## עצמים

תכנות מתקדם בשפת Java

אוהד ברזילי

אוניברסיטת תל אביב

## עצמים

המצגת מכילה קטעים מתוך מצגת של פרופ' עמירם יהודאי ע"פ הספר:

Object-Oriented Software Construction, 2nd edition, by Bertrand Meyer (Prentice Hall) .



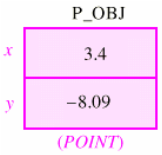
## עצמים

- עצם הוא מופע זמן-ריצה (instance) של מחלקה
- עצם הוא מופע ישיר של בדיוק מחלקה אחת, המחלקה היוצרת, ואולי מופע של מחלקות נוספות כתוצאה מירושה
- מבנה הנתונים המייצג עצם מכיל שדות נתונים בלבד ללא פונקציות (מתודות)



## POINT Class

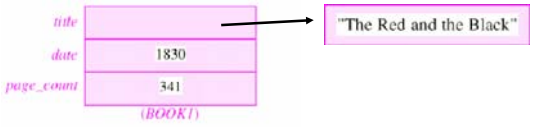
```
public class POINT {
  private float x, y;
  // Routines...
}
```






## Simple Book

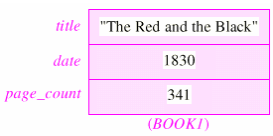
```
public class BOOK1 {
   private String title;
   private int date, page_count;
}
```






## Simple Book

```
public class BOOK1 {
   private String title;
   private int date, page_count;
}
```




התרשים פשטני – *מחרוזת* היא עצם ולכן השדה title מכיל רק הפנייה אליו

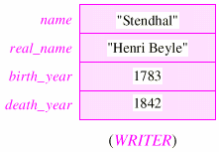
## עצמים

- נסווג את השדות לסוגים לפי טיפוסיהם:
  - טיפוסים יסודיים (פרימיטיבים): חלק משפת התכנות. כגון: int, char, float, bool (אבל לא string)
  - טיפוסים מורכבים (user defined types)
- שדה מורכב: עצם מוכל או עצם מוצבע?



## Writer Class

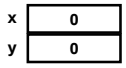
```
public class WRITER {
   private String name, real_name;
   private int birth_ year, death_ year;
}
```





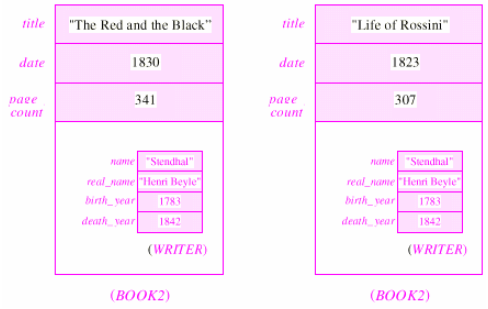

## עצמים, מצביעים והתייחסויות

```
Point p1 = new Point();
Point p2;
p2 = p1;
p1.setX(4);
```





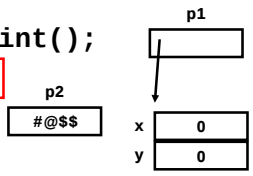

## תת עצמים






## עצמים, מצביעים והתייחסויות

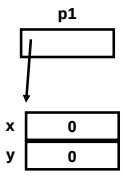
```
Point p1 = new Point();
Point p2;
p2 = p1;
p1.setX(4);
```





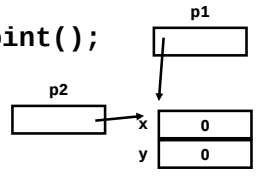

## עצמים, מצביעים והתייחסויות

```
Point p1 = new Point();
Point p2;
p2 = p1;
p1.setX(4);
```



## עצמים, מצביעים והתייחסויות

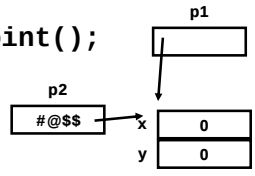
```
Point p1 = new Point();
Point p2;
p2 = p1;
p1.setX(4);
```






## עצמים, מצביעים והתייחסויות

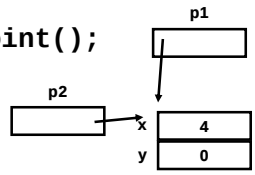
```
Point p1 = new Point();
Point p2;
p2 = p1;
p1.setX(4);
```






## עצמים, מצביעים והתייחסויות

```
Point p1 = new Point();
Point p2;
p2 = p1;
p1.setX(4);
```






## עצמים, מצביעים והתייחסויות

- נשים לב כי כל העצמים בשפת Java הם אנונימיים (חסרי שם)
  - להבדלי משפות אחרות (כגון C++)
- הפנייה ב Java היא ישות עצמאית, משתנה בפני עצמו, שעשויה להכיל כתובת של אובייקט קיים. ניתן לעדכן את ערכה של ההפנייה ובכך להחליף את העצם המוצבע.
- לא נתבלבל עם העברת נתונים לפונקציה by reference. העתקה או אי העתקת העצם לא תלויה בעצם עצמו אלא בהגדרת שיטת ההעברה בפונקציה

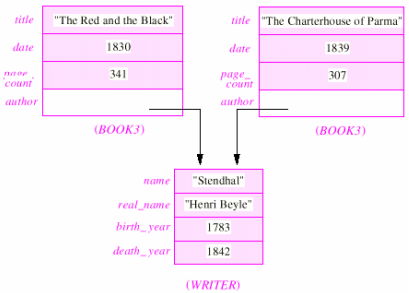


## עצם מוכל או מוצבע?

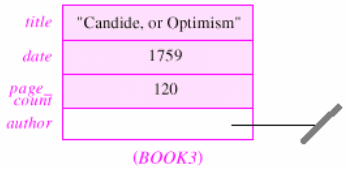
- תת עצמים תופסים יותר זכרון
- עצמים מוצבעים קל יותר לשתף בין מספר ישויות
- לא כל המצביעים מאותחלים אוטומטית (ל- null) כך שקיימות 3 אפשרויות:
  - המצביע מתייחס לעצם אחד מסוים
  - מצביע לכתובת 'זבל'
  - המצביע הוא null כלומר לא משויך לעצם מסוים
- בשפת Java השאלה לא קיימת מכיוון שאין עצמים מוכלים



## מוצבע



## לא משויך לאף עצם






## זהות של עצמים

- שני עצמים נפרדים עשויים להיות זהים
- שוויון שדות אינו תנאי מספיק לזהותם של שני עצמים

```
class WRITER {
   String name;
   int birth_year, death_year;
}

class BOOK {
   String title;
   int date;
   WRITER *author;
}
```



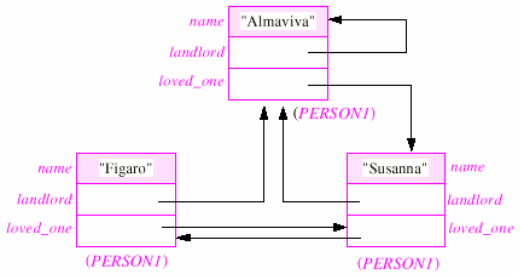
## מעגלים ביחס הלקוחות

- מחלקה עשויה להיות לקוחה של עצמה
- נבדיל בין מעגל של עצמים ומעגל של מחלקות

```
class PERSON1{
   string name;
   PERSON1 loved_one, landlord;
}
```



## מצביע לעצמו



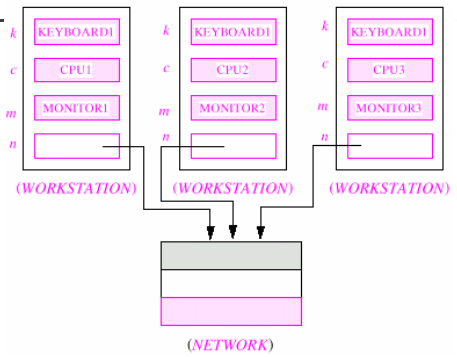



## יתרונות העצם המוכל

- **יעילות**
  - גישה לשדות מוכלים שלא דרך dereference של מצביע
- **מודל טוב יותר**
  - מצביע למחלקה S פירושו שהלקוח "יודע על" S
  - עצם כעצם מוכל מעיד על כך שהלקוח מכיל S
  - בפרט, הכלה מרמזת על אי-שיתוף
- **תמיכה אחידה בטיפוסים פרימיטיבים**
  - עצמים מכילים את הטיפוסים היסודיים עצמם ולא מצביע אליהם



## אובייקטים מורכבים



## עצמים מוכלים

- לא יתכנו מעגלי ספק-לקוח כאשר הספק הוא עצם מוכל (expanded type)
- יצירה והריסה של האובייקט המוכל תלויה בזמני היצירה וההריסה של העצם המכיל
- בהשמת העצם העוטף לא יופעל בהכרח אופרטור ההשמה של כל אחד משדותיו המוכלים
- כאשר עצם מכיל מצביעים לעצמים אחרים יש לשקול מתי לשחרר את הזכרון של עצמים אלו (למשל אם הם משותפים לכמה עצמים)



## הכלה או הצבעה?

- בשפת Java הוחלט **שלא לאפשר** הכלת עצמים
- כל ההתייחסויות לעצמים הן הפניות



## פונקציה יוצרת

- יצירת ואתחול עצמים מפורשת
- כאשר בפונקציה כלשהי מגיעים לשורה `T x = new T();` קורים הדברים הבאים:
  1. מוקצה מקום עבור עצם מטיפוס T (עבור כל אחד משדותיו)
  2. x מקבל את כתובתו של העצם שהוקצה



## מצביעים ומצבם

- מצביע יכול להיות קשור או לא קשור לעצם
- מצביע p יקשר לעצם ע"י:
  - p=new SomeClass(...)
  - או ע"י p=p1 , כאשר p1 הוא מצביע קשור
- p יהפך ללא קשור ע"י:
  - p=null
  - או ע"י p=p1 כאשר p1 הוא null
- הפעלת מתודה דרך מצביע שאינו קשור תגרום לטעות זמן ריצה



## פעולות על מצביעים

- השמה של מצביעים אינה מבצעת העתקה של האובייקט המוצבע
- ההשוואה בין מצביעים p1==p2 תחזיר true רק אם הם מצביעים על אותו עצם עצמו או ששניהם null
- ההשמה p=null שימושית כדי:
  - לבדוק בהמשך האם p!=null (שכן למצביע עלול להיות גם ערך 'זיבלי')
  - לעזור ל Garbage Collector להבין שהעצם המוצבע אינו נחוץ עוד

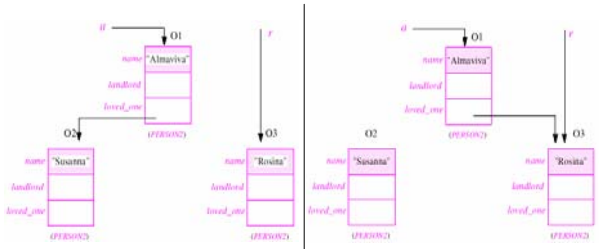


## קשירת מצביע

```
class PERSON2 {
  String name;
  PERSON2 loved_one, landlord;

  /**Attach the loved_one field of
  current object to l. */
  void set_loved (PERSON2 l){
     loved_one = l;
  }
}
```

## קשירת מצביע






## פונקציה יוצרת

- במחלקות שנכתוב ניתן (ורצוי?) להחליף את אתחול ברירת המחדל בפונקציה
- פונקציה זו נקראת הבנאי (constructor) של המחלקה
- ניתן להעמיס אותה
- ניתן להעביר לה פרמטרים
- שמה כשם המחלקה שאותה היא מאתחלת ואין לה ערך מוחזר

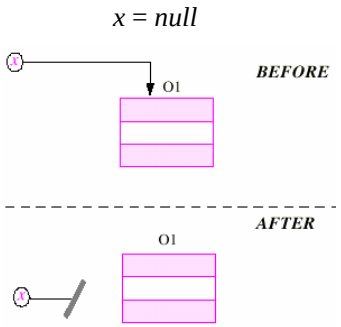


## ריבוי בנאים

- כדי לשמור על עיקביות בתחזוקת מספר גרסאות מועמסות של אותה הפונקציה מומלץ להשתמש בקריאות הדדיות בין הגרסאות המועמסות
- בהעמסת בנאים, הקריאה `this(args)` קוראת לבנאי מועמס שחתימתו תואמת את `args`



## איפוס מצביע (ללא שחרור!)






## בנאים ל Point

```
public class Point {
  //...
  public Point() {
     m_x = 0;
     m_y = 0;
  }

  public Point(int x, int y) {
     m_x = x;
     m_y = y;
  }
  //...
}
```



## בנאי יותר טוב ל Point

```
public class Point {
  //...
  public Point(int x, int y) {
     m_x = x;
     m_y = y;
  }
  public Point() {
     this(0,0);
  }
  //...
}
```

## עוד על בנאים

- פונקצית אתחול שהוגדרה כ private לא תאפשר ליצור באמצעותה עצמים
- למחלקה שלא נכתבו עבורה בנאים כלל מספק המהדר (הקומפיילר) בנאי ברירת מחדל ( default constructor) זהו בנאי חסר פרמטרים שאינו עושה דבר
- מרגע שנכתב בנאי כלשהו למחלקה לא ניתן עוד להשתמש בבנאי ברירת המחדל
- יש מקרים שבהם נהיה חייבים להשתמש בבנאי חסר פרמטרים ולכן נגדיר כזה לכל מחלקה



## יצירה ואתחול מפורשים

- עצמים נוצרים רק ע"י שימוש מפורש באופרטור new
- 2 יוצאי דופן:
  - אתחול מערך:

```
int [] arr = {1, 2, 3};
```

  - יצירת מחרוזות ע"י מרכאות:

```
String hello = "hello";
```