

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 - מצבים של התוכנית שבהם מותר לקרוא לשירות
- בד"כ, ההנחות הללו נוגעות רק לקלט שמועבר לשירות.
- תנאי הקדם יכול להיות מורכב ממספר תנאים שעל כולם להתקיים (AND)
- סימון:

@pre, precondition:

תנאי אחר (postconditions)

- 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 התוכנית תעוף / תתקע לאחר מכן ...
- ובכתיב לוגי: תנאי קדם \Leftarrow תנאי אחר,
(תנאי קדם) \Leftarrow ?

@post, postcondition:

סימון: •

כיצד נסמן?

- בקורס הנוכחי אנחנו מאפשרים גמישות בתחביר של כתיבת חוזים
- ניתן להשתמש ב:
 - תנאים בוליאניים בג'אווה ($x \geq 0$)
 - תגיות מהסגנון (שנלמד בהרצאה): `@pre`, `@post`, `$prev`, `$ret`, `$implies`
 - הסגנון שנראה היום (`precondition`, `postcondition`)
 - ביטויים ונוסחאות מתמטיים ($x \in [0,1]$)
 - שפה חופשית ("`M is a diagonal square matrix`")
 - שילובים של הנ"ל, ועוד
- בכתיבת חוזים חשוב לשמור על
 - התייחסות לכל המקרים שמתאימים לתנאי הקדם בתנאי האחר
 - תמציתיות, בהירות ודיוק! (בייחוד אם משתמשים בשפה טבעית)

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)