[A06] Tiefensuche

Aufgabe

Im Package A06_Tiefensuche finden Sie die notwendigen Basisklassen, um eine Tiefensuche in einem Baum zu implementieren. Auch passende JUnit-Tests, um Ihre Implementierung zu überprüfen, sind dort vorhanden. Eingefügt in den Baum werden Film-Objekte, als Sortierkriterium soll deren Länge dienen. Sie müssen also die compare()-Methode entsprechend implementieren.

- `public List<String> getNodesInOrder(Node<Film> node)`
- `public List<String> getMinMaxPreOrder(double min, double max)`

Lösung

In-Order-Tiefensuche

Die Methode `getNodesInOrder` implementiert eine Tiefensuche in der Reihenfolge (linkes Kind, Knoten, rechtes Kind), die eine Liste der Titel, welche im Baum gespeicherte Filme ausgibt.

```
public List<String> getNodesInOrder(Node<Film> node) {
  List<String> result = new ArrayList<>();
  if (node != null) {
    List<String> resultOfLeftChild = getNodesInOrder(node.getLeft());
    Film nodeValue = node.getValue();
    String title = nodeValue.getTitel();
    List<String> resultOfRightChild = getNodesInOrder(node.getRight());
    result.addAll(resultOfLeftChild);
    result.add(title);
    result.addAll(resultOfRightChild);
  }
  return result;
}
```

Pre-Order-Tiefensuche

Die Methode `getNodesPreOrder` implementiert eine Tiefensuche in der Reihenfolge (Knoten, linkes Kind, rechtes Kind) die eine Liste der Titel, der im Baum gespeicherten Filme, ausgibt. Außerdem gibt es noch die Einschränkung, dass nur jene Filme ausgegeben werden können, welche eine Länge zwischen den Werten `min` und `max` haben.

Die rekursive Methode `getNodesPreOrder` wird mithilfe von `getMinMaxPreOrder` auf dem Wurzelknoten `root` aufgerufen.

```
public List<String> getMinMaxPreOrder(double min, double max) {
  return getNodesPreOrder(root, min, max);
}
```

```
public List<String> getNodesPreOrder(Node<Film> node, double min, double max) {
  List<String> result = new ArrayList<>();
  if (node != null) {
    Film nodeValue = node.getValue();
    String title = nodeValue.getTitel();
    double nodeLength = nodeValue.getLaenge();
    if (nodeLength > min && nodeLength < max) {
      result.add(title);
    }
    List<String> left = getNodesPreOrder(node.getLeft(), min, max);
    result.addAll(left);
    List<String> right = getNodesPreOrder(node.getRight(), min, max);
    result.addAll(right);
  }
  return result;
}
```

Laufzeit

Bei den beiden Tiefensuchen ist der Algorithmus gleich, lediglich die Reihenfolge des Speicherns der Knoten in die Liste der Resultate ist verschieden. Deswegen haben beide Algorithmen die selbe Laufzeit von *O(n)*, wobei *n* der Anzahl der Knoten entspricht. Bei `getNodesPreOrder` gibt es zusätzlich zwei Vergleiche, die aber keinen wesentlichen Einfluss auf die Laufzeit haben, da sie mit einer konstanten Laufzeit (*O(1)*) erfolgen.