

## מבוא לתכנות II

הרצאה 5 – OOP III – מחלקות חסומות,  
אובייקט וסוג.

סמסטר 2

# מחלקה חסומה (sealed)

- לא ניתן לרשת ממחלקה חסומה.

```
sealed class Base  
{  
    ...  
}
```

```
class Derived : Base  
{
```

שגיאת  
קומפילציה

# פונקציה חסומה

- מילת המפתח *sealed* יכולה להיכתב לפני מילת המפתח *override* על מנת למנוע את שרשרת הרב צורתיות מלהמשיך למחלקות היורשות הבאות.
- זה אומר שהמימוש הבא של הפונקציה במחלקה הנגזרת חייב להיות *new*.

```
class Base
{
    public virtual void Print()
    {
        Console.WriteLine("in the Base class.");
    }
}

class Derived : Base
{
    public override sealed void Print()
    {
        Console.WriteLine("in the Derived class.");
    }
}
```

המשך בעמוד הבא

# פונקציה חסומה

```
class DerivedFromDerived : Derived
{
    //public override void Print() //ERROR beacuse sealed!
    public new void Print() //a new method
    {
        Console.WriteLine("in the Derived from Derived class.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Base b = new Base();
        b.Print();

        b = new Derived();
        b.Print();

        b = new DerivedFromDerived();
        b.Print(); //will print the Derived beacuse of the sealed can
        not be overriden

    }
}
```

in the Base class.

in the Derived class.

in the Derived class.

# סיכום

- *Virtual* - מימוש ראשוני.
- *override* - מימוש נוסף.
- *new* - מימוש חדש.
- *sealed* – מימוש אחרון. לאחר מכן חייב להיות *new*.

# מחלקת אובייקט

- כפי שאנו כבר יודעים, #C היא שפה מוכוונת עצמים וכל סוג הוא מחלקה.
- למעשה, כל מחלקה ב-#C יורשת באופן אוטומטי ממחלקת *object*.
- כל מחלקה נגזרת יורשת את מחלקת *object* בהיררכיה שלה.
- אז מה יש במחלקת *object*?

# מחלקת אובייקט

- מחלקת *object* מכילה מספר פונקציות סטנדרטיות אשר קיימות בכל מחלקה.
- רוב הפונקציות מוגדרות בתור *virtual* כדי שנוכל לממש את הפעולה המתאימה באמצעות *override* במידת הצורך.
- להלן כמה פונקציות לדוגמה:

```
public virtual bool Equals(object obj);  
public virtual string ToString();  
public Type GetType();  
public virtual int GetHashCode();
```

# Person class – דוגמה

```
public class Person
{
    private int id;
    private string name;
    private DateTime birthDate;
    ...
    public int Age
    ...
}
```



# ToString

```
public virtual string ToString();
```

- ToString – מחזירה תיאור של האובייקט בפורמט של מחרוזת.
- כאשר אין `override`, היא מחזירה את המסלול המלא של האובייקט, כולל הnamespace, לדוגמה: `Object.Person`

```
public override string ToString()
{
    string str =
        "ID: " + ID + "\n" +
        "Name: " + Name + "\n" +
        "BirthDate: " + BirthDate.ToShortDateString() + "\n" +
        "Age: " + Age;

    return str;
}
```

# Equals

```
public virtual bool Equals(object obj);
```

- Equals – מקבלת אובייקט כפרמטר ומחזירה אמת אם הוא שווה ערך לאובייקט שעליו הפעילו את הפונקציה.
- כאשר אין override זה עושה פעולה דומה לאופרטור ==.
- טיפ – זכרו כי ניתן להמיר את הפנית הבסיס להפניה לנגזרת.
- ברירת המחדל של Equals (ושל ==) בין אובייקטים היא בדיקה האם ההפניות שוות.

```
public override bool Equals(object obj)
{
    return ID == ((Person)obj).ID;
}
```

# GetHashCode

```
public virtual int GetHashCode();
```

- GetHashCode – מחזירה ערך ייחודי עבור האובייקט.
- טבלת גיבוב (Hash Table) – מבנה מידע (כמו מערך, מטריצה...) אשר לכל אובייקט המאוחסן בו יש ערך מפתח הייחודי לו בלבד, ושלא יכול להופיע יותר מפעם אחת. למשל, אדם- < ת"ז, עובד - < מספר עובד, מכונית - < מספר לוחית רישוי.
- החיפוש בטבלת גיבוב נעשה באופן מהיר ויעיל.
- הפונקציה הזו צריכה להחזיר מספר ייחודי אשר לא יחזור על עצמו באובייקט אחר.
- ישנן מחלקות אשר לא חייבות במפתח ייחודי, כמו: נקודה, מעגל, תאריך.
- כאשר אין override זה יחזיר מספר ייחודי אך חסר משמעות.

```
public override int GetHashCode()  
{  
    return ID;  
}
```

# GetHashCode

- לפעמים זה הולם להיעזר ב `GetHashCode()` בתוך `.Equals()`

```
public override bool Equals(object obj)
{
    int hashCode1 = obj.GetHashCode();
    int hashCode2 = GetHashCode();

    if (hashCode1 == hashCode2)
        return true;

    return false;
}
```

# GetType

```
public Type GetType();
```

- GetType - מחזיר את הסוג של האובייקט תוך שימוש במחלקת *Type* (פירוט בשקף הבא).
- לא virtual ולכן לא ניתן לעשות override.

```
Person p1 = new Person(564971324, "Avi", new DateTime(2001, 10, 9));  
Console.WriteLine(p1.GetType());
```

Object.Person

# מחלקת סוג

- מחלקת *Type* היא מחלקה מיוחדת.
- המחלקה הזו מכילה את כל הפרטים של סוג מסוים:
  - שם
  - Namespace
  - ציבורי או לא?
  - מה הפונקציות שלו.
  - ...

# מחלקת סוג - המשך

- ישנן 2 דרכים ליצור אובייקט *Type*:

– `Obj.GetType()` - משתמשים בזה כאשר האובייקט קיים


```
Person p1 = new Person(564971324, "Avi", new DateTime(2001, 10, 9));
```

```
Type t1 = p1.GetType();
```

```
Console.WriteLine(t1.Name);
```

```
Console.WriteLine(t1.Namespace);
```

```
Console.WriteLine(t1.IsPublic);
```



Person  
Object  
True

# מחלקת סוג - המשך

– `Typeof(className)` - משתמשים בזה כאשר אין אובייקט קיים

```
Type t = typeof(int);  
  
Console.WriteLine("Name: {0}", t.Name);  
Console.WriteLine("IsAbstract: {0}", t.IsAbstract);  
Console.WriteLine("IsArray: {0}", t.IsArray);  
Console.WriteLine("IsPublic: {0}", t.IsPublic);  
Console.WriteLine("IsValueType: {0}", t.IsValueType);  
Console.WriteLine("Namespace: {0}", t.Namespace);
```

Name: Int32  
IsAbstract: False  
IsArray: False  
IsPublic: True  
IsValueType: True  
Namespace: System



# מחלקת סוג - המשך

- כדי להשיג את מרכיבי המחלקה נוכל להיעזר בקוד הבא:  
(קריאת המידע מהמטא-נתונים אשר שוכנים באסמבלי  
נקראת שקיפות-reflection)

```
MemberInfo[] members = t2.GetMembers();  
for(int i=0; i<members.Length; i++)  
{  
    Console.WriteLine(members[i]);  
}
```

```
Int32 get_ID()  
Void set_ID(Int32)  
System.String get_Name()  
Void set_Name(System.String)  
System.DateTime get_BirthDate()  
Void set_BirthDate(System.DateTime)  
Int32 get_Age()  
Void Print()  
Int32 GetHashCode()  
Boolean Equals(System.Object)  
System.String ToString()  
System.Type GetType()  
Void .ctor()  
Void .ctor(Int32, System.String, System.DateTime)  
Int32 ID  
System.String Name  
System.DateTime BirthDate  
Int32 Age
```

# מחלקת סוג - המשך

- כדי לקבל רק את הפונקציות נוכל להיעזר בקוד הבא:

```
MethodInfo[] methods = t.GetMethods();  
for (int i = 0; i < methods.Length; i++)  
{  
    Console.WriteLine(methods[i]);  
}
```

```
Int32 get_ID()  
Void set_ID(Int32)  
System.String get_Name()  
Void set_Name(System.String)  
System.DateTime get_BirthDate()  
Void set_BirthDate(System.DateTime)  
Int32 get_Age()  
Void Print()  
Int32 GetHashCode()  
Boolean Equals(System.Object)  
System.String ToString()  
System.Type GetType()
```

# is ו- as

- *is* – מחזיר האם האובייקט הוא מסוג מסוים.
- *as* – מחזיר המרה של אובייקט לסוג מסוים, אם לא מצליח מחזיר null.
- (אפשרי גם באמצעות שימוש ב *cast* ותפיסת חריגות עם *catch*, אבל *is* אלגנטי יותר)

```
class Base
{
    public int x;

    public virtual void Print()
    {
        Console.WriteLine("Base print");
    }
}
```


המשך בדף הבא...

# is - המשך

```
class Derived : Base
{
    public int y;

    public override void Print()
    {
        Console.WriteLine("Derived print");
    }
}
class Program
{
    static void Main(string[] args)
    {
        Base b = new Derived();
        b.Print();

        //option 1:
        if (b is Derived)
            ((Derived)b).y = 10;
        b.Print();
        //option 2:
        Derived d = b as Derived;
        if (d != null)
            d.y = 10;
        b.Print();
    }
}
```



Derived print: 0  
Derived print: 10  
Derived print: 10

- צרו מחלקת מכונית עם השדות הבאים:

- License plate number

- Model

- Company ( Enum )

- Build year

- Color

- ממשו את כל פונקציות מחלקת Object במכונית.