

תוכנה 1

תרגול מס' 3
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 יקוימו 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;
}
```


דוגמא 5 (ללא precondition)

```
/*
 * precondition: true
 *
 * postcondition: If ((arr != null) &&
 *                  (arr.length > 0) &&
 *                  (arr contains only numbers))
 *                  returns the minimal value in arr.
 *                  Else, the return value is undefined.
 */
public static double min5(double[] arr) {
    if (arr == null)
        return 0;

    double m = Double.POSITIVE_INFINITY;

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

    return m;
}
```

תנאי אחר המגדיר תגובה רק לקלט פשוט. עבור קלטים אחרים - מתחייב להחזיר ערך כלשהו לא מוגדר (כלומר לסיים קריאה באופן תקין)

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

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;
}
```