Lösung - Binäre Suche

Aufgabe 4:

Lade dir die Vorlage zur Aufgabe 4 vom Server und fülle die markierten Stellen in der Datei **`MyBibliothek.java`** so, dass eine rekursive Methode zur binären Suche entsteht.

Iterative Version der binären Suche:

```
public static int binaereSuche(int[] feld, int start, int stopp, int wert) {

  if (feld!=null && feld.length > 0){

    do {
      int mitte = (start + stopp) / 2;

      if (feld[mitte] == wert) {
        return mitte;
      } else {
        if (wert < feld[mitte]) {
          stopp = mitte - 1;
        }else{
          start = mitte + 1;
       }
      }
    } while (start <= stopp);
  }
  return -1;
}
```

Wie bereits in der letzten Handreichung bemerkt, müssen wir als erstes die Schleife entfernen, die beiden Selbstaufrufe einfügen und das Abbruchkriterium (start<=stopp) wieder einbauen. Damit wären wir bei folgender Version:

```
public static int binaereSuche(int[] feld, int start, int stopp, int wert) {

  if (feld!=null && feld.length > 0) {

     if(start<=stopp){

       int mitte = (start + stopp) / 2;

       if (feld[mitte] == wert) {
         return mitte;
       } else {
         if (wert < feld[mitte]) {
           binaereSuche(feld, start, mitte-1, wert);
         }else{
           binaereSuche(feld, mitte+1, stopp, wert);
        }
       }
     }else{
       return -1;
    }
  }
  return -1;
}
```

Das Problem ist allerdings, dass das Programm in manchen Situationen korrekt arbeitet, in anderen aber nicht:

Korrekt:

Das sortierte Feld:

20 31 34 43 52 55 78 83 84 89

Ich suche die Zahl: 52

Ausgabe der binären Suche: Gefunden an Position: 4

Nicht korrekt:

Das sortierte Feld:

6 21 30 35 49 52 71 72 73 78

Ich suche die Zahl: 35

Ausgabe der binären Suche: Zahl existiert nicht!

Woran liegt das? Wenn das Element im ersten Aufruf (also direkt) gefunden wird, wird der Wert von **mitte** an die aufrufende Methode korrekt zurück geliefert. Wenn allerdings der Wert von **mitte** erst in einem späteren Rekursionsschritt gefunden wird, gibt es ein Problem. Das liegt daran, dass die Methode den Wert von **mitte** zwar zurück liefert, in der darüber liegenden Methode wird der Wert von **mitte** allerdings nicht mehr nach oben durchgereicht! Wir müssen also dafür sorgen, dass der Wert von **mitte** bis ganz nach oben durchgereicht wird. Dieses erreichen wir, indem wir vor den beiden Selbstaufrufen noch jeweils ein **return** einfügen!

Rekursive Version der binären Suche (nicht optimiert):

```
public static int binaereSuche(int[] feld, int start, int stopp, int wert) {

  if (feld!=null && feld.length > 0) {

     if(start<=stopp){
       int mitte = (start + stopp) / 2;

       if (feld[mitte] == wert) {
         return mitte;
       } else {
         if (wert < feld[mitte]) {
           return binaereSuche(feld, start, mitte-1, wert);
         }else{
           return binaereSuche(feld, mitte+1, stopp, wert);
        }
       }
     }else{
       return -1;
    }
  }
   return -1;
}
```

Wenn man sich von der, durch die iterative Version vorgegebene Struktur löst, kommt man allerdings zu einer deutlich kompakteren Lösung.

Rekursive Version der binären Suche (optimiert):

```
/**
 * Die Methode bekommt ein int-Feld und einen int-Wert übergeben und
 * prüft, ob der Wert in dem Feld vorkommt.
 * Kommt der Wert vor, wir die Position des ersten gefundenen
 * Vorkommens zurückgeliefert.
 * Kommt der Wert nicht vor, wird -1 zurückgegeben.
 * @param feld das Feld,
 * @param start der startindex
 * @param stopp der stoppindex
 * @param wert der zu suchende Wert
 *
 * @version 2.0 (optimiert)
 */

public static int binaereSuche(int[] feld, int start, int stopp, int wert) {

  if (feld==null || feld.length == 0 || start>stopp) return -1;

  int mitte = (start + stopp) / 2;

  if(feld[mitte] == wert) return mitte;
  if(wert < feld[mitte])  return binaereSuche(feld, start, mitte-1, wert);
  if(wert > feld[mitte])  return binaereSuche(feld, mitte+1, stopp, wert);

  return -1; // nicht erreichbar
}
```