SIT232: Object Oriented Development

Encapsulation: hiding the data

What is Encapsulation?

Encapsulation is one of the fundamental principles of OOP which focuses on bundling data and the methods that operates on that data into a single unit called class. It is the concept of enclosing data and related behaviors within a class and controlling access to that data through methods and properties. Encapsulation hides the internal implementation details of an object and provides a clean and well-defined interface for interacting with the object. This allows the object to control its own state and ensures data integrity and providing abstraction to the program outside.

Advantages: -

- **Data hiding:** internal state of an object, including its data and implementation details, is hidden from external code. For example, in the following code the fields are made private

```
private string _make;
private string _model;
private int _year;
private decimal _mileage;
```

  This prevents the direct access to the object's data and allows it to enforce proper usage through defined methods or properties.

- **Data protection:** by encapsulating the data within a class, access to that data can be controlled. by making the fields private and providing getter and setter methods, the code ensures that direct access to the private data is restricted, and modifications to the data can be performed through controlled methods, maintaining data protection and encapsulation principles. Following example code will help,

```
public string GetMake()
{
      return _make;
}
public string GetModel()
{
      return _model;
}
public int GetYear()
{
      return _year;
}
public decimal GetMileage()
{
      return _mileage;
}
public void SetMileage(decimal mileage)
{
      if (mileage >= 0)
        {
            _mileage = mileage;
        }
```

```
            else
            {
                Console.WriteLine("Mileage cannot be negative.");
            }
    }
```

- **Code organization:** Encapsulation helps in organizing code by grouping related data and behavior together within a class. This improves code maintainability, modularity, and reusability.

Objectives:

- **Data Hiding**
- **Abstraction:** by providing simplified interface to the outside world, by making fields private and providing controlled access through methods and properties.
- **Information hiding:** implementation details of the class is hidden by keeping the internal data of the class private, this maintains the stability and reliability.
- **Modularity:** Encapsulation promotes modularity by grouping related data and behaviors within a class.
- **Code Reusability:** Encapsulation supports code reusability by providing well-defined and self-contained classes. Encapsulated classes can be easily reused in different parts of an application or in other projects.

Summarize what you learned:

By achieving these objectives, encapsulation helps to create more robust, maintainable, and flexible software systems. It promotes the principles of data hiding, abstraction, information hiding, modularity, and code reusability, leading to improved code quality and development efficiency.

You can refer to the code I made that clears the concepts of encapsulation. It is present on GitHub

mailto:https://github.com/17012004/Encapsulation/blob/main/Program.cs

Please add your feedback and mail to me on:

mailto:vansh4856.be22@chitkara.edu.in

Please add the CC/BCC mail as

mailto:harsimran.kaur@chitkara.edu.in