Informatik mit Java — Bäume: Einfügen und Suchen

Gierhardt

Im Unterricht wurden die folgenden Klassen entwickelt:

```
1  public class Knoten
2  { int inhalt;
3    Knoten links, rechts;
4
5    public Knoten (int wert)
6    { this.inhalt = wert;
7      this.links = null;
8      this.rechts = null;
9    }
10 }
```

```
1  public class BinaerBaum
2  { Knoten wurzel;
3
4    public BinaerBaum()
5    {  wurzel = null;
6    }
7
8    public void rekFuegeEin (Knoten ast, int wert)
9    { if (wert < ast.inhalt)
10           { // links einfuegen
11             if (ast.links == null)
12                  ast.links = new Knoten(wert);
13             else rekFuegeEin(ast.links, wert);
14           }
15     else { // rechts einfuegen
16             if (ast.rechts == null)
17                  ast.rechts = new Knoten(wert);
18             else rekFuegeEin(ast.rechts, wert);
19           }
20   } // rekFuegeEin
21
22   public void fuegeEin (int wert)
23   { if (wurzel == null)
24           wurzel = new Knoten(wert);
25     else rekFuegeEin(wurzel, wert);
26   } // FuegeEin
27
28   public void rekLaufeDurch (Knoten ast)
29   { if (ast != null)
30           { rekLaufeDurch (ast.links);
31             Out.print(ast.inhalt); Out.println();
32             rekLaufeDurch (ast.rechts);
33           }
34   } // rekLaufeDurch
35
36   public void laufeDurch ()
37   { if (wurzel != null)
38           { rekLaufeDurch (wurzel.links);
39             Out.print(wurzel.inhalt); Out.println();
40             rekLaufeDurch (wurzel.rechts);
41           }
42   } // laufeDurch
43 } // class BinaerBaum
```

Ein dazugehöriges Hauptprogramm könnte so aussehen:

```
1  public class BinaerBaumBeispiel
2  { BinaerBaum baum;
3
4    public BinaerBaumBeispiel()
5    { baum = new BinaerBaum();
6    }
7
8    public void action()
9    { int zahl = 0;
10     Out.print("Bitte␣Zahlen␣eingeben.␣Ende␣mit␣-1.");
11     Out.println();
12
13     while (zahl != -1)
14     {     Out.print("Bitte␣Zahl␣eingeben:␣");
15           zahl = In.readInt();
16           if (zahl != -1)
17                baum.fuegeEin(zahl);
18     } // while
19     baum.laufeDurch();
20   }
21 } // class BinaerBaumBeispiel
```

Aufgaben:

1. In einer Klasse können zwei Methoden den gleichen Namen haben, wenn sie sich durch die Parameterliste unterscheiden. Dies bezeichnet man als *Überladen*. Die Auswahl der tatsächlich aufzurufenden Methode erfolgt zur Laufzeit anhand des Typs der aktuellen Parameter. Der Ergebnistyp muss allerdings immer gleich bleiben. Das dem Überladen übergeordnete Konzept der Objektorientierung ist die *Polymorphie*, was so viel wie „Vielgestaltigkeit“ heißt. Ein weiteres Beispiel für Polymorphie ist die Verwendung des +-Operators, der bei Zahlen die Addition und bei Strings die *Konkatenation* bewirkt.

   Wende das Überladen bei den Methoden der Klasse `BinaerBaum` an.

2. Die Methode `laufeDurch()` ist zu umständlich formuliert. Vereinfache sie.

3. Der Baum soll nach bestimmten Inhalten durchsucht werden. Formuliere dazu eine Funktion `gefunden(int zahl)` und teste die Methode.

   Tipp: Überladen und natürlich Rekursion machen die Lösung einfach.

4. Die Klasse `BinaerBaum` wird um drei Konstanten ergänzt und die Methode `laufeDurch` erhält einen weiteren Parameter. Formuliere damit die Methoden um, damit der Baum auf verschiedene Arten durchlaufen werden kann. Teste die verschiedenen Arten aus.

```
1  class BinaerBaum // ------------------------------------
2  { Knoten wurzel = null;
3
4    final int INORDER   = 1;
5    final int PREORDER  = 2;
6    final int POSTORDER = 3;
7    ...
8    void laufeDurch (int order)
9    ...
10 } // class BinaerBaum
```