

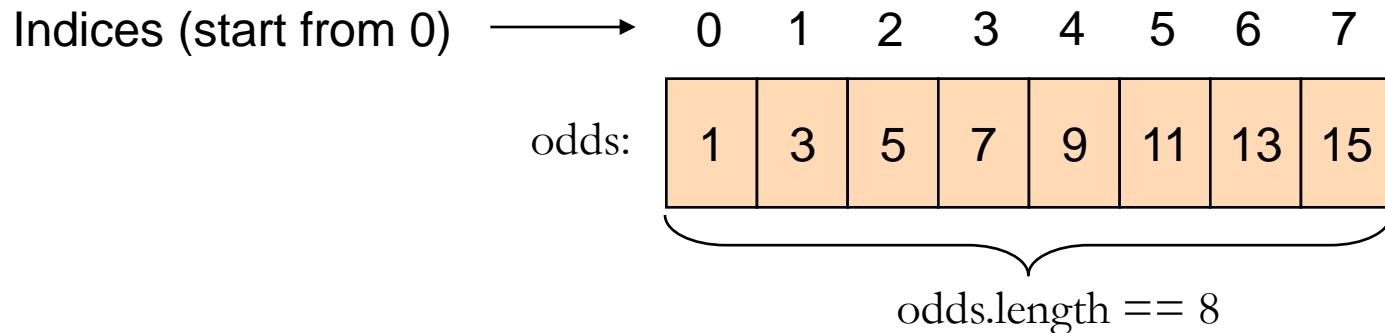
# תוכנה 1

תרגול 2: מערכים, מבני בקרה ושגיאות

# מערכים

- **Array:** A fixed-length data structure for storing multiple values of the same type

- Example: An array of odd numbers:



The type of all elements is `int`

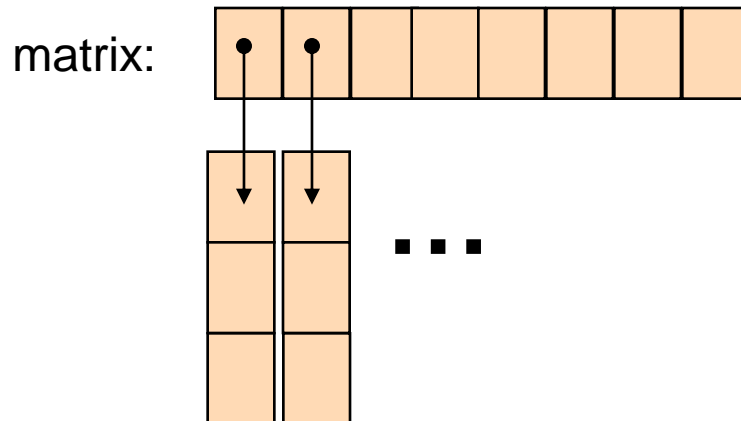
The value of the element at index 4 is 9: `odds[4] == 9`

# Array Declaration

- An array is denoted by the [] notation

- Examples:

- `int[] odds;`
- `int odds[];` // legal but discouraged
- `String[] names;`
- `int[][] matrix;` // an array of arrays



# 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

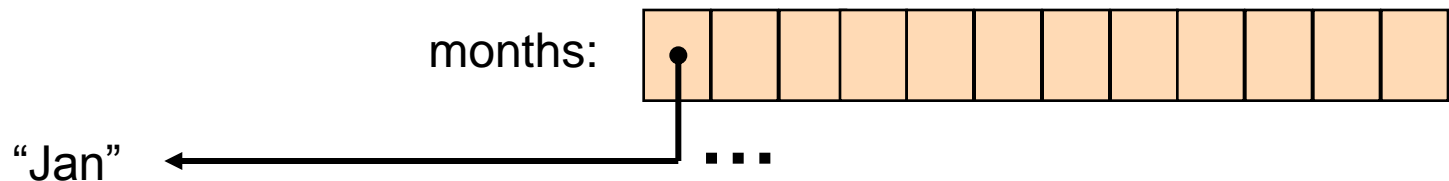
# Array Creation and Initialization

- 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" };
```



# Loop through Arrays

- By promoting the array's index:

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

The variable month is assigned the next element in each iteration

- foreach (since Java 5.0):

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

# Operations on arrays

---

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

# Copying Arrays

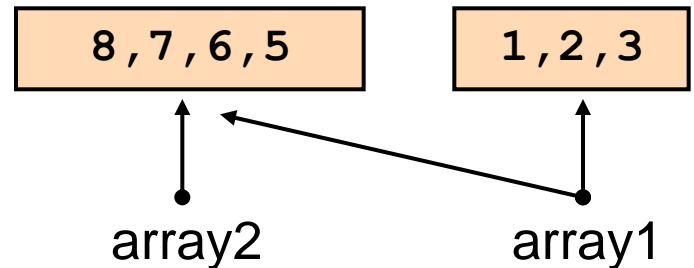
- Assume:

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

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

- Naïve copy:

```
array1 = array2;
```



- How would we copy an array?



# Copying Arrays

## ■ `Arrays.copyOf`

- the original array
- the length of the copy

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

## ■ `Arrays.copyOfRange`

- the original array
- initial index of the range to be copied, inclusive
- final index of the range to be copied, exclusive

- 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

# Other Manipulations on Arrays

- The [java.util.Arrays](#) class has methods for sorting and searching, assigning arrays e.g.
  - public static void **sort**(int[] a)
  - public static int **binarySearch**(int[] a, int key)
  - public static void **fill**(long[] a, long val)
- More details in JDK 6.0 documentation  
<http://java.sun.com/javase/6/docs/api/java/util/Arrays.html>

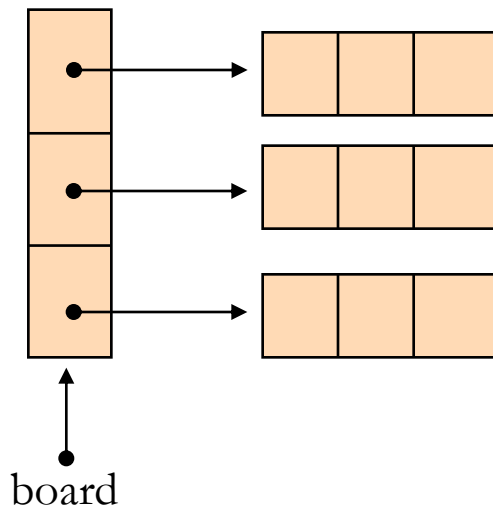
# 2D Arrays

- 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

- Building a multiplication table:

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

# Fibonacci

- Fibonacci series

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

- Definition:

- $\text{fib}(0) = 1$
- $\text{fib}(1) = 1$
- $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$


[en.wikipedia.org/wiki/Fibonacci\\_number](http://en.wikipedia.org/wiki/Fibonacci_number)



"Yes, you're right! We have increased our initial investment."

# סלט פיבונאצ'י

## סלט פיבונאצ'י

 (חשוב להצביע! הצבעות: 697)

פורסם בתאריך 7 בינואר 2009 - 9:02 | מאת מערכת דורבנות Share 

סלט שמרכיביו הם הסלט של אתמול והסלט של שלשום. סלט פיבונאצ'י ניתן למצוא בצה"ל, כיוון שהמטבחיים בצה"ל הם סחלה דה לה סחלה.

מקור השם הוא בסדרת פיבונאצ'י, בה כל איבר בסדרה הוא סכום של שני קודמיו. למשל:  
1, 1, 2, 3, 5, 8, 13, 21, 34

ע"ע ארוחת חבלים.

- "עד מתי? עד מתי נצטרך לאכול סלט פיבונאצ'י?"  
- "תגיד תודה. גם הבשר פיבונאצ'י היום."

נתרם ע"י: אסף שגיא.

קרא עוד מהנושאים: [מזון](#), [צבאי](#)

# 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 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:  
Dead Code

# For Loop

- 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;  
  
    for (int i = 2 ; i < n ; i++) {  
        curr = prev + prevPrev;  
        prevPrev = prev;  
        prev = curr;  
    }  
  
    curr = prev + prevPrev;  
    return curr;  
}
```

Assumption:  
 $n \geq 0$


# נתונים במקום חישוב

- בתרגום רקורסיה ללולאה אנו משתמשים במשתני עזר לשמירת המצב `curr, prev` ו-`prevPrev`
- הלולאה "זוכרת" את הנקודה שבה אנו נמצאים בתהליך החישוב
- דין: יעילות לעומת פשטות.
- עיקרון ה-KISS (**keep it simple stupid**)
- תרגיל: כתבו את השירות `computeElement` בעזרת `prev` ו-`prevPrev` בלבד (ללא `curr`)

# 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));  
    }  
}
```



It is better to use args[0]

# 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 האברים וגם להדפיס אותם
- כמו כן כדי לחשב איבר בסדרה איננו משתמשים בתוצאות שכבר חישבנו (של אברים 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)

# for vs. while

- The following two statements are almost equivalent:

Variable `i` is not defined outside the for block

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

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



# 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) ;
```

works since  $n \geq 1$

...פיו

# 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 ■

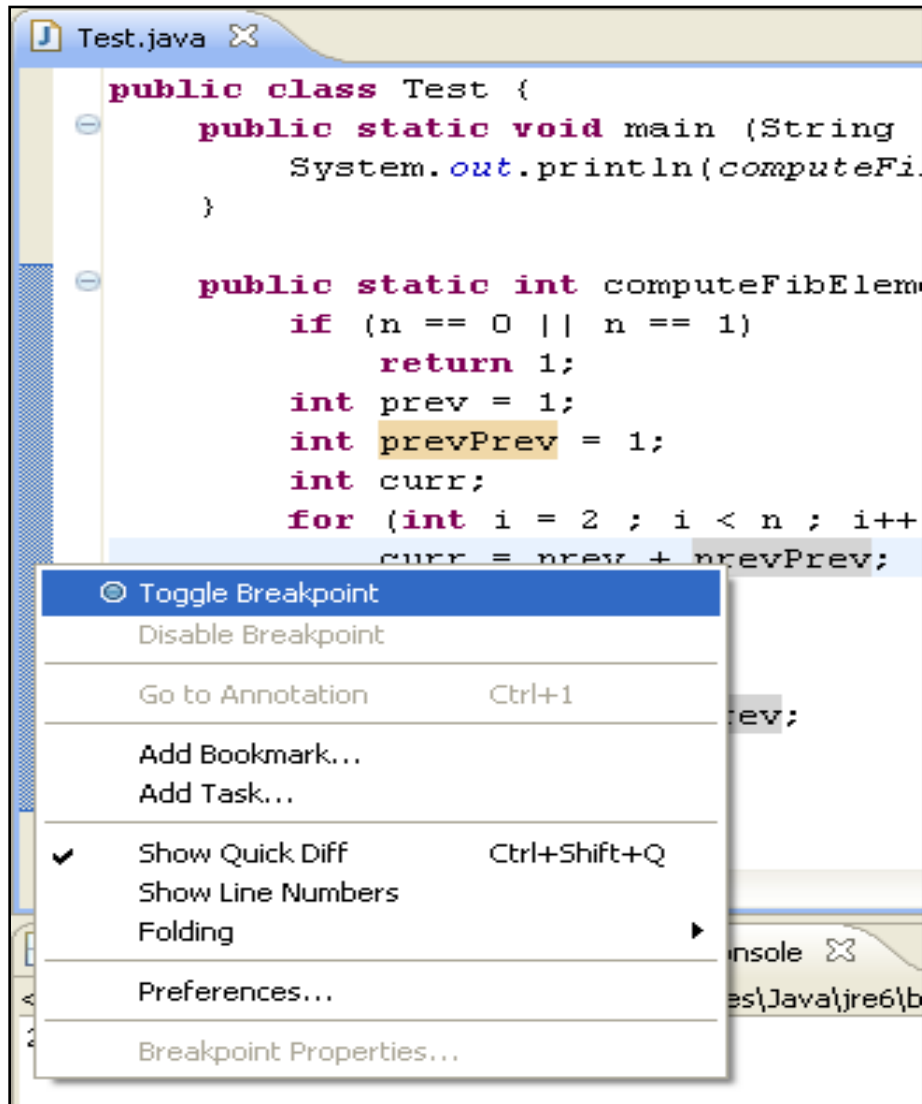
```
public class T {  
    /** calculate x! */  
    public static int factorial(int x) {  
        int f = 0;  
        for (int i = 2; i <= x; i++)  
            f = f * i;  
        return f;  
    }  
}
```

# The Debugger

---

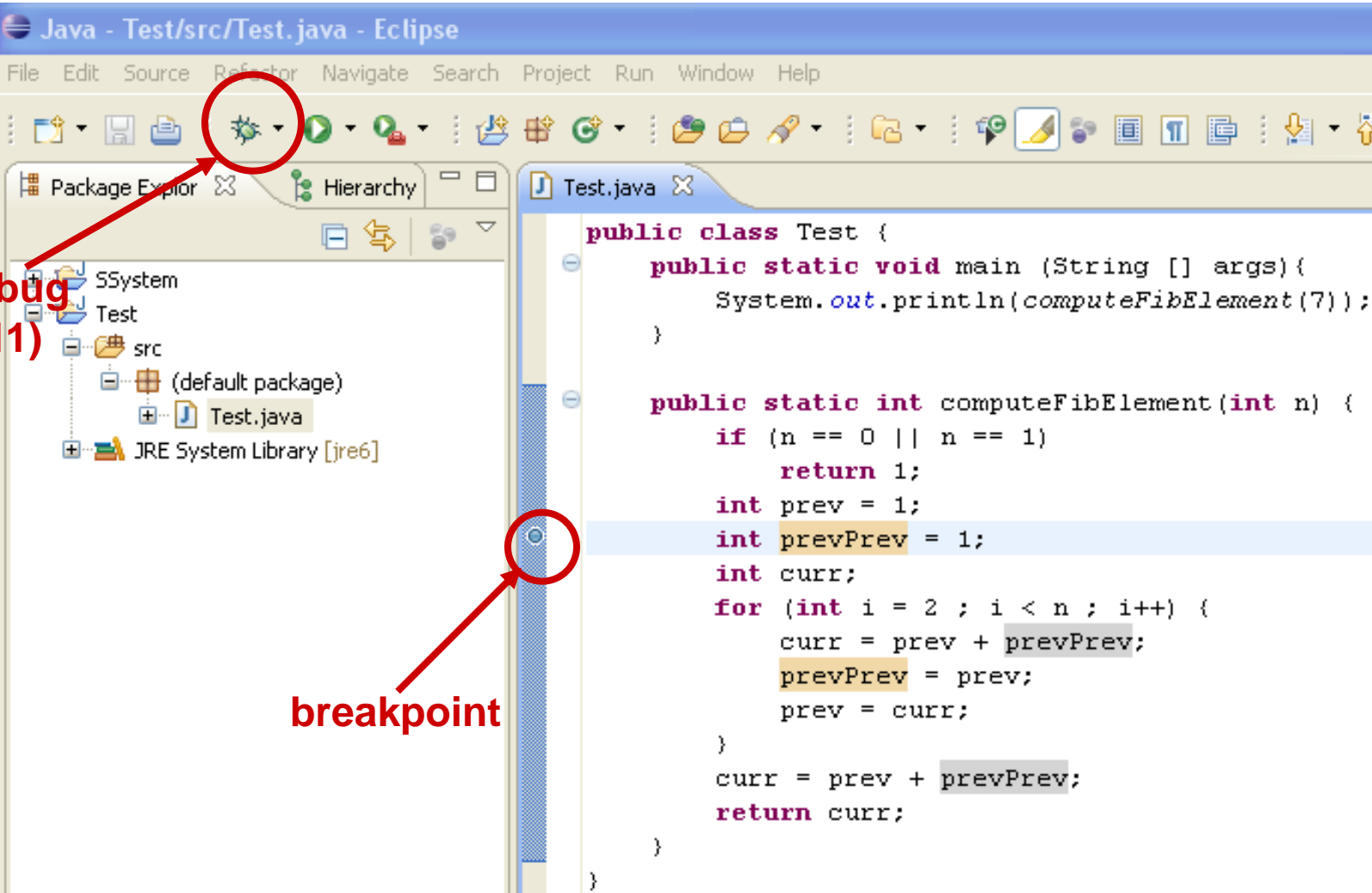
- Some programs may compile correctly, yet not produce the desirable results
- These programs are **valid** and **correct** Java programs, yet not the programs we meant to write!
- The debugger can be used to follow the program step by step and may help detecting bugs in an **already compiled** program

# Debugger – Add Breakpoint



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

# Debugger – Start Debugging



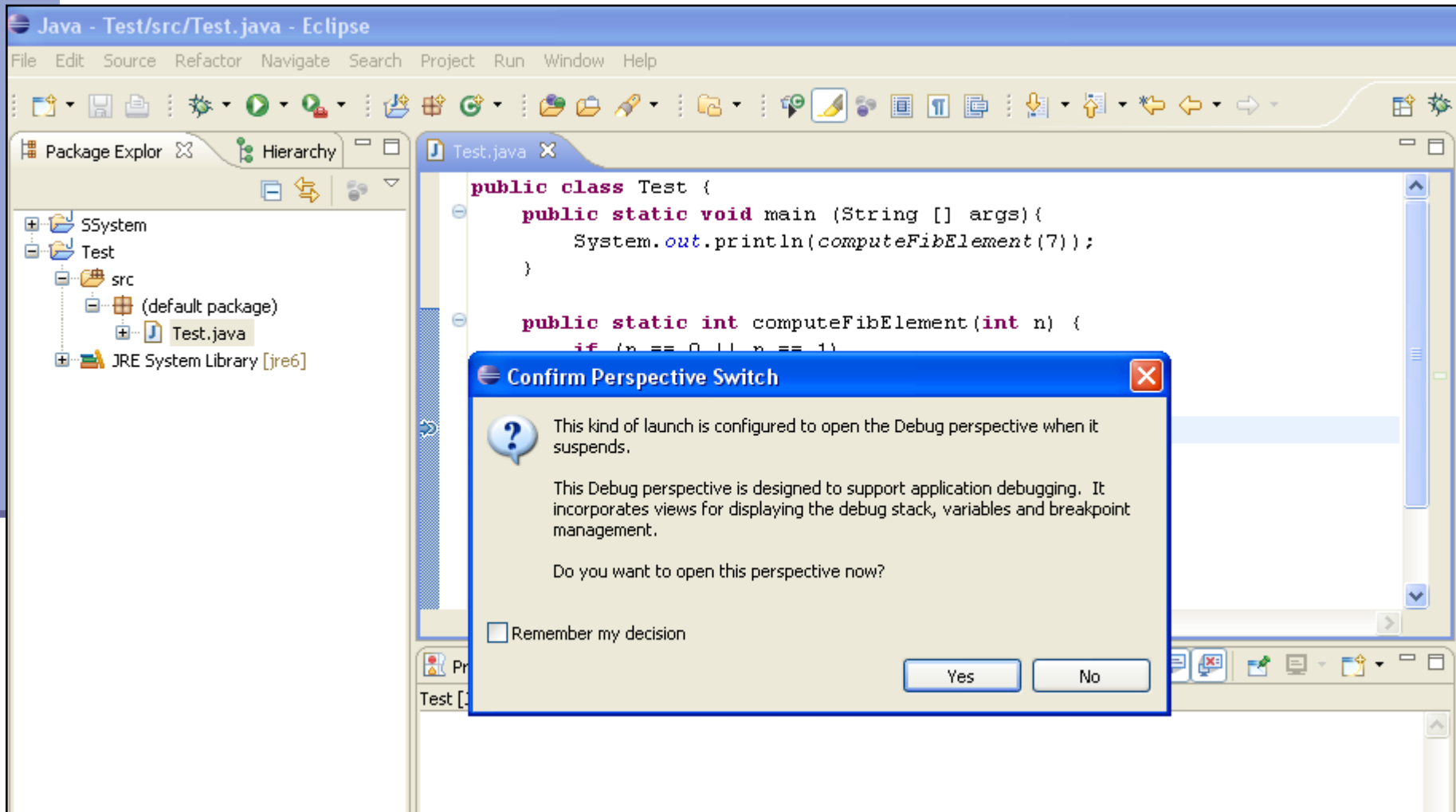
debug (F11)

breakpoint

```
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

The screenshot shows the Eclipse IDE in debug mode. The top toolbar has a 'Java' button circled in red, with a red arrow pointing to it from the text 'Back to Java perspective'. The 'Variables' view on the right shows the current state of variables:

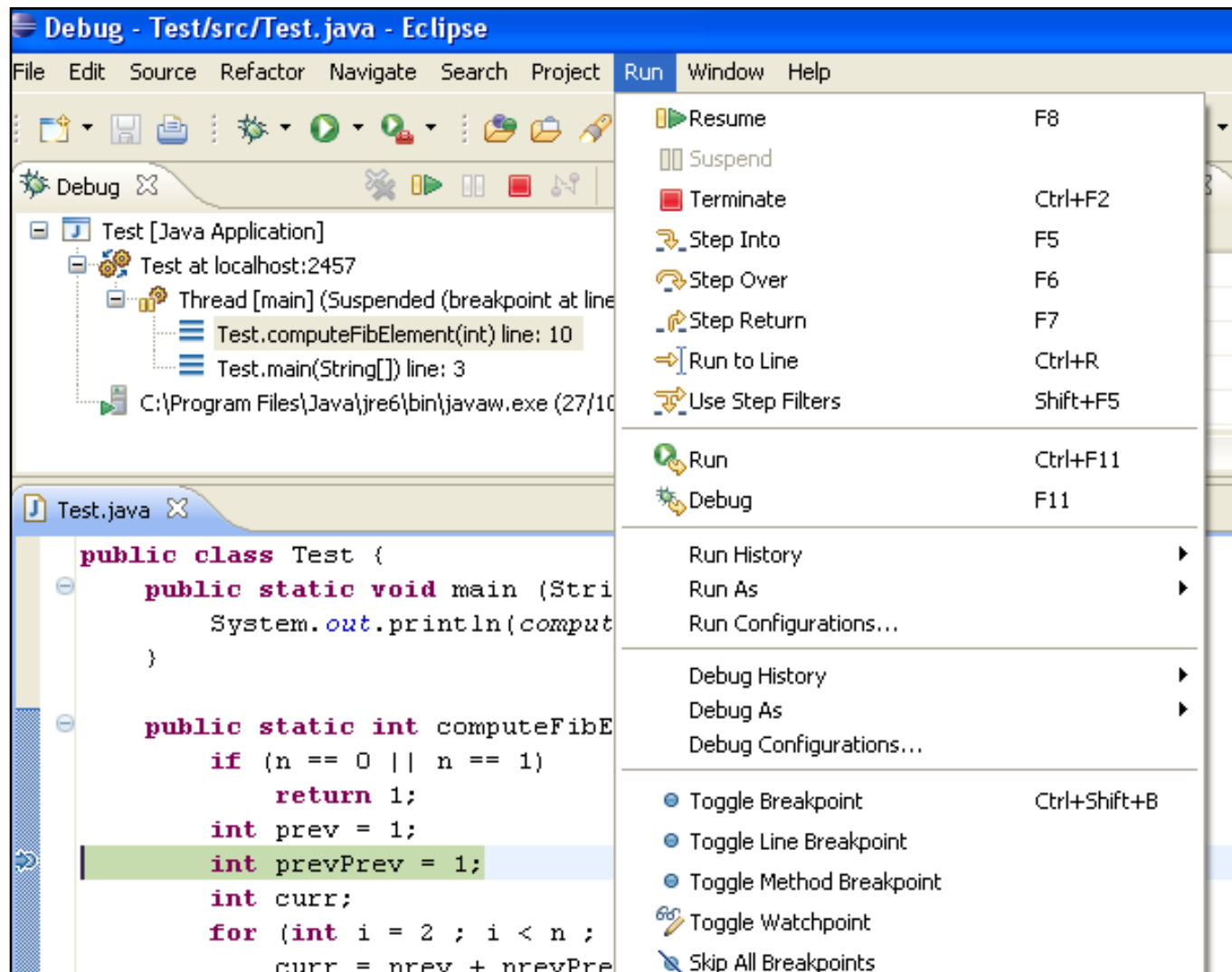
Name	Value
n	7
prev	1

The text 'Current state' is written in red below the variables table. The source code editor shows the following code:

```
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;  
    }  
}
```

The line `int prevPrev = 1;` is highlighted in green, with the text 'Current location' written in red below it. The Console view at the bottom shows the output of the application.

# 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 actions 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