

# תוכנה 1

סמסטר א' תשס"ט

תרגול מס' 4  
תיכון לפי חוזים והמערכת הבנקאית  
ליאור שפירא ומתי שמרת

# 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;
}
```



# 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)



# המערכת הבנקאית

# חשבון בנק - מצב הפנימי

- המצב הפנימי של עצם מיוצג ע"י נתוניו (שדותיו)
- שדות עצם הם לרוב עם הרשאת גישה פרטית
- במקרה של חשבון בנק:
  - מצב פנימי: מכיל בין היתר שדה לייצוג היתרה
  - מאיזה טיפוס?

```
public class BankAccount {  
    ...  
    private ??? balance;  
}
```

# שרותי המחלקה

ישנם 3 סוגי שירותים (מתודות, פונקציות, פרוצדורות)

## ■ שאילתות (queries, accessors)

- מחזירות ערך ללא שינוי המצב הפנימי
- כגון: בירור יתרה

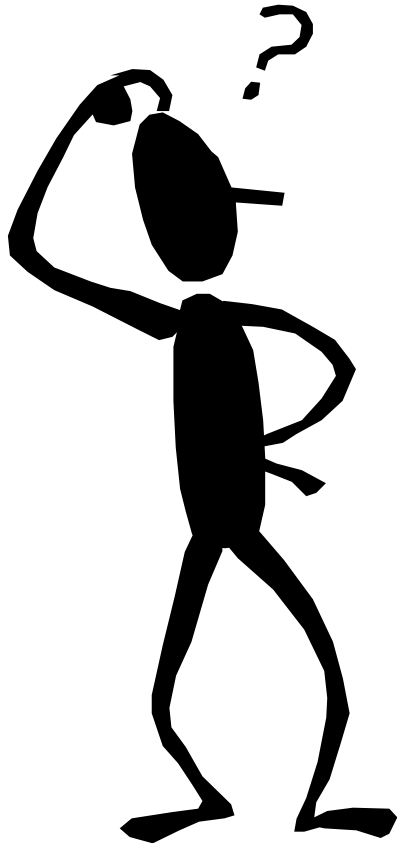
## ■ פקודות (commands, transformers, mutators)

- מבצעות שינוי במצב הפנימי של העצם
- כגון: משיכה, הפקדה

## ■ בנאים (constructors)

- יצירת עצם חדש
- כגון: יצירת חשבון חדש

# שאלות BankAccount



■ ברור יתרה:

- ארגומנטים?
- מה טיפוס הערך המוחזר?
- חוזה? (תנאי קדם? תנאי אחר?)

■ פרטים על החשבון:

- מספר חשבון?
- פרטים על בעל החשבון?
  - ❖ תעודת זהות?
  - ❖ גיל?

# שאלות BankAccount

```
public class BankAccount {  
    public double getBalance() {  
        return balance;  
    }  
  
    public long getAccountNumber() {  
        return accountNumber;  
    }  
  
    public Customer getOwner () {  
        return owner;  
    }  
}
```

שאלות

מצב  
פנימי

```
private double balance;  
private long accountNumber;  
private Customer owner;  
}
```

- מוסכמה: הגישה לשדה field תעשה בעזרת המתודה `.getField()`
- שמירה על מוסכמה זו הכרחית בסביבות JavaBeans ו- GUI Builders

# getter/setter

- יש חשיבות לגישה לנתונים דרך מתודות. מדוע?
- לא כל שדה עם נראות פרטית (`private`) צריך `getter/setter` ציבורי
- יצירה אוטומטית של שרותים אלו עבור כל שדה פוגמת בעקרון הסתרת המידע
- למשל: עבור השדה `balance`
  - האם דרוש `?getter`  
כן, זהו חלק מהממשק של חשבון בנק
  - האם דרוש `?setter`  
לא בהכרח, פעולות של משיכה או הפקדה אמנם משפיעות על היתרה, אבל פעולה של שינוי יתרה במנותק מהן אינה חלק מהממשק

# פקודת ה-'להפקיד'

■ המתודה: deposit

■ סכום הכסף המופקד מתווסף ליתרה בחשבון

■ ארגומנטים?

■ ערך מוחזר?

■ חוזה? (תנאי קדם?, תנאי אחר?)

מוסכמה: שמות פקודות  
הם שמות פועל





# פקודת ה-'להפקיד'

```
/**  
 * Makes a deposit to the account  
 * @pre ?????????????????????????????????????????????????????????  
 * @post ?????????????????????????????????????????????????????????  
 */  
public void deposit(double amount) {  
    balance += amount;  
}
```

# פקודת ה-'למשוך'

■ המתודה: withdraw

■ סכום הכסף המבוקש יורד מיתרת החשבון.

משיכת יתר (אוברדרפט) אינו אפשרי.

■ ארגומנטים?

■ ערך מוחזר?

■ חוזה? (תנאי קדם? תנאי אחר?)



# פקודת ה-'למשוך'

```
/**  
 * Withdraw amount from the account  
 * @pre ?????????????????????????????????????????????  
 * @post ?????????????????????????????????????????????  
 */  
public void withdraw(double amount) {  
    balance -= amount;  
}
```

# שיעור הבא

■ נשלים את דוגמת הבנק

■ נבדוק את הקשר בין המחלקות השונות במערכת

Bank .

Customer .

BankAccount .

■ נראה דוגמה לשימוש במערכת



# הסוף

