מבוא לתכנות II

הרצאה 7– OOP IV – מחלקה מופשטת וממשק

סמסטר 2

מחלקה מופשטת

- מחלקה מופשטת הינה מחלקה שקיימת רק כבסיס למחלקות אחרות, אך אין צורך באובייקטים משלה.
- כדי להצהיר על מופשטות המחלקה יש רק לרשום את מילת המפתח *abstract* לפני class.
- זה בלתי אפשרי ליצור אובייקט של מחלקה מופשטת.
- נוכל ליצור הפניה עבור מחלקה מופשטת (לצרכי פולימורפיזם).
- יכולה להכיל כל מה שמחלקה רגילה מכילה.

```
abstract class Base
{…}
class Program
{
        //Base b = new Base(); //ERROR
```

מחלקה מופשטת

- יכולה להכיל פונקציות מופשטות – פונקציה ללא מימוש (רק הכרזה).
- מחלקה רגילה לא יכולה להכיל פונקציה מופשטת.

```
abstract class Base
{
    public int ID { get; set; }
    public abstract void Print();
```

- כאשר אנו יורשים (במחלקה רגילה) מחלקה מופשטת, אנו מחויבים לממש את כל הפונקציות המופשטות שבה.
- זה נעשה באמצעות מילות המפתח *override*, שכן פונקציות מופשטות הן בהכרח *virtual* במרומז.
- לרוב, המחלקה העליונה ביותר בשרשרת ההורשות היא מחלקה מופשטת אשר מכילה את כל הפונקציות המשותפות. זה מאפשר לנו ליצור הפניה אליה ולרוץ על מערך המחלקות הנגזרות שלה.

מחלקה מופשטת

- לדוגמה, המחלקה "צורה" אמורה להיות מופשטת, מכיוון שלא ניצור אובייקטים שלה.
- כן ניצור אובייקטים של מחלקות שיורשות מ"צורה", כגון: משולש, ריבוע, עיגול...

```
abstract class Shape
{
        public abstract double GetArea();
```

- כפי שניתן לראות, הפונקציה צריכה להיות מופשטת, שכן אין היגיון בלממש חישוב שטח לצורה כללית.
- אבל, כשנממש צורות מוגדרות כגון ריבוע ומשולש, נוכל לממש עבורן באופן הגיוני את GetArea().
- כעת נוכל ליצור מערך מסוג "צורה" ולהריץ את GetArea() עבור כל הצורות.

דוגמה 1

```
abstract class Base
{
    public int ID { get; set; }
    public abstract void Print();
    //instead of the following - think of a function
    //which would return something
    //public virtual void Print()
    //{
    //    return;
    //}
}

class Derived : Base
{
    public override void Print()
    {
        Console.WriteLine("print");
    }
}
abstract class derived2 : Base
{
    //dont have to implement Print in an abstract class!

    public abstract int Num();
}
```

המשך דוגמה 1

```
class Derived3 : derived2
{
    public override void Print()
    {
        Console.WriteLine("print");
    }

    public override int Num() { return 7; }
}


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

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

        derived2 d2 = new Derived3();
        Console.WriteLine(d2.Num());
    }
}
```

02 - Shapes

- Show example 02 - Shapes

ממשק

- ממשק הוא מרכיב לוגיקה מופשטת אשר מכיל רק הכרזות.

```
Interface IPrint
{
        void Print();
}
```

- לא יכול להכיל שדות או מימושים.
- כאשר אנו מממשים (לא יורשים ממשק) ממשק, אנו חייבים לממש את כל ההכרזות שבו.
- אנו יכולים לרשת רק מחלקה <u>אחת</u> אבל לממש <u>הרבה</u> ממשקים.

ממשק

- לא אפשרי ליצור אובייקט של ממשק, רק הפנייה.
- נהוג להתחיל שמות של ממשקים עם האות I.
- מדוע?
- על מנת ליידע מחלקות אחרות לגבי פונקציה שאני מממש במחלקה שלי. "אני" מממש ממשק מסוים שיש לו את הפונקציה הזו!
- במילים אחרות: ממשקים נועדו למימוש על ידי מחלקות כדי שנדע שאובייקט של המחלקה יכול להריץ את הפונקציות אשר הוכרזו בממשק. אנו יוצרים "חוזה בין מחלקות". כאשר מחלקה מקבלת אובייקט של ממשק מסוים מבחוץ, היא יכולה לדעת שפונקציה של הממשק היא ברת הרצה/מימוש באובייקט.

דוגמה

```
class Person : IPrint
{
    public void Print()
    {
            Console.WriteLine("Print");
    }
}

…
IPrint p = new Person();
p.Print();
```

- שימו לב שסוג ההפניה הינה של הממשק.

ממשקים ידועים

- בסביבה של NET. ישנם מספר רב של ממשקים מוכרים אשר מציגים יכולות שימושיות, כגון:
  - IComparable
  - IComparer
  - IEnumerable
  - IEnumerator
- בואו נסתכל בIcomparable אשר בעל פונקציה אחת בלבד למטרת השוואת אובייקטים ובהסבר של הערך שהוא מחזיר:

```
public interface IComparable
{
        int CompareTo(object obj);
}
```

מחזיר:
ערך אשר מציין את הסדר היחסי של האובייקטים המושווים. משמעות הערכים מתחלקת כך:
ערך אשר קטן מ0 = מופע זה קטן מהאובייקט אליו משווים.
ערך השווה ל0 = המופע שווה ערך לאובייקט.
ערך אשר גדול מ0 = מופע זה גדול מהאובייקט אליו משווים.

IComparable

- מחלקת "מערך" מכילה פונקציה בשם Array.Sort().
  הintellisense מציין כי פונקציה זו ממיינת את המערך באמצעות המימוש של Icomparable.
- אם נמיין את הסוגים הבסיסיים כמו int, string... ממשק הIcomparable ממומש כבר עבורם.
- אך מה יקרה אם נרצה למיין מערך של אנשים?
- אז נצטרך לממש את Icomparable במחלקת האדם כדי לאפשר לArray.Sort() לעשות את כל העבודה.
- שימו לב שהפונקציה היחידה בIcomparable מתמודדת עם השוואת שני אובייקטים בלבד, לא מערך שלם.

דוגמה Icomparable

```
public class Person : IComparable
{
    private int age;
    public int ID { get; set; }
    public static bool sortById = false;
…
        public int CompareTo(object obj)
        {
            Person p = (Person)obj;
            if (sortById)
            {
                if (ID< p.ID)
                    return -1;
                if (ID > p.ID  )
                    return 1;
                return 0;
            }
            else
            {
                if (Age < p.Age)
                    return -1;
                if (Age > p.Age)
                    return 1;
                return 0;
            }
        }
…
```

דוגמה Icomparable

```
Person[] pArr = new Person[3];
pArr[0] = new Person(5,100);
pArr[1] = new Person(2,20);
pArr[2] = new Person(12,15);

for (int i = 0; i < pArr.Length; i++)
    pArr[i].Print();

Console.WriteLine("-sort by AGE--------------------");
Array.Sort(pArr);

for (int i = 0; i < pArr.Length; i++)
    pArr[i].Print();


Console.WriteLine("-sort by ID--------------------");
Person.sortById = true;
Array.Sort(pArr);

for (int i = 0; i < pArr.Length; i++)
    pArr[i].Print();
```

-מיון לפי גיל-
Age: 2, ID: 20
Age: 5, ID: 100
Age: 12, ID: 15

-מיון לפי ת"ז-
Age: 12, ID: 15
Age: 2, ID: 20
Age: 5, ID: 100

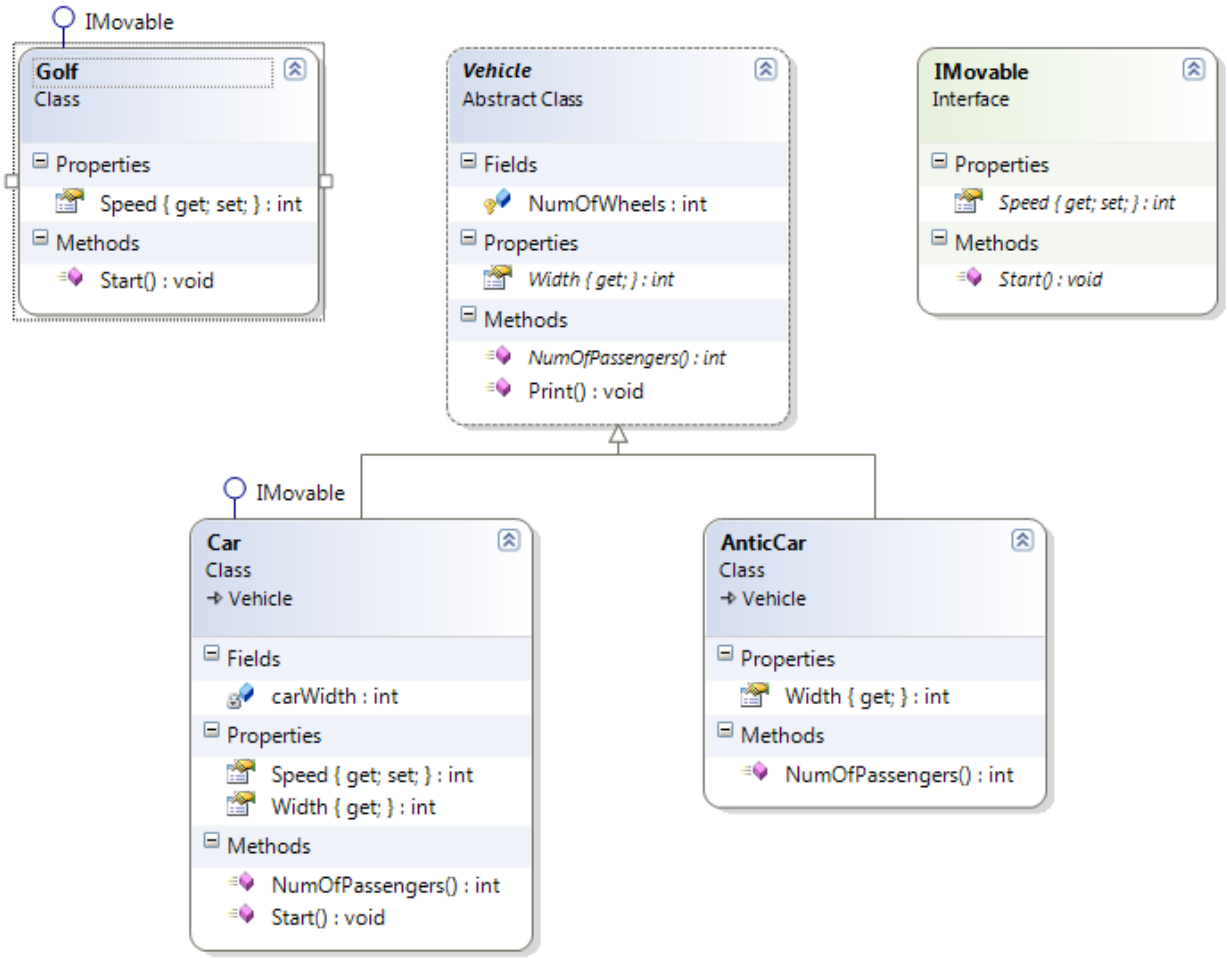
IMovable - 4 תרגיל

```
static void Main(string[] args)
{
    Car c = new Car();
    AnticCar t = new AnticCar();
    Golf g = new Golf();

    StartEverything(c);
    StartEverything(g);
    //StartEverything(t);
}

static void StartEverything(IMovable something)
{
    something.Start();
}
```

```
Car is now started!
Starting to swing...
Press any key to continue . . .
```



- *Italic* means not implemented ➔abstract\interface

תרגיל - צורה

- הוסיפו למחלקת הצורה את הממשק Icomparable.
- צרו מערך של צורות.
- הדפיסו לפני ואחרי המיון.