Access Modifiers

In this lesson, we will learn about the access modifiers and how to use them in our C# code.

WE'LL COVER THE FOLLOWING

In C#, we can impose access restrictions on different data members and member functions. The restrictions are specified through **access modifiers**. Access modifiers are tags we can associate with each member of the class to define which of its parts can be accessed by the program/application directly.

Types of Access Modifiers

There are the following six types of access modifiers in C#:

1. `private`
2. `public`
3. `protected`
4. `internal`
5. `protected internal`
6. `private protected`

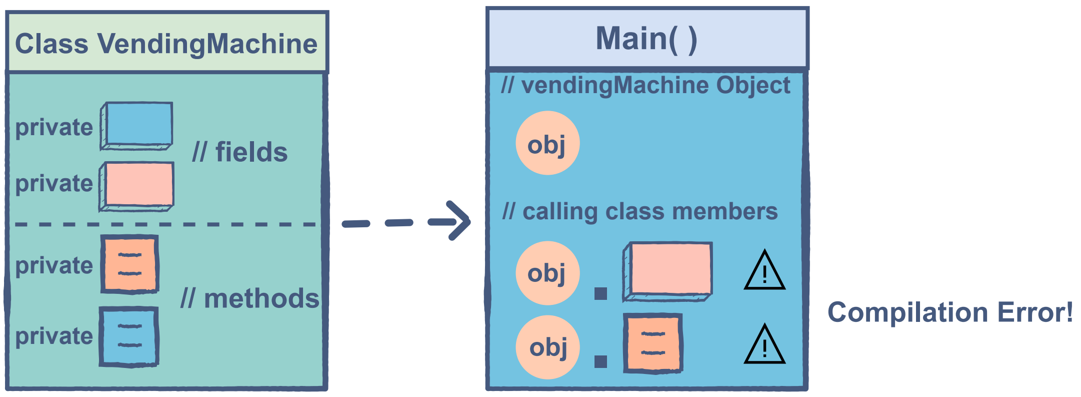
We'll only discuss `private` and `public` for now. The rest of the access modifiers will be discussed in the upcoming chapter on inheritance.

Private

A `private` modifier is used with a class member to specify that its access is limited to the code inside of the containing class only. In other words, private members can only be accessed by the methods of that class. A private member cannot be accessed directly from outside the class.

> If we don't specify any access modifier with a class member, it is recognized as `private` by default.

The aim is to keep it hidden from users and other classes. It is a popular practice to **keep data members private** since we do not want anyone manipulating our data directly. We can make members private by using the keyword `private`.



Private Access Modifier

> ⚠ Trying to access the `private` class members directly from another class will cause a compilation error.

Example

Let's look at an example by using the `private` access modifier in our code.

The following example **won't compile** because we are trying to access a private member of a class directly from another class.

```
class VendingMachine
```

```
{
    //Class fields
    private int _count = 30; //Private field

}

class Demo //Class containing the Main() method
{
    public static void Main(string[] args)
    {
        var vendingMachine = new VendingMachine(); //Object creation
        //Calling the private member of the VendingMachine class
        Console.WriteLine("The count of the products in the machine is: {0}", vendingMachine.

    }
}
```

> `VendingMachine._count' is inaccessible due to its protection level`, the error message is pretty much self-explanatory and tells us that we cannot access a private class member directly from another class.

Naming Convention

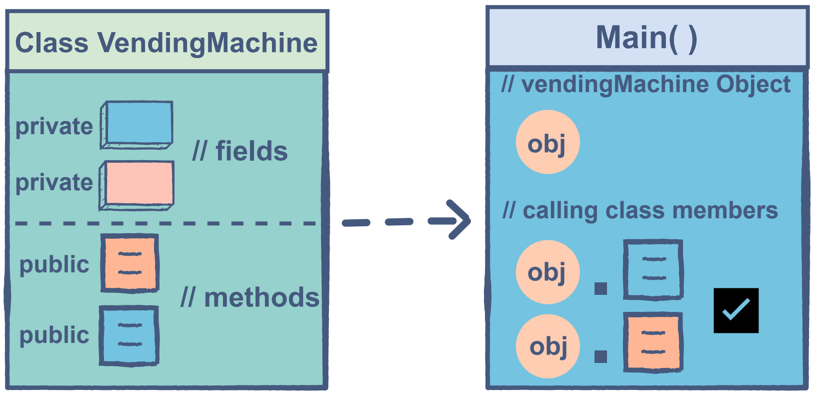
In C#, the name of a private class member should always start from an underscore `_` followed by `camelCase`.

Public

Members with this tag can be directly accessed by anything which is in the same scope as the class object. In other words, `public` members can be accessed from anywhere in the current assembly or any other assembly that references it.

Member methods are usually public as they provide the interface through which an application can communicate with our private members.

Public members can be declared using the keyword `public`.



Public Access Modifier

Example

Let’s look at an example by using the `private` access modifier in our code.

```
class VendingMachine
{
    //Class fields
    private int _count = 30; //Private field

    public int GetCount()   //Public method
    {
        return _count;
    }
}

class Demo //Class containing the Main() method
{
    public static void Main(string[] args)
    {
        var vendingMachine = new VendingMachine(); //Object creation
        //Calling the public member of the VendingMachine class
        Console.WriteLine("The count of the products in the machine is: {0}", vendingMachine.

    }
}
```

Public members of a class can be accessed by a class object using the `.` operator.

In general, it’s not a good idea to declare `public` fields in a class as they are mutable and can cause bugs in our application/program.

> they are mutable and can cause bugs in our application/program whereas member methods are often declared `public`.

Naming Convention

In C#, the name of a `public` field should follow `PascalCase`.

In the next lesson, we will discuss everything about the data members or fields.