

תוכנה 1

תרגול 2: מחרוזות, מערכים, לולאות והתמודדות עם שגיאות

מחרוזות

מחרוזות

- מחרוזות הן אובייקט המחזיק אוסף של תווים.
- דוגמאות לפונקציות מהמחלקה `:String`

```
String str1 = "Hello";  
char c = str1.charAt(0);  
String str2 = str1.toUpperCase();  
int strLength = str1.length();
```

`// c == 'H'`
`// str2 == "HELLO"`
`// strLength == 5`

■ אופרטור שרשור:

- `"Hello " + "World"` is `"Hello World"`
- `"19" + 8 + 9` is `"1989"`

עוד ב-

<http://docs.oracle.com/javase/8/docs/api/index.html?java/lang/String.html>

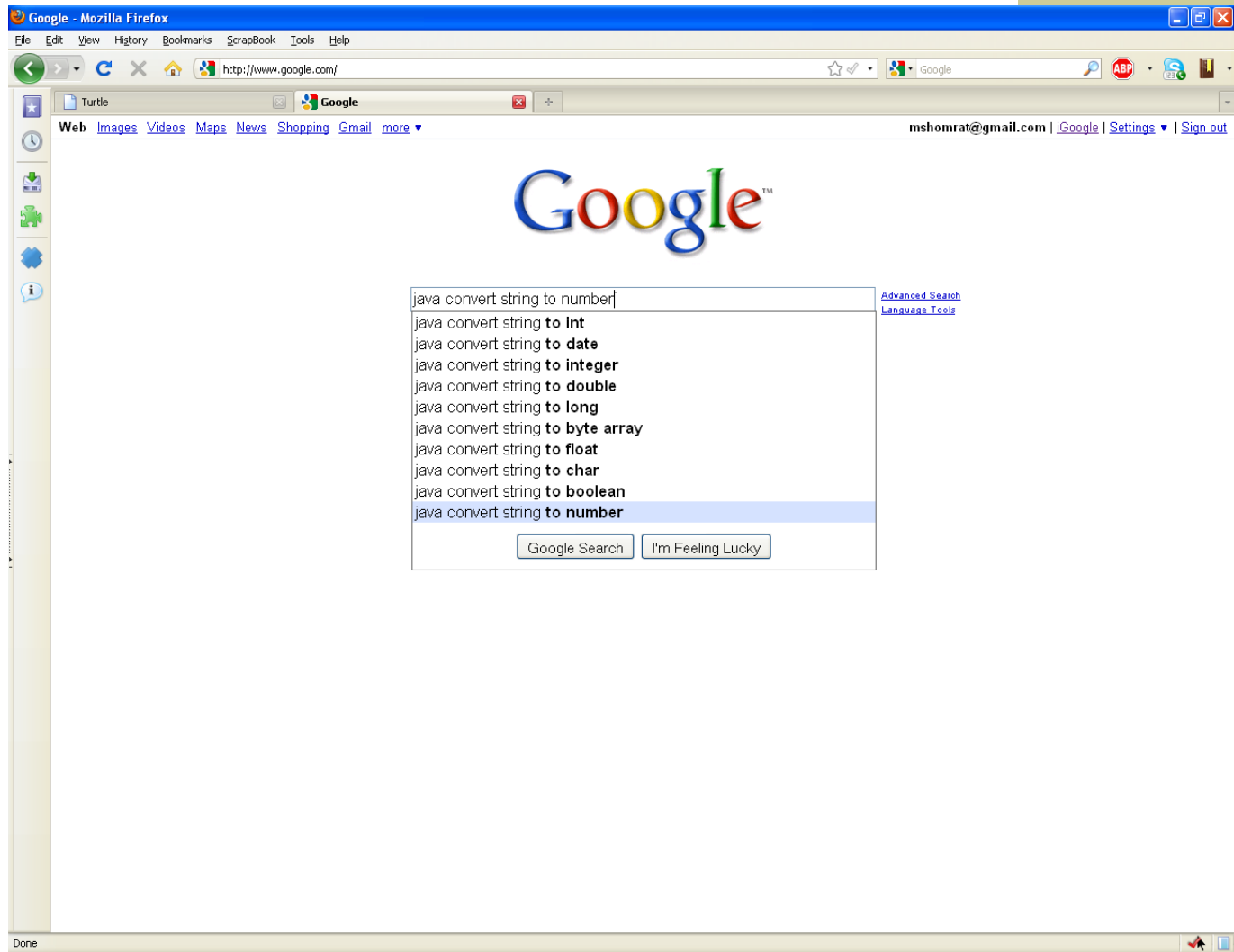
המרת מחרוזות למספרים

- `Long.parseLong`
- `Integer.parseInt`
- `Short.parseShort`
- `Byte.parseByte`
- `Double.parseDouble`
- `Float.parseFloat`
- `Boolean.parseBoolean`

```
public static void main(String[] args){  
    int i = Integer.parseInt("1");  
    double d = Double.parseDouble("-12.45e2");  
}
```

```
// i==1  
// d== -1245.0
```

איך נדע אילו פקודות קיימות...?



איך נדע אילו פקודות קיימות...?

OVERVIEW PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3
java.lang

Class String

java.lang.Object
 java.lang.String

All Implemented Interfaces:
Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are `String`s. `String`s are constant; their values cannot be changed after they are created. `String` buffers support n

```
String str = "abc";
```

מחרוזות ותווים

- כתוב תוכנית שמקבלת תו כארגומנט ומדפיסה:
 - את התו
 - את התו העוקב לו

```
public static void main(String[] args) {  
    char c = args[0].charAt(0);  
    char c1 = (char)(c + 1);  
    System.out.println(c + "\t" + c1);  
}
```

מחרוזות ותווים

■ ב-Java 17, ניתן להגדיר מחרוזת באמצעות **Text Blocks**

```
String textBlock1 = """
    red
    green
    blue
    """
```

// == red\ngreen\nblue\n

■ וללא ירידת שורה בסוף?

```
String textBlock2 = """
    red
    green
    blue"""
```

// == red\ngreen\nblue

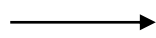
מערכים

מערכים

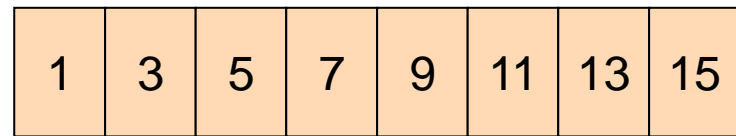
- **מערך:** מבנה נתונים בגודל קבוע מראש אשר שמור מספר איברים מאותו הטיפוס.
- לדוגמא: מערך עם ערכים אי זוגיים
- מדובר במערך של int בשם odds:

```
int[] odds = new int[8];
```

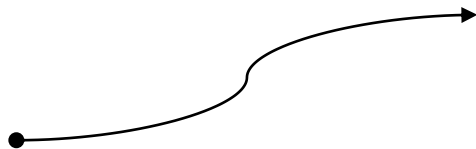
אינדקס מתחיל מ-0



0 1 2 3 4 5 6 7



odds
reference

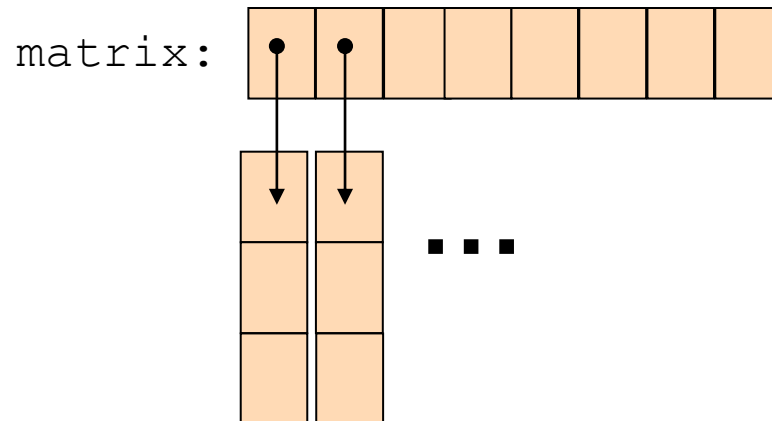


odds.length == 8

```
Odds [2] = 5
```

Array Variables

- An array is denoted by the `[]` notation
- Examples:
 - `int[] odds;`
 - `String[] names;`
 - `int[][] matrix; // an array of arrays`



Loop through Arrays

- By promoting the array's index:

```
for (Initialization int i = 0; condition i < months.length; step i++) {  
    System.out.println(months[i]);  
}
```

The variable month is assigned the next element in each iteration

- foreach:

```
for (String month: months) {  
    System.out.println(month);  
}
```

Array Creation and Initialization

- What is the output of the following code:

```
int[] odds = new int[8];  
for (int i = 0; i < odds.length; i++) {  
    System.out.print(odds[i] + " ");  
    odds[i] = 2 * i + 1;  
    System.out.print(odds[i] + " ");  
}
```

Array creation: all elements get the **default value** for their type (0 for `int`)

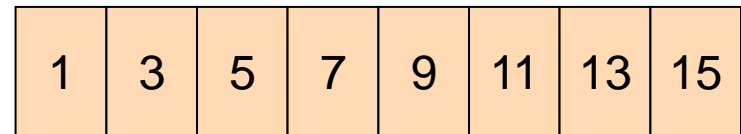
- Output:

0 1 0 3 0 5 0 7 0 9 0 11 0 13 0 15

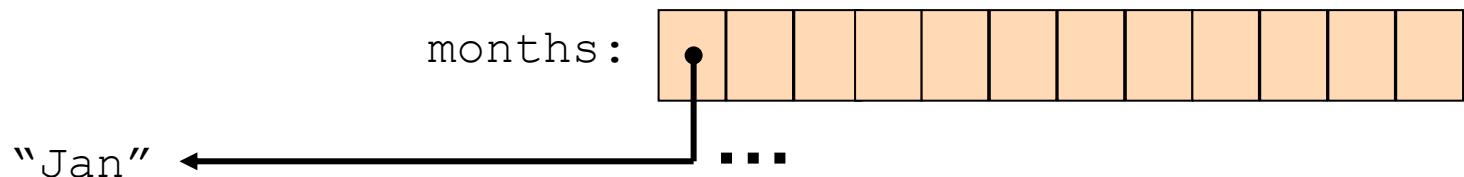
Array Creation and Initialization

- Creating and initializing small arrays with *a-priori* known values:

- `int[] odds = {1, 3, 5, 7, 9, 11, 13, 15};`



- `String[] months =
{"Jan", "Feb", "Mar", "Apr",
"May", "Jun", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"};`



Operations on arrays

- The class Arrays provide operations on array
 - Copy
 - Sort
 - binarySearch
 - Fill
 - ...
- [java.util.Arrays](http://docs.oracle.com/javase/6/docs/api/index.html?java/util/Arrays.html)
<http://docs.oracle.com/javase/6/docs/api/index.html?java/util/Arrays.html>

Copying Arrays

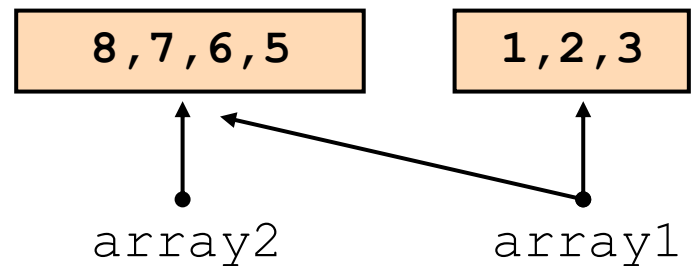
- Assume:

```
int[] array1 = {1, 2, 3};
```

```
int[] array2 = {8, 7, 6, 5};
```

- Naïve copy:

```
array1 = array2;
```



- What's wrong with this solution?

Copying Arrays

■ **Arrays.copyOf**

- 1st argument: the original array
- 2nd argument: the length of the copy

```
int[] arr1 = {1, 2, 3};  
int[] arr2 = Arrays.copyOf(arr1, arr1.length);
```

■ **Arrays.copyOfRange**

- 1st argument: the original array
- 2nd initial index of the range to be copied, inclusive
- 3rd argument: final index of the range to be copied, exclusive

Question

- What is the output of the following code:

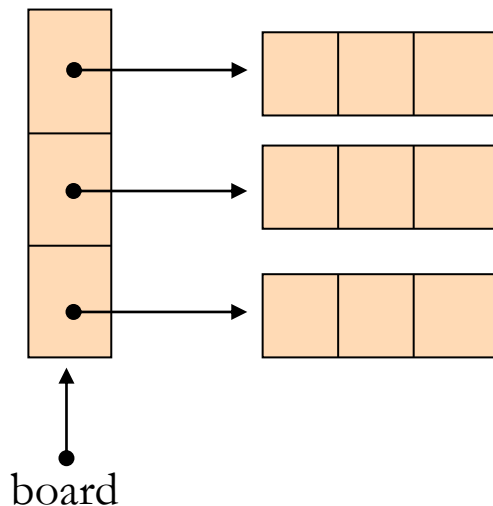
```
int[] odds = {1, 3, 5, 7, 9, 11, 13, 15};  
int newOdds[] =  
    Arrays.copyOfRange(odds, 1, odds.length);  
for (int odd: newOdds) {  
    System.out.print(odd + " ");  
}
```

Output: 3 5 7 9 11 13 15

2D Arrays

- There are no 2D arrays in Java but ...
- you can build array of arrays:

```
char [][] board = new char [3] [] ;  
for (int i = 0; i < 3; i++)  
    board[i] = new char [3] ;
```



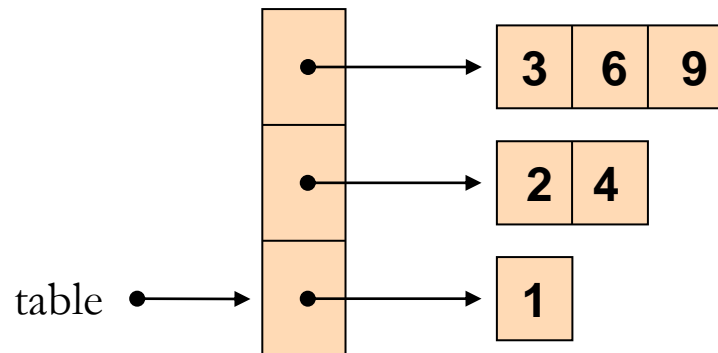
Or equivalently:

```
char [][] board = new char [3][3];
```

2D Arrays

- A more compact table:

```
int[][] table = new int[10][];  
for (int i = 0; i < 10; i++) {  
    table[i] = new int[i + 1];  
    for (int j = 0; j <= i; j++) {  
        table[i][j] = (i + 1) * (j + 1);  
    }  
}
```





לולאות ותנאים

Fibonacci

- Fibonacci series

1, 1, 2, 3, 5, 8, 13, 21, 34

- Definition:

- $\text{fib}(0) = 1$

- $\text{fib}(1) = 1$

- $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$



If-Else Statement

```
public class Fibonacci {  
    ...  
  
    /** Returns the n-th Fibonacci element */  
    public static int computeElement(int n) {  
        if (n==0)  
            return 1;  
        else if (n==1)  
            return 1;  
        else  
            return computeElement(n-1) + computeElement(n-2);  
    }  
}
```

Assumption:
 $n \geq 0$

Can be
removed

Switch Statement

```
public class Fibonacci {  
    ...  
  
    /** Returns the n-th Fibonacci element */  
    public static int computeElement(int n) {  
        switch(n) {  
            case 0:  
                return 1;  
            case 1:  
                return 1;  
            default:  
                return computeElement(n-1) + computeElement(n-2);  
        }  
    }  
}
```

Assumption:
 $n \geq 0$

can be placed
outside the switch

Switch Statement

```
public class Hello {  
    ...  
  
    /** Returns the n-th Fibonacci element */  
    public static int computeElement(int n) {  
        switch(n) {  
            case 0:  
                System.out.print("Hello");  
            case 1:  
                System.out.print("World");  
            default:  
                System.out.print("!");  
        }  
    }  
}
```

Switch Statement

```
public class Hello {  
    ...  
  
    /** Returns the n-th Fibonacci element */  
    public static int computeElement(int n) {  
        switch(n) {  
            case 0:  
                System.out.print("Hello");  
            case 1:  
                System.out.print("World");  
                break;  
            default:  
                System.out.print("!");  
        }  
    }  
}
```

Switch Statement

```
public class Fibonacci {  
    ...  
  
    /** Returns the n-th Fibonacci element */  
    public static int computeElement(int n) {  
        switch(n) {  
            case 0:  
                return 1;  
            case 1:  
                return 1;  
                break;  
            default:  
                return computeElement(n-1) + computeElement(n-2);  
        }  
    }  
}
```

Assumption:
 $n \geq 0$

Compilation Error:
Unreachable Code

Switch Expression

■ ב-17 Java אפשר גם:

```
public class Fibonacci {  
    ...  
  
    /** Returns the n-th Fibonacci element */  
    public static int computeElement(int n) {  
        int next_n = switch(n){  
            case 0 -> 1;  
            case 1 -> 1;  
            default -> computeElement(n-1) + computeElement(n-2);  
        };  
        return next_n;  
    }  
}
```

for vs. while

- The following two statements are almost equivalent:

```
for(int i = 0 ; i < n ; i++)  
    System.out.println(computeElement(i));
```

```
int i = 0;  
while (i < n) {  
    System.out.println(computeElement(i));  
    i++;  
}
```

Variable `i` is not defined
outside the for block

while vs. do while

- The following two statements are equivalent if and only if $n > 0$:

```
int i = 0;
while (i < n) {
    System.out.println(computeElement(i));
    i++;
}
```

```
int i = 0;
do {
    System.out.println(computeElement(i));
    i++;
} while (i < n);
```

עקרונות בתכנות

Iterative Fibonacci

- A loop instead of a recursion

```
static int computeElement(int n) {  
    if (n == 0 || n == 1)  
        return 1;  
  
    int prev = 1;  
    int prevPrev = 1;  
    int curr = 2;  
  
    for (int i = 2 ; i < n ; i++) {  
        prevPrev = prev;  
        prev = curr;  
        curr = prev + prevPrev;  
    }  
  
    return curr;  
}
```

Assumption:
 $n \geq 0$

Must be initialized.
Why?

~~1~~ ~~1~~ 2

prevPrev

~~1~~ ~~2~~ 3

prev

~~2~~ ~~3~~ 5

curr

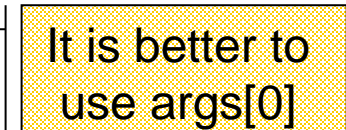
נתונים במקום חישוב

- בתרגום רקורסיה ללולאה אנו משתמשים במשתני עזר לשמירת המצב `prev` ו-`curr` - `prevPrev`
- הלולאה "זוכרת" את הנקודה שבה אנו נמצאים בתהליך החישוב
- **דין: יעילות לעומת פשטות.**
אצלינו בקורס, נעדיף פשטות

For Loop

- Printing the first n elements:

```
public class Fibonacci {  
    public static int computeElement(int n) {  
        ...  
    }  
  
    public static void main(String[] args) {  
        for(int i = 0 ; i < 10 ; i++) {  
            System.out.println(computeElement(i));  
        }  
    }  
}
```



מודולריות, שכפול קוד ויעילות

- יש כאן חוסר יעילות מסוים:
- לולאת ה-`for` חוזרת גם ב-`main` וגם ב-`computeElement`. לכאורה, במעבר אחד ניתן גם לחשב את האיברים וגם להדפיס אותם
- כמו כן כדי לחשב איבר בסדרה איננו משתמשים בתוצאות שכבר חישבנו (של איברים קודמים) ומתחילים כל חישוב מתחילתו

מודולריות, שכפול קוד ויעילות

- מתודה (פונקציה) צריכה לעשות דבר אחד בדיוק!
 - ערוב של חישוב והדפסה פוגע במודולריות (מדוע?)
- היזהרו משכפול קוד!
 - קטע קוד דומה המופיע בשתי פונקציות שונות יגרום במוקדם או במאוחר לבאג בתוכנית (מדוע?)
- את בעיית היעילות (הוספת מנגנון memoization) אפשר לפתור בעזרת מערכים (תרגיל)

התמודדות עם שגיאות

Compilation vs. Runtime Errors

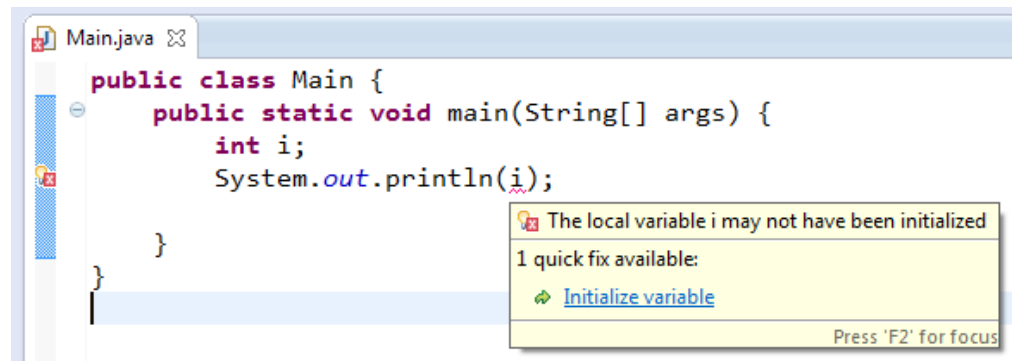
- שגיאות קומפילציה (הידור): שגיאות שניתן "לתפוס" בעת קריאת הקובץ והפיכתו ל-bytecode ע"י המהדר
- דוגמאות:

Syntax error on token "Class", class expected

```
Class MyClass {  
    void f() {  
        int n=10;  
  
        void g() {  
            int m = 20;  
        }  
    }  
}
```

Syntax error, insert "}" to complete MethodBody

```
...  
int i;  
System.out.println(i);  
...
```



בדרך כלל קשורות ל:

תחביר, תאימות טיפוסים, הגדרה לפני שימוש

Compilation vs. Runtime Errors

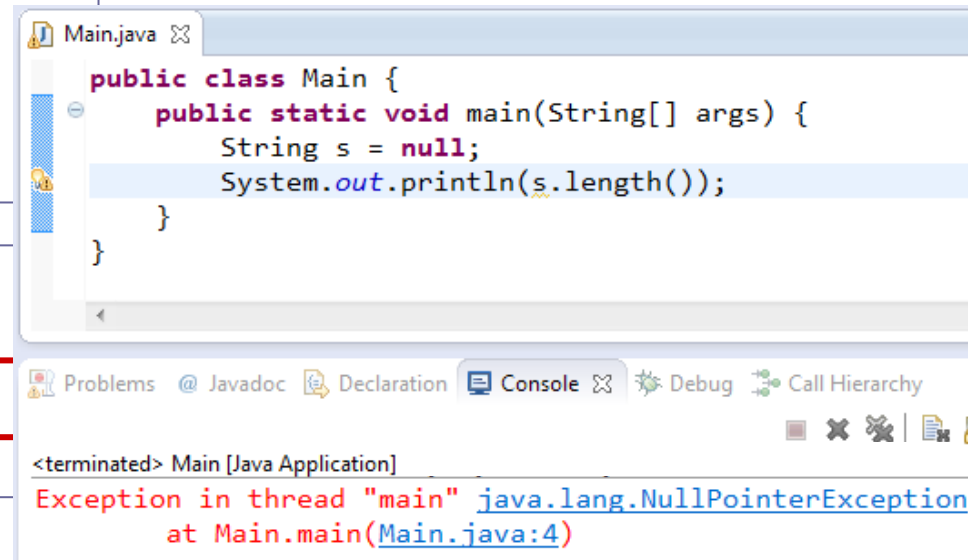
- שגיאות זמן ריצה: לא ניתן לדעת שתהיה שגיאה במקום ספציפי בזמן ההידור (קומפילציה)
- דוגמאות:

```
a = new int[20];
```

```
...  
int a[] = new int[10];  
...
```

```
a[15] = 10;
```

```
...  
String s = null;  
System.out.println(s.length());  
...
```



```
Main.java x  
public class Main {  
    public static void main(String[] args) {  
        String s = null;  
        System.out.println(s.length());  
    }  
}
```

Problems @ Javadoc Declaration Console x Debug Call Hierarchy

<terminated> Main [Java Application]
Exception in thread "main" java.lang.NullPointerException
at Main.main(Main.java:4)

- מתקשר למנגנון החריגים (exceptions), עליו נלמד בהמשך

Compilation vs. Runtime Errors

האם יש עוד סוג של טעויות? ■

כן, הכי גרועות, טעויות לוגיות בתוכנית ■

```
public class Factorial {  
    /** calculate x! */  
    public static int factorial(int x) {  
        int f = 0;  
        for (int i = 2; i <= x; i++)  
            f = f * i;  
        return f;  
    }  
}
```


The Debugger

- Some programs may compile correctly, yet not produce the desirable results
- These programs are **valid** and **correct** Java programs, yet not the programs we meant to write!
- The debugger can be used to follow the program step by step and may help detecting bugs in an **already compiled** program

Using the Debugger: Video Tutorial

■ מצגות וידאו

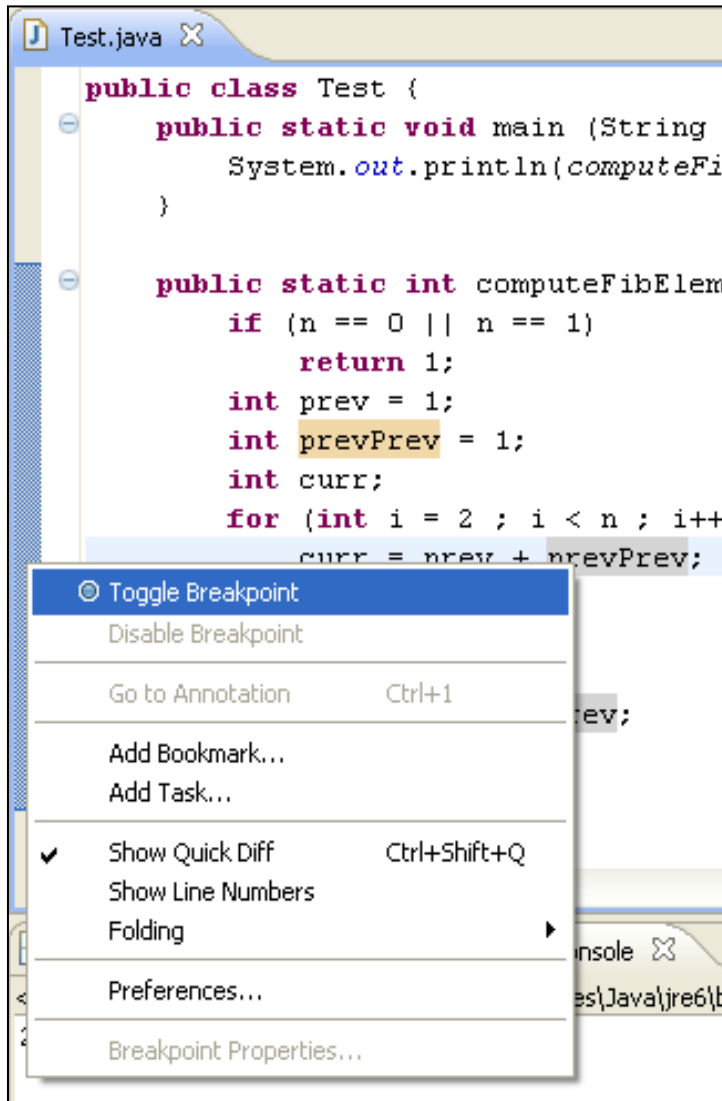
<http://eclipsetutorial.sourceforge.net/debugger.html>

■ מדריך עדכני יותר

<http://www.vogella.com/tutorials/EclipseDebugging/article.html>

■ הקישורים נמצאים גם באתר הקורס

Debugger – Add Breakpoint



- Right click on the desired line
- “Toggle Breakpoint”

Debugger – Start Debugging

Java - Test/src/Test.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy

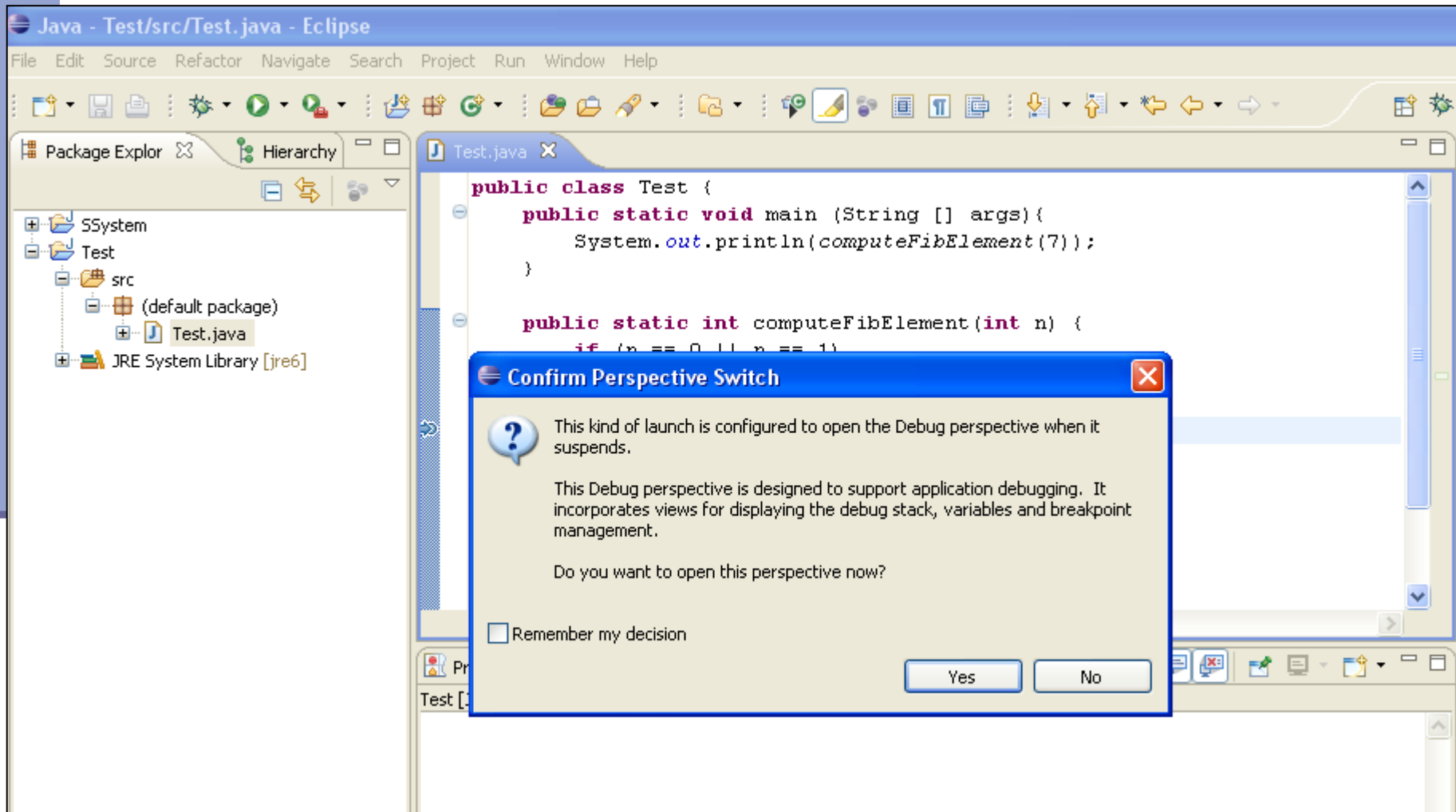
SSystem
Test
src
(default package)
Test.java
JRE System Library [jre6]

```
public class Test {  
    public static void main (String [] args){  
        System.out.println(computeFibElement(7));  
    }  
  
    public static int computeFibElement(int n) {  
        if (n == 0 || n == 1)  
            return 1;  
        int prev = 1;  
        int prevPrev = 1;  
        int curr;  
        for (int i = 2 ; i < n ; i++) {  
            curr = prev + prevPrev;  
            prevPrev = prev;  
            prev = curr;  
        }  
        curr = prev + prevPrev;  
        return curr;  
    }  
}
```

debug (F11)

breakpoint

Debugger – Debug Perspective



Debugger – Debugging

Debug - Test/src/Test.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]

- Test at localhost:2457
 - Thread [main] (Suspended (breakpoint at line 10 in Test))
 - Test.computeFibElement(int) line: 10
 - Test.main(String[]) line: 3

C:\Program Files\Java\jre6\bin\javaw.exe (27/10/2009 12:52:30)

Name	Value
n	7
prev	1

Current state

```
public class Test {  
    public static void main (String [] args){  
        System.out.println(computeFibElement(7));  
    }  
  
    public static int computeFibElement(int n) {  
        if (n == 0 || n == 1)  
            return 1;  
        int prev = 1;  
        int prevPrev = 1;  
        int curr;  
        for (int i = 2 ; i < n ; i++) {  
            curr = prev + prevPrev;  
            prevPrev = prev;  
            prev = curr;  
        }  
        curr = prev + prevPrev;  
    }  
}
```

Current location

Back to Java perspective

Console Tasks

Test [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (27/10/2009 12:52:30)

Debugger – Debugging

The screenshot displays the Eclipse IDE in a debug state. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Run menu is open, showing various debugging actions and their keyboard shortcuts. The Debug console on the left shows the execution flow: Test [Java Application] at localhost:2457, with Thread [main] suspended at a breakpoint on line 10 of Test.computeFibElement(int). The code editor shows the following code:

```
public class Test {  
    public static void main (String[] args) {  
        System.out.println("computeFibElement");  
    }  
  
    public static int computeFibElement(int n) {  
        if (n == 0 || n == 1)  
            return 1;  
        int prev = 1;  
        int prevPrev = 1;  
        int curr;  
        for (int i = 2 ; i < n ; i++)  
            curr = prev + prevPrev;  
        return curr;  
    }  
}
```

The Run menu options and their shortcuts are:

- Resume (F8)
- Suspend
- Terminate (Ctrl+F2)
- Step Into (F5)
- Step Over (F6)
- Step Return (F7)
- Run to Line (Ctrl+R)
- Use Step Filters (Shift+F5)
- Run (Ctrl+F11)
- Debug (F11)
- Run History
- Run As
- Run Configurations...
- Debug History
- Debug As
- Debug Configurations...
- Toggle Breakpoint (Ctrl+Shift+B)
- Toggle Line Breakpoint
- Toggle Method Breakpoint
- Toggle Watchpoint
- Skip All Breakpoints

הסוף...