

פולימורפיזם - רב צורתיות



מה בשיעור

פולימורפיזם – הגדרה והסבר

המרות

המרה כלפי מעלה

המרה כלפי מטה

שימוש בפעולות

פעולות ממחלקת האב

דריסה

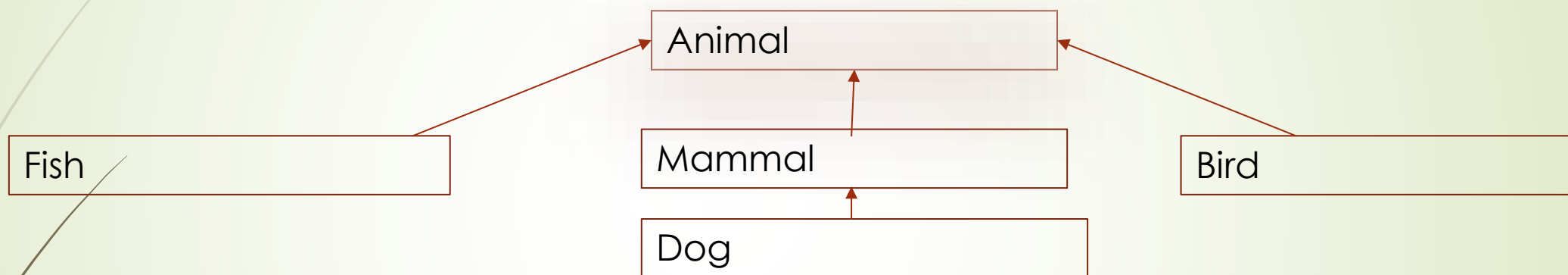
פעולות ממחלקת הבן

instance of



Polymorphism

פרוייקט דוגמא



בין תת מחלקה למחלקת העל שלה יש יחס של "סוג של" המאפשר לנו להסתכל על אובייקט גם כחלק מהמחלקה בה נוצר וגם כאובייקט של אחת המחלקות הנמצאות מעליו באותו עץ ירושה.

האפשרות להסתכל על אובייקט אחד בצורות שונות נקראת "רב צורתיות" או "פולימורפיזם" יתרונות הפולימורפיזם:

- אפשרות לטפל באופן אחיד בטיפוסים מסוגים שונים – כאילו נוצרו מאותו טיפוס.
- כל טיפוס יציג את מה שמוגדר במחלקה שלו.

מה זה?

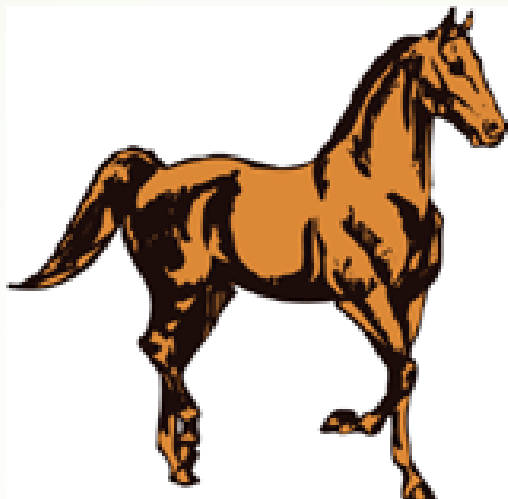


הכל תלוי בנקודת המבט שלנו!
על אותו אובייקט (כלב) נוכל להסתכל
בצורות שונות (=רב-צורתיות).



- כלב?
- יונק?
- בעל חיים?

מה זה?



• סוס?

• יונק?

• בעל חיים?

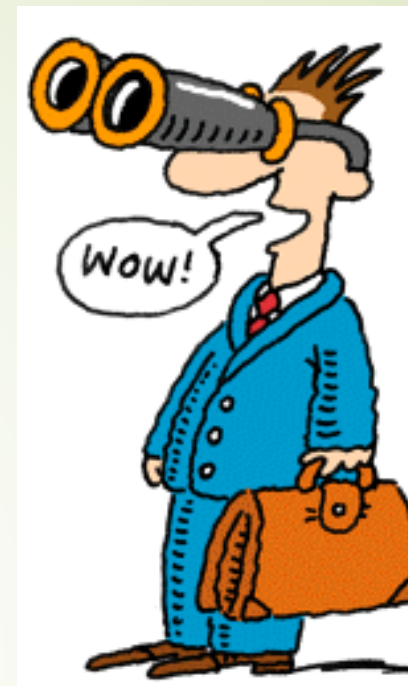
נוכל להסתכל על האובייקט בצורות שונות!

נעשה זאת לפי הצרכים שלנו.

מה זה?



אל אותו אובייקט (נשר) נוכל לפנות פעם
כאל עוף דורס, פעם כאל ציפור ופעם כאל
בעל חיים.



• עוף דורס?

• ציפור?

• בעל חיים?

רב-צורתיות – האפשרות להתייחס על ידי הפניה מטיפוס של מחלקת-העל גם לאובייקטים מתת-המחלקות



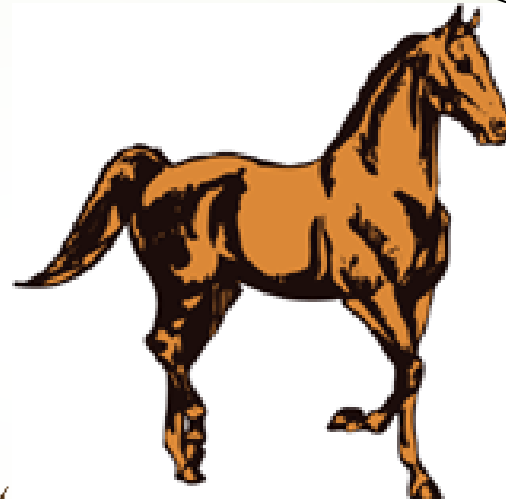
• בעלי חיים

• יונקים

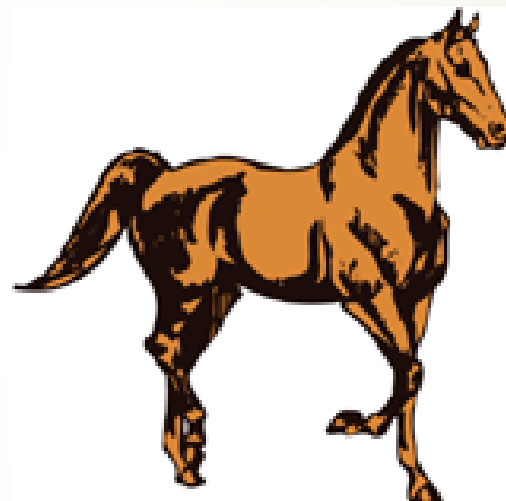
• כלבים

אם נשנה את נקודת המבט נשנה גם את קבוצת
האובייקטים עליה אנו מסתכלים ...

אם נסתכל על הטיפוס בעלי חיים,
נראה את ...




אם נסתכל על הטיפוס יונקים,
נראה את ...



אם נסתכל על כלבים אז נראה את ...





פולימופיזם – רב צורתיות - הגדרה

היכולת להסתכל על
אותו אובייקט בצורות
שונות



מה בשיעור

פולימורפיזם – הגדרה והסבר

המרות

המרה כלפי מעלה

המרה כלפי מטה

שימוש בפעולות

פעולות ממחלקת האב

דריסה

פעולות ממחלקת הבן

instanceof



וב java?



נשתמש בהמרה – העברת
עצם ממחלקה ("צורת
הסתכלות") אחת לשנייה.



המרה בטוחה

פולימורפיזם מבוסס על המרות בין טיפוסים שונים המופיעים באותו ענף של עץ ירושה נתון.
שני סוגי המרות:

המרה "כלפי מעלה" up casting – מאפשר להסתכל על עצם מתת מחלקה כאילו היה עצם ממחלקת העל שלו.

המרה "כלפי מטה" down casting – לקחת עצם שהמרנו כלפי מעלה ולהסתכל עליו שוב לפי הטיפוס המקורי שלו.

המרה כלפי מעלה



```
Mammal sara=new Mammal("sara","female",2,3);
```

```
Animal myAnimal=sara;
```

יצרנו עצם מטיפוס "יונק" שהפנייה sara מתייחסת אליו.

לאחר ההמרה נוכל להתייחס אליו כאל אובייקט ממחלקת העל.

Mammal sara

Name=Sara

Gender=Female

energy=2

milk=3

myAnimal

Name=Sara

Gender=Female

energy=2

כדי להסתכל על האובייקט כעל יונק נשתמש בהפנייה sara (מטיפוס יונק)
כדי להסתכל על אותו אובייקט כעל "בעל חיים" נשתמש בהפניה myAnimal
ניסיון לפנות לתכונה חלב דרך myAnimal – לא חוקית ! ניתן לגשת רק
לתכונות הטיפוס "בעל חיים".


המרה כלפי מטה down casting



המרה כלפי מטה מאפשרת לנו לשוב ולהסתכל על אובייקט שהמרנו כלפי מעלה על אובייקט מהמחלקה בה הוא נוצר במקור.

```
Mammal sara=new Mammal("sara","female",2,3);  
Animal myAnimal=sara; // up casting  
Mammal myMammal=(Mammal)myAnimal;
```



המרה כלפי מטה עלולה לגרום לשגיאות בזמן ריצה לכן בניגוד להמרה כלפי מעלה – יש לציין בעזרת שם הטיפוס (בסוגרים) 

המרה למטה – אזהרה!!

```
Animal sara=new Animal ("sara","female",5);
```

```
// sara is an animal
```

```
Mammal mamal=(Mammal) sara;// ???
```

ביצענו המרה מפורשת ולכן לא תהיה שגיאת הידור (יעבור קומפילציה).

אבל... **זהירות!!** בחרנו להסתכל על אובייקט שנוצר מהמחלקה Animal כאילו היה Mammal

כמובן – לא הגיוני - יגרום לשגיאת ריצה **Runtime Error**



סיכום ביניים... חשוב לזכור !!

- **המרה למעלה up casting** – (מיונק לבעל חיים) – המרה של הפניה ממחלקה למחלקת העל שלה – המרה בטוחה.
- **המרה למטה down casting** (מבעל חיים ליונק) המרה של הפניה ממחלקה לתת מחלקה - המרה לא בטוחה – יש צורך בהמרה מפורשת – אפשרי רק אם הטיפוס נוצר מראש בתת המחלקה.

מה בשיעור

פולימורפיזם – הגדרה והסבר

המרות

המרה כלפי מעלה

המרה כלפי מטה

שימוש בפעולות

פעולות ממחלקת האב

דריסה

פעולות ממחלקת הבן

instanceof





ביצוע המרות זימון שיטות – מחלקת האב

כל פעולה ממחלקת האב יכולה לשרת את המחלקה ואת תת המחלקות היורשות אותה.

לדוגמא אם הפעולה `getName()` מוגדרת כבר ב `Animal` ניתן לקרוא לה מכל תת המחלקות של `Animal`

ביצוע המרות זימון שיטות - דריסה (כתיבת פעולה בעלת אותה חותמת בדיוק בכל המחלקות)

גם לאחר המרה טיפוס המחלקה של האובייקט ולא של ההפניה הוא זה שיקבע איזו שיטה תזומן לדוגמא :

```
Mammal myMammal = new Mammal("joe","male",5,6);  
Animal ani = myMammal; // up casting  
System.out.println(ani.toString());
```

למרות שההפניה ani היא מטיפוס Animal (המרה למעלה) , הפעולה שתבצע היא toString במחלקה Mammal – המחלקה בה נוצר האובייקט.

ביצוע המרות וזימון שיטות - דריסה (כתיבת פעולה בעלת אותה חותמת בדיוק בכל המחלקות)

דריסה מאפשרת מימוש שונה בכל מחלקה והתייחסות למערך כאל טיפוס מסוג אחד למרות השוני בין הטיפוסים – לדוגמא `toString()` הממומש שונה בכל אחת מהמחלקות בפרוייקט. מאפשר קריאה אחידה לפעולה מכל אחד מהטיפוסים במחלקה.

```
for(int i=0;i<zoo.length;i++)  
{  
    System.out.println(zoo[i].toString());  
}
```

ומה כשפעולה לא מוגדרת בכל המחלקות?

instanceof

► **Instanceof** – אופרטור שבודק אם עצם הוא מופע של מחלקה מסויימת

```
Animal sara=new Animal ("sara","female",5);// bar is an animal
```

```
if (sara instanceof Mammal) {  
    Mammal mama=(Mammal) sara;// ???  
    mama.getMilk();  
}
```

► כדי למנוע שגיאות בזמן ריצה, נבצע זימון של פעולה לעצם שהומר כלפי מטה
(ממחלקת האב למחלקת הבן) רק לאחר שימוש באופרטור

אזהרות...

- אם יש מערך מסוג Animal שכל איבר במערך הוא מתת מחלקה אחרת של Animal כאשר רוצים לבצע פעולה במחלקה מסוימת (שאיננה Animal) יש לבצע המרה מפורשת.
- את ההמרה יש לבצע לאחר בדיקה באופרטור instanceof
- דוגמא : הפעולה myMammal - נמצאת במחלקה Mammal כדי להפעילה יש לבדוק האם האיבר הנבדק במערך נוצר כ Mammal אם כן – לבצע המרה ישירה לטיפוס ולהפעיל את הפעולה.

```
for(int i=0;i<zoo.length;i++)  
{  
    if(zoo[i] instanceof Mammal)  
    {  
        Mammal m=(Mammal)zoo[i];  
        m.myMammal();  
    }  
}
```



ואפשר גם בלי ליצור טיפוס חדש ?

כן ... אבל ... כרגיל.. צריך להזהר...

```
for(int i=0;i<zoo.length;i++)  
{  
    if(zoo[i] instanceof Mammal)  
    {  
        ((Mammal)zoo[i]).myMammal();  
    }  
}
```

חייבים לוודא שהטיפוס מתאים להמרה (instanceof)

להקיף בסוגרים את ההפניה ואת בקשת ההמרה כדי שיהיה ברור שמי שמומר זה ההפניה zoo[i] ולא מה שחוזר מהפעולה ! .

מה בשיעור

פולימורפיזם – הגדרה והסבר

המרות

המרה כלפי מעלה

המרה כלפי מטה

שימוש בפעולות

פעולות ממחלקת האב

דריסה

פעולות ממחלקת הבן

instance of




```
public class A
{
    private int myVal;
    public A (int val)    {myVal = val;}
    public int f ()      {return 1;}
}

public class B extends A
{
    private double x;
    public boolean validCode() {return x > 8.0;}
}
```

לפניך שתי הגדרות מהתכנית הראשית:

```
A code = new B(127 , 1.4);
```

```
A num = new A(613);
```

בעבור כל אחת מההוראות iv-i שלפניך, קבע אם היא תקינה או אינה תקינה.

אם אינה תקינה, נמק את קביעתך וכתוב אם זו שגיאת ריצה או שגיאת הידור (קומפילציה).

- i `boolean myBool = code.validCode();`
- ii `boolean myBool = num.validCode();`
- iii `boolean myBool = (B) code.validCode();`
- iv `boolean myBool = (B) num.validCode();`

```

public class A
{
    private int myVal;
    public A (int val)    {myVal = val;}
    public int f ()      {return 1;}
}

public class B extends A
{
    private double x;
    public boolean validCode() {return x > 8.0;}
}

```

A code = new B(127 , 1.4);

A num = new A(613);

בעבור כל אחת מהראיות iv-i שלפניך, קבע אם היא תקינה או אינה תקינה.

אם אינה תקינה, נמק את קביעתך וכתוב אם זו שגיאת ריצה או שגיאת הידור (קומפילציה).

- i boolean myBool = code.validCode();
- ii boolean myBool = num.validCode();
- iii boolean myBool = (B) code.validCode();
- iv boolean myBool = (B) num.validCode();

1. שגיאת קומפילציה!

אמנם הופעל בנאי של B אבל, בוצעה המרה מעלה ל A ולכן כדי לפנות לפעולה ב B יש לבצע המרה מפורשת חזרה מטה.

לפניך שתי הגדרות מהתכנית הראשית:

2. שגיאת קומפילציה num

הוא מסוג A ואין לו קשר לפעולה המבוקשת

3. עוד שגיאת קומפילציה ... שימו לב! ההמרה מתבצעת על תוצאות הפעולה !

כדי שיהיה תקין יש לרשום : myBool=((B)code).validCode()

4. שגיאת קומפילציה ...שוב.. ההמרה מתייחסת לתוצאות הפעולה ולא לטיפוס.

לו היה נרשם :

myBool=((B)num).validCode()

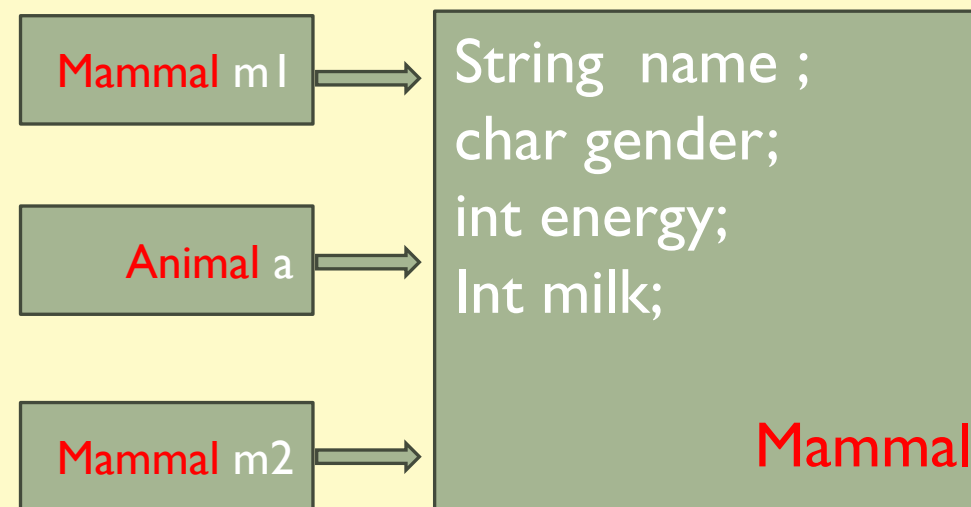
הייתה מתקבלת שגיאת הרצה – בוצעה המרה מפורשת מטה אבל הטיפוס לא נוצר כ B

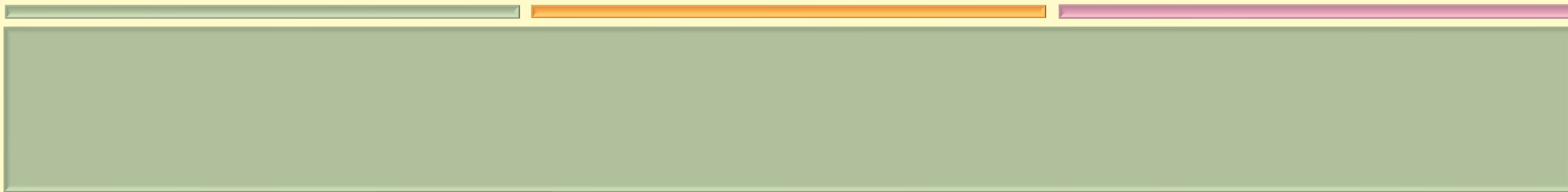
מעקב עצמים

```
Mammal m1 = new Mammal(...);
```

```
Animal a = m1;
```

```
Mammal m2 = (Mammal) a;
```





תודה רבה