*Encapsulation – Week 3*

Written by: Sergio Luilly Cabrera Dorado

1. Definition

A simple way to understand it is by understanding part of its name "capsule" since we do not see the ingredients inside the drug, but we know that what it contains will be for our benefit.

Encapsulation is one of the fundamental principles of OOP (object-oriented programming), its main function is to group data (fields) and methods (behaviors) that operate in classes, while restricting direct access to the internal data of the object, so that it can be interacted with from a controlled public interface.

To control access, **access modifiers** are used:

* private: The member (field or method) is only accessible from within the class itself.
* public: The member is accessible from any other part of the program.
* protected: The member is accessible from the class that defines it and any class that inherits from it.

1. Proceeds

A crucial benefit of encapsulation is that it **protects the integrity of data** and makes code **easier to maintain**:

* Data protection: applied in the field of finance, it protects personal data, especially the balance in the bank account, so that you can only withdraw what you have, in addition to your login details.
* Ease of maintenance: when modifications are made to the internal implementation of a class (such as the storage of new or modified data), while the public interface (properties and methods) will remain the same and keep the code without modifications that affect its operation. This allows you to modify and improve your code with less risk of introducing errors elsewhere.

1. Application and Example

Citing again the example of finance, the principle of encapsulation applies:

*public class UserAccount*

*{*

*private string \_secretKey = "ABC123";*

*private double \_balance = 1000;*

*public bool ValidateWithdrawal(double amount)*

*{*

*if (amount <= \_balance)*

*{*

*\_balance -= amount;*

*return true;*

*}*

*return false;*

*}*

*public bool CheckSecretKey(string inputKey)*

*{*

*return inputKey == \_secretKey;*

*}*

*public double GetBalance()*

*{*

*return \_balance;*

*}*

*}*

The data that is kept private is *\_secretKey* and *\_balance*. Which are the ones that have sensitive data, are kept hidden from the view of users, avoid using the command "you have been hacked".

In addition to other functionalities, such as the validation of the *ValidateWithdrawal balance*, key verification without revealing it, *CheckSecretKey* and *GetBalance* that shows the available balance without the possibility of modification by the user.

UserAccount user = new UserAccount();

if (user. CheckSecretKey("ABC123")) // ✅ Correct Key

{

if (user. ValidateWithdrawal(500)) // ✅ Sufficient Balance

{

Console.WriteLine("Successful withdrawal");

}

else

{

Console.WriteLine("Insufficient Balance"); // ❌ Validation Fails

}

}

Sensitive data (\_secretKey, \_balance) is PROTECTED within the class, and is only accessed by methods that validate and control access.