מבוא למדעי המחשב – כתה יא-1

תאריך: 26.2.2026

תכנות מונחה עצמים – ריבוי צורות (Polymporphism)

תירגול #3 – חלק א' – הדגמה והסבר

מבוא:

בשיעור הקודם למדנו על ירושה.
ראינו איך אפשר לחסוך קוד, על ידי כך שהפעולות והמשתנים המשותפים עוברים למחלקה שנקראת מחלקת בסיס (Base Class).

כך מחלקות אחרות, הנקראות מחלקות יורשות, יכולות להשתמש בקוד המשותף מבלי לשכפל אותו שוב ושוב.

אולם עדיין קיימת בעיה תכנונית.

כאשר הווטרינר רצה לגרום לחיה "לדבר", הוא **היה צריך לבדוק באיזה סוג חיה מדובר** (Dog או Cat).

בנוסף, הצליל שכל חיה משמיעה אינו תלוי באובייקט מסוים של אותה חיה, אלא מאפיין את סוג החיה כולו — כל כלב נובח "Warf Warf", וכל חתול מיילל "Meou".

לכן, אין סיבה שהמחלקה Vet תכיר את הצליל של כל סוג חיה.

היינו רוצים שהקוד החיצוני יוכל להפעיל את **הפעולה Talk() על אובייקט מטיפוס Pet**, ובאופן פלאי כלשהו, אם זה חתול נשמע (או יודפס) "Meou" ואם זה אובייקט שמייצג כלב אז נראה או נשמע "Warf Warf".

כאשר קוד חיצוני בודק את סוג האובייקט כדי להחליט כיצד לפעול זה סימן שהתכנון אינו מנצל נכון את תכנות מונחה העצמים.

בשיעור הזה נלמד את הנושא הזה שנקרא ריבוי צורות. ונתרגל אותו.

החומר נועד לשימוש אישי של תלמידי תיכון קריית שרת בלבד.
אין להעתיק, לשכפל או להפיץ ללא רשות

© צבי מלמד

תזכורת מהקוד של השיעור על ירושה

```
class Pet
{
    public void Talk(string sound)
    {
        Console.WriteLine(GetName() + " says: " + sound);
    }


class Vet
{
    public void CheckPet(Pet p)
    {
        Console.WriteLine("Checking: " + p.GetName());

        // כרגע הווטרינר מחליט איזה צליל להשמיע
        if (p.GetName().StartsWith("Dog"))
        {
            p.Talk("Warf Warf");
        }
        else
        {
            p.Talk("Meow");
        }
```

בקוד הזה (כאן למעלה):

- מי מחליט איזה צליל להשמיע?
- מי יודע שכלב נובח וחתול מיילל?

תשובה: המחלקה Vet – וזה לא מה שצריך להיות.

מה שהיינו רוצים זה שהקוד יהיה הרבה יותר פשוט:

```
public void CheckPet(Pet p)
{
    Console.WriteLine("Checking: " + p.GetName());
    p.Talk();
```

טיפוסים בירושה - IS-A and Has-A

נשים לב לקוד שביצע את הקריאה לפעולה CheckPet(Pet p).

```
        Dog d = new Dog("Luna", 3);
        Cat c = new Cat("Mizi", 2);

        Vet v = new Vet();

        v.CheckPet(d);
        v.CheckPet(c);
```

אנחנו יודעים שצריכה להיות התאמת טיפוסים בין הפרמטר של הפונקציה לבין הארגומנט שמשתמשים בו בקריאה לפונקציה. אז משהו כאן "מוזר" – אולי לא מסתדר לנו?

אבל זה עובד! הסיבה שזה עובד – כי בירושה מתקיים: Dog is a Pet וגם Cat is a Pet.

ולכן, פונקציה שמקבלת כארגומנט Pet יכולים לקרוא לה עם אובייקט מסוג כלב או חתול.

החומר נועד לשימוש אישי של תלמידי תיכון קריית שרת בלבד.
אין להעתיק, לשכפל או להפיץ ללא רשות

© צבי מלמד

<<< הדגמה והסברים בכיתה: התכנית Pet & Vet & Cat & Dog בריבוי צורות. >>>

<u>סיכום – ריבוי צורות(Polymorphism)</u>

- **אותה פעולה – התנהגות שונה** לפי סוג האובייקט בפועל
- במחלקת הבסיס מגדירים **פעולה וירטואלית:** `virtual`
- במחלקות היורשות עושים **דריסה:** `override`
- אפשר להחזיק משתנה מטיפוס בסיס:

```
Pet p1 = new Dog(...)   or
Pet p2 = new Cat(...)
```

- קריאה דרך משתנה בסיס: `p1.Talk()  or p2.Talk()` מפעילה את הפעולה המתאימה **לסוג האמיתי** של האובייקט

לא בודקים את הטיפוס עם `if` – כל מחלקה "יודעת" איך לבצע את הפעולה שלה.

<u>שאלה:</u>

כפי שהדגמנו הקוד הבא עובד:

```
public class Vet
{
    public void CheckPet(Pet p)
    {
        Console.WriteLine("Checking: " + p.GetName());
        p.Talk();
        Console.WriteLine();
    }
```

איך לדעתכם הקומפיילר יתייחס לקוד הבא:

```
        Pet d = new Dog("Luna", 3);
        Pet c = new Cat("Mizi", 2);

        Pet[] arr = new Pet[2];
        arr[0] = d;
        arr[1] = c;

        Console.WriteLine("=== Array demo ===");
        for (int i = 0; i < arr.Length; i++)
        {
            Console.WriteLine(arr[i]);   // ToString
            arr[i].Talk();               // polymorphism
        }
```

אם נניח שהקומפיילר מוכן לקבל את זה, עדיין נשאלה השאלה – האם זה נחוץ?

החומר נועד לשימוש אישי של תלמידי תיכון קריית שרת בלבד.
אין להעתיק, לשכפל או להפיץ ללא רשות

© צבי מלמד