, String nachname){
        linkerNachfolger.mitgliederAusgeben(vorname, nachname);
        if(daten.istVornameGleich(nachname) &&
               daten.istNachnameGleich(vorname)){
           daten.alleDatenAusgeben();
        }
         rechterNachfolger.mitgliederAusgeben(vorname, nachname);
   }
}

class ABSCHLUSS extends BAUMELEMENT{
    void mitgliederAusgeben(String vorname, String nachname){ }
}
```
12

2g
```
SELECT *
FROM mitglieder
WHERE vorname = “Laura“ AND nachname = „Meier“;
```
3

3a



4

Begründung: In einem ungerichteten Graphen kann man nicht erkennen, wenn eine Freundschaftsanfrage unbestätigt ist

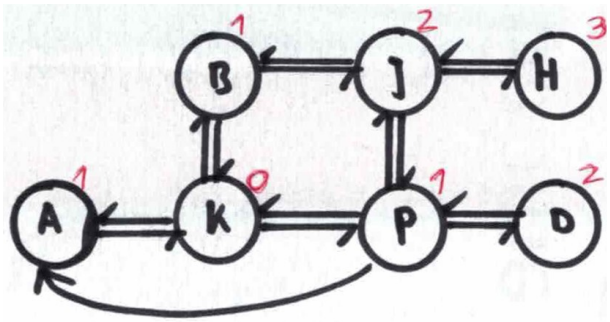
3b

	A	B	D	H	J	K	P
A						1	
B					1	1	
D							1
H					1		
J		1		1			1
K	1	1					1
P	1		1		1	1	

3

3c Für einen gegebenen Knoten kn wird gezählt, zu wie vielen Nachbarknoten eine „Freundschaft“, d. h. eine beidseitige Beziehung, besteht. Zurückgegeben wird die Anzahl der Freunde des zum Knoten gehörigen Mitglieds. 5

3d



2

Idee: Der Startknoten bekommt das Gewicht 0, alle Nachbarn das Gewicht 1, deren Nachbarn 2 usw. rekursiv weiter bis alle Knoten ein Gewicht haben.

3e Methode *freundeVonFreunden(kn):*

8

```
i = KnotennummerGeben(kn)
wiederhole für j von 0 bis n – 1
    wenn adjmatrix[j][i] == 1 und adjmatrix[i][j] == 1
        wiederhole für k von 0 bis n-1
            wenn (adjmatrix[k][j] == 1 und adjmatrix[j][k] == 1)
              und (adjmatrix[k][i] == 0 oder adjmatrix[i][k] == 0)
              und k != i
                Ausgabe von k
            endeWenn
        endeWiederhole
    endeWenn
endeWiederhole
```

80