

תוכנה 1

תרגול 3: מתודות ומחלקות
רובי בויים ומתי שמרת

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

Span

■ 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

מה בתכנית

פתרון לבעיית ה $\max\text{Span}$ ■

■ בדיקות קוד

■ שימוש באקליפס

■ שתי גישות לפתרון

■ מחרוזות

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;
}
```

ושדרוג הקוד Refactor(?) בבדיקה,

- 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

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;  
}
```

חלק 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);  
}
```

והשאר

```
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;
}
```

והשאר

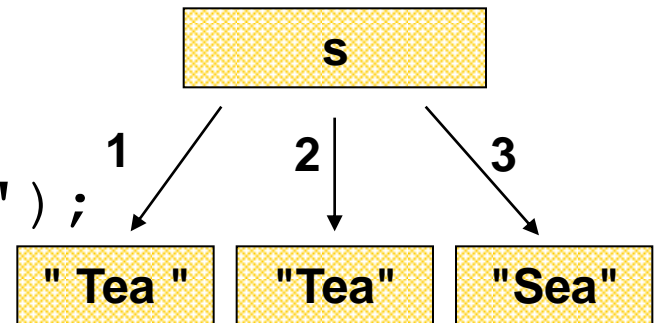
```
private static void add(int[] values, int position, int value) {
    values[position] = value;
}

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;
}
```

מחרוזות

- מחרוזת היא רצף של תווים
- הטיפוס String מייצג מחרוזת ב Java
- ב Java משהש מרגע שנוצרה מחרוזת היא אינה ניתנת לשינוי (immutable)
- ההפניה למחרוזת כמובן יכולה להשתנות

```
String s = " Tea ";  
s = s.trim();  
s = s.replace('T', 'S');
```



String Constructors

- Use implicit constructor:

```
String s = "Hello";
```

Instead of:

```
String s = new String("Hello");
```

פעולות על מחרוזות

■ המחלקה String מגדירה עשרות פעולות שניתן לבצע על מחרוזת

- charAt
- endsWith
- boolean equals(..)
- indexOf
- length
- split
- startsWith
- substring
- ...

שימוש במחרוזות - דוגמא

קריאה לשרות על מחרוזת לא תשנה את המחרוזת ■

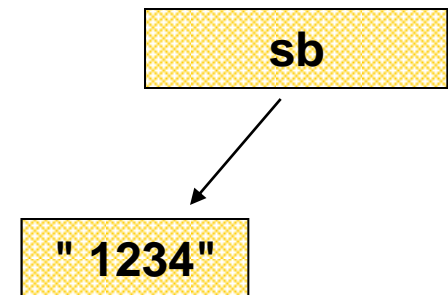
```
String testString = "agjSDRoir";
String validChars = "atgc";
testString = testString.toLowerCase();

for (int i = 0; i < testString.length(); i++)
{
    char c = testString.charAt(i);
    if (validChars.indexOf(c) == -1) {
        System.out.printf("Bad character[" + c +
            "]" + " at position " + i);
    }
}
```

המחלקה `StringBuilder`

- מייצג מחרוזות ניתנת לשנוי (mutable)
- מאפשר לבצע שינוי במחרוזת קיימת מבלי ליצור אובייקטים חדשים
- שירותים חשובים: `append` ו-`insert`

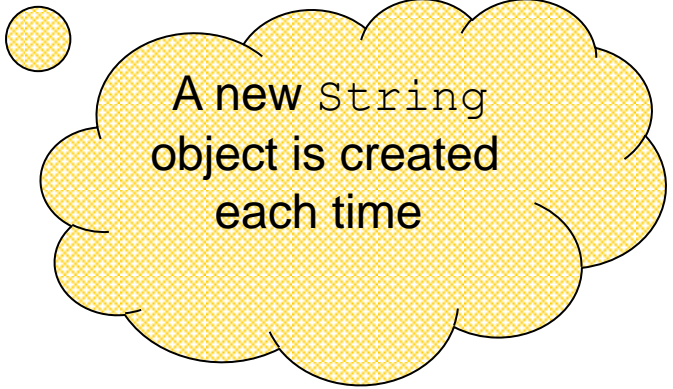
```
StringBuilder sb = new StringBuilder("123");  
sb.append(4);
```



StringBuffer vs. String

■ Inefficient version using String

```
public static String duplicate(String s, int times) {  
    String result = s;  
    for (int i = 1; i < times; i++) {  
        result = result + s;  
    }  
    return result;  
}
```



A new String
object is created
each time

StringBuffer vs. String (cont.)

- More efficient version with StringBuilder:

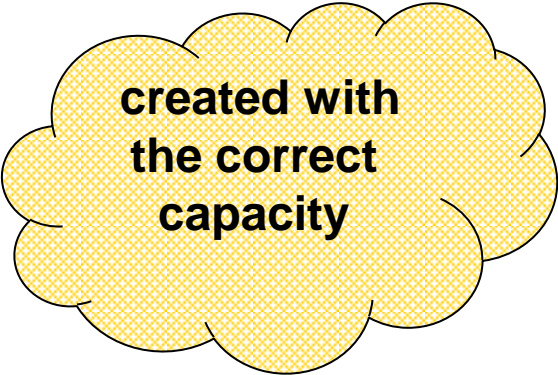
```
public static String duplicate(String s, int times) {  
    StringBuilder result = new StringBuilder(s);  
    for (int i = 1; i < times; i++) {  
        result.append(s);  
    }  
    return result.toString();  
}
```



StringBuffer vs. String (cont.)

■ Even more efficient version:

```
public static String duplicate(String s, int times) {  
    StringBuilder result =  
        new StringBuilder(s.length() * times);  
    for (int i = 0; i < times; i++) {  
        result.append(s);  
    }  
    return result.toString();  
}
```



created with
the correct
capacity