OOPs Concept in C#

Agenda

- What are the Problems of Functional Programming?
- How we can overcome the Functional Programming Problem?
- What Is Object-Oriented Programming in C#?
- What are the OOPs Principles?
- Why do we need Object Oriented Programming in C#?
- Why do we need real-world objects in a Project?
- What types of programming languages come under the OOP system?

What are the Problems of Functional Programming?

Reusability:
In Functional Programming, we need to write the same code or logic at multiple places which increases the code duplication. Later if we want to change the logic, then we need to change it at all places.

Extensibility:
It is not possible in functional programming to extend the features of a function. Suppose you have a function and you want to extend it with some additional features then it is not possible. You have to create a completely new function and then change the function as per your requirement.

Simplicity:
As extensibility and reusability are not possible in functional programming, usually we end up with lots of functions and lots of scattered code.

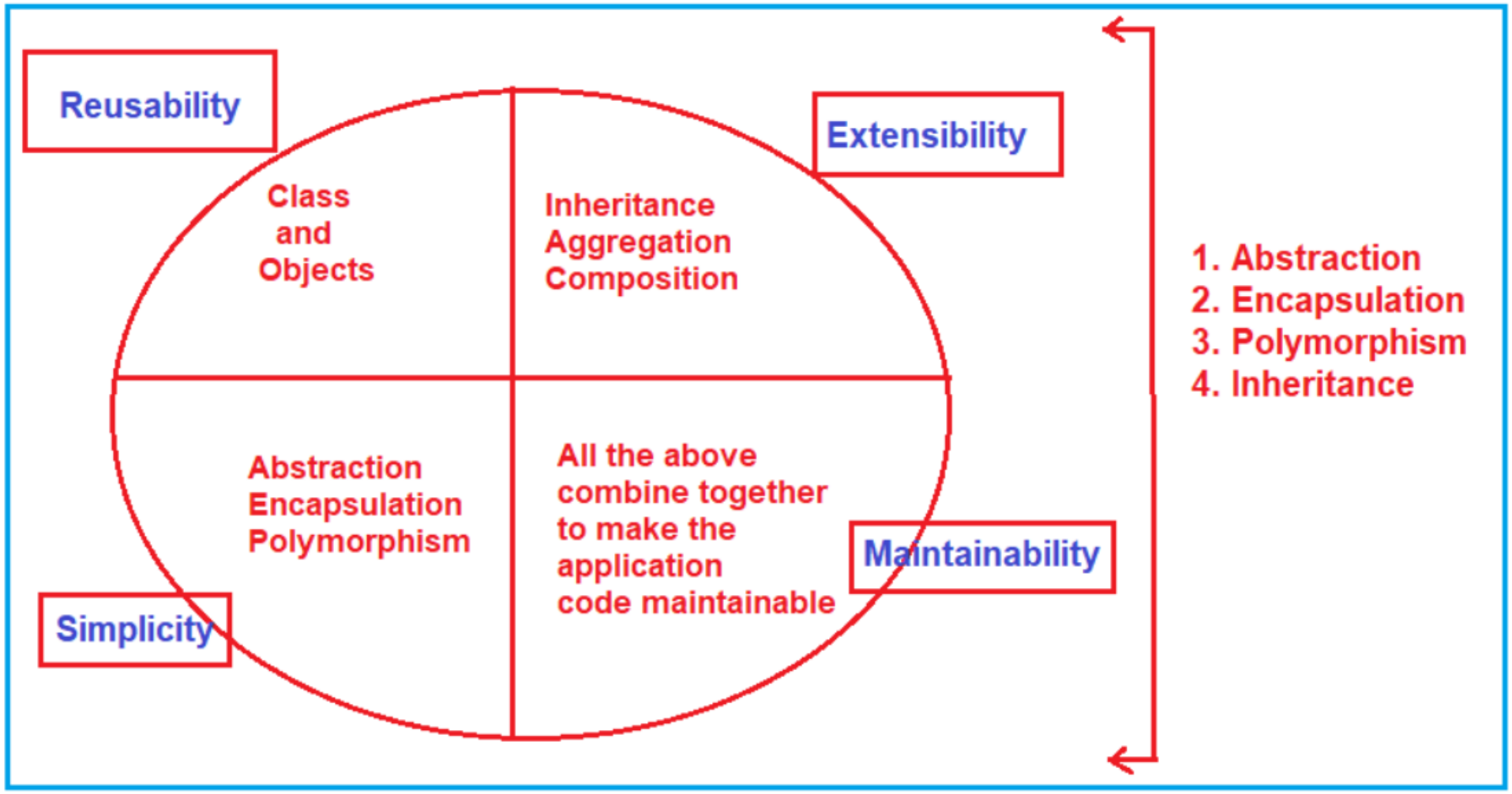
Maintainability:
As we don't have Reusability, Extensibility, and Simplicity in functional Programming, so it is very difficult to manage and maintain the application code.

How we can overcome Functional Programming Problems?

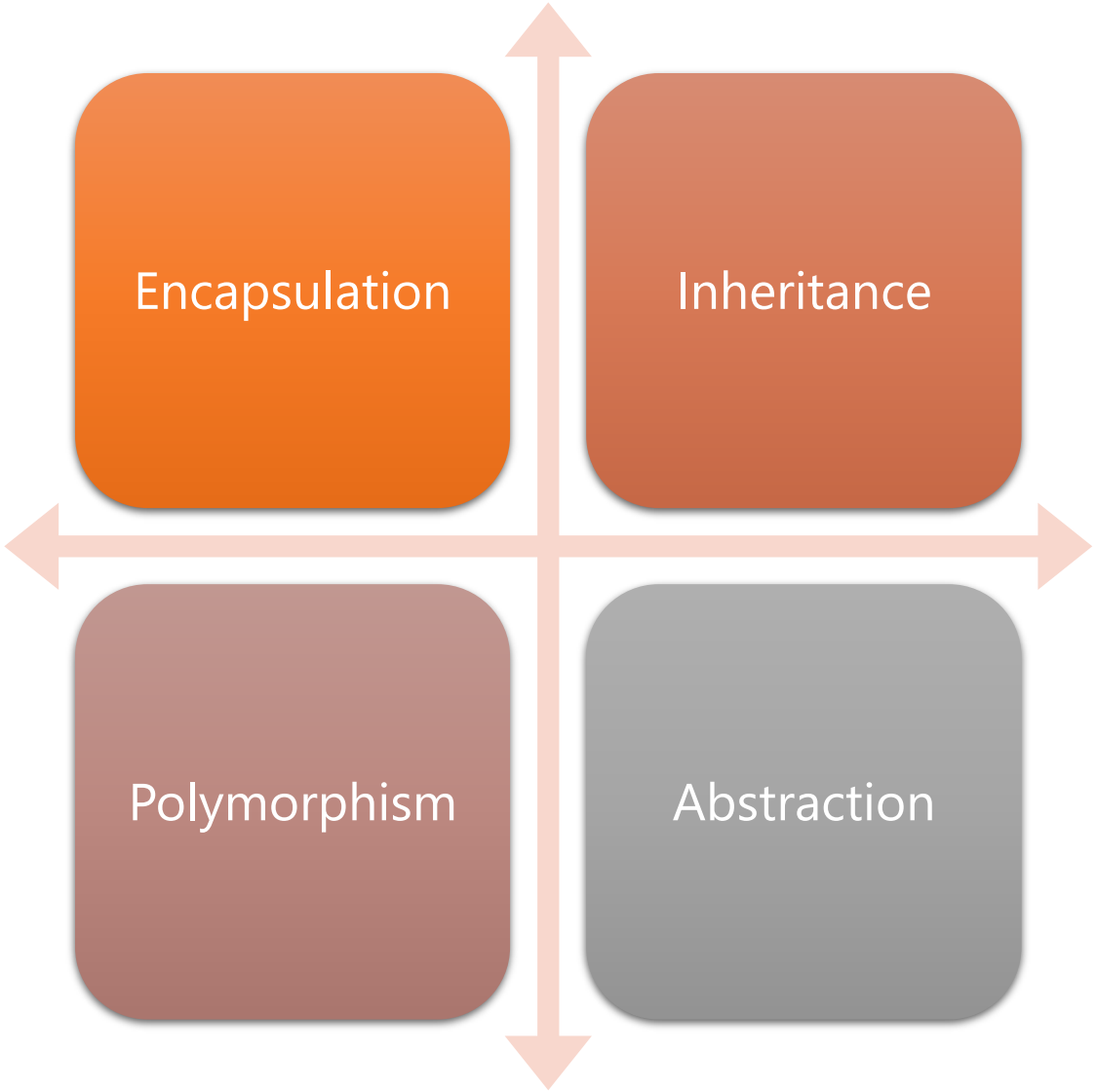
We can overcome the functional programming problems (Reusability, Extensibility, Simplicity, and Maintainability) using Object-Oriented Programming. OOPs provide some principles and using those principles we can overcome the functional programming problems.

What Is Object-Oriented Programming?

Object-Oriented Programming (OOPs) in C# is a design approach where we think in terms of real-world objects rather than functions or methods. Unlike procedural programming language, here in oops, programs are organized around objects and data rather than action and logic.



What are the OOPs Principles or OOPs Concepts in C#?

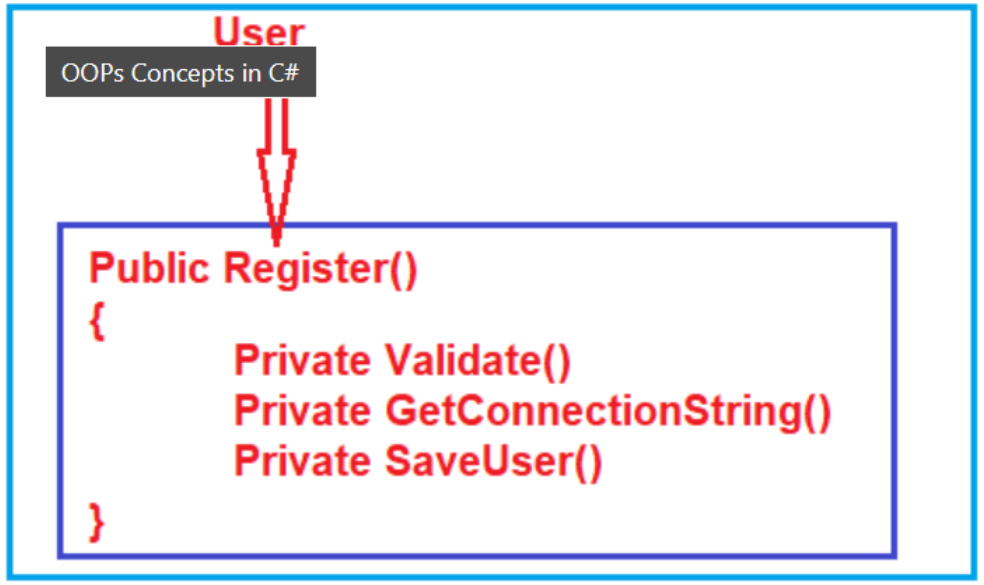


What are Abstraction and Encapsulation?

The process of representing the essential features without including the background details is called **Abstraction**. In simple words, we can say that it is a process of defining a class by providing necessary details to call the object operations (i.e. methods) by hiding or removing its implementation details.

The process of binding the data and functions together into a single unit (i.e. class) is called **Encapsulation**. In simple words, we can say that it is a process of creating a class by hiding its internal data members from outside the class and accessing those internal data members only through publicly exposed methods or properties.

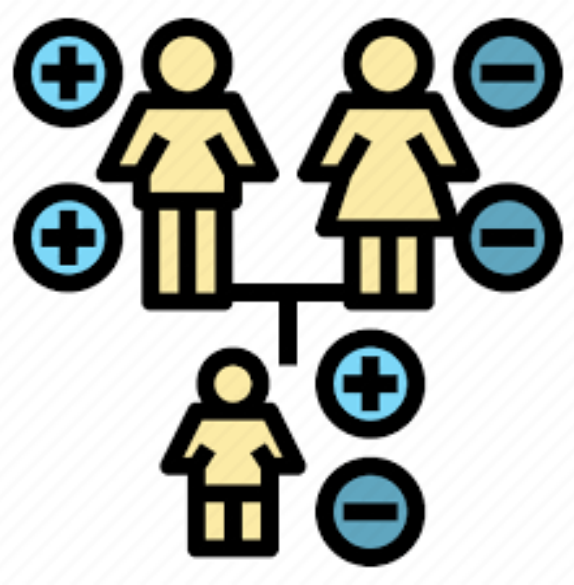
Abstraction and Encapsulation are related to each other. We can say that Abstraction is logical thinking whereas Encapsulation is its physical implementation.

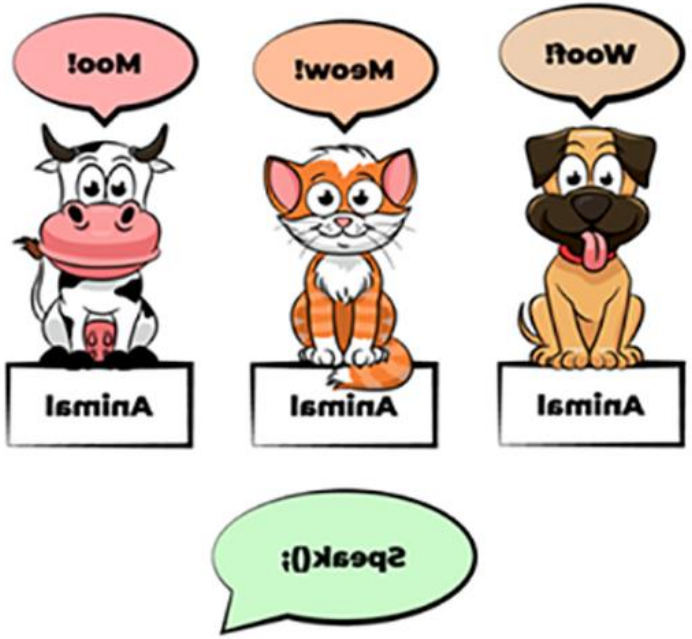


Abstraction	Encapsulation
Abstraction solves the problem in the design level.	Encapsulation solves the problem in the implementation level.
Abstraction is used for hiding the unwanted data and giving only relevant data.	Encapsulation is hiding the code and data into a single unit to protect the data from outer world.
Abstraction is set focus on the object instead of how it does it.	Encapsulation means hiding the internal details or mechanics of how an object does something.
Abstraction is outer layout in terms of design. For Example: - Outer Look of a iPhone, like it has a display screen.	Encapsulation is inner layout in terms of implementation. For Example: - Inner Implementation detail of a iPhone, how Display Screen are connect with each other using circuits

What is Inheritance?

The process by which the members of one class are transferred to another class is called **inheritance**. The class from which the members are transferred is called the Parent/base class and the class which inherits the members of the Parent class is called the Derived/ child class. We can achieve code **extensibility** through inheritance.





What is Polymorphism?

The word **Polymorphism** is derived from the Greek word, where Poly means many and morph means faces/ behaviors. So, the word polymorphism means the ability to take more than one form. Technically, we can say that when the same function/operator will show different behaviors by taking different types of values or with a different number of values called **Polymorphism**. There are two types of polymorphism

1.Static polymorphism/compile-time polymorphism/Early binding
2.Dynamic polymorphism/Run time polymorphism/Late binding

Static polymorphism is achieved by using **function overloading and operator overloading** whereas dynamic polymorphism is achieved by using **function overriding**.

Why do we need real-world objects in a Project?

We need real-world objects in a project because real-world objects are part of our business. As we are developing applications (software) for automating the business, we must have to create the business-related real-world objects in the project.

For example, to automate the Bank business we must create real-world objects like Customer, Manager, Clerk, Office Assistant, Marketing Executive, Computer, Printer, Chair, table, etc. So along with the Bank object, we must also have to create all the above objects because without all the above objects we cannot run a Bank business. Technically we call the above objects are business objects.

What types of programming languages come under the OOP system?

The programming languages which implement all the four principles provided by OOPs are called object-oriented programming languages. Examples: Java, .Net, C++, etc.