Übung 4: Set

Abgabetermin: 16.04.2013

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	Java-Programm	□	

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

Implementieren Sie eine sortierte Menge für Buchstaben als binären Suchbaum in der Klasse *BinaryTreeSet.* Erlaubt sind nur die Kleinbuchstaben 'a' bis 'z'. Die Schnittstelle ist durch die abstrakte Klasse *Set* gegeben (für Methodenbeschreibungen siehe Java-Dokumentation in der Vorgabedatei).

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

```
public abstract class CharIterator {
  public abstract boolean hasNext();
  public abstract char next();
}
```

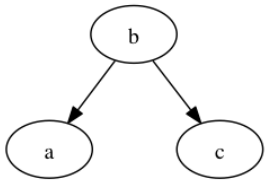
Implementieren Sie die Klassen *BinaryTreeSet* und *BinaryTreeIterator* im Paket *at.jku.students*. Verwenden Sie dazu die Klassen *Set* und *TreeNode* aus der Vorgabedatei.

```
package at.jku.students;
public class BinaryTreeSet extends Set {
  TreeNode root;
  public String makeDot() {
    return DotMaker.makeDotForBinaryTree(root);
  }
}
public class BinaryTreeIterator
    extends CharIterator {
  ...
}
```

```
Set s = new BinaryTreeSet();
s.add('b'); s.add('c'); s.add('c');
s.add('b'); s.add('a');
CharIterator it = s.iterator();
while (it.hasNext()) {
  Out.print(" " + it.next());
} // Ausgabe: a b c
Out.open("test.dot");
Out.print(
  ((BinaryTreeSet) s).makeDot());
Out.close();
```

Implementierungshinweise:

- Verwenden Sie die Vorgabedateien *ssw-pi2.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.
- Testen Sie Ihre Implementierung mit der Vorgabedatei *BinaryTreeSetTest.java* und vergleichen Sie Ihre Ergebnisse mit *BinaryTreeSetTest.Output.txt.*



test.dot

Abzugeben ist: Java-Programm, Testergebnisse