

תוכנה 1

תרגול 3:

עבודה עם מחרוזות (Strings)

מתודות (Methods)

מחרוזות (STRINGS)

מחרוזות

- מחרוזות הן אובייקטים המכילים רצף של תווים.

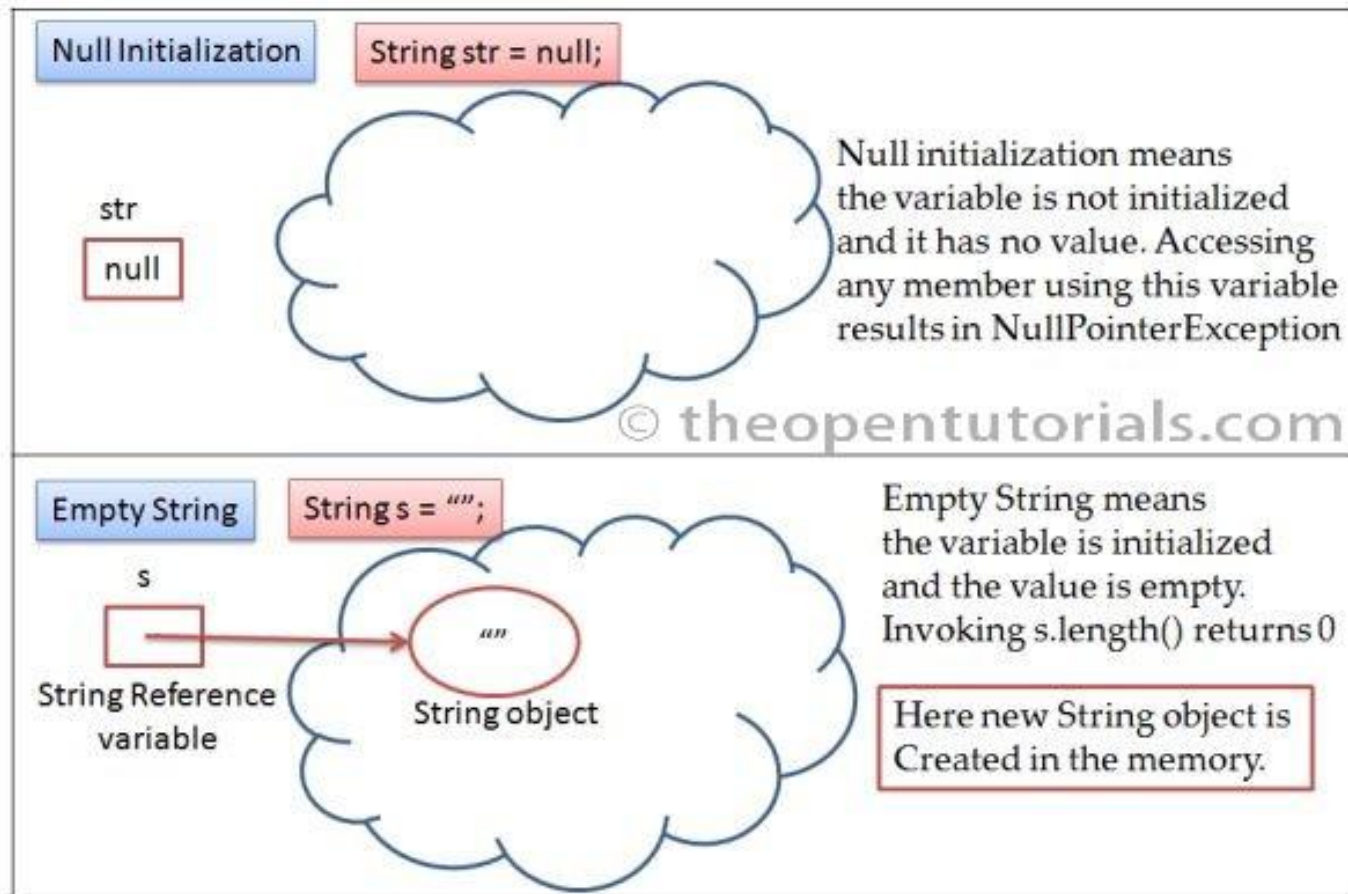
String s = "Hello";

index	0	1	2	3	4
character	H	e	l	l	o

- כל אלמנט במחרוזת הוא מסוג char.
- האינדקס של התו הראשון הוא 0.
- אורך המחרוזת מוחזר ע"י הפונקציה length().
- שרשור מחרוזות נעשה ע"י האופרטור +

String s2 = s + " World" + 5 // "Hello World5"

Null Initialization V/s Empty String



מחרוזות – פונקציות בדיקה

Method	Description
<code>equals(str)</code>	whether two strings contain the same characters
<code>equalsIgnoreCase(str)</code>	whether two strings contain the same characters, ignoring upper vs. lower case
<code>startsWith(str)</code>	whether one contains other's characters at start
<code>endsWith(str)</code>	whether one contains other's characters at end
<code>contains(str)</code>	whether the given string is found within this one

כדי להשוות שתי מחרוזות מבחינת תוכן יש להשמש בפונקצייה `equals()` ולא באופרטור `==` שבודק אם מדובר באותו אובייקט

מחרוזות – פונקציות שימושיות

Method name	Description
<code>indexOf(str)</code>	index where the start of the given string appears in this string (-1 if not found)
<code>substring(index1, index2)</code> or <code>substring(index1)</code>	the characters in this string from <i>index1</i> (inclusive) to <i>index2</i> (<u>exclusive</u>); if <i>index2</i> is omitted, grabs till end of string
<code>toLowerCase()</code>	a new string with all lowercase letters
<code>toUpperCase()</code>	a new string with all uppercase letters

המימוש של הפונקציות לעיבוד מחרוזות יחזיר תמיד מחרוזת חדשה ולא יבצע שינויים על המחרוזת המקורית שעליה נקראה הפונקציה (**Strings are immutable**)
(in Java).

מחרוזות – פיצול לחלקים

Method name	Description
<code>split(DelimiterString)</code>	Splits the string into tokens using the given delimiter string. Returns an array of Strings.

```
String str= "Another useful example";  
String[] tokens = str.split(" ");  
//tokens = {"Another", "useful", "example"}
```

הדפסת מחרוזות ומספרים

```
int a=1805;
```

```
double d=123.456789;
```

```
System.out.println ("a=" + a);           //"a=1805";
```

```
System.out.format ("a=%d\n",a);          //"a=1805";
```

```
System.out.format ("d=%.2f\n",d);        //"d=123.46"
```

```
System.out.format ("d=%20.10f\n",d);     //"d=    123.4567890000"
```

%n - platform-specific line separator

%d – decimal

%f – float

<http://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

מתודות (METHODS)

בניית תוכנית תוך שימוש ראוי במתודות

Span - הגדרה

- בהינתן מערך של מספרים וערך כלשהו נגדיר את ה- span של הערך כמספר האברים (כולל) בין שני המופעים הקיצוניים של הערך במערך.

• דוגמאות:

- המערך $[1, 2, 1, 1, 3]$ והערך 1 – ה span הוא 4
- המערך $[1, 4, 2, 1, 1, 4, 1, 4]$ והערך 1 – ה span הוא 7
- המערך $[1, 4, 2, 1, 1, 4, 1, 4]$ והערך 2 – ה span הוא 1

Max Span

- Max-Span יהיה ה span המקסימלי על פני כל הערכים במערך מסוים
- נרצה לממש פונקציה שבהינתן מערך של מספרים שלמים תחזיר את ה Max-Span שלו
- דוגמאות:
 - המערך $[1,2,1,1,3]$ – ה-maxSpan הוא 4
 - המערך $[1,4,2,1,1,4,1,4]$ – ה-maxSpan הוא 7

נתחיל לעבוד

- נפתח פרויקט חדש בשם MaxSpan
- נתחיל לכתוב תכנית בדיקה לפתרון שלנו



תכנית בדיקה

- נגדיר מחלקה חדשה עבור הבדיקות
`il.ac.tau.cs.sw1.maxspan.tests.TestMaxSpan`
- החלק הראשון - חבילה (package)
http://en.wikipedia.org/wiki/Java_package
- כעת נכתוב את המקרים שנרצה לבדוק:

תכנית בדיקה

```
int[] array = null;
int maxSpan;

array = new int[]{1, 2, 1, 1, 3};
maxSpan = MaxSpan.maxSpan(array);
if (maxSpan != 4) {
    System.out.println(Arrays.toString(array) + " expected: 4, result: " + maxSpan);
} else {
    System.out.println(Arrays.toString(array) + " correct!");
}
array = new int[]{1, 4, 2, 1, 1, 4, 1, 4};
maxSpan = MaxSpan.maxSpan(array);
if (maxSpan != 7) {
    System.out.println(Arrays.toString(array) + " expected: 7, result: " + maxSpan);
} else {
    System.out.println(Arrays.toString(array) + " correct!");
}
```

למה המהדר כועס?

- לא מכיר את Arrays?

- `import java.util.Arrays;`

- לא מכיר את MaxSpan?

- `import il.ac.tau.cs.sw1.maxspan.MaxSpan;`

- אבל לא מוגדרת מחלקה כזו...מה לעשות?

- בואו נקשיב להמלצה של אקליפס (QuickFix)

- קיצור מקשים: Ctrl+1

ועכשיו לפתרון

```
public static int maxSpan(int[] array) {
    int max = 0;
    for (int i = 0; i < array.length; i++) {
        int j = array.length - 1;
        for ( ; j >= i; j--) {
            if (array[i] == array[j]) {
                break;
            }
        }
        int span = j - i + 1;
        if (max < span) {
            max = span;
        }
    }
    return max;
}
```


בדיקה, Refactor ושדרוג הקוד (?)

- נבדוק שתכנית הבדיקה עובדת
- בואו נכתוב את הפונקציה בצורה יותר "נכונה"
- ראשית נשנה את שם המחלקה, נשתמש ב-Refactor
- דיון: כתיבת הפונקציה בצורה "נכונה"
 - יעילות
 - מודולריות, פתרון Top-down
 - הבנת הקוד
 - אפשרות לשינויים עתידיים

"top-down" תכנון

```
int maxSpan(int[]) {...}
```

For each value in the array

Compute its span in the array

Return the largest span found

```
int[] values(int[]) {...}
```

```
int span(int, int[]) {...}
```

Create an empty output array

For every element in the input array

Find the first occurrence of value

Find the last occurrence of value

Span = last - first + 1

If the output array does not already contain the current value, add it to the output array

```
int lastIndexOf(int, int[]) {...}
```

```
int firstIndexOf(int, int[]) {...}
```

```
boolean contains(int[], int, int)
```

```
void add(int[], int, int) {...}
```

We also need to adjust the output array size...

הפונקציה הראשית

```
public static int maxSpan(int[] nums) {  
    int max = 0;  
    for (int value: values(nums)) {  
        max = Math.max(max, span(value, nums));  
    }  
    return max;  
}
```

וחלק מפונקציות העזר

```
private static int span(int value, int[] nums) {  
    return lastIndexOf(value, nums) - indexOf(value, nums) + 1;  
}
```

```
private static int[] values(int[] nums) {  
    int[] values = new int[nums.length];  
    int nextIndex = 0;  
  
    for (int i = 0; i < nums.length; i++) {  
        if (!contains(values, nextIndex, nums[i])) {  
            add(values, nextIndex++, nums[i]);  
        }  
    }  
  
    return Arrays.copyOf(values, nextIndex);  
}
```

```

private static int lastIndexOf(int value, int[] nums) {
    for (int i = nums.length - 1; i >=0; i--) {
        if (nums[i] == value) {
            return i;
        }
    }
    // should never get here
    return -1;
}

private static int firstIndexOf(int value, int[] nums) {
    int index = -1;
    for (int i = 0; i < nums.length; i++) {
        if (nums[i] == value) {
            index = i;
            break;
        }
    }
    return index;
}

```

```

private static void add(int[] values, int position, int value) {
    values[position] = value;
}

private static boolean contains(int[] temp, int tempLength, int value) {
    for (int i = 0; i < tempLength; i++) {
        if (temp[i] == value) {
            return true;
        }
    }
    return false;
}

```