Übung 9: Hashing

Abgabetermin: 19.05.2015

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: Hashtabelle mit quadratischem Probieren (24 Punkte)

Implementieren Sie eine Hashtabelle zur Assoziation von *String*-Schlüsseln mit *Object*-Werten. Als Kollisionsstrategie verwenden Sie quadratisches Probieren mit dem in der Vorlesung beschriebenen Trick, um bei der Suche alle Tabellenplätze zu besuchen. Die Schnittstelle ist durch die abstrakte Klasse *Map* gegeben, Methodenbeschreibungen finden Sie in der Java-Dokumentation in der Vorgabedatei.

```
package at.jku.ssw;
abstract class Map {
 abstract boolean containsKey(String key);
 abstract Object get(String key);
 abstract void put(String key, Object value);
 abstract Object remove(String key);
}
```

```
class Element {
 final String key;
 final Object value;
 boolean isDeleted;
 Element(String key,
         Object value) {...}
}
```

Implementieren Sie im Paket *at.jku.students* eine Klasse *HashMap* wie folgt:

```
class HashMap extends Map {
 Element[] table;
 HashMap(float fillFactor) {  ... }
 String makeDot() {  ... }
 String makeDot(float width) {  ... }
 ...
}
```

```
Map m = new HashMap(0.9f);
m.put("Flughafen Wien", "01 7007 0");
m.put("BAWAG", "05 99 05");
m.put("Hypo Alpe Adria", "05 02 02-0");
m.put("BUWOG", "01 878 28 1130");
m.put("Meinl Bank AG", "01 53 18 80");
Out.println(m.get("Hypo Alpe Adria"));
// Ausgabe: 05 02 02-0
Out.open("hashtable.dot");
Out.println(((HashMap)m).makeDot());
Out.close();
```

Implementierungshinweise:

a) Dimensionieren Sie die Hashtabelle anfangs mit 7 Speicherplätzen. Immer dann, wenn der im Konstruktor angegebene maximale Füllfaktor überschritten wird, vergrößern Sie automatisch auf mindestens die doppelte Größe. Der maximale Füllfaktor muss größer 0.0 und kleiner oder gleich 1.0 sein. Nach jedem Vergrößern ist ein *rehashing* notwendig.

b) Beachten Sie, dass die **Größe der Tabelle sowohl die Formel $4 \cdot j + 3$ (für j in $\mathbb{N}$) erfüllen als auch eine Primzahl sein muss,** damit die Suche korrekt funktioniert. Mit der Methode *PrimeNumbers.isPrime(n)* aus der Vorgabedatei können Sie prüfen, ob eine Zahl prim ist.

c) Berechnen Sie den Hash der *String*-Schlüssel durch Aufruf von deren *hashCode*()-Methode.

d) Verwenden Sie die Methoden *DotMaker.makeDotForHashtable(Element[] table)* und *DotMaker.makeDotForHashtable(Element[] table, float width)*, um GraphViz-Bilder Ihrer Hashtabelle zu erzeugen.

Abzugeben ist: Java-Programm, Testergebnisse