

תזכורת – מופעי מחלקה

- אפשר ליצור מופעים של מחלקה מסוימת (גם: עצמים מטיפוס המחלקה) בעזרת ביטוי `new`.
`BankAccount account1 = new BankAccount(...);`
- כל מופע יכול להכיל ערכים שונים של **שדות מופע**
 - בניגוד לשדות סטטיים, אשר שייכים למחלקה
- כל מופע יכול לקרוא **לשירתי מופע**
 - מתוך שירותים אלה יש גישה למשתנה `this`, אשר מצביע על העצם הקורא, וממנו ניתן לגשת לשדות ושירותי מופע נוספים
 - בניגוד לשירותים (פונקציות/מתודות) סטטיים, אשר אינם מקושרים למופע ספציפי אלא רק למחלקה

2

תוכנה 1

תרגול מס' 6
מחלקות, עצמים, וקצת חוזים

שירותי מופע

- ישנם 3 סוגי שירותים (מתודות, פונקציות, פרוצדורות):
- שאלות (queries, accessors)
 - מחזירות ערך ללא שינוי המצב הפנימי
 - שאלות צופות (observers): מחזירות פרט מידע הקשור לעצם (למשל, בירור יתרה)
 - שאלות מפקות (producers): מחזירות עצם מאותו טיפוס (למשל, חשבון חיסכון המקושר לחשבון עובר ושב)
 - פקודות (commands, transformers, mutators)
 - מבצעות שינוי במצב הפנימי של העצם
 - גנון: משיכה, הפקדה
 - בנאים (constructors)
 - יצירת עצם חדש
 - גנון: יצירת חשבון חדש

4

המצב הפנימי של אובייקט

- המצב הפנימי של עצם מיוצג ע"י נתוניו (שדותיו)
- שדות עצם יהיו 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

getter/setter

- יש חשיבות לגישה לנתונים דרך מתודות. מדוע?
 - לא כל שדה עם נראות פרטית (`private`) צריך `getter/setter` ציבורי
 - למשל: עבור השדה `balance`
 - האם דרוש `getter`?
 - כן, זהו חלק מהממשק של חשבון בנק
 - האם דרוש `setter`?
- ```
public void setBalance(double balance) {
 this.balance = balance;
}
```
- לא בהכרח, פעולות של משיכה או הפקדה אמנם מופיעות על היתרה, אבל פעולה של שינוי יתרה במנותק מהן אינה חלק מהממשק

6

## שאלות BankAccount

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

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

5



## צ'אניט, אחזקות, נראות ומה שפניהם

14

## העמסת בנאים

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

תזכורת: העמסה = יצירת מתודה בעלת שם זהה אך עם ארגומנטים שונים. באופן דומה ניתן להגדיר בנאים עם ארגומנטים שונים.

`this()` כאן משמש לא כמשתנה אלא כקריאה לבנאי אחר של אותה מחלקה שיבצע אתחול ראשוני על העצם שאנו מייצרים. ניתן להשתמש בתחביר זה רק מתוך בנאי!

15

## המחלקה OtherClass

```
public class OtherClass {
 public static void othersPublicStaticMethod() {
 System.out.println("In othersPublicStaticMethod");
 }

 private static void othersPrivateStaticMethod() {
 System.out.println("In othersPrivateStaticMethod");
 }

 public void othersPublicMethod() {
 System.out.print("In othersPublicMethod >> ");
 othersPrivateMethod();
 }

 private void othersPrivateMethod() {
 System.out.println("In othersPrivateMethod");
 }
}
```

16

## המחלקה CurrentClass

```
public class CurrentClass {

 public static void myPublicStaticMethod() {
 System.out.println("In myPublicStaticMethod");
 }

 private static void myPrivateStaticMethod() {
 System.out.println("In myPrivateStaticMethod");
 }

 public void myPublicMethod() {
 System.out.print("In myPublicMethod >> ");
 myPrivateMethod();
 }

 private void myPrivateMethod() {
 System.out.println("In myPrivateMethod");
 }
}
```

קריאה למתודה פרטית ממתודה פומבית (גם ההפך זה בסדר)

17

## מסקנות

- מתודה סטטית אינה יכולה לקרוא למתודה שאינה סטטית
  - חייבים לציין מיהו העצם שהשירות משויך אליו
  - `myPublicMethod()` מתוך מתודה סטטית
  - `currentClass.myPublicMethod()`!
- נראות מגדירה מאיזה מקום בקוד ניתן לגשת למתודה
  - נראות פרטית = ניתן לגשת רק מהקוד של אותה מחלקה
  - נראות פומבית = ניתן לגשת מכל מחלקה (אם היא לא באותה חבילה, יש להוסיף הצהרת `import`)
  - נלמד על עוד שני סוגים בהמשך

18

## נוסוף main ל-CurrentClass

```
public class CurrentClass {
 public static void main(String[] args) {
 CurrentClass.myPublicStaticMethod(); // Prints: In myPublicStaticMethod
 myPublicStaticMethod(); // Prints: In myPublicStaticMethod
 CurrentClass.myPrivateStaticMethod(); // Prints: In myPrivateStaticMethod
 CurrentClass.myPublicMethod();
 CurrentClass currentClass = new CurrentClass();
 currentClass.myPublicMethod(); // Prints: In myPublicMethod >> In myPrivateMethod
 currentClass.myPrivateMethod(); // Prints: In myPrivateMethod
 currentClass.myPublicStaticMethod(); // Has a warning, Prints: In myPublicStaticMethod

 OtherClass.othersPublicStaticMethod(); // Prints: In othersPublicStaticMethod
 OtherClass.othersPrivateStaticMethod();
 OtherClass otherClass = new OtherClass();
 otherClass.othersPublicMethod(); // Prints: In othersPublicStaticMethod
 OtherClass.othersPrivateMethod();
 }
 ...
}
```

19

## דוגמא

```

public class BankAccount {
 public static final String BANK_NAME = "BNP"; //static constant
 private static int LastAccountId = 0; //static field
 private int id;

 public BankAccount() {
 id = ++LastAccountId; // unique ID for every account
 }

 /* static method */
 public static void main(String[] args) {
 System.out.println(LastAccountId);
 System.out.println(id);
 BankAccount account = new BankAccount();
 System.out.println(account.id);
 }

 /* instance method */
 public void printStuff() {
 System.out.println(LastAccountId);
 System.out.println(id);
 }
}

```

20

## Instance vs. Class (static) Fields

### Instance fields

- למה? ■
- ייצוג פנימי של המופע ■

מתי? ■

- מאותחלים עם יצירת האובייקט ■

כמה? ■

- אחד לכל מופע ■

מאיפה? ■

- נגישים אך ורק ממתודות מופע! (למה?) ■

### Class (static) fields

למה? ■

- קבועים ■

- ערכים המשותפים לכל מופעי המחלקה ■

מתי? ■

- מאותחלים לפי הסדר עם טעינת המחלקה ■

כמה? ■

- יש רק 1 בכל התוכנית! (0 לפני טעינת המחלקה) ■

מאיפה? ■

- נגישים ממתודות סטטיות ומתודות מופע ■

19