Praktikum Algorithmen und Datenstrukturen, Teil 3	Hochschule Bochum Bochum University of Applied Sciences
Praktikum Nr. 3: Hashing-Datenstrukturen	

Generelles:

Die Aufgabenstellung besteht aus einer Hauptaufgabe, in der Sie Hashtabellen betrachten sollen, in denen Verkettung verwendet wird. Als **optionale Zusatzaufgabe** können Sie (für 1/3 Punkt) dann eine zweite, alternative Implementation der Hash-Struktur realisieren, die statt Verkettung offenes Hashing verwendet.

Aufgabenbeschreibung: Hash-Datenstruktur mit Verkettungen

Hauptaufgabe(1 Punkt): Gegeben sei die Applikation HashApp.java, die Strings in eine Hashtabelle gemäß einer definierten Hashfunktion einfügt.

1. Auf welche „Hash-Buckets“ werden zunächst die Strings jeweils abgebildet, und wie lautet die ursprünglich hier im Programmcode verwendete Hashfunktion? Dokumentieren sie die entsprechende Methode geeignet mit den Antworten auf die obigen Fragen.
2. Schreiben Sie die 4 Funktionalitäten *Element suchen*, *Element löschen*, *Liste eines Buckets ausgeben*, und *listenLaengen* mit den Eigenschaften:
    a. *Element suchen*: sucht einen einzugebenden String in der Hashtabelle und gibt - falls vorhanden- die Nummer des Korbes an.
    b. *Element löschen*: löscht ein Datenelement aus der Hashtabelle sofern existent.
    c. *Liste eines Buckets ausgeben*: Gibt alle Elemente der Datenstruktur aus, die sich im Bucket mit dem angegebenen Index befinden.
    d. *listenLaengen*: gibt die Länge der Listen aller Buckets aus.
3. Programmieren Sie eine eigene (sinnvollere) Hashfunktion. Dokumentieren Sie die neue Methode, und äußern Sie sich zu Vorteilen gegenüber der bisherigen Hashfuntion.
4. Wie sehen die Strings (bzw.: deren Format) aus, die beim automatischen Befüllen der Listen im Menüpunkt 7 verwendet? Dokumentieren Sie die entsprechende Code-Passage mit der Antwort auf diese Frage.

Optionale Zusatzaufgabe (1/3 Punkt): Realisieren Sie eine zweite, alternative Implementation HashAppOpen.java der Hash-Struktur aus HashAppOpen.java. Die alternative Implementierung soll statt Verkettung Hashing mit offener Adressierung verwenden. Auch die alternative Implementierung soll Methoden zum Einfügen, Löschen, und Suchen von Strings in der Hashtabelle bereitstellen.

```
package hash;

import javax.swing.*;

public class HashApp {

   public static void main (String args []) {
      final int B = 5;
      String  menue = "Eingabe\n" ;
      menue += " 1) Element einfuegen\n";
      menue += " 2) Element suchen\n";
      menue += " 3) Zelle loeschen\n";
      menue += " 4) Laenge der Liste von Korb k\n";
      menue += " 5) Liste von Korb k ansehen\n";
      menue += " 6) Laenge der Listen von allen Koerben\n";
      menue += " 7) Listen zufaellig fuellen\n";
      String ausgabe="";
      Liste hashTabelle[];
      hashTabelle = new Liste [B];
      for (int i = 0; i < B;i++)
         hashTabelle[i] = new Liste();
      while (true) {
         String menueeingabe = JOptionPane.showInputDialog(menue);
         if (menueeingabe == null)
            break;
         String dialogEingabe;
         if (menueeingabe.equals("1")) {
            dialogEingabe = JOptionPane.showInputDialog("Datenstring?");
            int h = hashTabelle[0].hashFunktion(dialogEingabe, B); //Welcher Korb?
            boolean existsInHashtable = false;
            for (int j = 1; j <= hashTabelle[h].laenge(); j++) {
               if (hashTabelle[h].inhalt(j).equals(dialogEingabe)) {
                  ausgabe = "Element in Korb " + h + " gefunden, daher nicht eingefuegt";
                  existsInHashtable = true;
               }
               break;
            }
            if (!existsInHashtable) {
               hashTabelle[h].einsetzenAnfang(dialogEingabe);
               ausgabe = "Eingabe eingefuegt in bucket " + h;
            }
            JOptionPane.showMessageDialog(null, ausgabe);
         }

         if (menueeingabe.equals("2")) {
            dialogEingabe = JOptionPane.showInputDialog("Welches Element soll gesucht werden?");
            ausgabe = "";
            //...
         }

         if (menueeingabe.equals("3")) {
            dialogEingabe = JOptionPane.showInputDialog("Welches Element soll geloescht werden?");
            ausgabe = "";
            //...
         }

         if (menueeingabe.equals("4")) {
            dialogEingabe = JOptionPane.showInputDialog("Von welchem Korb soll die Laenge angezeigt werden?");
            ausgabe = "";
            int k = Integer.parseInt(dialogEingabe);
            int l = hashTabelle[k].laenge();
            ausgabe = "Liste hat Laenge " + l;
            JOptionPane.showMessageDialog(null, ausgabe);
         }

         if (menueeingabe.equals("5")) {
            //...
         }

         if (menueeingabe.equals("6")) {
            //...
         }

         if (menueeingabe.equals("7")) {
            dialogEingabe = JOptionPane.showInputDialog("Anzahl?");
            ausgabe = "";
            int zahl = Integer.parseInt(dialogEingabe);
            for (int j = 1; j <= zahl; j++) {
               String s = "";
               int laenge = (int) ((Math.random() * 8) + 3);

               for (int k = 1; k <= laenge; k++) {
                  char zufall = (char) ((Math.random() * 26) + 65);
                  s += zufall;
               }
               int h = hashTabelle[0].hashFunktion(s, B); //Welcher Korb?
               hashTabelle[h].einsetzenAnfang(s);
            }
         }
      }
   }
}
```

```
// Liste
package hash;

public class Liste {
    Zelle anfang;
    Zelle cursor;

    int laenge () {
        Zelle cur = anfang;
        int l = 0;
        while (cur != null) {
            l++;
            cur = cur.next;
        }

        return l;
    }

    int hashFunktion (String x,int b) {
        return b-1;
    }

    boolean istGueltigePosition (int p) {
        return (p >= 1) && (p <= laenge () );
    }

    void setzeCursor (int p){
        cursor = null;
        if (istGueltigePosition (p) ) {
            Zelle cur = anfang;
            for (int i = 1; i < p;i++)
                cur = cur.next;
            cursor = cur;
        }
    }


    void einsetzenAnfang (String e){
        Zelle z = new Zelle (e,anfang);
        anfang = z;
    }


    void loesche (int p) {
        if (istGueltigePosition(p)){
            if (p == 1) // Loesche 1. Zelle
                anfang = anfang.next;
            else {
                setzeCursor(p-1);
                cursor.next = cursor.next.next;
            }
        }
    }


    void loescheElem (String e) {

        for (int i = 1; i <= laenge(); i++) {

            if(inhalt(i).equals(e))
                loesche(i);
        }

        //Loesche(Suche (e));
    }

    String inhalt (int p){
        setzeCursor (p);
        return cursor.inhalt;
    }


}
```

```java
package hash;

public class Zelle {
    String inhalt; // Beachten Sie: inhalt ist sowohl der Schluesselwert, als auch der komplette zugehörige Dateneintrag
    Zelle next;

    // Konstruktor:
    Zelle (String el) {
        inhalt = el; next = null;
    }
    Zelle (String el, Zelle z) {
        inhalt = el; next = z;
    }
}
```