Im Unterricht wurden die folgenden Klassen entwickelt:

```
class Knoten  // ------------------------------------------
{  int inhalt;
   Knoten links, rechts;
   Knoten (int wert)
   { this.inhalt = wert;  this.links = null;  this.rechts = null; }
} // class Knoten

class BinaerBaum // ---------------------------------------
{ Knoten wurzel = null;

  void rekFuegeEin (Knoten ast, int wert)
  { if (wert < ast.inhalt)
         { if (ast.links == null) ast.links = new Knoten(wert);
           else rekFuegeEin(ast.links, wert);
         }
    else { if (ast.rechts == null) ast.rechts = new Knoten(wert);
           else rekFuegeEin(ast.rechts, wert);
         }
  } // rekFuegeEin

  void fuegeEin (int wert)
  { if (wurzel == null) wurzel = new Knoten(wert);
    else rekFuegeEin(wurzel, wert);
  } // fuegeEin

  void rekLaufeDurch (Knoten ast)
  { if (ast != null)
         { rekLaufeDurch (ast.links);
           Out.print(ast.inhalt); Out.println();
           rekLaufeDurch (ast.rechts);
         }
  } // rekLaufeDurch

  void laufeDurch ()
  { if (wurzel != null)
         { rekLaufeDurch (wurzel.links);
           Out.print(wurzel.inhalt); Out.println();
           rekLaufeDurch (wurzel.rechts);
         }
  } // laufeDurch
} // class BinaerBaum
```

Eine Main-Klasse kann so aussehen:

```
class BinBaumBsp0
{ static BinaerBaum baum;

  public static void main(String[] arg)
  { int zahl = 0;
    String muell;
    baum = new BinaerBaum();
    Out.print("Bitte Zahlen eingeben. Ende mit -1."); Out.println();
    while (zahl != -1)
    { Out.print("Bitte Zahl eingeben: ");
      zahl = In.readInt();    muell = In.readLine();
      if (zahl != -1) baum.fuegeEin(zahl);
    } // while
    baum.laufeDurch();
  } // main
} // class BinBaumBsp0
```

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.

```
class BinaerBaum // ------------------------------------
{ Knoten wurzel = null;

  final int INORDER   = 1;
  final int PREORDER  = 2;
  final int POSTORDER = 3;
  ...
  void laufeDurch (int order)
  ...
} // class BinaerBaum
```