## Aufgabe 1: "StrinX" [12 Punkte]

Realisieren Sie eine Klasse **`StrinX`**, welche die Klassen `String` bzw. `StringBuffer` in Teilen nachahmt. Die Klasse `StrinX` soll dazu **intern den Inhalt als `char`-Array repräsentieren**.

Die Klasse soll dazu mindestens folgende **Konstruktoren** aufweisen:

Erzeugen eines leeren `StrinX`.

```
public StrinX() { /* creates an empty string */ }
```

Erzeugen eines `StrinX` mit den Zeichen des übergebenen Arrays.

```
public StrinX(char[] data) { /* Inits with characters in data */ }
```

Erzeugen eines `StrinX` Objekts, das die Buchstaben eines anderen `StrinX` übernimmt.

```
public StrinX(StrinX str)  { /* Inits with the characters of str */ }
```

Erzeugen eines `StrinX` Objekts, das die Buchstaben eines `String` übernimmt.

```
public StrinX(String str)  { /* Inits with the characters of str */ }
```

Außerdem soll die Klasse mindestens folgende **Methoden** aufweisen:

**length** – Berechnet und returniert die Länge des `StrinX`.

```
public int length() { /* Returns the length of the string */ }
```

**toString** – Wandelt den `StrinX` zu `java.lang.String` um und gibt ihn zurück.

```
public String toString() { /* Returns string as java.lang.String */ }
```

**substring** – Liefert einen Teilstring des `StrinX` ab Index `beginIndex`, mit Länge `count`. Das Zeichen an der Stelle `beginIndex` ist dabei noch Teil des Teilstrings. Werden ungültige Werte übergeben, soll `null` zurückgegeben werden.

```
public StrinX substring(int beginIndex, int count) { /* Returns a
substring consisting of count characters, starting at beginIndex */ }
```

**endsWith** – Überprüft ob der `StrinX` mit der übergebenen `suffix` endet. Falls `sufffix` den Wert `null` hat oder leer ist, soll `false` zurückgegeben werden.

```
boolean endsWith(char[] suffix) { /* Tests if this string ends with
the specified suffix. */ }
```

**compareTo** – Vergleich zwei als Parameter übergeben `StrinX` miteinander und gibt das Ergebnis des Vergleichs zurück. Achten Sie hier besonders auf die Entscheidung Klassen-Methode vs. Instanzmethode und deklarieren Sie die Methode entsprechend.

```
boolean startsWith(StrinX s1, StrinX s2) { /* Compares two strings
lexicographically. The result is a negative integer if s1
lexicographically precedes s2. The result is a positive integer if s1
lexicographically follows s2. The result is zero if s1 and s2 are
equal. */ }
```

Ihre `StrinX` soll wie folgt verwendet werden können:

```
class class StrinXDemo{
      public static void main(String [] args) {

            Out.print("\nConstructors---------------------------------");
            StrinX s1 = new StrinX();
            Out.print("\ns1: " + s1.toString());           // empty StrinX

            char [] hugo = {'H','u','g','o'};
            StrinX s2 = new StrinX(hugo);                 // "Hugo"
            Out.print("\ns2: " + s2.toString());

            StrinX s3 = new StrinX(s2);                   // also "Hugo"
            Out.print("\ns3: " + s3.toString());

            String str = null;
            StrinX s4 = new StrinX(str);                  // ""
            Out.print("\ns4: " + s4.toString());

            str = " Boss";
            StrinX s5 = new StrinX(str);                  // " Boss"
            Out.print("\ns4: " + s5.toString());

            Out.print("\nLENGTH---------------------------------");
            Out.print("\nlength: " + s2.length());         // 4

            Out.print("\nSUBSTRING---------------------------------");
            StrinX s6 = s2.substring(2,4);                          // "go"
            Out.print("\nsubstring: " + s6 + "\n");

            Out.print("\nSUBSTRING---------------------------------");
            s6 = s2.substring(-1,9999);                             // null
            Out.print("\nsubstring: " + s6 + "\n");

            Out.print("\nENDSWITH---------------------------------");
            char [] boss = {'B', 'o', 's', 's'};
            boolean found = s2.endsWith(boss);                  // true
            Out.print("\nendsWith: "+s2+" 'B', 'o', 's', 's'->"+found+"\n");

            char [] bosx = {'B', 'o', 's', 'X'};
            found = s2.endsWith(bosx);                          // false
            Out.print("\nendsWith: "+s2+ " 'B', 'o', 's', 'X'->"+found+"\n");
```

```
            Out.print("\nCOMPARE--------------------------------");
            StrinX s7 = s2;
            int comp = StrinX.compareTo(s2, s7);
            // 0
            Out.print("\n COMPARE: " + s2 + " with " + s7 + "-> " + comp + "\n");

            StrinX a = new StrinX("A");
            StrinX b = new StrinX("B");
            comp = StrinX.compareTo(a, b);                              // -1
            Out.print("\n COMPARE: " + a + " with " + b + "-> " + comp + "\n");
            comp = StrinX.compareTo(b, a);                              // 1
            Out.print("\n COMPARE: " + a + " with " + b + "-> " + comp + "\n");
            comp = StrinX.compareTo(b, null);                           // 1
            Out.print("\n COMPARE: " + a + " with " + b + "-> " + comp + "\n");


        }
}
```

Sie können sich das `StrinXDemo` über **MOODLE** herunterladen.

***Abgabe:***

➔ **`StrinX.java` mit der Klasse `StrinX`.**

## Aufgabe 2: "Map" [12 Punkte]

Gesucht ist eine Klasse **`Map`**, die es ermöglicht Schlüssel/Wert-Paar zu speichern. Als Schlüssel, sowie für die Werte sollen `Strings` vorgesehen werden. Unter einem eindeutigen Schlüssel soll der dazu gehörende Wert gespeichert werden. Schlüssel müssen in der Map eindeutig sein, Werte können mehrfach vorkommen.

Realisieren Sie die **`Map`** auf Basis von Arrays. Es soll denoch möglich sein theoretisch beliebig viele Schlüssel/Wert-Paare zu speichern.

Folgende Funktionalität soll Map unterstützen:

- **`store(key, value)`: Hinzufügen** eines Schlüssel/Wert-Paares zur Map **bzw. Ändern** des Wert zu einem bereits gespeicherten Schlüssel. Zeigen Sie mit dem Rückgabewert an, ob der Wert erfolgreich eingefügt werden konnte.
- **`del(key)`**: Löschen eines Schlüssel/Wert-Paares aus dem Map. Das als Parameter übergebene Schlüssel soll gesucht und die Schlüssel/Wert-Zuordnung aus der Liste entfernt werden. Zeigen Sie mit dem Rückgabewert an, ob der Schlüssel gefunden und somit aus der Map entfernt werden konnte.
- **`get(key)`**: Gibt den Wert, welcher zu dem Wert hinterlegt ist zurück; ansonsten `null`.
- **`size()`**: Gibt die Anzahl der Schlüssel/Wert-Paare in der Map zurückgibt.
- **`keys()`**: Gibt die Schlüssel der Map als neues Array zurückgibt. Die Reihenfolge der Schlüssel in der Liste soll dabei egal sein.

Ihre `Map` sollen wie folgt verwendet werden können:

```
class MapDemo{
  public static void main(String [] args) {

    Out.print("\nConstructors----------------------------------");
    Map map = new Map();
    Out.print("\nmap: " + map.toString());          // empty Map

    map.add("1", "A");                              // 1,A is no contained
    Out.print("\nmap: " + map.toString());          // (1,A)

    map.add("2", "B");                              // 2,B is no contained
    Out.print("\nmap: " + map.toString());          // (1,A)(2,B)

    map.add("2", "X");                              // 2 is changes to X
    Out.print("\nmap: " + map.toString());          // (1,A)(2,X)

    map.del("1");
    Out.print("\nmap: " + map.toString());          // (2,X)

    map.add("3", "C");                              // 3,C is no contained
    Out.print("\nmap: " + map.toString());          // (2,X)(3,C)

    Out.print("\nmap size: " + map.size());         // 2


    Out.print("\nmap get: " + map.get("2"));   // X
    Out.print("\nmap get: " + map.get("3"));   // C
    Out.print("\nmap get: " + map.get("1"));   // null
    Out.print("\nmap get: " + map.get(null));  // null
    String [] keys = map.keys();
    Out.print("\nmap keys: ");
    for (int i = 0; i < keys.length; i++) {
     Out.print("\n   " + keys[i]);                 // 2 3
    }


  }
}
```

Sie können sich das `TimerDemo` über **MOODLE** herunterladen.

***Abgabe:***

➔ **`Map.java` mit der Klasse `Map`.**