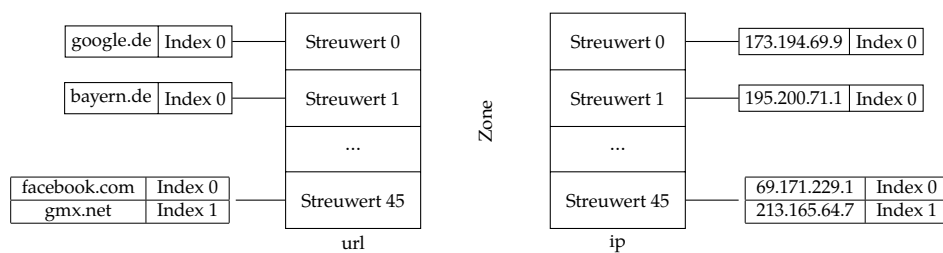


Aufgabe 6 Streutabellen (Hash-Tables)

Um die URL (zum Beispiel google.de) und die zugehörige IP des Servers (hier 173.194.69.9) zu verwalten, werden Streutabellen verwendet, die eine bestimmte Zone von Adressen abbilden. Die Streutabellen werden als zwei dynamische Arrays (in Java: ArrayLists) realisiert. Kollisionen innerhalb einer Zone werden ebenfalls in dynamischen Arrays verwaltet.



Um zu einer URL die IP zu finden, berechnet man zunächst mittels der Funktion `hash()` den entsprechenden Streuwert, entnimmt dann den Index der Tabelle URL und sucht schließlich an entsprechender Stelle in der Tabelle IP die IP-Adresse.

- (a) Erläutern Sie am vorgestellten Beispiel, wie ein Hash-Verfahren zum Speichern großer Datenmengen prinzipiell funktioniert und welche Voraussetzungen und Bedingungen daran geknüpft sind.
- (b) Nun implementieren Sie Teile dieser IP- und URL-Verwaltung in einer objektorientierten Sprache Ihrer Wahl. Verwenden Sie dabei die folgende Klasse (die Vorgaben sind in der Sprache Java gehalten):

```

1  class Zone {
2  private ArrayList<ArrayList<String>> urlList =
3  new ArrayList<ArrayList<String>>();
4  private ArrayList<ArrayList<String>> ipList =
5  new ArrayList<ArrayList<String>>();
6  public int hash(String url) { /* calculates hash-value h, >=0 */
7  }

```

- (i) Prüfen Sie in einer Methode `boolean exists(int h)` der Klasse `zone`, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert vorhanden ist. Falls `h` größer ist als die derzeitige Größe der Streutabelle, existiert der Eintrag nicht.

```

56  /**
57   * Prüfe, ob bereits mindestens ein Eintrag für einen gegebenen
58   * Streuwert vorhanden ist. Falls h größer ist als die derzeitige Größe
59   * der Streutabelle, existiert der Eintrag nicht.
60   *
61   * @param h Der Index-Wert, der durch die Hashfunktion erzeugt
62   * wird.

```

```

63      * @return Wahr, wenn in beiden Streutabellen an einer
        ↳ bestimmen Index-Position
64      *
        ↳ mindestens ein Wert hinterlegt ist, sonst falsch.
65      */
66      boolean exists(int h) {
67          if (urlList.size() - 1 < h || ipList.size() - 1 < h)
68              return false;
69
70          ArrayList<String> urlCollisionList = urlList.get(h);
71          ArrayList<String> ipCollisionList = ipList.get(h);
72          if (urlCollisionList.size() == 0 || ipCollisionList.size() ==
        ↳ 0)
73              return false;
74
75          return true;
76      }

```

- (ii) Die Methode `int getIndex (string url, ArrayList<String> urlList)` soll den Index einer URL in der Kollisionsliste berechnen. Ist die URL in der Kollisionsliste nicht vorhanden, soll `-1` zurückgeliefert werden.

```

68          return false;
69
70          ArrayList<String> urlCollisionList = urlList.get(h);
71          ArrayList<String> ipCollisionList = ipList.get(h);
72          if (urlCollisionList.size() == 0 || ipCollisionList.size() ==
        ↳ 0)
73              return false;
74
75          return true;
76      }
77
78      /**
79      * Berechne den Index einer URL in der Kollisionsliste. Ist die
        ↳ URL in der
80      * Kollisionsliste nicht vorhanden, soll -1 zurückgeliefert
        ↳ werden.
81      */
82      int getIndex(String url, ArrayList<String> urlList) {
83          for (int i = 0; i < urlList.size(); i++) {
84              if (urlList.get(i).equals(url))
85                  return i;
86          }
87          return -1;
88      }

```

- (iii) Ergänzen Sie die Klasse Zone um eine Methode `String lookup (String url)`, die in der Streutabelle die IP-Adresse zur `url` zurückgibt. Wird eine nicht vorhandene Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.

```

90      /**
91      * Gib in der Streutabelle die IP-Adresse zurück. Wird eine
        ↳ nicht vorhandene
92      * Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.
93      *
94      * @param url Die gesuchte URL.
95      *
96      * @return Die entsprechende IP-Adresse.

```

```

97     */
98     String lookup(String url) {
99         int h = hash(url);
100         int collisionIndex = getIndex(url, urlList.get(h));
101         if (collisionIndex == -1)
102             return
103             ↪ "Die URL kannte nicht in der Tabelle gefunden werden";
104         return ipList.get(h).get(collisionIndex);
105     }

```

```

3 import static org.junit.Assert.assertEquals;
4
5 import java.util.ArrayList;
6
7 import org.junit.Test;
8 import org.junit.Before;
9
10 public class ZoneTest {
11
12     Zone zone;
13
14     @Before
15     public void legeZoneAn() {
16         zone = new Zone();
17         zone.addUrl("google.de", "173.194.69.9");
18         zone.addUrl("bayern.de", "195.200.71.1");
19         zone.addUrl("facebook.com", "69.171.229.1");
20         zone.addUrl("gmx.net", "213.165.64.7");
21     }
22
23     @Test
24     public void methodExists() {
25         assertEquals(true, zone.exists(0));
26         assertEquals(true, zone.exists(1));
27         assertEquals(false, zone.exists(2));
28         assertEquals(true, zone.exists(45));
29         assertEquals(false, zone.exists(46));
30     }
31
32     @Test
33     public void methodGetIndex() {
34         ArrayList<String> urlCollisionList = zone.getUrlCollisionList(45);
35         assertEquals(0, zone.getIndex("facebook.com", urlCollisionList));
36         assertEquals(1, zone.getIndex("gmx.net", urlCollisionList));
37         assertEquals(-1, zone.getIndex("bschlangaul.org", urlCollisionList));
38     }
39
40     @Test
41     public void methodLookup() {
42         assertEquals("173.194.69.9", zone.lookup("google.de"));
43         assertEquals("195.200.71.1", zone.lookup("bayern.de"));
44         assertEquals("69.171.229.1", zone.lookup("facebook.com"));
45         assertEquals("213.165.64.7", zone.lookup("gmx.net"));
46         assertEquals("Die URL kannte nicht in der Tabelle gefunden werden",
47             ↪ zone.lookup("bschlangaul.org"));
48     }
49 }

```