Übung 9 – Hashing

Implementieren Sie eine Klasse *DoubleHashingHashTable*, die eine Hashtabelle verwaltet. Das Verfahren zur Kollisionsbehebung soll *double hashing* sein.

```
public class DoubleHashingHashTable {
  public DoubleHashingHashTable()
    // default-Konstruktor
  public DoubleHashingHashTable(int size)
    // Konstruktor mit Angabe der Groesse der Hashtabelle

  public boolean contains(String s);
    // suche nach einem String
  public void insert(String s);
    // fuege neuen String ein
  public void remove(String s);
    // loesche einen String aus der Hashtabelle
  public void printHashTable();
    // schreibt den Inhalt der Hashtabelle auf die Konsole
}
```

Es soll möglich sein, Strings einzufügen, zu finden und zu löschen. Zudem soll für Kontrollzwecke eine Methode *printHashTable* zur Verfügung stehen, die die Hashtabelle in folgender Form auf die Konsole schreibt:

```
DoubleHashingHashTable:
0     -
1     last
2     -
3     famous
4     words
...
```

Jeder Eintrag auf eine separate Zeile.
Zuerst der Index gefolgt von einem Tabulator, dann der Inhalt des Strings.
Falls ein Eintrag leer ist, geben Sie "-" aus.

Hinweise:

- Verwenden Sie die `hashCode`-Methode der eingefügten Strings um einen ersten Hashcode zu erhalten.
- Zur Bereichsbegrenzung einer neuen, zweiten Hashfunktion *h2* verwenden Sie z.B. die besprochene Formel:
  $hash2(s) = p - (h2(s) \bmod p)$ mit p < N und N, p prim, N Tabellengrösse.

Antwortstruktur:

1. Beantworten Sie zuerst folgende Fragen:
   a) Wie sieht Ihre zweite Hashfunktion h2 aus? Begründen Sie Ihren Entscheid.
   b) Welches Problem entsteht, wenn man beim Löschen einen Eintrag einfach entfernt? Geben Sie ein Lösungskonzept an.
   Wir gehen davon aus, dass die Antworten auf maximal 1-2 Seiten Platz finden.
2. Programmieren Sie die Klasse DoubleHashingHashTable (ohne remove)
3. Freiwillig: implementieren Sie *remove*
4. Freiwillig: Testen Sie Ihre Klasse mit dem Werk von Shakespeare