

## חוזה בין ספק ללקוח

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



2

## תוכנה 1

תרגול מס' 5: חוזים (הבנק חלק שני)  
נעמה מאיר ומתי שמרת

## תנאי אחר (postconditions)

- מגדיר את המחויבות 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 יוחזר ערך שגוי
  - אולי השרות יתקע בלולאה אינסופית
  - אולי התוכנית תעוף מייד, אולי השרות יסתיים ללא בעיה אך והתוכנית תעוף / תתקע לאחר מכן

... ■ 4

## תנאי קדם (preconditions)

- מגדירים את הנחות הספק
- ההנחות הללו מתארות מצבים של התוכנית שבהם מותר לקרוא לשירות
- 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)

3

## 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;
}
```

בהשואה לחוזה מדוגמא 1:  
חוזה מתירני יותר מבחינת הלקוח

6

## 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;
}
```

הימוש אינו בודק את קיומם של תנאי הקדם

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

## דוגמא 4 (ללא precondition)

```
/*
 * 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;
}
```

מוכן לכל מקרה

תנאי אחר המגדיר תגובה לכל קלט אפשרי מסבך את הקוד.

8

## 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 לחזרה מדוגמא 2: טיפול שונה במקרה קצה (קיום ערכי NaN)

7

## פקודת ה-'להפקיד'

```
/**
 * Makes a deposit to the account
 * @pre amount > 0
 * @post getBalance() == $prev(getBalance()) + amount
 */
public void deposit(double amount) {
    balance += amount;
}
```

הערך של הביטוי לפני ביצוע הפונקציה

10

## 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 קליטים אחרים - מתחייב להחזיר ערך כלשהו לא מוגדר (כלומר לסיים קריאה באופן תקין)

## העברה בנקאית

אפשרות א: מתודה סטטית שתקבל שני חשבונות בנק ותבצע ביניהם העברה:

```
/**
 * Makes a transfer of amount from one account to the other
 * @pre 0 < amount <= from.getBalance()
 * @pre from != null
 * @pre to != null
 * @pre from != to
 * @post to.getBalance() == $prev(to.getBalance()) + amount
 * @post from.getBalance() == $prev(from.getBalance()) - amount
 */
public static void transfer(double amount,
    BankAccount from,
    BankAccount to) {
    from.withdraw(amount);
    to.deposit(amount);
}
```

12

## פקודת ה-'למשוך'

```
/**
 * Withdraw amount from the account
 * @pre 0 < amount <= getBalance()
 * @post getBalance() == $prev(getBalance()) - amount
 */
public void withdraw(double amount) {
    balance -= amount;
}
```

נשתמש רק בממשק הפומבי של המחלקה לתיאור החזרה

11

## שמורת המחלקה (Class Invariant)

- צריכה להתקיים "תמיד"
- לפני ואחרי ביצוע כל מתודה ציבורית
- אחרי הבנאי
- במחלקה חשבון בנק:
  - חשבון חייב להיות עם יתרה אי שלילית
  - לכל חשבון קיים מספר מזהה במערכת
  - לכל חשבון יש בעלים

14

## העברה בנקאית

- אפשרות א: מתודה סטטית שתקבל שני חשבונות בנק ותבצע ביניהם העברה:

```
/**
 * Makes a transfer of amount from one account to the other
 * @pre 0 < amount <= from.getBalance()
 * @pre from != null
 * @pre to != null
 * @post to.getBalance() == $prev(to.getBalance()) + amount
 * @post from.getBalance() == $prev(from.getBalance()) - amount
 */
public static void transfer(double amount,
                             BankAccount from,
                             BankAccount to) {
    if (from != to) {
        from.withdraw(amount);
        to.deposit(amount);
    }
}
```

13

## בנאי BankAccount

```
/**
 * Constructs a new account and sets its owner and identifier
 * @pre id > 0
 * @pre customer != null
 * @post getOwner() == customer
 * @post getAccountNumber() == id
 * @post getBalance() == 0
 */
public BankAccount(Customer customer, long id) {
    accountNumber = id;
    owner = customer;
}
```

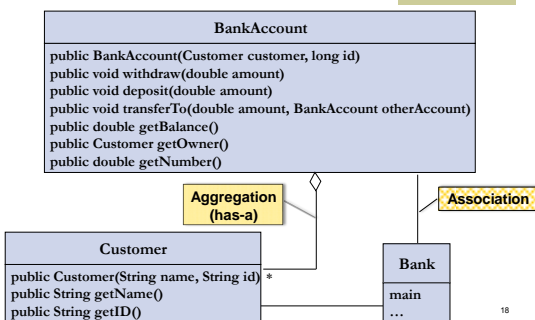
16

## שמורת BankAccount

```
/**
 * @inv getBalance() >= 0
 * @inv getAccountNumber() > 0
 * @inv getOwner() != null
 */
public class BankAccount {
    ...
}
```

15

## Class Diagram



18

## המערכת הבנקאית

- נתאר את מערכת התוכנה שלנו בעזרת דיאגרמות
- דיאגרמות סטטיות:
  - תיאור היחסים בין המחלקות השונות במערכת
  - דיאגרמות דינאמיות:
    - תיאור ההתנהגות של המערכת בזמן ריצה
    - מצב האובייקטים
    - תיאור של תרחיש



17

## Toy Bank Program

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

20

## המחלקה Customer

```
public class Customer {
    public Customer(String name, String id) {
        this.name = name;
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public String getID() {
        return id;
    }
    private String name;
    private String id;
}
```

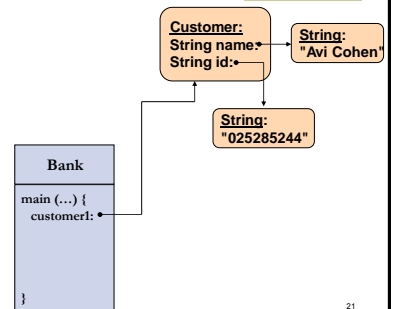
19

## Toy Bank Program

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

22

## Object Diagram



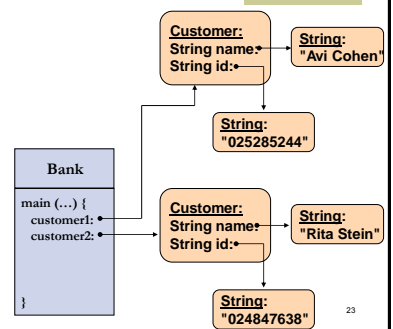
21

## Toy Bank Program

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

24

## Object Diagram



23

## Message Sequence Chart

```

public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");

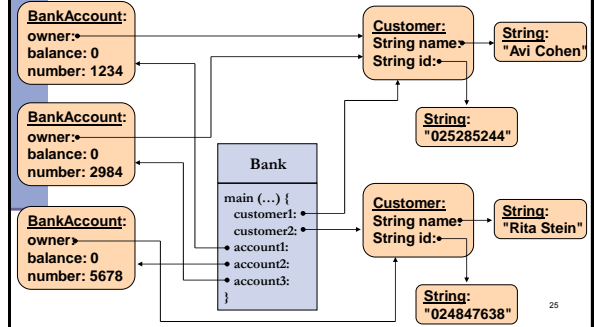
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);

        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);

        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
    
```

26

## Object Diagram



25

## Output

```

public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");

        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);

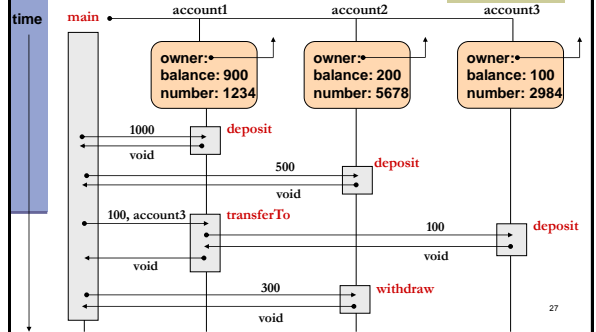
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);

        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
    
```

**output:** account1 has 900.0  
account2 has 200.0

28

## Message Sequence Chart



27