

תוכנה 1

תרגול מס' 4
מתודות ותיכון לפי חוזים

חוזה בין ספק ללקוח

- חוזה בין ספק ללקוח מגדיר עבור כל שרות:
 - תנאי ללקוח - "תנאי קדם" - precondition
 - תנאי לספק - "תנאי אחר" - postcondition.



תנאי קדם (preconditions)

- מגדירים את הנחות הספק
- ברוב המקרים, ההנחות הללו מתארות מצבים של התוכנית שבהם מותר לקרוא לספק
- במקרים פשוטים (ונפוצים), ההנחות הללו נוגעות רק לקלט שמועבר לשירות.
- במקרה הכללי ההנחות הללו מתייחסות גם למצב התוכנית, כגון משתנים גלובליים.
- תנאי הקדם יכול להיות מורכב ממספר תנאים שעל כולם להתקיים (AND)

תנאי אחר (postconditions)

- מגדיר את המחוייבות של הספק
- אם תנאי הקדם מתקיים, הספק חייב לקיים את תנאי האחר
- ואם תנאי קדם אינו מתקיים? לא ניתן להניח דבר:
 - אולי השרות יסתיים ללא בעיה
 - אולי השרות יתקע בלולאה אינסופית
 - אולי התוכנית תעוף מייד
 - אולי יוחזר ערך שגוי
 - אולי השרות יסתיים ללא בעיה אך והתוכנית תעוף / תתקע לאחר מכן
 - ...
- ובכתיב לוגי: תנאי קדם \Leftarrow תנאי אחר,
(תנאי קדם) \Leftarrow !?

דוגמא 1

```
/*
 * precondition:
 *     1) arr != null
 *     2) arr.length > 0
 *     3) arr contains only numbers (no NaN or ±infinity)
 *
 * postcondition: Returns the minimal element in arr
 */
public static double min1(double[] arr) {
    double m = Double.POSITIVE_INFINITY;

    for (double x : arr)
        m = (x < m ? x : m);

    return m;
}
```

המימוש אינו בודק את קיומם של תנאי הקדם

מה יקרה אם בקריאה ל- `min1` לא יקויימו כל התנאים בתנאי הקדם?
?arr==null
?arr.length == 0
?arr מכיל NaN
?arr מכיל Infinity או -Infinity

דוגמא 2 (אותו קוד, חוזה שונה)

```
/*
 * precondition:  arr != null
 *
 * postcondition:
 *   If ((arr.length==0) || (arr contains only NaNs))
 *       returns Infinity.
 *   Otherwise, returns the minimal value in arr.
 */
public static double min2(double[] arr) {
    double m = Double.POSITIVE_INFINITY;

    for (double x : arr)
        m = (x < m ? x : m);

    return m;
}
```

בהשוואה לחוזה מדוגמא 1:
חוזה מתירני יותר מבחינת הלקוח

דוגמא 3 (טיפול שונה ב- NaN)

```
/*
 * precondition:  arr != null
 *
 * postcondition: If (arr.length=0) returns Infinity.
 * Otherwise,   if arr contains NaN - returns NaN.
 * Otherwise,   returns the minimal value in arr.
 */
public static double min3(double[] arr) {
    double m = Double.POSITIVE_INFINITY;

    for (double x : arr) {
        if (Double.isNaN(x))
            return x;
        m = (x < m ? x : m);
    }

    return m;
}
```

השוואה לחוזה מדוגמא 2:
טיפול שונה במקרה קצה
(קיום ערכי NaN)

דוגמא 4 (ללא precondition)

מוכן לכל מקרה

תנאי אחר המגדיר תגובה לכל קלט אפשרי מסבך את הקוד.

```
/*
 * precondition: true
 *
 * postcondition: If ((arr==null) || (arr.length==0))
 *                 returns NaN
 * Otherwise, if arr contains only NaN - returns Infinity.
 * Otherwise, returns the minimal value in arr, ignoring any NaN.
 */
public static double min4(double[] arr) {
    if (arr == null || arr.length == 0)
        return Double.NaN;

    double m = Double.POSITIVE_INFINITY;

    for (double x : arr)
        m = (x < m ? x : m);

    return m;
}
```


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

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...

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

```
...  
short x = 5;  
short y = 10;  
short z = x * y;  
...
```

Type mismatch: cannot convert from int to short

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

The local variable i may not have been initialized

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:

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

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

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

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

מתקשר למנגנון החריגים (exceptions), עליו נלמד בהמשך

Compilation vs. Runtime Errors

האם יש עוד סוג של טעויות? ■

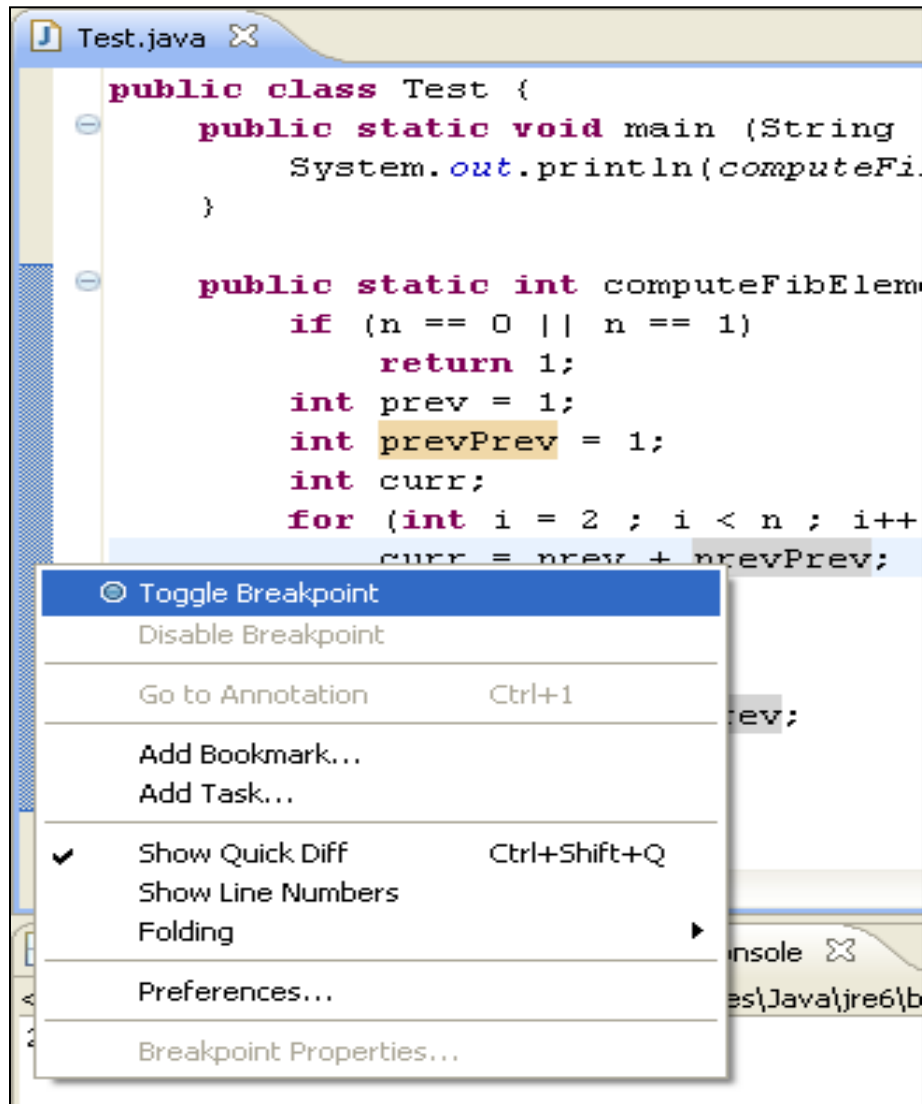
כן, הכי גרועות, טעויות לוגיות בתוכנית ■

```
public class T {  
    /** 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

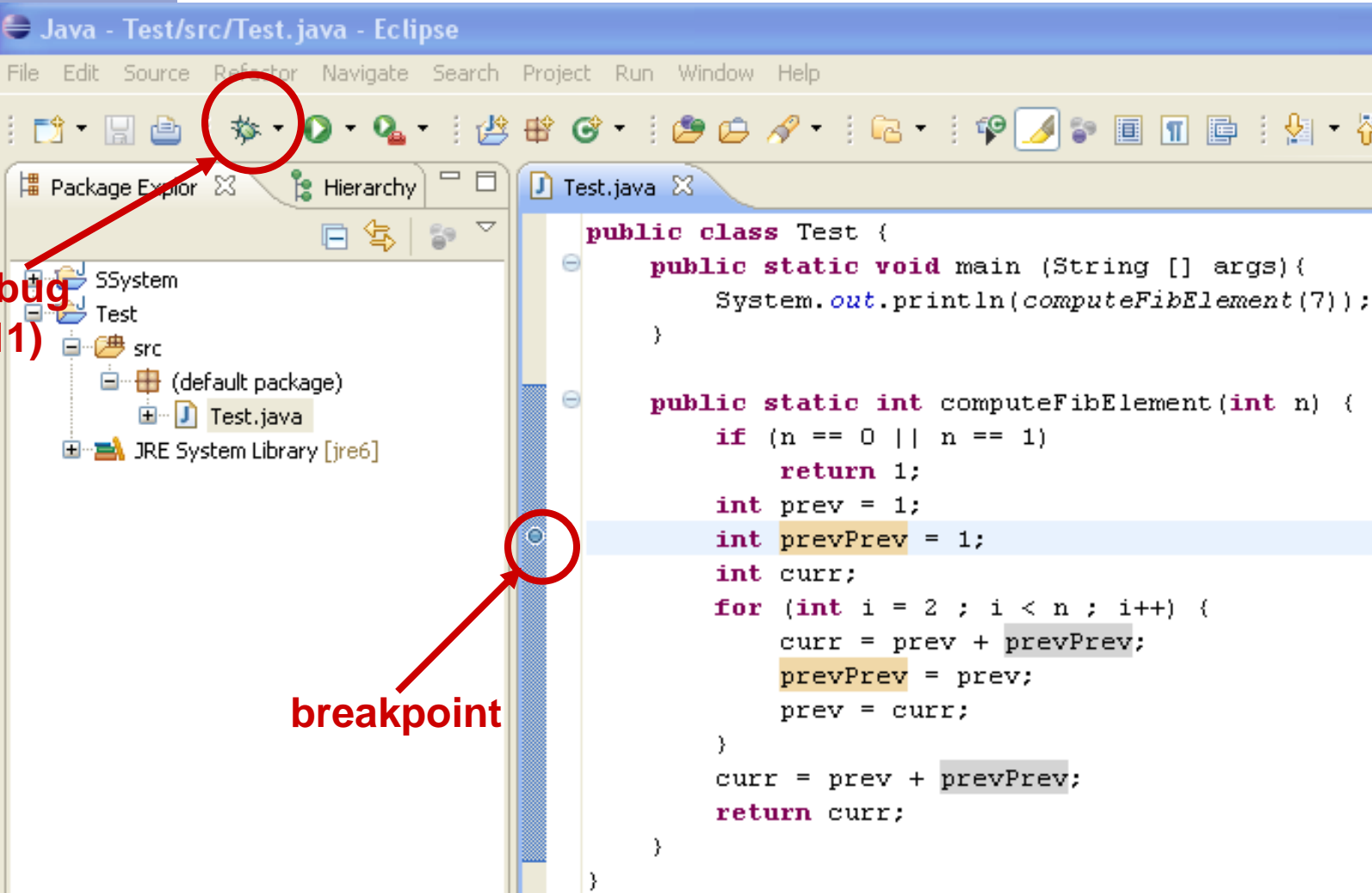
- Some programs may compile correctly, yet not produce the desirable results
- These programs are **valid** and **correct** Java programs, yet not the programs we meant to write!
- The debugger can be used to follow the program step by step and may help detecting bugs in an **already compiled** program

Debugger – Add Breakpoint



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

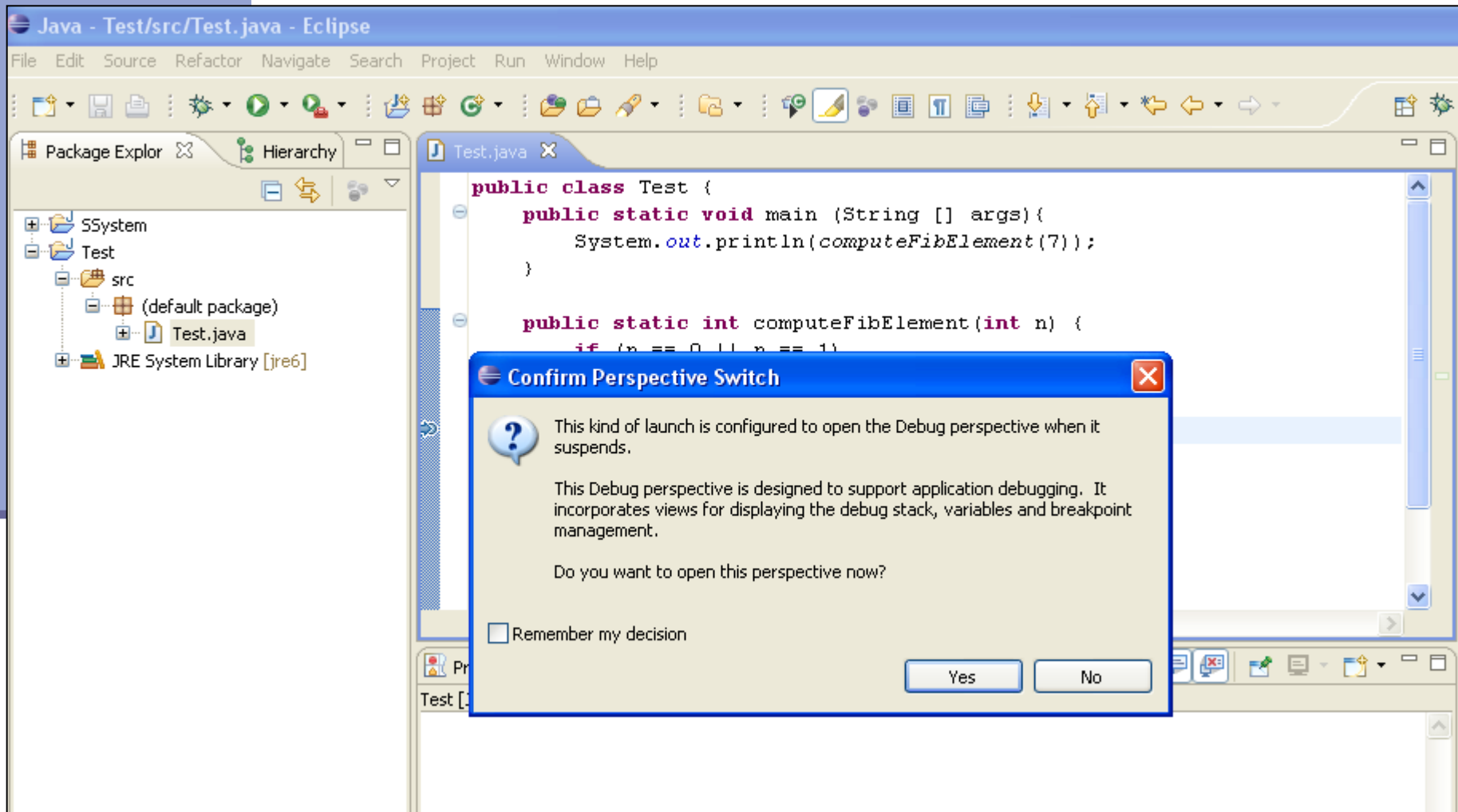
Debugger – Start Debugging



The screenshot shows the Eclipse IDE interface for a Java project. The title bar reads "Java - Test/src/Test.java - Eclipse". The menu bar includes "File", "Edit", "Source", "Refactor", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons, with the "debug" icon (a green bug) circled in red. A red arrow points from the text "debug (F11)" to this icon. Below the toolbar, the Package Explorer on the left shows the project structure: "SSystem" containing "Test", which contains "src" (default package) containing "Test.java". The "JRE System Library [jre6]" is also visible. The main editor window shows the source code of "Test.java". A breakpoint is set on line 14, which is circled in red. A red arrow points from the text "breakpoint" to this circle. The code in the editor is as follows:

```
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;  
    }  
}
```

Debugger – Debug Perspective



Debugger – Debugging

Debug - Test/src/Test.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Test [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 shows the Eclipse IDE interface with a Java application named 'Test' being debugged. The 'Run' menu is open, displaying various debugging options and their keyboard shortcuts. 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

Using the Debugger: Video Tutorial

■ תוכלו למצוא מצגות וידאו מצוינות המדריכות כיצד להשתמש ב debugger באתר:

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

■ מומלץ לצפות לפחות בארבעת הסרטונים הראשונים

* הקישור מופיע גם באתר הקורס בחלק על סביבת הפיתוח