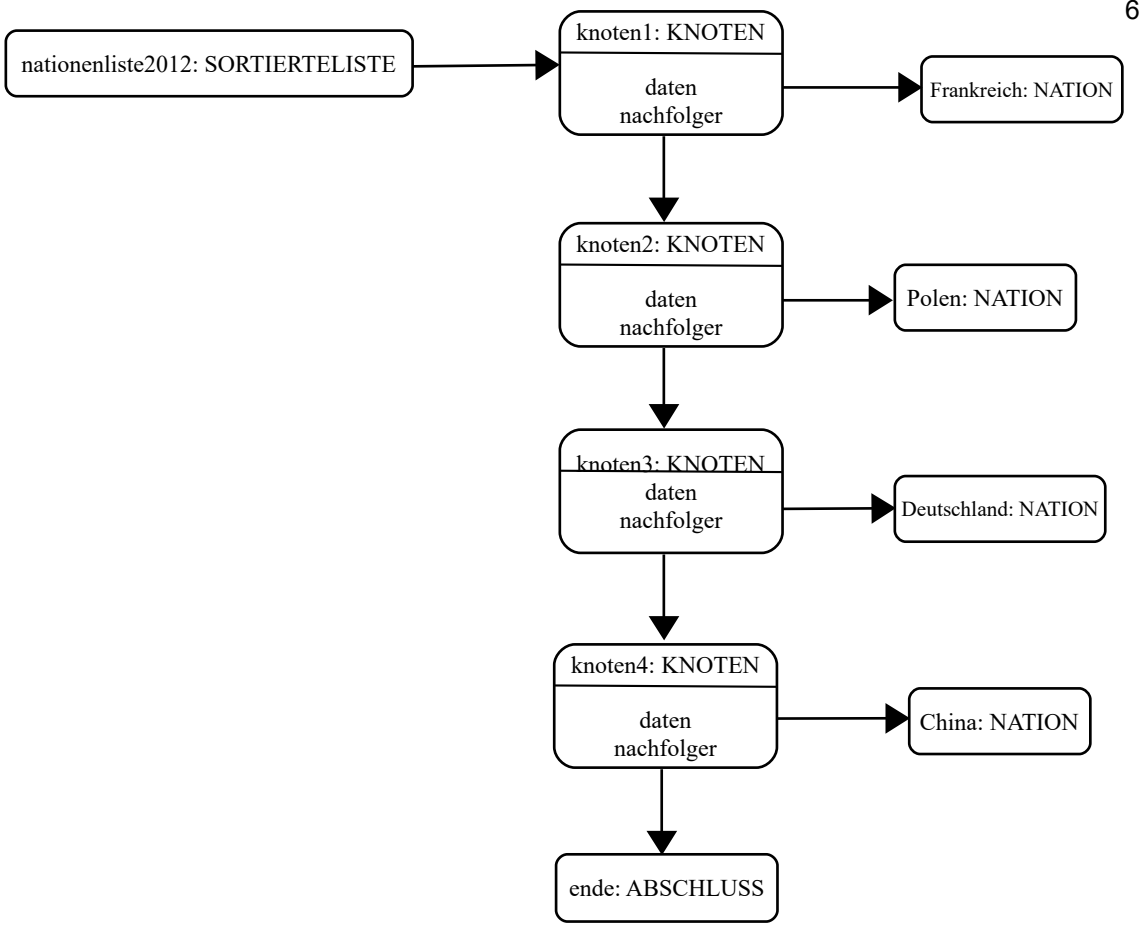
Informatik Abitur Bayern 2012 / I - Beispiellösung

Autor:
Jetzinger

1a 6



1b 20

```
public class SORTIERTELISTE
{
  private LISTENELEMENT anfang;

  public int anzahlAllerAthletenGeben()
  {
    return anfang.anzahlAllerAthletenGeben();
  }

  public void einfuegen(NATION nationNeu)
  {
    anfang = anfang.Einfuegen(nationNeu);
  }
}


public abstract class LISTENELEMENT
{
  public abstract int anzahlAllerAthletenGeben();

  public abstract KNOTEN Einfuegen(NATION nationNeu);
}
```

```
public class KNOTEN extends LISTENELEMENT
{
  private LISTENELEMENT nachfolger;
  private NATION nation;

  public KNOTEN(NATION nationNeu, LISTENELEMENT nachfolgerNeu)
  {
    nation = nationNeu;
    nachfolger = nachfolgerNeu;
  }

  public int AnzahlAllerAthletenGeben()
  {
    return nation.AnzahlAthletenGeben() +
       nachfolger.AnzahlAllerAthletenGeben();
  }

  public KNOTEN Einfuegen(NATION nationNeu)
  {
    if(nation.istKleiner(nationNeu))
    {
       return new KNOTEN(nationNeu, this);
    }
    else
    {
      nachfolger = nachfolger.Einfuegen(nationNeu);
      return this;
    }
}


public class ABSCHLUSS extends LISTENELEMENT
{
  public int AnzahlAllerAthletenGeben()
  {
    return 0;
  }

  public KNOTEN Einfuegen(NATION nationNeu)
  {
    return new KNOTEN(nationNeu, this);
  }
}


public class NATION
{
  private int anzahlAthleten;



  public int AnzahlAthletenGeben()
  {
    return anzahlAthleten;
  }
}
```

1c Man führt zwei weitere Klassen GRIECHENLAND und GASTGEBER ein, die von der Klasse KNOTEN erben und sich ähnlich wie der Abschluss anders verhalten. Das Attribut nation in der Klasse KNOTEN sollte dann auf protected gesetzt werden. 6

```
public class GRIECHENLAND extends KNOTEN
{
   public GRIECHENLAND(NATION nationNeu, KNOTEN nachfolger)
   {
     nation = nationNeu;
     nachfolger = nachfolgerNeu;
   }

   public KNOTEN Einfuegen(NATION nationNeu)
   {
      nachfolger = nachfolger.Einfuegen(nationNeu);
      return this;
   }
}

public class GASTGEBER extends KNOTEN
{
  public GASTGEBER(NATION gastgeberland)
  {
    super(gastgeberland, new ABSCHLUSS());
  }

  public KNOTEN Einfuegen(NATION nationNeu)
  {
      return new KNOTEN(nationNeu, this);
  }
}
```

Die Liste verwaltet dann wie eine Warteschlange einen Knoten anfang und einen Knoten ende, die eine Referenz auf ein Objekt der Klasse GRIECHENLAND bzw. GASTGEBER verwalten:

```
public class SORTIERTELISTE
{
  KNOTEN ende;
  KNOTEN anfang;

  public SORTIERTELISTE(NATION gastgeber, NATION griechenland)
  {
    ende = new GASTGEBER(gastgeber);
    anfang = new GRIECHENLAND(griechenland, ende);
  }
}
```

Man könnte anstatt der beiden neuen Klassen auch die Methode istKleiner() in der Klasse NATION anpassen, so dass diese Griechenland und das Gastgeberland als Sonderfälle behandelt.

Alternativ wäre es möglich die Nationen Griechenland und das Gastgeberland als lokale Referenzen in der Liste zu speichern. Beim Ausgeben der Anzahl der Athleten müsste dann in der Liste die Anzahl der beiden Länder addiert werden. Wie oben werden Griechenland und das Gastgeberland am besten direkt bei der Erstellung der Liste im Konstruktor übergeben. Hier wären also nur folgende Änderung in der Klasse SORTIERTELISTE nötig:

```
public class SORTIERTELISTE
{
  private KNOTEN anfang;
  private NATION griechenland;
  private NATION gastgeberland;

  public SORTIERTELISTE(NATION griechenland, NATION gastgeberland)
  {
```

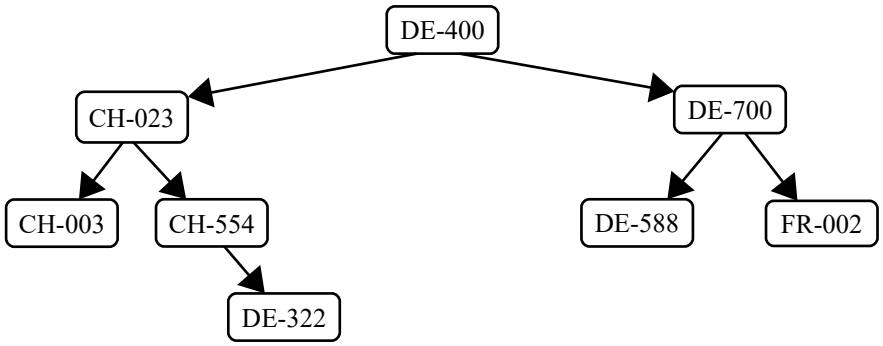
```
    this.griechenland = griechenland;
    this.gastgeberland = gastgeberland;
  }

  public int anzahlAllerAthletenGeben()
  {
    return anfang.anzahlAllerAthletenGeben() +
           griechenland.AnzahlAllerAthletenGeben() +
           gastgeberland.AnzahlAllerAthletenGeben();
  }
```

2
```
public boolean istBesser(NATION nation)
{
  return
         medaillen[0] > nation.MedaillenGeben()[0] ||
         medaillen[0] == nation.MedaillenGeben()[0] &&
              medaillen[1] > nation.MedaillenGeben()[1] ||
         medaillen[0] == nation.MedaillenGeben()[0] &&
         medaillen[1] == nation.MedaillenGeben()[1]  &&
         medaillen[2] > nation.MedaillenGeben()[2];
}
```
6

Hinweis: Wegen Vorrang des &&-Operators vor dem ||-Operator kann die Klammerung entfallen.

3a 7



mögliche Einfügereihenfolge: DH-400, CH-023, CH-003, CH-554, DE-322, DE-700, DE-588, FR-002

3b Ein balancierter Baum mit n Knoten hat $\log_2 n$ Ebenen. Das heißt, es sind maximal $\log_2(12000) = 13{,}551$ Vergleiche nötig. Ein Vergleich dauert 20 Nanosekunden, d.h. es dauert maximal $13{,}551 \cdot 20 \cdot 10^{-9} s = 2{,}7102 \cdot 10^{-7} s \approx 2{,}7 \cdot 10^{-7} s$
Im ungünstigsten Fall sind 12000 Vergleiche notwendig. Die Zeit errechnet sich daher wie folgt: $12000 \cdot 20 \cdot 10^{-9} s = 2{,}4 \cdot 10^{-4} s$ 6

Die beiden Zeiten unterscheiden sich um einen Faktor von 10^{-3}. Bei einer einzelnen Anfrage wird der Benutzer den Unterschied kaum merken. Werden jedoch viele Anfragen gestellt, macht er sich bemerkbar. 12500 Abfragen dauern im schlechtesten Fall drei Sekunden, im besten Fall dauern sie nur 0.003375s.

3c Die Methode gibt alle deutschen Athleten mit ihrer Sportart in der Form [Name: Sportart] aus. Die Ausgabe erfolgt in Inorder-Reihenfolge, also sortiert nach dem Schlüssel. 6

3d
```
SELECT Name, Sportart
FROM athlet
WHERE Nation = "Deutschland";
```
3

4a 4

4b 4

	A	B	C	D	E	F	G	H	I
A		3							
B	3						3		
C				6		6			
D			6		4				
E				4					
F			6						
G		3							3
H									5
I							3	5	

4c Der Graph ist nicht zusammenhängend und kann somit nicht als Baum dargestellt werden. 2

5 10