Access Modifiers

A C# nyelv hozzáférési módosítói (access modifiers) határozzák meg az osztályok, mezők, metódusok, tulajdonságok, vagy más típusok láthatóságát és elérhetőségét a program különböző részeiről.

Hozzáférési módosítók leírása

1. **public**
   Az elem bárhonnan elérhető, nincs korlátozás az elérhetőségére.
2. public class MyClass
3. {
4. public int MyProperty { get; set; }
5. }
6. **private**
   Az elem csak az osztályon belül érhető el, amelyben definiálták.
7. public class MyClass
8. {
9. private int _secretNumber = 42;
10. }
11. **protected**
    Az elem csak az osztályon belül és a leszármazott osztályokban érhető el.
12. public class BaseClass
13. {
14. protected string Name = "BaseName";
15. }
16. public class DerivedClass : BaseClass
17. {
18. public void PrintName()
19. {
20. Console.WriteLine(Name); // Elérhető a protected mező

```
    }
}
```

3. **internal**
   Az elem csak ugyanabban az assembly-ben érhető el.

```
internal class MyClass
{
    internal void Display()
    {
        Console.WriteLine("Internal method");
    }
}
```

4. **protected internal**
   Az elem ugyanabban az assembly-ben bárhonnan elérhető, illetve más assembly-kből csak a leszármazott osztályokon keresztül.

```
public class MyClass
{
    protected internal void Show()
    {
        Console.WriteLine("Protected internal method");
    }
}
```

5. **private protected**
   Az elem csak az aktuális assembly-ben lévő osztályokon belül érhető el, és csak a leszármazott osztályokon keresztül.

```
public class BaseClass
{
    private protected int Value = 10;
}
public class DerivedClass : BaseClass
{
```

```
46.    public void ShowValue()
47.    {
48.        Console.WriteLine(Value); // Elérhető, mert leszármazott
49.    }
50. }
```

Feladatok

Feladat 1: Public és Private használata

Készíts egy osztályt, amely rendelkezik egy **public** tulajdonsággal és egy **private** mezővel. A tulajdonságon keresztül érhető el az érték.

Elvárások:

- Az osztály neve legyen Person.
- A privát mező legyen _age, amely az életkort tárolja.
- Egy public tulajdonság (Age) vezérelje az _age értékét.

Feladat 2: Protected és Leszármazás

Készíts egy bázisosztályt és egy származtatott osztályt. A bázisosztály tartalmazzon egy **protected** mezőt, amelyet a származtatott osztály módosít.

Elvárások:

- A bázisosztály neve legyen Animal.
- A származtatott osztály neve legyen Dog.
- Az Animal osztály tartalmazza a protected string Name mezőt.
- A Dog osztály állítsa be a Name értékét és írassa ki a konzolra.

Feladat 3: Internal és Assembly szintű hozzáférés

Hozz létre egy osztályt, amely egy internal metódust tartalmaz. Próbáld elérni egy másik osztályból ugyanabban az assembly-ben.

Elvárások:

- Az osztály neve legyen InternalExample.
- Az internal metódus neve legyen PrintMessage.

Feladat 4: Protected Internal használata

Hozz létre egy osztályt, amely tartalmaz egy **protected internal** metódust. Készíts egy másik osztályt, amely ugyanabban az assembly-ben található, és hívja meg ezt a metódust.

Elvárások:

- Az első osztály neve legyen ParentClass.
- A metódus neve legyen ShowDetails.
- Egy másik osztályból hívd meg a metódust.

Feladat 5: Private Protected viselkedése

Mutasd be a **private protected** módosító használatát egy osztályhierarchiában.

Elvárások:

- A bázisosztály neve legyen BaseCalculator.
- A private protected mező neve legyen result.
- Egy származtatott osztály, AdvancedCalculator, módosítsa és írassa ki a result mezőt.

Feladat 1: Public és Private használata

```
using System;


public class Person
{
  private int _age;


  public int Age
  {
    get { return _age; }
    set
    {
```

```
            if (value >= 0)
            {
                _age = value;
            }
            else
            {
                Console.WriteLine("Age cannot be negative!");
            }
        }
    }
}

class Program
{
    static void Main()
    {
        Person person = new Person();
        person.Age = 25;
        Console.WriteLine($"Age: {person.Age}");
        person.Age = -5; // Hibaüzenetet ad
    }
}
```

Feladat 2: Protected és Leszármazás

```
using System;

public class Animal
{
```

```
    protected string Name;
}

public class Dog : Animal
{
    public void SetName(string name)
    {
        Name = name;
    }

    public void PrintName()
    {
        Console.WriteLine($"Dog's name: {Name}");
    }
}

class Program
{
    static void Main()
    {
        Dog dog = new Dog();
        dog.SetName("Buddy");
        dog.PrintName();
    }
}
```

Feladat 3: Internal és Assembly szintű hozzáférés

```
using System;

internal class InternalExample
{
    internal void PrintMessage()
    {
        Console.WriteLine("Internal method called!");
    }
}

class Program
{
    static void Main()
    {
        InternalExample example = new InternalExample();
        example.PrintMessage(); // Hozzáférhető ugyanabban az assembly-ben
    }
}
```

Feladat 4: Protected Internal használata

```
using System;

public class ParentClass
{
    protected internal void ShowDetails()
    {
        Console.WriteLine("Protected internal method called!");
```

```
    }
}

public class AnotherClass
{
    public void CallShowDetails()
    {
        ParentClass parent = new ParentClass();
        parent.ShowDetails(); // Hozzáférhető ugyanabban az assembly-ben
    }
}

class Program
{
    static void Main()
    {
        AnotherClass another = new AnotherClass();
        another.CallShowDetails();
    }
}
```

Feladat 5: Private Protected viselkedése

```
using System;

public class BaseCalculator
{
    private protected int result;
```

```
    public BaseCalculator()
    {
        result = 100; // Alapérték
    }
}

public class AdvancedCalculator : BaseCalculator
{
    public void Add(int value)
    {
        result += value;
        Console.WriteLine($"Result after addition: {result}");
    }
}

class Program
{
    static void Main()
    {
        AdvancedCalculator calculator = new AdvancedCalculator();
        calculator.Add(50);
    }
}
```

Kihívás feladat: Két assembly kezelése

Ezt a feladatot két külön projektben (pl. egy Class Library és egy Console App) kell megvalósítani. Íme egy példa:

Library project (AccessModifiersLibrary)

```
using System;

namespace AccessModifiersLibrary
{
  public class LibraryClass
  {
    internal void InternalMethod()
    {
      Console.WriteLine("Internal method in LibraryClass.");
    }

    protected internal void ProtectedInternalMethod()
    {
      Console.WriteLine("Protected internal method in LibraryClass.");
    }

    private protected void PrivateProtectedMethod()
    {
      Console.WriteLine("Private protected method in LibraryClass.");
    }
  }
}
```

Console App project

```
using AccessModifiersLibrary;
using System;

class Program
{
```

```
    static void Main()
    {
        LibraryClass libraryClass = new LibraryClass();

        // libraryClass.InternalMethod(); // Nem érhető el másik assembly-ből
        // libraryClass.PrivateProtectedMethod(); // Nem érhető el másik assembly-ből
        // Csak a ProtectedInternalMethod érhető el, ha ugyanabban az assembly-ben származtatunk

        Console.WriteLine("Kérjük, teszteld leszármazott osztállyal az egyes módosítók viselkedését!");
    }
}
```
