Aufgabe 7

(a) Führen Sie *„Sortieren durch Einfügen“* lexikographisch aufsteigend und *in-situ* (*in-place*) so in einem Schreibtischlauf auf folgendem Feld (Array) aus, dass gleiche Elemente ihre relative Abfolge jederzeit beibehalten (also dass z. B. A_1 stets vor A_2 im Feld steht). Jede Zeile stellt den Zustand des Feldes dar, nachdem das jeweils nächste Element in die Endposition verschoben wurde. Der bereits sortierte Teilbereich steht vor |||. Gleiche Elemente tragen zwecks Unterscheidung ihre *„Objektidentität“* als Index (z. B. `"A1".equals("A2")` aber `"A1" != "A2"`)

L	A_1	B_1	F	A_2	B_2

L	A_1	B_1	F	A_2	B_2
A_1	L	B_1	F	A_2	B_2
A_1	B_1	L	F	A_2	B_2
A_1	B_1	F	L	A_2	B_2
A_1	A_2	B_1	F	L	B_2
A_1	A_2	B_1	B_2	F	L

(b) Ergänzen Sie die folgende Methode so, dass sie die Zeichenketten im Feld `a` lexikographisch aufsteigend durch Einfügen sortiert. Sie muss zum vorangehenden Ablauf passen, d. h. sie muss *iterativ* sowie *in-situ* (*in-place*) arbeiten und die relative Reihenfolge gleicher Elemente jederzeit beibehalten. Sie dürfen davon ausgehen, dass kein Eintrag im Feld null ist.

```
1  void sortierenDurchEinfuegen(String[] a) {
2    // Hilfsvariable:
3    String tmp;
4  }
```

```
5    static void sortierenDurchEinfuegen(String[] a) {
6      // Hilfsvariable:
7      String tmp;
8      for (int i = 1; i < a.length; i++) {
9        tmp = a[i];
10       int j = i;
11       while (j > 0 && a[j - 1].compareTo(tmp) >= 1) {
12         a[j] = a[j - 1];
13         j = j - 1;
14       }
15       a[j] = tmp;
16     }
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/InsertionSort.java