Algorithmen und Datenstrukturen

Hashing

Aufgabe 1: Hashtabelle [2 Punkte]

Die zehn nach der Einwohnerzahl grössten Städte Italiens werden in einer HashTabelle der Grösse 13 abgelegt. Für die Kollisionsauflösung ist „Quadratic Probing“ festgelegt.

Die Städte und die für sie berechneten Hash-Codes sind:

5	Bari
8	Bologna
3	Catania
9	Firenze
0	Genova
12	Milano
7	Napoli
7	Palermo
7	Roma
5	Torino

Wie sieht die Hash-Tabelle aus, nachdem alle zehn Städte in alphabetischer Reihenfolge wie oben eingefügt wurden?

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

Aufgabe 2: Competitor Methoden [2 Punkte]

Bei der Auswertung nach einem Wettrennen müssen die Einträge in der Teilnehmerliste mit der Ankunftszeit vervollständigt werden. Es muss **anhand des Namens** und **des Jahrgangs** der Teilnehmer schnell bzw. effizient gefunden werden. Das effizienteste Verfahren basiert auf Hashtabellen. Implementieren Sie equals, compareTo und hashCode Methoden im Competitor so, dass sie dem *Object-Contract* gehorchen.

Aufgabe 3: Verwaltung einer Teilnehmerliste [3 Punkte]

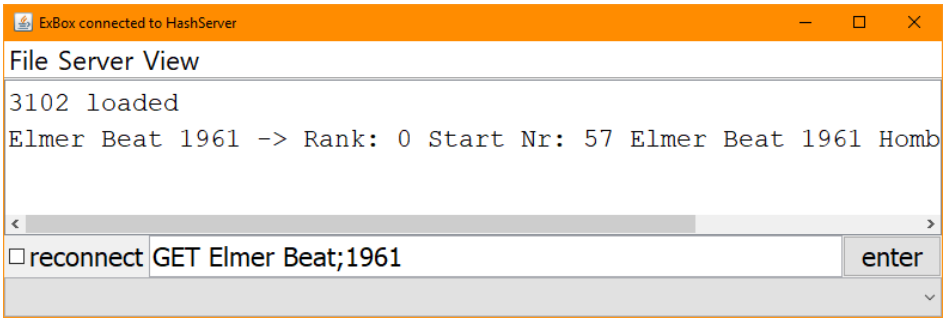
Schreiben Sie eine HashServer-Klasse, die den CommandoExecutor implementiert und mittels der Sie die Teilnehmerliste des Zürich-Marathons in einer Hashtabelle verwalten können. Als Schlüssel soll der Namen plus Jahrgang genommen werden. Es sollen die Operationen *Hinzufügen* und *Suchen (GET <key>)* von Teilnehmern implementiert werden.

Hinweise:

- Damit sie die Rangliste einfach (wie gehabt) einlesen können, soll die Defaultoperation von `execute` (ohne vorangestellten GET-Befehl) «einlesen der Daten» sein.

```
if (arg.toUpperCase().startsWith("GET")) {
    …
}
else {
    …
}
```

- Verwenden Sie das Map-Interface aus dem java.util Paket und die Klasse HashMap.
- Falls Sie die Hashtabelle in der Klasse HashServer speichern, müssen Sie das Programm ohne die Option «reconnect» ausführen.



Aufgabe 4: Eine eigene Hashtable [3 Punkte]

Ersetzen sie die HashMap durch eine eigene Klasse MyHashtable, die das Map-Interface implementiert. Implementieren Sie hinzufügen, löschen (REMOVE) und suchen (GET). Rufen Sie die hashCode Methode zur Bestimmung des Hashcodes auf. Mit HashTest können Sie Ihre Implementation testen.

Hinweise:

- Verwenden Sie das MyHashtable Gerüst (siehe Vorlage):

```
public class MyHashtable<K,V> implements java.util.Map<K,V> { private Object[] keys; private
   Object[] values; public MyHashtable(int size) { // to be done } public V put(Object key,
   Object value) { // to be done } public V get(Object key) { // to be done } public V
   remove(Object key) { // optional: to be done } ...
}
```

- In der Vorlage werfen alle Methoden die UnsupportedOperationException damit die Vorlage compiliert.
- Implementieren Sie die Methoden mit "to be done" im Kommentar