OOP-C#

Object-Oriented Programming

Lesson 2

©איילת ויץ 2023

הרשאות גישה

הרשאות גישה משמשות להגדרת רמת הגישה עבור מחלקות/שדות/שיטות ומאפיינים.
סוגי הרשאות גישה ב #C:

public – מחלקה, שדה, מתודה שניתן לגשת אליהם מכל מקום

private – שדה או מתודה שניתן לגשת אליהם רק מתוך מתודות של אותה מחלקה בלבד או של מחלקה חברה (friend).

protected - שדה או מתודה שניתן לגשת אליהם מתוך מתודות של אותה מחלקה בלבד ושל המחלקות שיורשות ממנה.

Internal - מחלקה, שדה, מתודה שניתן לגשת אליהם מכל מתודה או מחלקה שנמצאת איתם באותו אסמבלי (פרוייקט).
ובלי קשר ל namespace

©איילת ויץ 2023

ב C# גם למחלקה עצמה יש הרשאת גישה.

הרשאת גישה נכתבת לכל שדה/ מתודה /מחלקה בנפרד, משמאל לשורת ההגדרה ואם לא צויינה אזי נלקח ערך ברירת המחדל של ההרשאה.

ברירת המחדל של הרשאות ב C#:

עבור חברי מחלקה/מבנה (שדות ומתודות) – private

```
internal class MyClass
{
        String name;
}
```

```
static void Main(string[] args)
{
    MyClass m=new MyClass();
    m.name = "Sara";
}
```

אין גישה לתכונה מכיוון שהרשאת הגישה אליה לא הוגדרה, וברירת המחדל היא private

עבור המחלקה/מבנה עצמה – internal

```
internal class MyClass
{
}
```


©איילת ויץ 2023

מאפיינים

כאשר הרשאת הגישה לתכונות מוגדרת כ private, לא ניתן לגשת לתכונות מחוץ למחלקה.
לכן כאשר זו היא הגדרת הגישה לתכונות נשתמש בפעולות Get ו Set עבור גישה לתכונות.

ביצד נגדיר אותן? הדרך המוכרת:

```
private String name;

        public String GetName()
{
            return name;
}

        public void SetName(String name)
{
            this.name = name;
}
```

הגישה לתכונות תראה כך:

```
m.SetName("Rachel");
Console.WriteLine(m.GetName());
```


©איילת ויץ 2023

הדרך הנכונה, ע"י **מאפיינים**:

```
private String name;

public String Name
{
            get { return name; }
            set { name = value; }
}
```

ואפשר גם כך:

```
 public String Name
{
            get => name;
            set => name = value;
}
```

הגישה לתכונות תעשה ע"י המאפיינים, ותראה כך:

```
 MyClass m=new MyClass();
 m.Name = "Sara";
```


©איילת ויץ 2023

שדה סטטי ומתודה סטטית

שדה סטטי שייך למחלקה ולא לאובייקט מסוים. כלומר, השדה הסטטי שייך לכל האובייקטים של המחלקה.
שדה סטטי יכול לקבל ערך כבר בהגדרת השדה:

```
public static int count = 5;
```

או מתוך הבנאי:

```
        public MyClass(String name)
{
            this.name = name;
            count++;
}
```

כמובן שגם מאפיין השדה יהיה סטטי:

```
public static int Count
{
            get => count;
}
```

הגישה לשדה/מתודה סטטית יעשה דרך שם המחלקה . שם השדה/פונקציה.
לדו':

```
Console.WriteLine(MyClass.count) ;
```


©איילת ויץ 2023

פונקציה סטטית:

```
 public static int func()
{
            return count * 2;
}
```

הפעלת פונקציה סטטית:

```
 MyClass.Func();
```

שימי לב: מתודות סטטיות יכולות להשתמש רק בשדות סטטיים ולא מכירות את **this**.

©איילת ויץ 2023

מחלקה סטטית

מחלקה סטטית היא מחלקה שבשורת ההגדרה שלה רשומה הרשאת הגישה **public** וכל השדות והמתודות המוכלים בה הינם סטטיים.
לא ניתן לרשת ממחלקה סטטית
אין לה בנאי רגיל, אך ניתן לבנות בה בנאי סטטי.

יתרון למחלקה סטטית:
ניתן להשתמש בפונקציות של מחלקה סטטית גם בלי ליצור מופע למחלקה.
לדו':

```
public static class Calculator
{
public static int Plus(int x, int y)
{
            return x + y;
}
}

Calculator.Plus(5, 2);
```


©איילת ויץ 2023

```
public static class Calculator
{
        public static int num;
public static int Plus(int x, int y)
{
            return x + y;
}
        public static int Subtraction(int x, int y)
{
            return x - y;
}
        public static int Multiplication(int x, int y)
{
            return x * y;
}
        public static int Division(int x, int y)
{
            return x / y;
}
}
```


©איילת ויץ 2023

שדה קבוע const

הגדרה של שדה עם ערך קבוע.

שדה קבוע שייך למחלקה ולא לאוביקט (לעצם)

גישה לשדה ע"י שם המחלקה . שם השדה

חייב לקבל ערך כבר בשורת ההגדרה

שימי לב: שדה עם ערך קבוע-לא ניתן לשינוי!

```
public const int num= 5;
```


©איילת ויץ 2023

שדה לקריאה בלבד readonly

שדה המשמש לקריאה בלבד ומרגע הגדרת הערך עבורו לא ניתן לשנות את ערך השדה.

היתרון שלו על פני const שהוא לא סטטי למחלקה אלא ניתן להגדיר ערך שונה לכל מופע ע"י הגדרת הערך בבנאי.

```
public readonly int numRead;


public MyClass(int numRead)
{
        this.numRead = numRead;
}
```


©איילת ויץ 2023

הורשה Inheritance

ב- C# אפשר לרשת שדות ושיטות ממחלקה אחת לאחרת. נחלק את מושג "הורשה" לשתי קטגוריות:

Derived Class מחלקה נגזרת (ילד)

Base Class מחלקת הבסיס (הורה)

כדי לרשת ממחלקה נשתמש ב ":"
לדו':

```
class Animal
{
        private String name;
        private int bornYear;
}

class Sheep:Animal
{

        private int woolWeight;
}
```


©איילת ויץ 2023

פולימרפיזם Polymorphism

פולימורפיזם פירושו "צורות רבות", והוא מתרחש כאשר יש לנו מעמדות רבים הקשורים זה לזה על ידי הורשה.

ירושה מאפשרת לנו לרשת שדות ושיטות ממחלקה אחרת. פולימורפיזם משתמש בשיטות אלו לביצוע משימות שונות. זה מאפשר לנו לבצע פעולה בודדת בדרכים שונות.
לדוגמא, נניח שקיימת מחלקת Animal, והרבה מחלקות יורשות ממנה-Dog, Fish ועוד.

למחלקת Animal יש פונקציה להשמעת קול, ()AnimalSound. אך גם למחלקות היורשות ממנה מוגדרת הפונקציה הזו להשמעת קול בהתאם למחלקה היורשת (ע"פ סוג החיה)

©איילת ויץ 2023

```
 class Animal
{
public void AnimalSound()
{
            Console.WriteLine("I am animal");
}
}
class Sheep:Animal
{
        public void AnimalSound()
{
            Console.WriteLine("I am sheep");
}
}
class Fish:Animal
{

        public void AnimalSound()
{
            Console.WriteLine("I am Fish");
}
}
```


©איילת ויץ 2023

הפתרון:

```
 class Animal
{
Public virtual void AnimalSound()
{
            Console.WriteLine("I am animal");
}
}
class Sheep:Animal
{
        public override void AnimalSound()
{
            Console.WriteLine("I am sheep");
}
}
class Fish:Animal
{

        public override void AnimalSound()
{
            Console.WriteLine("I am Fish");
}
}
```


©איילת ויץ 2023

-END-

©איילת ויץ 2023