Aufgabe 1

Algorithmus: Kontakt im Suchbaum finden anhand eines Nachnamens

Setze Referenz „knoten“ auf die Wurzel des Suchbaums.

Wiederhole, bis knoten leer ist:
    Hole Referenz „kontakt“ von knoten.
    Falls der gesuchte Nachnamen mit Nachname von kontakt übereinstimmt,
        Gib kontakt zurück und beende die Methode.
    Sonst, falls der gesuchte Nachname alphabetisch VOR kontakt liegt,
        setze knoten auf seinen linken Nachfolger.
    Sonst
        setze knoten auf seinen rechter Nachfolger.

Nach der Schleife gib null zurück (Name nicht gefunden).

Implementierung:

```
public Kontakt suchen(String pname)
{
   BinaryTree<Kontakt> knoten;
   Kontakt kAkt;

   knoten = wurzel;
   while (!knoten.isEmpty())
   {
      kAkt = knoten.getContent();
      if (pname.equals(kAkt.getNachname()))
      {
          return kAkt;
      }
      else if (pname.compareTo(kAkt.getNachname()) < 0)
      {
         knoten = knoten.getLeftTree();
      }
      else
      {
         knoten = knoten.getRightTree();
      }
   }
   return null;
}
```

Aufgabe 2

Algorithmus: Kontakt in Suchbaum einfügen

Setze Referenz „knoten“ auf die Wurzel des Suchbaums.

Wiederhole, bis knoten leer ist:
    Hole Referenz „kontakt“ von knoten.
    Falls der gesuchte Nachnamen mit Nachname von kontakt übereinstimmt,
        Breche die Methode ab (gleicher Nachname nicht vorgesehen).
    Sonst, falls der neue Nachname alphabetisch VOR kontakt liegt,
        setze knoten auf seinen linken Nachfolger.
    Sonst
        setze knoten auf seinen rechter Nachfolger.

Nach der Schleife (leerer Knoten gefunden) setze neuen Kontakt als Inhalt dieses Knoten.

Implementierung:

```
public void einfügen(Kontakt pk)
{
   BinaryTree<Kontakt> knoten;
   Kontakt kAkt;

   knoten = wurzel;
   while (!knoten.isEmpty())
   {
      kAkt = knoten.getContent();
      // Sonderfall: Kontakt bereits enthalten
      if (pk.getNachname().equals(kAkt.getNachname()))
      {
         return;
      }
      if (pk.getNachname().compareTo(kAkt.getNachname()) < 0)
      {
         knoten = knoten.getLeftTree();
      }
      else
      {
         knoten = knoten.getRightTree();
      }
   }
   knoten.setContent(pk);
}
```