C# - תכנות מכוון עצמים

ממשק interface

מהו ממשק (interface)?

- ממשק הוא מבנה אבסטרקטי ללא תכונות המכיל חתימת מתודות בלבד
- כל השיטות בממשק הן אבסטרקטיות וציבוריות
- שם הממשק מתחיל באות I (קיצור של Interface) ובנוסף נהוג שיסתיים ב- able
  - למשל: IComparable, IClonable, IPrintable
- ממשק מכיל אוסף התנהגויות בעלי מכנה משותף
  - למשל גם Person וגם Circle יודעים להדפיס את עצמם, כלומר שניהם ניתנים להצגה, כלומר IPrintable. האם זה נכון להגיד שאדם הוא סוג Printable? לא, וכנ"ל גם Circle ולכן אין זו ירושה.

שימושים לממשקים

- מחלקה יכולה לממש אינסוף ממשקים (בניגוד למחלקות, שניתן לרשת רק אחת)
  - זו תהייה הדרך שלנו "לרשת מכמה מחלקות"
- מאחר וממשק הוא מעין מחלקה אבסטרקטית, לא ניתן להגדיר אובייקטים מטיפוס הממשק, אבל כן ניתן להגדיר הפניות Pointers
- מחלקה שמחליטה לממש ממשק חייבת לממש את כל השיטות שיש בממשק (ממשק מגדיר חוזה) אלא אם היא מחלקה אבסטרקטית

דוגמא: הממשק IPrintable


```
interface IPrintable
{
    void print();
}
```

```
class Rectangle : IPrintable
{
    private int height, width;

    public Rectangle(int height, int width)
    {
        this.height = height;
        this.width = width;
    }


    #region IPrintable Members

    public void print()
    {
        for (int i = 0; i < height; i++)
        {
            for (int j = 0; j < width; j++)
                Console.Write("*");
            Console.WriteLine();
        }
    }

    #endregion
}
```

זהו ממשק בשם IPrintable
שמכיל מתודה אחת בשם print
שהיא public abstract

ציון מימוש הממשק

מימוש השיטה
המוגדרת בממשק

דוגמא: הממשק IPrintable (2)


```
class Person : IPrintable
{
    protected int id;
    protected string name;

    public Person(int id, string name)
    {
        this.id = id;
        this.name = name;
    }



    #region IPrintable Members

    public void print()
    {
        Console.WriteLine("Id: {0}, Name: {1}", id, name);
    }

    #endregion
}
```

דוגמה ל- Main

- Implicit casting (Up Casting)

```
Person person = new Person(1, “John”);
person.Print();

IPrintable printable = person; //this is polymorphism
printable.Print();
```

- Explicit Casting (Down Casting)

```
Object obj = new Rectangle(5, 10);
Rectangle rect = (Rectangle)obj; //this is polymorphism but we need to check if obj is Rectangle before
rect.Print();

IPrintable printable = (IPrintable)obj; // this is polymorphism but we need to check if obj is IPrintable before
printable.Print();
```

תחביר

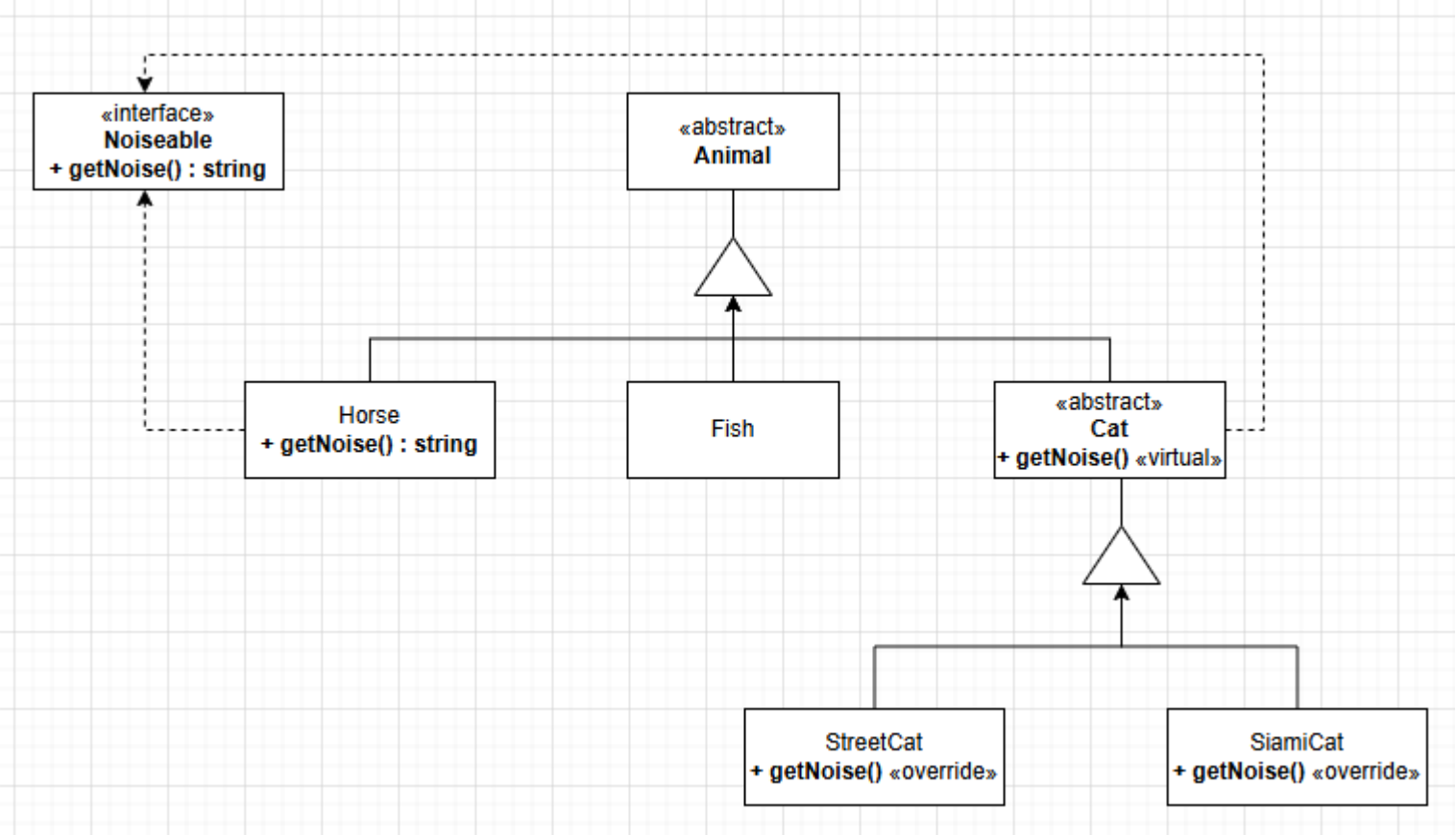
- מחלקה יכולה לרשת מחלקה אחת בלבד
- מחלקה יכולה לממש כמה ממשקים שהיא רוצה
- כאשר מחלקה יורשת מחלקה וממשת ממשקים, הממשקים יבואו לאחר המחלקה
- ממשק יכול לרשת ממשק אחד או יותר לצורך הרחבה

דוגמא: המחלקה Animal והממשק Noiseable

- לא כל החיות משמיעות קול, לכן לא נרצה לאפשר שירות זה עם מימוש ריק במחלקת הבסיס Animal
- מאחר והתוספת לחיות שעושות קולות הינה התנהגותית, נייצר ממשק שמכיל את היכולת getNoise
- חיות שיודעות להשמיע קול ירשו מ- Animal וימשו את הממשק Noiseable

```
interface Noiseable
{
    string getNoise();
}
```

דוגמא: המחלקה Animal והממשק Noiseable – תרשים UML



המחלקה Animal


```
abstract class Animal
{
    protected string color;
    protected int numOfLegs;

    public Animal(string color, int numOfLegs)
    {
        this.color = color;
        this.numOfLegs = numOfLegs;
    }

    public override string ToString()
    {
        return GetType().Name + ": Color: " + color +
               "\tNumOfLegs: " + numOfLegs + "\t";
    }

    public int GetNumOfLegs()
    {
        return numOfLegs;
    }


    public string GetColor()
    {
        return color;
    }
}
```

המחלקה Horse


```
class Horse : Animal, Noiseable
{
    public const int NUM_OF_LEGS = 4;
    private int tailLen;

    public Horse(string color, int tailLen)
        : base(color, NUM_OF_LEGS)
    {
        this.tailLen = tailLen;
    }

    public void ride()
    {
        Console.WriteLine("I'm riding :-)");
    }

    public override string ToString()
    {
        return base.ToString() + " TailLen: " + tailLen + "cm";
    }


    #region Noiseable Members
    public string getNoise()
    {
        return "Hiyaaa!";
    }
    #endregion
} // class Horse
```

המחלקה Fish


```
class Fish:Animal
{
    public const int NUM_OF_LEGS = 0;

    private int swimSpeed;

    public Fish(string color, int swimSpeed)
        : base(color, NUM_OF_LEGS)
    {
        this.swimSpeed = swimSpeed;
    }

    public override string ToString()
    {
        return base.ToString() + " Swim Speed: " + swimSpeed + "kmh";
    }

} // class Fish
```

המחלקה Cat


```
abstract class Cat: Animal, Noiseable
{
    public const int NUM_OF_LEGS = 4;

    protected int mustachLen;

    public Cat(string color, int mustachLen)
        : base(color, NUM_OF_LEGS)
    {
        this.mustachLen = mustachLen;
    }

    public void scratch()
    {
        Console.WriteLine("Scratching!!");
    }

    public override string ToString()
    {
        return base.ToString() + " MustachLen: " + mustachLen + "cm";
    }

    #region Noiseable Members
    public virtual string getNoise()
    {
        return "Miyaooo!";
    }
    #endregion
} // abstract class Cat
```

המחלקה StreetCat


```
class StreetCat : Cat
{
    private int numOfFights;

    public StreetCat(string color, int mustachLen, int numOfFights)
        : base(color, mustachLen)
    {
        this.numOfFights = numOfFights;
    }

    public override string getNoise()
    {
        return base.getNoise() + ", I want to see a dog!";
    }

    public void fight()
    {
        Console.WriteLine("I want to have a good fight! Any volunteer?");
    }

    public override string ToString()
    {
        return base.ToString() + " NumOfFights: " + numOfFights;
    }
} // class StreetCat
```

המחלקה SiamiCat


```
class SiamiCat : Cat
{
    public SiamiCat(string color, int mustachLen)
        : base(color, mustachLen)
    {
    }

    public override string getNoise()
    {
        return base.getNoise() + ", I'm so spoiled!";
    }
} // class SiamiCat
```

ה- main – פולימורפיזם בעזרת ממשקים


```
static void Main(string[] args)
{
    Animal[] allAnimals = new Animal[4];
    allAnimals[0] = new Horse("Brown", 120);
    allAnimals[1] = new SiamiCat("Gray", 5);
    allAnimals[2] = new StreetCat("Black", 7, 23);
    allAnimals[3] = new Fish("Gold", 20);

    for (int i = 0; i < allAnimals.Length; i++)
    {
        if (allAnimals[i] is Noiseable)
        {
            Console.WriteLine("Animal {0} noise: {1}", i,
                    ((Noiseable)allAnimals[i]).getNoise());
        }
    }
}
```

פולימורפיזם

במקום לבדוק האם החיה היא חתול או סוס (וטיפוסים שונים בהמשך), נבדוק האם האובייקט מממש את ההתנהגות המבוקשת

פולימורפיזם

```
Animal 0 noise: Hiyaaa!
Animal 1 noise: Miyaooo!, I'm so spoiled!
Animal 2 noise: Miyaooo!, I want to see a dog!
```

ירושת ממשקים


```
interface Swimmable
{
    void swim();
}
```

```
interface Rideable
{
    void ride();
}
```

```
interface Smokeable
{
    void smoke();
}
```

```
interface BetterSwimmable : Swimmable
{
    void swimFast();
}
```

ממשק זה יכלול את כל השיטות שבמשק שאותו הרחיב + השיטות הנוספות

```
interface Athletable : BetterSwimmable, Rideable
{
    void breath();
}
```

הרחבה של יותר ממשק אחד

```
class Person : Object, Athletable, Smokeable
```

How many methods Person must implement?

ירושת ממשקים ומחלקה


```
class Person : Object, Athletable, Smokeable
{
    #region Smokeable Members
    public void smoke()   {Console.WriteLine("I'm smoking!");}
    #endregion

    #region BetterSwimmable Members
    public void swimFast() {Console.WriteLine("I'm swimming fast!");}
    #endregion

    #region Swimmable Members
    public void swim()    {Console.WriteLine("I'm just swiming!");}
    #endregion

    #region Athletable Members
    public void breath()    {Console.WriteLine("I'm breathing!");}
    #endregion

    #region Rideable Members
    public void ride()         {Console.WriteLine("I'm riding!");}
    #endregion
} // class Person
```

קודם תצויין הירושה
ולאחר מכן הממשקים