

פולימורפיזם

להסתכל על אותו אובייקט בצורות שונות.

אובייקט של תת-מחלקה הוא גם אובייקט של מחלקת העל (כלה הוא גם סט-ל-חיים וגם יונק).



פולימורפיזם מאפשר להשיג שתי מטרות:

- לטפל בצורה אחידה באובייקטים מסוגים שונים כאילו כולם מאותו טיפוס (גיבוי אחיד לכל העליון החיים)
(מתת הוואחלקות של Animal)
- התגובה של כל אובייקט לטיפול האחיד תהיה ייחודית לו.

טיפוס הכרמטר לפצולה

אובייקט מתת מחלקה יכול להיות מועבר כפרמטר לפעולה אם הפרמטר של הפעולה הוא מסוג מחלקת העל שלו. למשל: פעולה שהפרמטר שלה הוא Animal יכולה לקבל כל אובייקט מתת המחלקות של Animal כי כולם סוג של בע"ח.

Mammal mam = **new** Mammal(...); : הוגדר האובייקט
void method (Animal a) : הוגדרה הפעולה

משפט הזימון הבא יהיה נכון: method (mam) : מתבצעת המרה (אוטומטית) כלפי מעלה
 upcasting

נגדיר מערך zoo של בעלי חיים: `Animal [] zoo = new Animal [7];`

ניתן לשים בכל תא במערך אובייקט של המחלקה Animal או אובייקט של אחת המחלקות היורשות מ-Animal.

טיפול אחיד: כל תא יכול הפנייה לטיפוס נתונים אחר בעץ ההורשה. נוכל להתייחס לכל האובייקטים שנוצרו מתת המחלקות שונות כאילו היו מטיפוס מחלקת העל שלהם (כי כולק סוס *se* *הסל-חיים* - Animal).

גמישות: אם נרצה בעתיד ליצור עוד תת-מחלקות של בעלי-חיים, לא נצטרך לשנות בתכנית, ונוכל להגדיר עצמים נוספים מסוג בעלי-חיים במערך zoo.

תגובה ייחודית: אם לאחת המחלקות יש מימוש שונה לפעולה שמוגדרת ב-Animal, יפעיל העצם את הפעולה שבמחלקה שלו.

```
Animal [] zoo = new Animal [7];
```

```
zoo[0] = new Animal ("animal", "male");
zoo[1] = new Dog ("Dasty", "female", 0.5, 3);
zoo[2] = new Fish ("Gupy", "male", 0.3);
zoo[3] = new Bird ("Twitty", "female", 3.5);
zoo[4] = new Boxer ("Rexy", "male", 3.2, 3, "house");
zoo[5] = new Labrador ("Toto", "male", 2.5, 4, new Person("Hila"));
zoo[6] = new Mammal ("Mammal", "female", 2.3);
```

שימו ♥:

- דרך הפנייה מסוג Animal ניתן לגשת רק לתכונות ולפעולות המוגדרות ב- Animal (ולכן הן משותפות לכל האובייקטים מהמחלקות היורשות).
 - לא ניתן לגשת לתכונות או פעולות שמוגדרות בתת החלקות (כי הפניה מסוג *הסל-חיים* לא מכירה אותן).
 - אי אפשר לבחור טיפוס אחר לאיברי המערך, למשל Dog כי בעלי החיים האחרים אינם סוג של כלב.
 - **מי שקובע אלו מהפעולות ניתן להפעיל זה טיפוס ההפניה** (רק פעולות המוגדרות במחלקה של טיפוס ההפניה). **מי שקובע איזו פעולה תופעל זה טיפוס העצם** (אבל - רק פעולות שטיפוס ההפניה מכיר).
- כלומר - ניתן לבקש מכל העצמים במערך להפעיל את הפעולה `say()` כי היא מוגדרת ב-Animal. כל איבר יפעיל את הפעולה הקרובה לו ביותר בעץ ההורשה. (הבוקר יפעיל את הפעולה שמוגדרת בבוקר. האברור והכא יפעילו את הפעולה הכתובה בכא. היונק יפעיל פעולה והסל חיים יפעילו כל אחד את הפעולה המוגדרת במחלקה שלו).

זימון פולימורפי של פונקציות

תגובה ייחודית: אם לאחת המחלקות יש מימוש שונה לפעולה שמוגדרת ב-Animal, יפעיל העצם של את הפעולה שבמחלקה שלו.

ל-Bird ו-Fish יש פעולות display() משלהם, ולכן הם יפעילו את הפעולה להם.

כל אובייקט שבמחלקה שלו מומשה הפעולה say(), יפעיל את ה-say() שלו.

אם לא קיים מימוש במחלקה שלו, יחפש את ה-say() הקרוב אליו ביותר בעץ ההורשה.

```
for (int i = 0 ; i < zoo.length; i++)
{
    System.out.print(zoo[i].display());
    System.out.println(":\t "+zoo[i].say());
}
System.out.println();
```

כל בע"ח במערך יפעיל את ה-say() וה-display() שלו

הפעולה שתבצע נקבעת לפי טיפוס האובייקט ולא לפי טיפוס ההפנייה אליו.

```
Dasty  female :   I am a Dog
Gupy   male   :   I am a Fish
Twitty female :   I am a Bied
Rex    male   :   I am a Boxer
Toto   male   :   I am a Dog
Mammal female :   I am a Mammal

num of males is: 4
num of Mammal is: 4
```

הצבת אובייקט מסוג תת-מחלקה במערך מסוג מחלקת העל, אינה משנה את האובייקט, אלא רק את הצורה שבה נתייחס אליו. ניתן להפעיל על האובייקט את כל הפעולות המוגדרות ומוכרות במחלקת-העל (במחלקה של טיפוס ההפניה), אך הוא יגיב בצורה הייחודית לו.

פולימורפיזם מבוסס על המרות

המרה = שינוי נקודת ההסתכלות על העצם.

ההמרה מתבצעת על ענף בעץ ההורשה (כלב הוא סוג של יונק ויונק הוא סוג של בעל-חיים. אבל לא סוג של ציפור או עט)

המרה כלפי מעלה - up casting :

הסתכלות על העצם כאילו היה מטיפוס מחלקת העל. הסתכלות על הכלב ועל היונק כאילו היו בע"ח. ההמרה נעשית על ידי הפניה מטיפוס מחלקת-העל שאליה ביצענו את ההמרה.

המרה כזו מאפשרת את הטיפול האחד לאיברים מטיפוסים שונים. (למשל: יצירת מערך של עצמים).

המרה כלפי מעלה נחשבת "המרה בטוחה" כי אובייקט מסוג תת-מחלקה הוא גם סוג של אובייקט מסוג

מחלקת העל (אבל היותך פנימי האמצעות הפנייה מסוג מחלקת-העל אפשרות שאינה מושגת במחלקת העל. שגיאה

מתקלה במהלך הקומפילציה).

המרה כלפי מעלה היא המרה אוטומטית שאינה מחייבת תוספת קוד מיוחדת.

דוגמא :

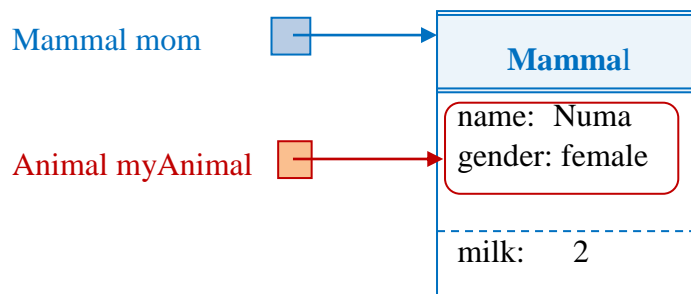
```
Mammal mom = new Mammal ("Numa", "female", 100, 2) ;
```

```
Animal myAnimal = mom;
```

ההפנייה myAnimal אינה משנה את העצם mom אלא רק את נקודת ההסתכלות עליו.

באמצעות mom נוכל לבצע את כל הפעולות המוגדרות על יונק.

באמצעות myAnimal נוכל לבצע רק את הפעולות המוגדרות במחלקה בע"ח.



אינה חוקית

myAnimal.getMilk()

הפעולה :

כי milk והפעולות עליו מוגדרות רק ב-Mammal ואילו myAnimal הינו הפנייה מסוג Animal.

המרה כלפי מטה - Down Casting :

כיצד נוכל להסתכל על בעלי החיים השונים שבמערך zoo כעל אובייקטים מסוג המחלקה שלהם?

בשתי דרכים :

- **המרה מפורשת :**

ניצור הפנייה מהסוג הרצוי (הפנייה מסוג שבין מחלקת העל והמחלקה של העצם - למשל : עבור Dog נוכל ליצור הפנייה מסוג Animal, Mammal, Dog), נמיר את העצם ונציבו בהפניה :

```
Mammal myMammal = (Mammal) myAnimal;
```

```
Labrador myLab = (Labrador) zoo[5];
```

```
Dog myDog = (Dog) zoo[5]; ; וכדי
```

לאחר ביצוע ההמרה, נוכל לפנות לתכונות ולפעולות של העצמים המומרים, לפי סוג ההמרה, כלומר - להתעניין בכמות החלב של myMammal, myLab, myDog ובכמות העצמות של myLab ו-myDog

- **המרה מרומזת לצורך מקומי :**

הסתכלות רגעית לצורך ביצוע הפעולה הנוכחית בלבד :

```
((Mammal)myAnimal).nurseFrom (mom)
```

שימו לב! יש חשיבות לסדר הביצוע ולכן חשיבות לסוגריים סביב העצם וטיפוס ההמרה.

*אזא אטרייט אלו. תתבצע תחילה הפעולה ואז ינסה להמיר את תוצאת הפעולה אטרייט יונק.
גור שוא כאובן לא אשפי!*

ניסיון לבצע המרה על ידי השמה בלבד תגרור הודעת שגיאה :

```
Mammal myMammal = myAnimal ; // שגיאה !!!
```

שגיאה מסוג אי התאמה בין טיפוסים : incompatible types.

המרה כלפי מטה נחשבת לא בטוחה כי אובייקט ממחלקת העל אינו בהכרח אובייקט של תת מחלקה. ההמרה תתבצע רק אם האובייקט המומר אמנם שייך לתת המחלקה אליה מבצעים את ההמרה. אם אינו שייך לענף שבין מחלקת העל ותת המחלקה, תתקבל **שגיאת זמן ריצה**, התכנית תעצר ותתקבל הודעת השגיאה :

Class Cast Exception - המרה בלתי חוקית.

דוגמא לשגיאה :

```
Animal rob = new Animal ("Rob", "male", 20);
```

```
Mammal mamRob = (Mammal) rob ; // run time Error
```

rob הוא מטיפוס Animal ולכן לא ניתן להמיר אותו לטיפוס שהוא תת-מחלקה של Animal. השגיאה תתגלה רק בזמן הריצה ולא בשלב ההידור.

מבחינת המהדר המתכנת ביצע המרה מפורשת. רק בזמן הריצה יתגלה ש- rob אינו מסוג יונק ולכן תיעצר התכנית ותתקבל הודעה שההמרה בלתי חוקית.

האם קטע הקוד הבא תקין?

- (1) Mammal goofy = new Mammal ("Goofy", "male", 250, 0);
- (2) Object obj = goofy ;
- (3) System.out.println (obj.getGender());
- (4) Animal ann = (Animal) obj ;
- (5) System.out.println (ann.getGender());
- (6) System.out.println ((Bird)obj).getName());

האם ההמרות שבקטע הקוד יגרמו לשגיאות הידור? לשגיאות ריצה? הסבר !!

- (2) המרה כלפי מעלה היא המרה בטוחה.
- (3) obj אינו מכיר את הפעולות של Animal (או של Mammal) ולכן תתקבל שגיאת הידור (שגיאת קומפילציה).
- (4) ביצענו המרה מפורשת לכן אין שגיאת הידור.
- (5) מכיוון שההמרה כלפי מטה שבשורה 4 היא המרה לטיפוס בעל-חיים, ויונק הוא סוג של בעל-חיים, הרי שההוראה תקינה (getGender() היא פעולה המוגדרת ב- Animal).
- (6) שגיאת זמן ריצה: obj הוא מטיפוס יונק וכן לא ניתן לבצע עליו המרה לטיפוס עוף. מבחינה תחבירית, רשמנו קוד תקין ולכן לא תתקבל שגיאת קומפילציה. השגיאה תתגלה רק בזמן הריצה, כשהמחשב ינסה לבצע א ההמרה.

האופרטור instanceof

אופרטור המאפשר לבדוק בזמן הריצה אם הפנייה מסויימת מפנה לעצם מטיפוס מסוים.

reference	instanceof	Type
משתנה/הפניה	instanceof	טיפוס

הביטוי מחזיר ערך בוליאני.

כדי להימנע מהמרה בלתי חוקית נרשום:

```
if (rob instanceof Mammal)
```

```
{
    Mammal mamRob = (Mammal) rob ;
    mamRob.addMilk(10);
}
```

```
count = 0;
for (int i = 0 ; i < zoo.length; i++)
    if (zoo[i] instanceof Mammal)
        count ++;
System.out.println("num of Mammal is: " + count);
num of Mammal is: 4
```

נשים לב שכל מי שהוא יונק או יורש מיונק עונה להגדרה זו

כיצד נוכל ליצור מערך של טיפוסים בסיסיים מסוג: *int, double, float, byte, long* ??

לכל אחד מהטיפוסים מחלקה עוטפת המגדירה פעולות בסיסיות שניתן לבצע על טיפוס זה.

מחלקת העל המשותפת היא Object, ולכן ניצור מערך של Object

באותה מידה נוכל להכניס גם ערכים מסוגים טקסטואליים וטיפוסים אחרים למערך זה.

```
Animal a1 = new Boxer ("Rexy", "male", 3.2, 3, "house");
Animal a2 = new Labrador ("Toto", "male", 2.5, 4, new Person("Hila"));

a1.say(); // Boxer הגדיר מחדש את say() ולכן יופעל say() של Boxer
a2.say(); // Labrador אין say() ולכן יופעל say() של Dog

Mammal m1 = (Mammal) a1; // המרה מפורשת
int milk1 = m1.getMilk();
int milk2 = ((Mammal)a2).getMilk() // המרה מרומזת

Boxer b1 = (Boxer) a2; // שגיאת הידור! a2 מטיפוס Labrador
// מתבצעת ההמרה אבל בעת ניסיון ההשמה ל-b1 מתגלה חוסר ההתאמה

Person owner = ((Labrador) a1).GetPerson(); // שגיאת זמן ריצה!
// a1 מטיפוס Boxer ולכן אין לו בעלים
// השגיאה תתגלה רק בזמן הריצה
```

