```java
/*
 * HSR − Uebungen 'Algorithmen & Datenstrukturen 2'
 * Version: Mon Oct 22 12:16:24 CEST 2018
 */

package uebung06.as.aufgabe02;

import java.util.Arrays;
import java.util.LinkedList;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * A Radix−Sort which uses internally a Bucket−Sort to sort a list of arrays of
 * strings.
 *
 * @author mbuehlma
 *
 */
public class RadixSort {

  // buckets used for bucket sort
  private final LinkedList<String>[] buckets;

  @SuppressWarnings("unchecked")
  RadixSort() {
    // create LinkedList for buckets
    buckets = (LinkedList<String>[]) new LinkedList<?>[1 + ('z' − 'a' + 1)];

    // TODO Implement here...

  }

  public void radixSort(String[] data) {

    // TODO Implement here...

  }

  protected void bucketSort(String[] data, int index) {

    // TODO Implement here...

  }
```



```java

  public static void main(String[] args) {

    // unsorted data
    final String[] data = new String[] { "bruno", "brach", "auto", "auto",
        "autonom", "clown", "bismarck", "autark", "authentisch",
        "authentische", "autobahn", "bleibe", "clan" };

    new RadixSort().radixSort(data);

    // verification array, for test purpose only
    final String[] verification;
    // sort the verification array
    verification = data.clone();
    Arrays.sort(verification);

    // print and verify output
    AtomicInteger i = new AtomicInteger(0);
    Arrays.stream(verification).forEachOrdered(verificationStr −> {
      if (verificationStr.equals(data[i.get()])) {
        System.out.println(data[i.get()]);
      } else {
        System.err.println("test failed: " + data[i.get()]);
      }
      i.incrementAndGet();
    });

  }

}

/* Session−Log:

autark
authentisch
authentische
auto
auto
autobahn
autonom
bismarck
bleibe
brach
bruno
clan
clown

*/
```

```
1  /*
2   * HSR − Uebungen 'Algorithmen & Datenstrukturen 2'
3   * Version: Mon Oct 22 12:16:24 CEST 2018
4   */
5
6  package uebung06.as.aufgabe02;
7
8  import static org.junit.Assert.assertArrayEquals;
9
10 import java.util.Arrays;
11 import java.util.stream.IntStream;
12
13 import org.junit.Test;
14
15 public class RadixSortJUnitTest {
16
17   @Test
18   public void testRadixSort() {
19
20     final int LOOPS = 100;
21     final int MIN_STRING_LEN = 1;
22     final int MAX_STRING_LEN = 10;
23     final int ARRAY_LEN = 100;
24
25     IntStream.range(0, LOOPS).forEach(loop −> {
26       String[] strArr = new String[ARRAY_LEN];
27       IntStream.range(0, ARRAY_LEN).forEach(i −> {
28         int len = random(MIN_STRING_LEN, MAX_STRING_LEN);
29         char[] charArr = new char[MAX_STRING_LEN];
30         IntStream.range(0, len).forEach(
31           j −> charArr[j] = (char) random('a', 'z'));
32         String str = new String(charArr, 0, len);
33         strArr[i] = str;
34       });
35       String[] strArrSorted = strArr.clone();
36       Arrays.sort(strArrSorted);
37       new RadixSort().radixSort(strArr);
38       assertArrayEquals(strArrSorted, strArr);
39     });
40   }
41
42   /**
43    * Returns a random−number in the range from..to.
44    *
45    * @param from
46    *          The Lower−Bound (inclusive).
47    * @param to
48    *          The Upper−Bound (inclusive).
49    * @return The generated random−number.
50    */
51   private int random(int from, int to) {
52     return from + (int) (Math.random() * (to − from + 1));
53   }
54
55 }
56
57
58
```