Übungen 3

# Aufgabe 4

|  |
| --- |
| public static int binSearch(T[] data, T value) {  int l = -1, h = data.length;  while (l + 1 != h) {  int m = l + (h - l) / 2;  if (data[m] < value) {  l = m;  } else if (data[m] > value) {  h = m;  } else {  return m;  }  }  return NOT\_FOUND;  } |

# Aufgabe 5

|  |
| --- |
| package ch.swaechter.fhnw.algd1.search.binobjectsearch;  public class BinaryObjectSearch {  public static final int NOT\_FOUND = -1;  public static <T extends Comparable<? super T>> int binSearch(T[] data, T value) {  int l = -1, h = data.length;  while (l + 1 != h) {  int m = l + (h - l) / 2;  int compare = data[m].compareTo(value);  if (compare < 0) {  l = m;  } else if (compare > 0) {  h = m;  } else {  return m;  }  }  return NOT\_FOUND;  }  } |

# Aufgabe 6

## Teilaufgabe A

* Iteration 1
  1. l = -1
  2. h = 14
  3. m = (-1 + 14) / 2 = 6.5 = 6 (Wert 8)
  4. l wird auf 6 gesetzt
* Iteration 2
  1. l = 6
  2. h = 14
  3. m = (6 + 14) / 2 = 10 (Wert 22)
  4. l wird auf 10 gesetzt
* Iteration 3
  1. l = 10
  2. h = 14
  3. m = (10 + 14) / 2 = 12 (Wert24)
  4. Es wird 11 zurückgegeben

Das mittlere der drei Werte, also Index 11, wird gefunden – nicht aber der erste mit dem Index 10

## Teilaufgabe B

* Offensichtliche Lösung: Wenn mir der binären Suche ein Index für das Element gefunden werden konnte, wird der Index so lange dekrementiert bis er ändert oder aber der Listenanfang erreicht wird:
  + Ändert der Wert des Elements gleich am Anfang, so handelt sich um ein kleineres Element und keinen Vorläufer. Der nicht dekrementierte Wert wird dann als Index zurückgegeben
  + Ändert der Wert des Elements nicht, handelt es sich um Vorläufer, sprich um ein Element mit demselben Wert. Der Index wird jetzt solange weiter dekrementiert, bis der Wert ändert (Siehe oben) oder der Listenanfang erreicht ist. Dieser Wert wird dann als Index zurückgegeben
* Da Sie erwähnt haben, dass die Lösung kürzer als der aktuelle Binary Search sei, kann der obige Ansatz nicht der beste sein. Hinzu kommt, dass er nicht den Index von nichtexistierenden Elementen zurückgibt (Sprich der Index vor dem nächst grösseren Element)
* Überlegungen:
  + Eine if/else if/else Konstrukt führt dazu, dass bei data[m] == value nur gerade der passende m Index, nicht aber der vorläuferlose Index zurückgegeben wird 🡪 Das return im while muss also Weg (Zumal mich das return m in der Schleife schon immer gestört hat)
  + Da wir immer den ersten Index möchten, benötigen wir eine Laufvariable die manipuliert und am Ende zurückgeben wird
    - Möchte man den Index des ersten vorläuferlosen Elementes, muss die obere Grenze h verwendet werden
    - Möchte man den Index des letzten vorläuferlosen Elementes, muss die untere Grenze l verwendet werden
  + Wir verwenden also h als Index und manipulieren ihn so lange, bis die Abbruchbedingung l + 1 == h eintritt. Da wir nie den erstbesten passenden Wert verwenden, repräsentiert h den Index

|  |
| --- |
| package ch.swaechter.fhnw.algd1.search.binsearchfirst;  public class BinSearchFirstElement {  public static int binSearch(int[] data, int value) {  int l = -1, h = data.length;  while (l + 1 != h) {  int m = l + (h - l) / 2;  if (data[m] < value) {  l = m;  } else {  h = m;  }  }  return h;  }  } |