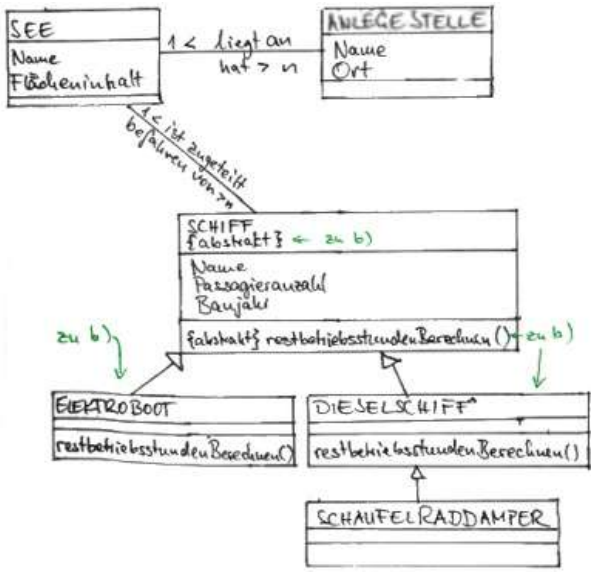
Informatik Abitur Bayern 2018 / II - Lösung

Autor:
Reinold

1a



5

1b siehe 1a

5

1c

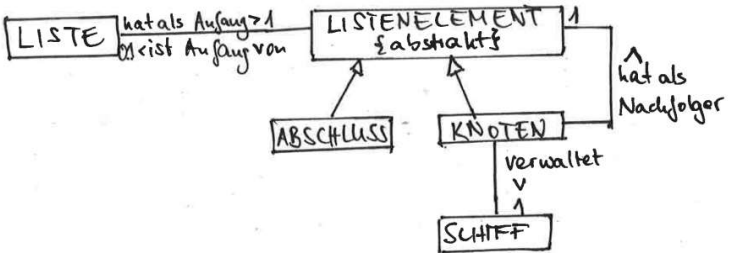
```
public int anzahlTage(DATUM saisonstart, DATUM saisonende)
{
  if(this.saisonende.istVor(this.wartungstermin)) //volle Saison
  {
   return this.saisonstart.dauer(this.saisonende);
  }
  else
  {
   if(this.wartungstermin.istVor(this.saisonstart)) // Wartung startet
zuvor
   {
    return 0;
   }
   else
   {
    return this.saisonstart.dauer(this.wartungstermin)-1;
   }
  }
}
```

5

1d Softwaremuster sind bewährte Standardverfahren zur Lösung häufig auftretender, vergleichbarer Problemstellungen der Softwareentwicklung. Die Zurückführung auf Standardverfahren erleichtert die Softwareentwicklung, da das Grundmodell nicht neu entwickelt werden muss.

3

1e



Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.

1f *Klasse LISTE* 14

```
public int anzahlSchiffeBestimmen()
{
 return this.anfang.anzahlSchiffeBestimmen();
}

void einfuegen(SCHIFF s)
{
  this.anfang=this.anfang.einfuegen(s);
}
```

Klasse LISTENELEMENT

```
public abstract int anzahlSchiffeBestimmen();
public abstract KNOTEN einfuegen(SCHIFF s);
```

Klasse KNOTEN

```
public KNOTEN(SCHIFF sNeu, LISTENELEMENT nfNeu)
{
  this.schiff = sNeu;
  this.nachfolger = nfNeu;
}

public int anzahlSchiffeBestimmen()
{
  return 1+this.nachfolger.anzahlSchiffeBestimmen();
}

public KNOTEN einfuegen(SCHIFF sNeu)
{
  if(schiff.WartungsterminGeben().istVor(sNeu.WartungsterminGeben())
  {
    this.nachfolger=this.nachfolger.einfuegen(sNeu);
    return this;
  }
  else
  {
    return new KNOTEN(sNeu, this);
  }
}
```

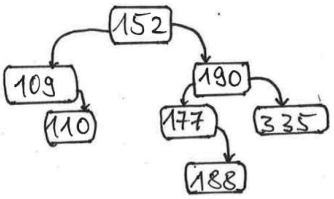
Klasse ABSCHLUSS

```
public int anzahlSchiffeBestimmen()
{
  return 0;
}
public KNOTEN einfuegen(SCHIFF sNeu)
{
  return new KNOTEN(s, this);
}
```

2a 3

```
SELECT persNr, name, vorname
FROM mitarbeiter
WHERE dienstort="Königssee"
```

2b 6



Postorder-Ausgabe (erst liNf, dann reNf, dann Wurzel): 110, 109, 188, 177, 335, 190, 152

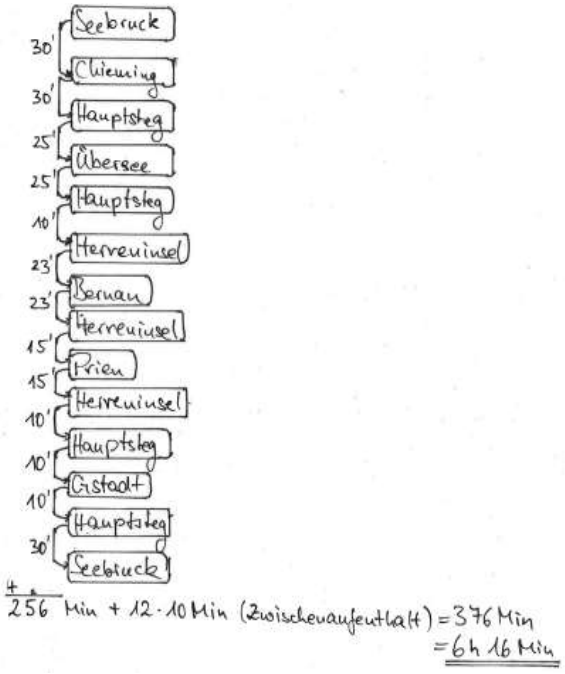
Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.

2c Alle nachfolgenden Knoten werden immer rechts eingeordnet, wodurch der Baum zur Liste entartet. Um diese Entartung zu vermeiden, kann z. B. der jeweils nächste einzufügende Datensatz zufällig ausgewählt werden. 4

3a Der Graph ist gerichtet, gewichtet, zyklisch und zusammenhängend 5

	Chieming	Seebruck	Hauptsteg	Übersee	Herreninsel	Bernau	Prien	Gstadt	Nordsteg
Chieming			30						
Seebruck	30								
Hauptsteg		30		25	10			10	
Übersee			25						
Herreninsel			10			23	15		
Bernau					23				
Prien					15				
Gstadt			10						5
Nordsteg								5	

3b 4



3c 6

```
class LINIENNETZ
{
  private KNOTEN[] knotenfeld;
  private int[][] matrix;

  LINIENNETZ(int maxAnzahlKnoten)
  {
    this.knotenfeld = new KNOTEN[maxAnzahlKnoten];
    this.matrix=new int[maxAnzahlKnoten][maxAnzahlKnoten];
    for(int i=0; i<maxAnzahlKnoten; i=i+1)
     for(int j=0; j<maxAnzahlKnoten; j=j+1)//Klammerung bei Einzeiler kann
                                           //entfallen
       this.matrix[i][j]=-1;
  }
}
```

Ein Attribut aktuelleAnzahlKnoten kann im speziellen Fall entfallen, da man diese aus

Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.

knotenfeld ermitteln kann. Das dies allerdings nicht vorteilhaft ist, zeigt sich bei 3d und 3e.

3d
```
public int indexSuchen(KNOTEN k)
{
  for(int i=0; i<this.knotenfeld.length; i=i+1)
  {
    if(this.knotenfeld[i]==null) return -1;
    else if(this.knotenfeld[i]==k) return i;
  }
  return -1;
}
```
5

3e
```
class KNOTEN
{
  private boolean istBesucht;

  KNOTEN(){ this.istBesucht = false;}

  public boolean istBesuchtGeben(){return this.istBesucht;}
  public boolean istBesuchtSetzen(boolean b){this.istBesucht = b;}
}

class LINIENNETZ
{
  public void traversieren(KNOTEN start)
  {
    for(int i=0; i<knotenfeld.length; i=i+1) //Initialisierung
    {
      this.knotenfeld[i].istBesuchtSetzen(false);
    }
    this.Tiefensuche(indexSuchen(start));
  }

  public void Tiefensuche(int knotenNummer)
  {
   this.knotenfeld[knotenNummer].besuchtSetzen(true);
   System.out.println(this.knotenfeld[knotenNummer].NameGeben()
                                                    +“ wird besucht“;
   for(int i=0; i<this.knotenfeld.length;i=i+1)
   {
    if(this.knotenfeld[i]==null) return;
    else
    {
     if(this.matrix[knotenNummer][i]!=-1 && !this.knoten[i].besuchtGeben())
     {
       this.Tiefensuche(i);
     }
    }
   }
}
```
10

Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.