

# תוכנה 1

תרגול מס' 3

עבודה עם מחרוזות (Strings)

מתודות (Methods)

העברת פרמטרים

# שלבי הפיתוח - חזרה קצרה

Java source files  
(.java)

```
class Foo {  
    /* ... */  
}
```

javac

Java bytecode files  
(.class/.jar)

```
...  
iconst_0  
iaload  
istore_1  
jsr 19  
iload_1  
...
```

ישנם שני שלבים נפרדים:

שלב פיתוח התוכנית

- בשלב זה אנו משתמשים במהדר (קומפיילר) כדי להמיר קבצי .java (קבצי טקסט הקריאים למתכנת) לקבצי .class שנועדו עבור המפרש (אינטרפרטר).

קומפילציה

# שלבי הפיתוח - חזרה קצרה

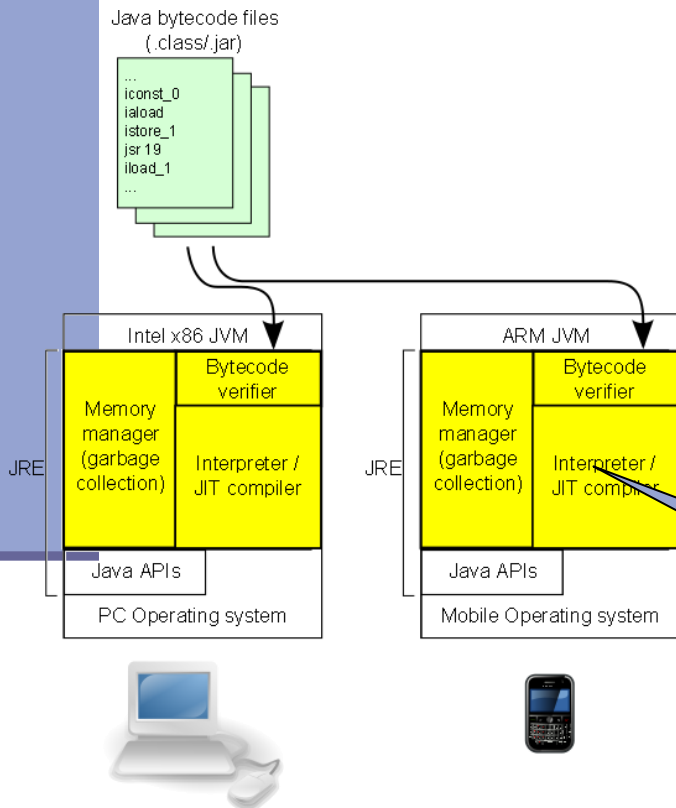
ישנם שני שלבים נפרדים:

שלב הרצת התוכנית

בשלב זה אנו משתמשים במפרש כדי להריץ את קבצי ה-class שיצרנו.

ב-Java אותו קובץ class יכול לרוץ בסביבות שונות אם קיים עבורו מפרש.

הרצה



# שלבי הפיתוח – חזרה קצרה

■ ה-**JDK** (Java Development Kit) נדרש לתהליך הפיתוח

■ קומפיילר

■ ה-**JRE** (Java Runtime Environment) נדרש להרצת תוכניות

■ JVM (Java Virtual Machine)

■ הספריות הסטנדרטיות

# Java Virtual Machine

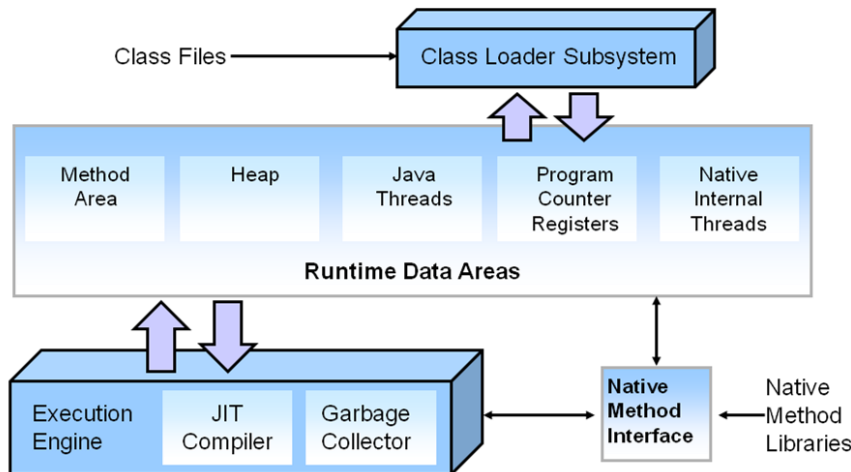
■ ה-JVM היא "מכונה וירטואלית" המריצה תוכניות  
Java

■ יודעת לטעון תוכניות

■ יודעת לוודא את תקינות הקבצים הנטענים

■ מכילה את המפרש (Interpreter)

HotSpot JVM: Architecture



Java תכנות מתקדם בשפת  
אוניברסיטת תל אביב

# העברת פרמטרים

# מה יהיה פלט התכנית הבאה?

```
public class Main {
```

```
    public static void main(String[] args) {  
        int x = 3;  
        System.out.println(x);  
        inc(x);  
        System.out.println(x);  
    }
```

```
    private static void inc(int x) {  
        x++;  
    }
```

```
}
```

תשובה:

3

3

למה??

# By value, By reference

- העברת פרמטרים by value:
- הפרמטר למעשה מועתק, נוצר עותק נוסף שלו אשר נשלח למתודה.
- משתנים פרימיטיביים מועברים by value
- שינוי הפרמטר בתוך המתודה לא יראה מחוץ למתודה.
- אנלוגיה: שליחת מסמך word במייל, אם המקבל משנה את העותק שלו, העותק הנוכחי לא מושפע מכך.



# מה יהיה פלט התכנית הבאה?

```
5 public class Main {  
6  
7 public static void main(String[] args) {  
8     int [] arr = {1,2,3};  
9     System.out.println(Arrays.toString(arr));  
0     inc(arr);  
1     System.out.println(Arrays.toString(arr));  
2 }  
3  
4 private static void inc(int[] arr) {  
5     for(int i =0; i<arr.length; i++)  
6         arr[i]++;  
7  
8 }  
9  
0  
1 }  
2
```

תשובה:  
[1,2,3]  
[2,3,4]  
למה??

# By value, By reference

- העברת פרמטרים by reference:
- הכתובת של האובייקט מועברת כפרמטר.
- אובייקטים מועברים by reference
- שינוי הפרמטר בתוך המתודה נשמר גם מחוץ למתודה
- אנלוגיה: שליחת כתובת של גוגל דוק, כל שינוי שצד אחד עושה במסמך, נראה גם אצל הצד השני.

# מה יהיה פלט התכנית הבאה?

```
5 public class Main {  
6  
7 public static void main(String[] args) {  
8     int [] arr = {1,2,3};  
9     System.out.println(Arrays.toString(arr));  
0     change(arr);  
1     System.out.println(Arrays.toString(arr));  
2 }  
3  
4 private static void change(int[] arr) {  
5     arr = new int [4];  
6  
7 }  
8  
9  
0 }  
1
```

תשובה:  
[1,2,3]  
[1,2,3]  
למה??

# By reference

■ כאשר מעבירים משתנה **By reference**, הכתובת עצמה מועתקת **by value**.

■ נחזור לדוגמת הגוגל דוק: נניח ששלחנו למישהו כתובת של מסמך גוגל דוק, והוא מחק את הכתובת. האם אצלנו הכתובת נמחקה גם??

# מחרוזות (STRINGS)

# מחרוזות - חזרה

■ 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"

# מחרוזות - השוואה

■ נניח ונרצה להשוות שתי מחרוזות (לבדוק האם הן שוות).

```
public static void main(String[] args) {  
    String s1 = new String("hello");  
    String s2 = new String ("hello");  
    System.out.println(s1.equals(s2));  
    System.out.println(s1 == s2);  
}
```

true

false

■ מה יודפס למסך? למה?

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

# 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



# 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> (exclusive); 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 – decimanl*

*%f – float*

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

בניית תוכנית תוך שימוש ראוי במתודות

# מתודות (METHODS)

# 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](http://en.wikipedia.org/wiki/Java_package)

■ כעת נכתוב את המקרים שנרצה לבדוק:



# תכנית בדיקה

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

# 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"
- דיון: כתיבת הפונקציה בצורה "נכונה"
- יעילות
- מודולריות, פתרון Top-down
- הבנת הקוד
- אפשרות לשינויים עתידיים

# הפונקציה הראשית

נרצה לעבור על כל ערך  
פעם אחת בלבד  
(יעילות)

```
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);  
}
```

# השאר

```
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;
}
```

# השאר

```
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;
}
```

# "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...



# סיכום

- מה ההבדל העיקרי בין שני הפתרונות לבעיית ה-  
maxSpan?
- מדוע הפיתרון השני, על אף היותו ארוך יותר, הוא  
עדיף?
- דרך העבודה על תכנית צריכה להיות top-down.  
נתחיל מבדיקות: נגדיר מהי התנהגות נכונה של  
התכנית. רק לאחר מכן נעבור למימוש עצמו.
- נחלק לפונקציות בצורה בה כל פונקציה אחראית על  
פעולה אחת בלבד. נבנה תכנית מודולרית ככל הניתן.

הסוף...