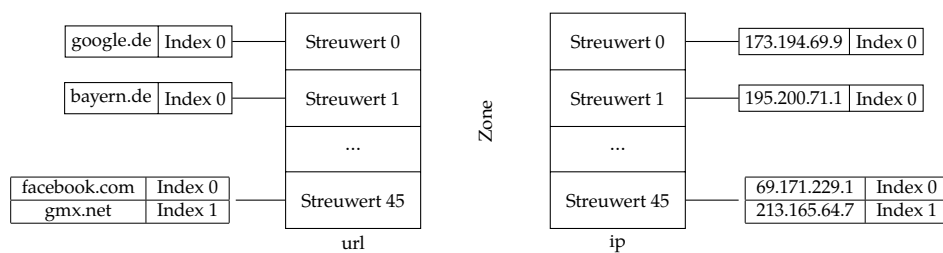


Aufgabe 6 Streutabellen (Hash-Tables)

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
   ↪ Streuwert
58   * vorhanden ist. Falls h größer ist als die derzeitige Größe
   ↪ der Streutabelle,
59   * existiert der Eintrag nicht.
60   *
61   * @param h Der Index-Wert, der durch die Hashfunktion erzeugt
   ↪ wird.
62   *

```

```

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 java.util.ArrayList;
4
5  class Zone {
6
7      private ArrayList<ArrayList<String>> urlList = new
8          ↳ ArrayList<ArrayList<String>>();
9
10     private ArrayList<ArrayList<String>> ipList = new
11         ↳ ArrayList<ArrayList<String>>();
12
13     /**
14      * Der Konstruktor initialisiert die Streutabellen mit 46 Buckets, damit
15      ↳ wir den
16      * Code testen können.
17      */
18     public Zone() {
19         for (int i = 0; i <= 45; i++) {
20             urlList.add(new ArrayList<String>());
21             ipList.add(new ArrayList<String>());
22         }
23     }
24
25     /**
26      * Diese Methode wird zum Testen der Methode getIndex gebraucht.
27      */
28     public ArrayList<String> getUrlCollisionList(int h) {
29         return urlList.get(h);
30     }
31
32     /**
33      * Nicht wirklich eine Hashfunktion. Gibt die Werte wie im Schaubild
34      ↳ zurück.
35      * Diese Methode muss nicht implementiert werden. Sie ist nur dazu da, um
36      ↳ die
37      * Code testen zu können.
38      *
39      * @param url Eine URL.
40      *
41      * @return Ein Hashwert, der größer gleich 0 ist.
42      */
43     public int hash(String url) {
44         /* calculates hash-value h, >=0 */
45         switch (url) {
46             case "google.de":
47                 return 0;
48
49             case "bayern.de":
50                 return 1;
51
52             case "facebook.com":

```

```

48         case "gmx.net":
49             return 45;
50
51         default:
52             return 42;
53     }
54 }
55
56 /**
57  * Prüfe, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert
58  * vorhanden ist. Falls h größer ist als die derzeitige Größe der
↪ Streutabelle,
59  * existiert der Eintrag nicht.
60  *
61  * @param h Der Index-Wert, der durch die Hashfunktion erzeugt wird.
62  *
63  * @return Wahr, wenn in beiden Streutabellen an einer bestimmen
↪ Index-Position
64  *         mindestes 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 }
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 }
89
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 "Die URL kannte nicht in der Tabelle gefunden werden";
103     return ipList.get(h).get(collisionIndex);
104 }
105

```

```

106  /**
107   * Nicht verlangt. Zum Einfügen von Testwerten gedacht.
108   *
109   * @param url Eine URL.
110   * @param ip  Eine IP-Adresse.
111   */
112  public void addUrl(String url, String ip) {
113      int h = hash(url);
114      urlList.get(h).add(url);
115      ipList.get(h).add(ip);
116  }
117
118  public static void main(String[] args) {
119      Zone zone = new Zone();
120      zone.addUrl("google.de", "173.194.69.9");
121      zone.addUrl("bayern.de", "195.200.71.1");
122      zone.addUrl("facebook.com", "69.171.229.1");
123      zone.addUrl("gmx.net", "213.165.64.7");
124  }
125  }

```