Übung 4: Binärer Suchbaum

Abgabetermin: 14.04.2015

Name: _____ **Matrikelnummer:** _____

Gruppe: ☐ **G1** Di 10:15 ~~☐ **G2** Di 11:00~~ ☐ **G3** Di 12:45

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Punkte
Aufgabe 1	24	☐		Java-Programm Testfälle und Ergebnisse	☐	

Aufgabe 1: Binärer Suchbaum für sortierte Menge von Zahlen (24 Punkte)

Implementieren Sie eine sortierte Menge für Zahlen als binären Suchbaum in der Klasse *BinaryTreeSet.* Die Schnittstelle ist durch die abstrakte Klasse *Set* gegeben (Details in Vorgabedatei).

```
package at.jku.ssw;
public abstract class Set {
  public abstract void add(int value);
  public abstract boolean contains(int value);
  public abstract boolean remove(int value);
  public abstract int size();
  public abstract IntIterator iterator();
  public abstract Set union(Set other);
  public abstract Set intersect(Set other);
  public abstract Set diff(Set other);
  public abstract Set subtract(Set other);
}
```

```
public abstract class IntIterator {
  public abstract boolean hasNext();
  public abstract int next();
}
```

Implementieren Sie davon abgeleitet folgende Klassen im Paket *at.jku.students:*

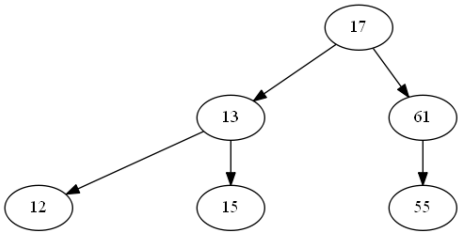
```
package at.jku.students;
class BinaryTreeSet extends Set {
  TreeNode root;
  public Iterator levelOrderIterator() { /* TODO */ }
  public String makeDot() {
    return DotMaker.makeDotForBinaryTree(root);
  }
}
class BinaryTreePreOrderIterator extends IntIterator {
  ...
}
class BinaryTreeLevelOrderIterator extends IntIterator {
  ...
}
```

```
Set s = new BinaryTreeSet();
s.add(17); s.add(13); s.add(61);
s.add(12); s.add(13); s.add(55);
s.add(15);
IntIterator it = s.iterator();
while (it.hasNext()) {
  Out.print(" " + it.next());
}  // Ausgabe: 17 13 12 15 61 55
Out.open("test.dot");
Out.print(
  ((BinaryTreeSet) s).makeDot());
Out.close();
```

Die Klasse *BinaryTreePreOrderIterator* liefert die Werte des Baums in *pre-order*-Ordnung (Tiefensuche). *BinaryTreeLevelOrderIterator* durchläuft die Baumebenen von links nach rechts (Breitensuche). Implementieren Sie das Durchlaufen wie in der Übung gezeigt mit Stack und Queue und verwenden Sie *LinkedListStack* und *LinkedListQueue* aus der Vorgabedatei. Die Methode *iterator* von *BinaryTreeSet* soll einen *pre-order*-Iterator erzeugen, während *levelOrderIterator* einen *level-order*-Iterator erzeugen soll.

Implementierungshinweise:

- Verwenden Sie die Vorgabedateien *ssw-pi2-ue04.jar* und die zugehörige Java-Dokumentation von der LVA-Website.
- Definieren Sie für alle Klassen, Methoden und Felder die geeignete Sichtbarkeit (private, protected, package, public).
- Implementieren Sie die Methoden *add, contains* und *size* mit rekursiven Algorithmen.
- Verwenden Sie die Methode *DotMaker.makeDotForBinaryTree* um GraphViz-Bilder Ihres Binärbaums zu erstellen.



test.dot

- Testen Sie Ihre Implementierung mit der Vorgabedatei *BinaryTreeSetTest.java* und vergleichen Sie Ihre Ergebnisse mit *BinaryTreeSetTest.Output.txt.*

Abzugeben ist: Java-Programm, Testergebnisse