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.

```
int compareTo(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. */ }
```

toCharArray – Gibt den Inhalt des `StrinX` als `char[]` zurück .

```
char[] toCharArray() { /* Returns the content as char array.
*/ }
```

Ihre `StrinX` soll wie in `StrinXDemo` gezeigt verwendet werden können. Sie können sich dazu `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 dennoch möglich sein theoretisch beliebig viele Schlüssel/Wert-Paare zu speichern.

Folgende Funktionalität soll Map unterstützen:

- **`store(key, value)`: Hinzufügen** eines *neuen* Schlüssel/Wert-Paares zur Map. Zeigen Sie mit dem Rückgabewert an, ob der Wert erfolgreich (Schlüssel darf noch nicht in der Map vorhanden sein) eingefügt werden konnte.
- **`update(key, value)`: Ändern** des Wert zu einem bereits gespeicherten Schlüssel. Zeigen Sie mit dem Rückgabewert an, ob der Wert erfolgreich geändert werden konnte.
- **`remove(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.
- **`value(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 `MapDemo` gezeigt verwendet werden können. Sie können sich dazu `MapDemo` über **MOODLE** herunterladen.

Abgabe:

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