Einführung

Die Elemente einer sortierten Liste sind ähnlich angeordnet wie in einem Array. Allerdings ergeben sich bei Strukturierung mit einem Array verschiedene Probleme:

- Beim Einfügen eines Elementes müssen alle nachfolgenden Elemente um eine Stelle verschoben werden.
- Beim Löschen eines Elementes muss die Datenstruktur umgekehrt entsprechend angepasst werden.
- Die Liste kann in einem Array nicht dynamisch wachsen und schrumpfen.

Die einfachste Form einer sortierten dynamischen Liste besteht aus Knoten mit Inhalt und einem Verweis auf den nachfolgenden Knoten.
Diese Strukturierung ist durch Verweise auf den jeweiligen Vorgänger erweiterbar. Dann erhält man eine *doppelt verkettete Liste.*

Einfügen

Beim Einfügen in eine sortierte Liste ergeben sich verschiedene Fälle, die beachtet werden müssen:

1. Es wird in eine leere Liste eingefügt.
2. Es wird vor dem ersten Element der Liste eingefügt.
3. Es wird zwischen zwei Elementen eingefügt.
4. Es wird am Ende der Liste eingefügt.

Zeichne für jeden der vier Fälle eine Graphik mit Zeigern, die die Operationen der im Anhang angegebenen Methode `insert` darstellt. Zeige damit die Korrektheit der Methode für alle Fälle.

Löschen

Beim Löschen in einer sortierten Liste ergeben sich verschiedene Fälle, die beachtet werden müssen:

1. Es wird versucht, in einer leeren Liste zu löschen.
2. Es wird das erste Element der Liste gelöscht.
3. Es wird zwischen zwei Elementen gelöscht.

4. Es wird am Ende der Liste gelöscht.

5. Es wird versucht, ein nicht vorhandenes Element zu löschen.

Zeichne für jeden der fünf Fälle eine Graphik mit Zeigern, die die Operationen der im Anhang angegebenen Methode `delete` darstellt. Zeige damit die Korrektheit der Methode für alle Fälle.

Suchen

Schreibe eine Methode, die feststellt, ob ein bestimmtes Element in der Liste vorhanden ist.

Aufgabe

Schreibe geeignete Klassen für eine Telephonnummernliste (mit Namen). Informiere dich dazu über geeignete Vergleichsoperationen für Strings und schreibe eine Methode, die als booleschen Funktionswert angibt, ob ein String kleiner als ein anderer ist oder nicht. Schreibe ein Hauptprogramm zum Testen der Klassen mit einer Menüstruktur zum Einfügen, Löschen und Ansehen der Einträge.

Anhang

```
1  class ZahlenListenTest
2  { ZahlenListe liste;
3
4    public ZahlenListenTest()
5    { liste = new ZahlenListe();
6      action();
7    }
8
9    public void action()
10   { liste.gibAllesAus();
11     liste.insert(3);     liste.gibAllesAus();
12     liste.insert(7);     liste.gibAllesAus();
13     liste.insert(12);    liste.gibAllesAus();
14     liste.insert(2);     liste.gibAllesAus();
15     liste.insert(8);     liste.gibAllesAus();
16     liste.insert(13);    liste.gibAllesAus(); // jetzt loeschen
17     liste.delete(7);     liste.gibAllesAus();
18     liste.delete(2);     liste.gibAllesAus();
19     liste.delete(3);     liste.gibAllesAus();
20     liste.delete(12);    liste.gibAllesAus();
21     liste.delete(12);    liste.gibAllesAus();
22     liste.delete(13);    liste.gibAllesAus();
23     liste.delete(8);     liste.gibAllesAus();
24   }
25 } // class ZahlenListenTest
```

```
1  public class Knoten
2  {
3    private int wert; // Stark vereinfachterr Knoten!
4                      // hier werden spaeter statt Zahlen
5                      // beliebige Objekte verwaltet
6
7    private Knoten nachfolger;
8
9    public Knoten (int zahl, Knoten next)
10   { wert = zahl;
11     nachfolger = next;
12   }
13
14   public void setNachfolger(Knoten next)
15   {  nachfolger = next; }
16
17   public Knoten getNachfolger()
18   {  return nachfolger; }
19
20   public void setWert(int zahl)
21   {  wert = zahl; }
22
23   public int getWert()
24   {  return wert; }
25
26 } // class Knoten
```

```
1  public class ZahlenListe
2  { private Knoten erster;
3
4    public ZahlenListe()
5    { erster = null;  }
6
7    public void insert (int zahl)
8    { Knoten lauf   = erster;
9      Knoten vorher = null;
10     while (lauf!=null && zahl>lauf.getWert())
11     {  vorher = lauf;
12        lauf   = lauf.getNachfolger();
13     }
14     // Einfuegestelle oder Listenende gefunden
15     Knoten neu = new Knoten(zahl, null);
16     neu.setNachfolger(lauf);       // nach vorne "verknoten"
17     if (lauf==erster)
18          erster = neu;
19     else vorher.setNachfolger(neu); // von hinten "verknoten"
20   } // insert
21
22   public void delete (int zahl)
23   { Knoten lauf   = erster;
24     Knoten vorher = null;
25     while (lauf!=null && zahl!=lauf.getWert()) // kurze Auswertung
26     {  vorher = lauf;
27        lauf   = lauf.getNachfolger();
28     }
29     // Ende der Schleife, wenn zahl=lauf.getWert() oder lauf==null
30     if (lauf!=null)
31          if (lauf==erster)
32               erster = lauf.getNachfolger();
33          else vorher.setNachfolger(lauf.getNachfolger());
34   } // delete
35
36   public void gibAllesAus()
37   { Knoten lauf = erster;
38     if (lauf==null)
39          { System.out.println("Liste␣ist␣leer.");
40            return;
41          }
42     while (lauf!=null)
43     {  System.out.print (lauf.getWert() + "␣␣");
44        lauf = lauf.getNachfolger();
45     }
46     System.out.println();
47   } // gibAllesAus
48
49 } // class ZahlenListe
```