Aufgabenblatt

Verkettete Objekte

Programmierung Praktikum
Prof. Dr. Dirk Eisenbiegler
Hochschule Furtwangen

Aufgabe 1 - Einfach verkettete Liste

Die Klasse Liste realisiert eine Liste von int-Werten.

```
public class Liste {

  public int element;
  public Liste nachfolger;

  public Liste (int w) {
     element = w;
     nachfolger = null;
  }

  public void hinzufuegen (int w) {
     if (nachfolger == null)
        nachfolger = new Liste(w);
     else
        nachfolger.hinzufuegen(w);
  }
}
```

```
public class ListeTester {
  public static void main(String[] args) {
     Liste x = new Liste(3);
     x.hinzufuegen(5);
     x.hinzufuegen(4);
     x.hinzufuegen(7);
  }
}
```

A) Tippen Sie die beiden Klassen ab und gehen Sie mit dem Debugger Zeile für Zeile durch die main-Methode von ListeTester.

B) Fügen Sie zur Klasse *Liste* die Methode *laenge* hinzu, die die Länge der Liste bestimmt. Die Methode *laenge* ist eine Objektmethode. Sie hat keinen Parameter und einen Rückgabewert vom Typ int.

C) Fügen Sie zur Klasse *Liste* die Methode *entfernen* hinzu, die das letzte Element der Liste entfernt. Die Methode *entfernen* ist eine Objektmethode. Sie hat keinen Parameter und keinen Rückgabewert (void). Enthält die Liste nur ein einziges Element, so soll die Methode *entfernen* keine Änderung vornehmen.

D) Fügen Sie zur Klasse Liste die Methode *toString* hinzu, die die Liste in Form eines Strings zurückgibt. In dem String sollen die Elemente

durch Kommata getrennt aufgelistet werden. Die Methode *toString* ist eine Objektmethode. Sie hat keinen Parameter und einen Rückgabewert vom Typ String.

E) Fügen Sie zur Klasse Liste die Methode *summe* hinzu, die von einer Liste die Summe der Werte zurückgibt. Die Methode *summe* hat keinen Parameter und einen Rückgabewert vom Typ int.

F) Fügen Sie zur Klasse Liste die Objektmethode *addiere* hinzu, die zu jedem Element der Liste einen vorgegebenen Wert hinzuaddiert. Die Methode *addiere* hat einen Parameter vom Typ *int* und keinen Rückgabewert.

Aufgabe 2 - *Binärer Suchbaum*

In dieser Aufgabe soll in Java ein Graph realisiert werden: ein binärer Suchbaum.

Binärer Suchbaum:

- ✗ Jeder Knoten enthält eine andere Zahl.
- ✗ Für jeden Knoten gilt:
  Die Zahlen im linken Teilbaum sind kleiner als der Wert im Knoten, und die Zahlen im rechten Teilbaum sind größer als der Wert im Knoten.

Implementierungsidee:
Die Knoten des Baums sind Instanzen einer Klasse mit dem Namen Knoten. Ein Baum ist somit eine Menge von Knoten-Objekten. Den Kanten zwischen den Knoten entsprechen Referenzen: Eine Kante von Knoten a zu Knoten b wird durch ein Attribut in a realisiert, in der die Objektreferenz auf b gespeichert wird. Mit anderen Worten: Ein Attribut von a zeigt nach b.
Eine Klasse mit dem Namen Baum gibt es nicht explizit. Jeder Knoten steht für einen Baum: der Baum, dessen Wurzel er ist.

A) Implementieren Sie eine Klasse mit dem Namen *Knoten*. Die Klasse *Knoten* soll ein Attribut *x* vom Typ int haben, in dem die Zahl gespeichert wird. Ferner soll die Klasse Knoten die beiden Attribute *links* und *rechts* vom Typ Knoten haben. *links* und *rechts* sind Referenzen auf die Nachbarknoten im linken bzw. rechten Teilbaum. Existiert kein solcher Teilbaum, so sollen deren Werte *null* sein.

B) Implementieren Sie einen Konstruktor für die Klasse *Knoten*. Der Konstruktor habe einen Parameter vom Typ int - die Zahl, die in dem Knoten gespeichert werden soll. Der Konstruktor soll die Attribute *links* und *rechts* auf null setzen. Es entsteht ein binärer Suchbaum mit einem Knoten.

C) Implementieren Sie in der Klasse *Knoten* die Objektmethode *einfuegen*. Die Methode *einfuegen* habe genau einen Parameter vom Typ int. Sie fügt ein neues Element in den Baum ein, dessen Wurzel der Knoten ist. Sollte das Element bereits im Baum enthalten sein, so soll die Methode keine Wirkung haben.

D) Implementieren Sie in der Klasse *Knoten* die Objektmethode *toString*.

Die Methode *toString* habe keinen Parameter. Sie soll den Baum, dessen Wurzel der Knoten ist, in einen String umwandeln. In dem String sollen jeweils der linke Teilbaum, der Zahlenwert und der rechte Teilbaum durch runde Klammern umschlossen werden.
Beispiele:

(3)	Baum mit einem Knoten dessen Wert 3 ist.
((1) 3)	Baum, dessen Wurzel den Wert 3 hat und dessen linker Teilbaum den Wert 1 hat.
(((1) 2 (5)) 8 (12))	größerer Baum mit den Werten 1, 2, 5, 8 und 12

E) Implementieren Sie in der Klasse *Knoten* die Objektmethode *suchen*.
Die Methode *suchen* habe genau einen Parameter vom Typ *int* und einen Rückgabewert vom Typ *boolean*. Sie soll prüfen, ob der angegebene Wert im Baum enthalten ist, dessen Wurzel der Knoten ist.
Rückgabewert true: das Element ist enthalten.
Rückgabewert false: das Element ist nicht enthalten.