Object Oriented in C#

Methods

1. Pass by reference: means that you pass the variable itself into the function.
2. Pass by value: means that you pass the actual value of the variable into the function.

- Local variable: a variable that is only accessible within a specific part of a program.
- Instance variable: a variable which is declared in a class but outside of constructor, methods or blocks.

Encapsulation

Internal: the details of the variables and methods of the class that defines it.

External: the services that an object provides and how the object interacts with the rest of the system.

An encapsulated object can be thought of as a black box – its inner workings are hidden(**Abstracted**) from the client.

Visibility Modifiers

	Public	Private
Variable	Violate encapsulation	Enforce encapsulation
methods	Private services to client	Support other methods in the class

---

Accessors & Mutators

Accessor method: return the current value.

Mutator method: change the value of a variable.

Constructor

is a special method that is executed when a new instance of the class is created.

Property

replace the setter & getter methods.

Method Overloading

more than one method with the same name in the same class.

Methods can share the same name as long as:

1. The number of parameters should be different.
2. The type is different.

This

pointer reference to an object.

We use this to:

1. Calling one constructor from another.
2. If variable names are the same, **this** is used for different instances from local variables.

Members: are attributes and behaviors(variables & methods).

Static

Static variable: class variable, only one copy shared between objects.

non static variable: instance variable multiple copies.

Static methods: call it from class only.

When to use static variables:

1. Only on copy from the variable accessible by all objects.
2. You can use static variables and methods without creating an object.

Dependency

a class uses another class attributes and methods(client,supplier).

We use the supplier by two ways:

1. As a parameter
2. As a local variable

Aggregation & Composition

Aggregation: if the whole die the parts may remain, and they instance reference variables.
We create the object outside the class but use it inside the class(we use it as input parameter).

Composition: if the whole die the parts will die too,is a stronger form of a aggregation
We create the object inside the class and use it inside the same class.

Notes:

- **Abstraction:** hide details.
- **Encapsulation:** protect data.
- Aggregation & composition at **run time**
- Protected: a visibility modifier used to inherit variables to a child class.
- Child class does not have access to private members.
- Child class can access public members, but it will violate the encapsulation.
- # : in uml means protected

Inheritance

1. Parent class: general
2. Child/Derived class: sebisifice

- We use inheritance to reuse code
- Inheritance: at **compile time**

visibility / keyword	Containing Classes	Derived Classes	Containing Assembly	Anywhere outside the containing assembly
public	yes	yes	yes	yes
protected internal	yes	yes	yes	no
protected	yes	yes	no	no
private	yes	no	no	no
internal	yes	no	yes	no

- Method inside public class cant access private members from the super class.
- Constructor is not inherited.
- Base reference: we use it to call parent constructor by child's constructor.
- If a child inherits from 1 parent called **single inheritance.**

Method Overriding

child overrides the definition of an inherited method.

- New method must have the same signature of the parent's method, but a different body.
- Virtual: is an access modifier used to make the methods overridden in the parent class.
- Override modifier: override method in child class.
- base.methodName: call the base method not the override one.
- Parent constructor can't be overridedd

Overloading	Overriding
Deals with multiple methods with the same name in the same class, but with different signature	Deals with two methods, one in a parent class and one in a child class, that gave the same signature

- Class hierarchies: parent can be a child and so on
- Siblings: Two children with the same parent
- Each class by default it will inherit from class object
- Class object have some properties :
    - ToString Method : return class name, but u can override it to return the class properties and methods.
    - Equals: true or false answer, if the two pointers are pointing at the same object (aliases).

Abstraction

Abstract Class: class you can't create objects from it(cannot be instantiated).

Abstract Method: it ends with semicolon, method with no definition, static and private methods can't be abstract.

Abstract classes can have non abstract methods, but abstract methods should be in abstract class.

The child of an abstract class must override the abstract methods of the parent, or it too will be considered abstract.

Sealed methods: a method child can't override it.

Polymorphism

you can create an object from parent type the give him value of child type (Parent parent = new child)

Polymorphism is generally used to store many related items, but with slightly different types (subclasses of the same superclass) in one storage container (for example : an array) and then do common operations on them through overridden functions

---

Interface

Interface: contain properties and abstract methods, we don't use override.

- An abstract method can be declared using the modifier abstract, but because all methods in an interface are abstract, usually it is left off.
- Public visibility by default public.
- A class implements an interface (give a body to all abstract methods).
- The override is not needed when defining an interface method in a class.
- A class must implement multiple all methods in all interfaces listed in the header.
- Interfaces can implement a multiple inheritance.
