

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;  
}
```

2

תוכנה 1

סמסטר א' תשע"ב

תרגול מס' 5
מימוש מחלקות לפי מפרט
יעל אמסטרדמר ואלכסיי זגלסקי

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

4

המצב 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 double balance;  
}
```

3

שאילתות BankAccount

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

שאלות

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

6

שאילתות 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?

5

פקודת ה-'להפקיד'

- **המתודה:** deposit
- **סכום הכסף המופקד מתווסף ליתרה בחשבון**
 - ארגומנטיים?
 - ערך מוחזר?
 - חוזקה? (תנאי קדם?, תנאי אחר?)

**מוסכמה: שמות פקודות
הם שמות פועל**



getter/setter

- יש חשיבות לגישה לנתונים דרך מתודות. מדוע?
- לא כל שדה עם נראות פרטית (**private**) צריך **getter/setter** ציבורי
- יצירה אוטומטית של שדות 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 יתרה במנותק מהן אינה חלק מהממשק

פקודת ה-'להפקיד'

```
/**  
 * Makes a deposit to the account  
 * @pre amount > 0 _____  
 * @post ??????????????????????????????????????  
 */  
  
public void deposit(double amount) {  
    ...;  
}
```

בגישת תכנות לפי חוזה תנאיי הקדם לא יבדקו בגוף המתודה. אחרת: תכנות מתגונן.

10

פקודת ה-'להפקיד'

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

מימוש ה-'להפקיד'

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

12

פקודת ה-'להפקיד'

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

1

פקודת ה-'למשוך'

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

14

פקודת ה-'למשוך'

■ המתודה: withdraw

■ סכום הכסף המבוקש יורד מיתרת החשבון.
משיכת יתר (אוברדראפט) אינו אפשרי.

■ ארגומנטים?

■ ערך מוחזר?

■ חוזה? (תנאי קדם? תנאי אחר?)



13

פקודת ה-'למשוך'

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

16

פקודת ה-'למשוך'

```
/**
 * Withdraw amount from the account
 * @pre 0 < amount <= getBalance()
 * @post ?????????????????????????????????
 */
public void withdraw(double amount) {
    ...
}
```

15

דיון – העברה בנקאית

■ מספר חלופות למימוש העברת סכום מחשבון לחשבון:
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()
 * @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);
}
```

18

פקודת ה-'למשוך'

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

17

דיון – העברה בנקאית

- אפשרות ג: העמסת deposit ו/או withdraw שיקבלו שני ארגומנטים (סכום והפנייה לחשבון נוסף):

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

20

דיון – העברה בנקאית

- אפשרות ב:

```
/**
 * Makes a transfer of amount from the current account to
 * the other one
 */
public void transferTo(double amount,
    BankAccount other) {
    other.deposit(amount);
    balance -= amount;
}
```

19

שמורת BankAccount

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

22

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

21

בנאי BankAccount

```
/**
 * Constructs a new account and sets its owner and identifier
 * @pre ????????
 * @pre ????????
 * @post ????????
 * @post ????????
 * @post ????????
 */
public BankAccount(Customer customer, long id) {
    accountNumber = id;
    owner = customer;
}
```

24

בנאי

- תפקיד: ליצור עצם חדש המקיים את שמורת המחלקה
- בנאי לא אמור לכלול לוגיקה נוספת פרט לכך
- במחלקה BankAccount:
 - בנאי ברירת המחדל יוצר עצם שאינו מקיים את השמורה!
- יש דברים שאינם באחריות המחלקה. למשל:
 - מי דואג לתקינות מספרי חשבון? (למשל שיהיו שונים)
 - מי מנהל את מאגר הלקוחות?

23

בנא '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;
}
```

26

בנא 'BankAccount

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

25

final

- חשבון בנק מזוהה חד-חד ערכית עם עצם לא משתנה של `accountNumber`. לכן, נהפוך שדה זה ל-`:final`

```
final private long accountNumber;
```

Blank final field: a final field that hasn't been initialized at creation

- שדה `blank final` יש לאתחל פעם אחת בדיוק, בתוך הבנאי של המחלקה.
- אכיפה ע"י הקומפיילר: במקרה של השמות נוספות למשתנה `final` תהיה שגיאת קומפילציה

27