

## what's the (OOP)

in the past there were ways how to think with programming:

### 1- procedural      2- functional      3- OOP

- C language supports procedural programming
- The main unit for procedural programming is a **function** called another function.
- each function executes one action (Logic)

- more info (**programming paradigms**) can be broadly classified into two main categories:

1- Imperative Paradigm : (procedural - OOP - Structured)

2- Declarative Paradigm : ( Logic Programming - functional)

so we can define **oop** : is a programming paradigm based on the concept of "objects," which can fields and methods to create modular, reusable, and scalable code.

### OOP:

- We need to think in application like we think in reality. First, I need to identify the parties involved in the problem, then provide a description for each party.
- the main unit for OOP is a **class**
- describe : actions + attributes → **class\ class members**: Fields-methods-properties
- define any class using one of the three keywords: (**Class, struct , enum** ).

### // How to declare a class :

```
class (name of class)
{
    // (attributes)Variables - Functions(actions)
}
```

ex: class car      // example of declaration

```
{
    double price; //Fields
    string number;
    void move forward() //Method
{
}
```

### Object :

- Object is a **physical** representation
- The data is stored within the physical representation (Object)
- An object doesn't necessarily need to be tangible; it can also represent a concept or a process like course for instance

-when i speak in general is a **class** , when i specify is an **object**

- How to declare an object from the class car for instance:

```
car c = new car();
```

- Let's say I was required to develop an application for an **airline booking website** and an **aircraft maintenance website**. Naturally, the features present in each class would differ. I will talk about the seats and their categories, such as business class or economy, whether they are reclining, and other features. However, when discussing maintenance, I will focus on the motors, the type of wheels, the number of motors,
- The wings, and whether these aspects differ in business class. this is a concept called (**Abstraction**)

#### – How to apply abstraction in programming?

- Create a class and include the attributes and methods that belong to the data type for business, which define the basic functions without the need to show the detailed implementation of how these functions are executed.

يعني ممكن نقول ان ال **abstraction** انا بتحكم في ايه الحاجات اللي هتبقي موجودة معايا و انا بكتب الكود بناء على ال business اللي عندي وفيه حاجات عايز ال **user** يشوفها و حاجات لا

#### – What's the relation between a class and an object ?

- Object is a physical representation from class but class is a logical representation.

**Note:** table is object ✓ from class wood ✗ from class Table.

door is object ✓ from class wood ✗ from class Door.

فكرة انهم بيحملوا نفس الصفات بس القيم اختفت فده بقى باب و دى بقى ترابيزة الفكره دي غلط  
الصح مينفعش احنا الاتنين نطلع من نفس ال class و انت عندك صفات مختلفة عنى يعني مثلا الترابيزة ليها صفات غير الباب و كمان فكده ال class مختلف .

There are **four Main** concepts in OOP  
**(Abstraction - Encapsulation - Inheritance - Polymorphism)**

**Encapsulation :** is the process of wrapping data and methods into a single unit (class) to protect it from misuse.

Class Car // declare class

```
{
    double price;
    string number;
    void moveForward()
    {
    }

    void stop()
    {

    }
}
Car c1 = new Car(); // declare object
Car c2 = new Car();
c1.price = 1000;
c2.price = 2000; // It shows an error
```

– Why does that error appear?

- Price is a member variable inside the Car class. It can only be accessed through an instance of its own object of the class.

Additionally, its visibility is controlled using access modifiers like public or private

– The correct way to implement the code based on previous example

Class Car

```
{  
    public double price;  
    private string number;  
    void moveForward()  
    {  
  
    }  
    void stop()  
    {  
  
    }  
}  
Car c1 = new Car();  
Car c2 = new Car();  
c1.price = 1000;  
c2.price = 2000;  
Console.WriteLine(c1.price)  
// c1.number = "123"; I can't access it because it's private
```

**Note:** If I don't write an access modifier next to the member Whether it's a **variable or a function**, by default it will be **private**. ex: double price; void moveForward()

طبع لو انا عندي private variable و عايز access عليه اقدر اعمل كده ان احط ال public جوا و بكته اقدر access عليه عادي بطريقة غير مباشرة بدل من كل هب و دب ي accesses عليها function

EX:

```
public double price;  
private string number;  
public void moveForward()  
{  
    price = 10;  
    number = "20"; // private  
}  
c1.moveForward() // access private variable throw public function
```

## – What's the difference between abstraction and encapsulation ?

- In abstraction, when designing the class, you decide what things will be exposed. In encapsulation, you decide what things will be hidden. At the level of abstraction, you decide whether this thing is present in your design or not based on business requirements. But , At the level of encapsulation, the thing exists, but you decide whether people outside the class can interact with it or not.

If I have a business model that says the car price must be **greater than 5000**, if I make it public, anyone can set any value and it will be accepted without validation. – The solution is to make it private and provide access to it through a public function. This way, I am providing indirect access, and I can use a setter and getter.

```
public class Car
{
    private double Price;

    // Public method to set price with validation
    public void SetPrice(double price)
    {
        if (price >= 5000)
        {
            Price = price;
        }
        else
        {
            Price = 5000;
        }
    }

    // Optional getter to access the price (if needed)
    public double GetPrice()
    {
        return price;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Car c1 = new Car();
        c1.SetPrice(10000); // Setting price using the SetPrice method
    }
}
```

- you can implement only a setter, you can implement only a getter, or both a setter and a getter, Depending on the business req.

## (Data Representation In Memory)

### Operating system fundamentals Before diving deep:

- Any application consists of two parts: **code – data**.
- **code → processing → output**.
- Any application, in order to run, gets loaded from the **hard drive** into the **RAM**.
- Any application in memory is loaded into two places: one for the code and one for the data.

### (The data segment)

- The data segment is divided into two places in memory: (**stack –heap**).
- The presence of data in the stack or heap affects their duration in memory.
- **Stack:** A place in memory for each application, where the last thing entered is the first thing to exit, meaning it follows the **Last In, First Out** system.
- **What type of data will be placed in the stack?**
  - Any **local variable** defined inside a function will be placed in the stack.
  - Any **parameter** for a function will also be stored in the stack.

#### EX: Local variable

```
static void Main(string [] args ) // The Main function will execute first.
```

```
{
```

```
    int num = 5; // local variable it means will be placed in the stack
```

```
}
```

#### EX: assuming we have a function named test with a defined parameter.

```
static void test(int y) // int y is a parameter
```

```
{
```

```
    int x = 3; // local variable
```

```
}
```

```
static void Main(string [] args ) // The Main function will execute first.
```

```
{
```

```
    int num = 5; // local variable.
```

```
    test();}
```

للي هيحصل كالتالي هنف الأول الدالة Main بالتالي هيكون جوه ال varibale name, stack اسمه num و قيمته 5

بعدين جوه ال stack اللي اسمه y

بعدين هيende للدالة test و اللي فيها variable x قيمته 3 و بکده تكون دالة Main خلص

**Important note:** Every time a function is called, it loads from the beginning, and every time a new variable is loaded and worked on.

- **Heap:** A place in memory used for dynamic memory allocation, where data is stored and managed without following any specific order. Memory is allocated and freed as needed, and unlike the stack, it doesn't follow systems. It allows for more flexible memory usage.

### (Value Types and Reference Types)

- In any language, there are data types, whether built-in type or user-defined type , and both are categorized under two main types: **value types** and **reference types**.
- In C#, we can define any data type using three keywords(**class – struct – enum** ).
- Anything defined using a class will be a **Reference Type**.
- Anything defined using an enum or a struct will be a **Value Type**.
- 
- The **Value Type** holds a value, whereas the **Reference Type** does not hold a value.

- **car c;** is A reference from a class **car** is not the object itself. It's just a reference that holds the **address of the object**.So, a **Reference Type** is the one that holds the address of the object.

- **car c = new car(); // Here,I created a reference pointing to the object's address.**

**Another EX :**

**car c = new car(); // here I created an object and made a reference pointing to the object's address, and everything is fine.**

**c = new person(); // Here, I made (c) point to a completely different object (person). Thus, the object named (car) will no longer be accessible. However, it will remain in memory until the Garbage Collector (GC) reclaims its memory when no references are pointing to it."**

**Constraints & rules:**

- A local variable cannot be used, such as printing or passing it to a function, unless it has been initialized, even if it is a value type or a reference type.
- If my variable is a reference type and I don't want it to point to the object, I can set it **to null**. **car c = null; // It's empty**, passed **compile✓ runtime X**
- Any value type cannot be initialized with null; it must be given a value.
- When I create an object, the member variables are automatically initialized.

**Ex : initialization when creating an object:**

```
class car

{
    public int price; // This will be automatically initialized to 0
    public string number; // This will be automatically initialized to null
    public void move()
    {
        int speed =100;
    }
}

car c = new car ();

Console.WriteLine(c.price);  // value type will print 0 by default (int)
Console.WriteLine(c.number);  // string reference type will print by default Null
```

- It's possible to have **multiple references** pointing to the same object.

**Ex: multiple references pointing to the same object**

```
public int price

public string number;

public void move()

car c = new car();

c.price = 1999;

car c2 = c; // made both equal to each other.

c2.price = 3000, // changed the value of price in c but it's still the same object.
```

**– What's The benefit of having multiple references pointing to the same object ?**

- When transferring data from one function to another is that any changes made to the object in one function will reflect in all references, as they all point to the same memory location. this can help avoid duplicating data and reduce memory usage when passing large objects around in your program.

- **passing a parameter by value and passing a parameter by reference** are part of the concept of multiple references.

### **EX: passing parameter by value**

```
static void test(int y) // y is a parameter
{
    int x = 3; // local variable
}

static void Main(string[] args) // The Main function will execute first.

{
    int num = 5; // local variable
    test(num); // Here,we passed the value of num to y in the test function
}
```

### **EX: passing by reference**

```
class Car
{
    public int price;
}

static void test(Car car) // Receive a reference to a Car object
{
    int x = 3;
    car.price = 90000; // Modify the price of the object
}

static void Main(string[] args)
```

## (Relationships between Classes)

- What is the point of knowing this relationship? Why not just figure it out intuitively?
- academic concept
- It's crucial for writing clean, structured, and maintainable code.
- Memory and Performance
- Avoiding Bugs
- Scalability

There are four relationships between classes :

## (Association- Aggregation– Composition– Inheritance)

- The implementation of these relationships is similar, so what makes me say that this relationship will be like this is **logic**.
- Each relationship we talk about can be described in a sentence.
- The strength of the relationship between classes will gradually increase.
- The **main idea** is the level of connection that will occur between the two objects coming from the classes. This level of connection largely depends on the object creation process. Am I, as class **X**, responsible for creating an object of class **Y**? Or am I not responsible? The answer to this question determines the type of relationship between the classes.

### Association:

**Description :** When a class **uses** another class as a dependency.

- the relationship is weak or short-term
- If class **X** uses class **Y**, this means that we will need an object of class **Y** inside Class **X**

**Ex :** I have an **instructor** who wants to **write with a marker** on the board.

### 👉 Step-by-Step Implementation:

- The first thing I will create is a class called Instructor.
- Then, I will create a function inside it called Write OnBoard.
- I will create a class called Marker and define the marker's attributes in it.
- Inside the WriteOnBoard function, I will put a reference to an object of the Marker class
- Then, I will create an object from Marker and an object from Instructor.
- I will call the WriteOnBoard function and pass the object to it. (**This is where the relationship begins**)

```

class Instructor

{
    public void WriteOnBoard(Marker marker )
    {
    }

    Class Marker

}

// Attributes

}

private static void Main(string[] args)

{
    Marker m = new Marker();

    Instructor ins = new Instructor();

    ins. WriteOnBoard(m);

}

```

**Note:** If we remove the Instructor object from memory, will the Marker object be affected? No, it will not be affected. And if the Marker object is deleted, will it affect the Instructor? No, because it is a short-term relationship inside WriteOnBoard only. **This represents the concept of Association**

### Aggregation :

**Description :** A class **has** an object of another class.

- A slightly long-term relationship, For example, a room and a student. The student can eat and spend some time in the room.
- **What's the similarity between Association and Aggregation?**
  - In both relationships, they are not dependent on each other, neither the classes nor the objects
- **When should I choose between Association and Aggregation?**
  - It depends on the business model and the requirements.

**EX: I have an instructor who wants to enter a room and turn on the light.**

```
class Room

{
    Instructor instructor;

    public Room()

    {
        instructor = null;
    }

    public void InstructorEnteredRoom(Instructor ins )

    {
        instructor = ins;
    }

    public void InstructorTurnLightOn()

    {
        // instructor
    }
}

class Instructor

{
}

private static void Main(string[] args)

{
    Room room = new Room();

    Instructor ins = new Instructor();

    room.InstructorEnteredRoom(ins);}
```

**Note :** In association, the relationship exists only at the function level. However, if the relationship extends beyond the function level and is maintained at the class level, it becomes aggregation.

### **Composition :**

**Description :** A class **consists of** objects of another class.

– Continuous and strong relationship, For example, Human and head. A human cannot live without a head

- **Note : If the object is created inside the class, then it is Composition.**

### **Ex :**

### **Ex :**

```
class Human Body
```

```
{
```

```
    Head head;
```

```
    public HumanBody()
```

```
{
```

```
    head = new Head(); // object in side the class
```

```
}
```

```
    public void Think()
```

```
{
```

```
}
```

```
}
```

```
    internal class Head
```

```
{
```

```
}
```

```
HumanBody body = new HumanBody();
```

## Inheritance :

**Description : is a relationship** , Example human is a creature , car is a vehicle

– There is a set of attributes and actions, such as in humans and animals, for example. They eat, drink, and sleep, and they have age, weight, and height, all derived from a **creature**. Therefore, we can define **inheritance**: **is a concept in which a class or data type inherits a set of attributes from another type and extends it with additional features, resulting in a new, more specialized type.**

## EX :

// (:) meaning **inherit** at the code level

```
class Creature
```

```
{
```

```
    protected int weight;
```

```
    public string Name;
```

```
    private int Age;
```

```
    public void Move()
```

```
{
```

```
    Console.WriteLine("Creature is moving...");
```

```
}
```

```
    private void Eat()
```

```
{
```

```
    Console.WriteLine("Creature is eating ...");
```

```
}
```

```
}
```

```
class Human : Creature // Human will inherit from Creature.
```

```
{
```

```
    public int ID
```

```
    public void think()
```

```
{
```

```
    }  
}
```

Human h = new Human();

h.Name = "Kareem" // I can access everything is **public** only in Creature.

**Note:** I cannot access members with the **protected** or **private** access modifiers through an **object**, whether it belongs to Creature or Human.

The **protected** access modifier allows access from both the Human **class** and the Creature **class**, but it cannot be accessed from an object

access modifier	Creature class	Human class	object Human	object Creature
public	✓	✓	✓	✓
private	✓	✗	✗	✗
protected	✓	✓	✗	✗

– How can I access a variable from the Creature class inside an object of Human and inside the Human class? What happens behind the scenes at the code level?

- As soon as I create an object of **Human**, The compiler automatically creates an object of Creature inside the Human object. However, the reference to this part is hidden. I can access it through the (**base**) keyword

– **base** : is a keyword that allows me to access the parent class's members, such as methods and constructors, from within the child class(**human**). It acts as a **reference** to call the inherited members instead of rewriting them.

Ex : To access a protected member inside the **Human** class, I must use the base reference. I had the **Weight** member as **Protected**, so I used **base.Weight**.

دي كلمة اللي اقدر اوصل ليها للاوبجكت اللي عمله ال Compiler لأنها reference

يعني كده اقدر الـ accesses protected جوا class Creature اللي جوا Class human

بس الكلام ده مش هييدي كده و خلاص ده فيه حوار ليه ؟ باختصار علشان احنا معتمدين على ال default Constructor اللي عملها ال Compiler لكن لو انا كاتب ال Compiler بنفسي اقدر Constructors constructor مش هيعمل

فلشن اعمل call لـ create object من Creature اعمل من Constructors واحد call لـ

انا دلوقتي مثلا عندي Constructor من Creature ومن Human علي سبيل المثال:

```

class Creature
{
    public Creature() // default Cons
    {
    }

    public Creature(int age) // parameterized Cons
    {
        Age = age;
    }
}

Class Human
{
    public Human() // :base () or base(age)
    {
    }

    public Human(int id, int age) // :base() or base (age)
    {
        ID = id;
    }
}

```

جوا class Creature هنا هيلافي 2 Constructors فيهقولي انا هنا هعملك object بدلالة انهي Constructor هو من غير ما يسانني هو بيختار ال default طب لو انا معنديش ال default ده جوا Creature عندي Parameter المنطق بيقول يختار اللي فيه parameter بس انا بابدي اقوله اختيار ال Constructor اللي انا عايزه من خلال الـ **: base** هنا بنعمل ال constructor chain من خلال chaining

هاجى جوا class Human و اعمل ال chaining كده **base()** او **base()** براحتي حسب البيزنس

**(Search for overloading and Overriding)**

**Polymorphism :** The idea is that it makes it easier for you to deal with objects from different classes as if they were one, as long as they follow the same inheritance hierarchy or implement the same interface. Instead of creating separate methods for each class, you create a single method that handles them all, and it automatically recognizes each object's type and executes the appropriate method for it.

يعني بدل ما تعمل functions لكل class لوحده، بتعمل واحدة تتعامل مع الكل، وهي بقى لوحدها بتعرف كل object جاي منين و بتشغل ال function المناسبة ليه

**There are two types of polymorphism :**

1- dynamic (Run-time polymorphism) It is implemented through inheritance, abstract classes, and interfaces.

2- static (Compile time ) it's implemented throw (Methods & Operators overloading)

**Dynamic by inheritance EX:**

```
class Animal
{
    public void MakeSound()
    {
        Console.WriteLine("Animal make sound");
    }
}

class Dog : Animal
{
}

class Cat : Animal
{
}
```

## Main Void

```
Animal Dog1 = new Dog(); // Dog = Animal  
dog1.MakeSound();
```

```
Animal Cat1 = new Cat(); // Cat = Animal  
Cat1.MakeSound();
```

كده لما اجي اعمل **calling** لـ function **make sound** في المريتين هيظهر Animal make sound انا عايز اخلي كل animal يصدر صوت مختلف في الحالة دي هحط **class** function لكل **Console.WriteLine** ( ) عمل جوا ال ( ) بعدين اعمل **overriding** فهاجي عند ال class وال اخلي ال **method virtual** وال الابن اعمل **overriding**

```
class Animal  
{  
    public virtual void MakeSound()  
    {  
        Console.WriteLine("Animal make sound");  
    }  
}  
  
class Dog : Animal  
{  
    public overriding void MakeSound()  
    {  
        Console.WriteLine("dog barks");  
    }  
}
```

كده في ال Main function لما اعمل **calling** لـ **MakeSound** هينفذ بتاعت ال dog نفس الكلام لو عايز انفذ بتاعت ال cat

### **static (Compile time )**

```
class Calculator

{
    public int Add(int a , int b)

    {
        return a +b;
    }

    public double Add(double a , double b) //غيرت من int الى double

    {
        return a +b;
    }

    public double Add(double a , double b , double c)// حطيت parameter تاني

    {
        return a +b + c;
    }
}
```

### **Main void**

```
Calculator c1 = new Calculator();
```

وبكده اكون عملت execute اللي انا عايز اعلمها method هحدد ال ctrl + shift + space جوا الأقواس () overloading

```
var result = c1.Add(10;5.5+7.3);
```

```
Console.WriteLine(result);
```

### **search for polymorphism with array**

## Abstract Classes :

Class Creature

```
{  
    public void Move()  
    {  
        Console.WriteLine("Creature is moving...");  
    }  
}
```

هل كده الكلام ده صح ؟ لا ;

لية لما عملت **Creature object** من مش صح ؟

لان النوع ده نوع ناقص لبعض الصفات انا عارف هو بيعمل ايه بس مقررش احط قيمة مقررش احدد ال **Creature** ده بيتمشى ازاي لاني مشفتوش قبل كده انت عمرك شوفت **Creature** ماشي في الشارع

معني كده انها عبارة عن **function** بس ملهاش **implementation** في الحالة دي بيكون **abstract class**

يعني انا عارف ان هو بيتحرك بس معرفش بيتحرك الزاي ;

كنا في الاول بنعمل كده **Class Creature** هتحول و هتبقي كده **abstract** لو هو

معني كده ان ال **class** يحتوي علي **abstract function** او اكتر ومش شرط يكون كله **abstract function** ممكن **abstract class** اسمها **die** مثلانا هنا عارف ال **creature** بيموت الزاي و ممكن عادي تبقى موجودة جوا **function**

## Another example

**الكلب و القطط طالعين من كلاس ايه ؟**

ابوة برافو عليك **Animal** ؟ غلط ليبيه ؟ ارجع للسلайд رقم (3) مينفعش احنا الاتنين نطلع من نفس ال **class** و انت عندك صفات مختلفة عنني. الكلب و القطط مش نفس الصفات إذا كلب من **class** كلب و القطط من **class** قطة وبالتالي ال **class** اللي اسمه **animal** هيكون **abstract** هو كمان ليه ؟ انت عمرك شوفت **animal** ماشي في الشارع قولت اه ده اهو شوفته ؟ طيب تقدر توصفهولي بيتحرك الزاي بينط بيجرى بطئ سريع بيطير ثابت؟ ده مجرد **concept** في دماغنا فهيكون **abstract**

**There are two main types of classes.**

- **Concrete Class** I can create an object from it
- **Abstract Class** I can't create an object from it

### **Inherited Methods from the Object Class :**

```
class Human
```

{

```
public int ID;
```

```
public string Name;
```

}

```
Human h = new Human();
```

h. معرفش جت منین functions هلاقی مجموعه من ال //

these functions coming from a **class called Object**

**القاعدة :** أي data type في الدنيا لازم ي inherit من class اسمه object هو اسم ال class كده

فكرة اني محتاج احط standardization الناس اللي شغالة معانيا تمشي عليه بانا نحط كل الحاجات الـ common في class واحد و خلي الناس ت inherit منه إذا class object هو الـ parent class لأن أي data type وبالتالي أي class object inherit من class object لازم ي

### Some conclusions:

دۇل functions و بالتالى ال. These functions that appeared must have a **public access modifier**.  
لەنی انا قادر اعمل عليهم abstract مستحيل يكۈن معمول لىيەم access

علشان اعرف كل ال functions اللي موجودة جوا هعمل كده

```
Human h = new Human();
```

**object** // I will click on it to select it, then right-click and choose **Go to Definition**

هلاقی 2 functions static و 4 functions public

```
public virtual bool Equals (Object obj);
```

```
public virtual int GetHashCode();
```

Public type GetType();

```
public virtual string ToString();
```

نمسك الاول اللي معمول ليهم public و نشوف ال implementation بتاعهم الزاي و فايلدته ايه

**operator overloading** قبل ما نشوفهم خلينا نعمل حاجه و كمان قبل ما نبدأ فيهم لازم تكون عارف

Class Employee : Human

```
{  
}
```

کده عامل inherit من object class ف انا ممکن اخليه کده

```
object h = new Human();
```

h. و مش هشوف ولا ال functions کده هشوف بس ال 4 // ID , Name

Class program

```
{
```

```
    static void Test (object obj) //
```

```
{
```

**obj . // i can access any of the four functions**

```
}
```

```
    static void Main (String args)
```

```
{
```

```
        Human h = new Human();
```

```
        object h = new Human();
```

**// i can call the function with any of these values**

```
        Test(h);
```

```
        Test(456);
```

```
        Test("demo");
```

```
        Test(new Employee());
```

```
}
```

```
}
```

لیه اقدر اعمل call بای value لانی انا عارف ان ای data type من inherit عامل object class

## Static Building & Dynamic Building

**Dynamic Building** : I do not know the type of the object I am dealing with at compile time, but it can be determined at runtime.

**Ex :**

```
static void Test (object obj)
```

```
{
```

```
}
```

**Static Building** : I know the type of the object I am dealing with at compile time, and it is determined before runtime.

**Ex :**

```
static void MyFunc(Employee emp)
```

```
}
```

here i know that the object must be of type Employee

```
}
```

**Note:** These examples are only to convey the idea. If you want to delve deeper, search for the

---

## First Function

يتبع لك object وعندني Equals() هينده object هبيقي عندي obj

على المثال بناء على Human بما ان object يمكن ادتها عن طريق inherit من Human object من Human و اعمل human passing

وكمان بما ان ال function دى بتعمل if condition ل return bool ممكن احطها جوا

```
Human h1 = new Human();
```

```
h.ID = 1;
```

```
Human h2 = new Human();
```

```
h2.ID = 1;
```

```
h.Equals(h2); // object calling function & object passing
```

```

if (h.Equals(h2));

Console.WriteLine ("Equal");

else

Console.WriteLine("Not equal ");

```

هنا ال reference هيقارن بين reference و reference مكتوب جوا ال function هو مكتوب object لكن object ايه هو ميعرفش ف اكيد مش هيعرف ال data بقىته data يخليه reference على ال reference بدل ما هو بقىته default implementation

- لو انا جيت جوا ال function و علمت **return this ID == obj**. مش هقدر اشوف ال ID لأنه هو بيتتعامل مع casting على انه عامل class object من inherit class human

```

class Human

{
    public int ID;

    public string Name;

    public override bool Equals(object obj)

    {
        Human h = (Human)obj; // casting

        return this ID == h.ID;
    }
}

void Main

if (h.Equals(h2));

Console.WriteLine ("Equal");

else

Console.WriteLine("Not equal ");

```

كده انا بقارن data ب data بس عندي مشكلة هنا ان طريقة ال casting دي مش safe ليه ؟ لأن انا لو غيرت هنا if (h.Equals("kareem")); هيعديها تمام لكن هيضرب في ال compiler int او string مثلًا خلبيته run time

الحل اني اعمل casting safe و فيه اكتر من طريقة اعمل بيها

Human h = obj **as** Human ;                      **(as )**

معناها ان لو الحاجه دي طلعت نوعها human تمام مطلعش human مش هيضرب في ال run time بس هيختلي قيمة ال Exception بس انا لو استخدمته وقولت كده reference Human ;return this ID == null.ID هيضرب  
بردو الحل اني احط if condition ويبقى كده لو ال casting نفع هيقارن الاثنين ببعض لو منفعش ال h هتساوي null و هيرجع false و ان الاثنين مش متساوين

public override bool Equals(object obj)

```
{  
    Human h = obj as Human ;  
  
    if (h == null)  
        return false;  
  
    return this ID == h.ID;  
}
```

**Another method (is) :**

احط ال casting قبل ما اعمل condition

```
public override bool Equals(object obj)  
{  
    if (obj is Human )  
    {  
        Human h = (Human)obj;  
  
        return this ID == h.ID;  
    }  
  
    else  
  
    return false ;  
}  
}
```

## Second Function

جاي منين object هدفها انها بترجعلي جملة تعبر عن الـ **ToString()**;

```
main void
```

```
Human h1 = new Human();
```

```
h.ID = 1;
```

```
Human h2 = new Human();
```

```
h2.ID = 1;
```

```
string msg = h1.ToString();
```

```
Console.WriteLine(msg);
```

```
Console.WriteLine(h2.ToString());
```

في الحالتين هيعمل function execute لـ **ToString()** اللي جوا object هيدبني اسم ال namespace + اسم ال class ده ال default implementation

بس بما ان ال **ToString()** دى **virtual** ف انا اقدر اعلمها **override**

```
class Human
```

```
{
```

```
    public int ID;
```

```
    public string Name;
```

```
    public override string ToString()
```

```
{
```

```
    return $"ID : {ID} \t Name : {Name}";
```

```
}
```

```
}
```

```
main void
```

```
string msg = h1.ToString();
```

الزاي ال **Console.WriteLine()** بتتعامل مع ال **object** function

بـ **Console.WriteLine(h1);** / compiler هنا مش هيعرض لان دى **ToString()** function overload

كده اقدر ابتلها أى **object** من اى **data type** object وبالتالي أى **object** يكون جواه **ToString()** فهنه عليهما

## **third Function**

```
public virtual int GetHashCode();
```

فيه علاقة بين الـ `GetHash` function و الـ `equals` function bool `equals` يتعمل على `hashCode` الـ `hashCode` من غير `override` بـ `default` بس هيبي فيه عملت لـ `hashCode` function `equals` override warning

لو اديتك 2 object مثلا ID و name و ID و name و طلع ال hash code متساوي للاثنين يعني المفروض الاتنين دول يكونو متساويين

**الخلاصة :** انا دلوقتى رحت عملت equals function override لـ hashCode function ولـ equals function اعتمدت على الـ default implementation ده كده هيسب مشكلة ليه ؟ لأنى كده ممكن اديك 2 objects و اسالك هما متساوين تقولي اه بس لما اجي اصى على الـ hash code الاقييم مختلفين فده مش صح . الصح ان لو الـ objects بتاعتي هتستخدم الـ hash generate زى ما اعمل hashCode اعمل كمان لـ equals function و code function بالطريقة اللي انا عايزها

### **fourth function**

**Public type GetType();**

مفيهاش override و بتعمل return من class type اسمه object ال type هكون محتوي علي شوية معلومات تخص ال data type يعني ممكن نقول انها بتعمل return لـ meta data الخاصة للاوبيجكت مش بس اعرف ال access modifiers و ال functions اللي جوا و ال namespace و اسم ال class لا كمان اعرف ال variables هيفيني في حاجه اسمها reflection و ده عباره عن Reverse engineering في ال run time بيكون معايا object في ال run time و اقدر من خلاله استخلص معلومات زي مثلا معايا variable اسمه ID فممكن اقوله هاتلي ال value بتاعت ID

### **fifth function**

**protected Object MemberwiseClone();**

طبع الـ `function` دى لازمتها ايه و ليه معمولة `protected` ؟ دي بتعمل `copy` من الـ `object` اللي معانيا بكل حاجه فيه و ترجعي الـ `object` جديد معمولة `protected` علشان مش دايما هشتغل بيها بنفس الشكل ده طريقة الـ `implementation`

انا مثلًا عايز `h1` انقله في `object` جديد بكل الـ `data` اللي فيه

عمل `Human` class `Human` function `Clone` public الآتي `Human` جوا ويعدين عمل

```
public Human Clone() // function
{
    return(Human)this.MemberWiseClone();
}

main void
```

Human h3 = h1.Clone(); // reference h3 الجديد ب كده اقدر اشاور على ال

طب ليه من الاول ال public دى مش function ؟ لأنانا من جوا ال class اللي بعمل control على ال object انا اللي  
بسح لحد انه هو بعمل clone لـ object بتاعي سبب تاني ان ال Member Wise Clone هيتخرج منه مشاكل استخدامها مش  
هيكون الأمثل لأنها بتعمل Shallow Copying انا عندي نوعين من ال deep copy وCopying Shallow :  
دور علي الفرق بينهم

we can define any class using one of the three keywords: (**Class**, **struct** , **enum** ). slide2

### Enum :

انا لو عايز اعبر عن gender جوا class هو male و female ولا string و Int و  
عالی نفترض اني عندي class Employee char

```
class Employee
```

```
{
```

```
    public int ID;
```

```
    public string Name;
```

```
    public string Gender; // made it a string.
```

```
}
```

احنا كده هنعمله ب **enum** بس ليه لما علمته ب string فيه مشاكل خلينا نكمل

```
void Main
```

```
Employee e1 = new Employee();
```

```
e1.Gender = "Male";
```

المشاكل هنا ان مش كل الناس هتكتب كل الكلمة زي ما انا عايز و كمان ال processing validation صعب عليه

لكن مثلا الملاحظات زي عنوان الشقة و تفاصيل ال location string عادي لأن مش هحتاج اعمل عليها processing validation ولا processing validation يفضل نبعد عن string علشان مثلا ممكن يحط حرف غلط او يكتب الكلمة غلط و نفس الحوار في ال char اللي هو ممكن مثلا يكتب حرف small او capital

تمام نيجي لل int بقى هاجي جوا ال function Main و اعمل كده

```
void Main
```

```
const int Male = 0;
```

```
Const int Female = 1;
```

```
Employee e1 = new Employee();
```

```
e1.Gender = Male ;
```

هذا الجملة تمام و زي الفل ولو حد كتبتها غلط ال compiler هيعترض لأنها متعرف هنا على اساس انه هو variable بس عيبه ان هو متعرف على مستوى ال function و local variable accesses عليهم من اي مكان تاني يعني لو فيه بشهندس شغال معاليا تاني هبيقى لازم يعرف نفس القيمة دي تاني فيه عمل Code Duplication و ده مش صح ف الحل اني اخليه enum

### enum : set of constant values

declaration inside **namespace**

**namespace** EmployeeData

{

enum Gender // عباره عن constant يعتبر

{

Male = 0; // **Values**

Female = 1;

}

}

main void

Gender g = Gender.Male;

class Employee

{

public int ID;

public string Name;

public **Gender** Gender; // **data type**

}

**Gender** g = Gender.Male; // variable form data type Gender

Employee e1 = new Employee();

e1.gender = Male;

ال **data type** gender يعني اقدر اعمل منه variable فالجملة دي معناها هعرف variable اسمه g من نوع Gender و هيكون معاه قيمة واحدة بس من القيم اللي فوق و مقدرش اديله int

ال **values** اللي بتكون جوا لا enum by default تكون integer وينفع يكون اي حاجه من مشتقات ال integer زي ال short و ال long مثلًا لكن sting او double لا

**Another EX :The colors consists of three main colors RGB**

### [Flags]

Enum color

{

Red = 1 ;

Green= 2;

Blue = 4;

}

فيه حوار وراء اختيار الارقام دي بنفع اكتب values و بنفع مكتش خالص لما مكتش هيتعبر ال red بزيرو و يزود 1 و بعدين ال green ب 1 و ال blue ب 2 كلام جميل . لكن لو انا مدي value لل red بس ؟ هيزود هو واحد علي القيمة اللي انا مدعيها لل red مثلانا عامل كده Red =4 هو هيخلى ال green ب 5 و ال blue ب 6 . طيب لو مدي قيمة لل في النص بس اللي هو ال green ب 20 مثلانا كده ال Red هيكون ب 0 و ال blue ب 21 .

main void

color c = color.Red; ب c خليت ال red

طب لو عندي color بيحتوي على قيمتين هنا يستخدم ال ( or operator | )

### ex

color c = color.Red | color.Green; //bitwise or

Console.WriteLine(c); // 3 هيدبني

ايه اللي بيحصل هنا : ال red قيمته ب 1 و ال green قيمته ب 2 دور ال bitwiseing انها بتعمل bitwiseing و بتحول الرقمين الى Binary و تبدا تعمل بينهم or معنى bitwiseing هتم علي مستوى ال bits

ال Red بكام ب 1 لما نحوله Binary هيكون كده 0001

ال green بكام ب 2 لما نحوله Binary هيكون كده 0010

هناخد ال digit اللي فوق مع اللي تحت

c هيبقى 3 اللي هي قيمة ال int الناتج ده لو حولته الى 0011

طب لو كنت سايب القيمة بتاعت ال Red ب 0 ال binary بتاع ال 0 = 0000 هيجي يعمل green مع ال green هيطلعل

بردو و ده طبعا مش صح

امتنى اعمل oring لو كان ال data type الواحد فيه اكتر من value زي ال color لكن في ال gender هي واحدة بس بس female يا male

لو هعمل oring اختار دايما الارقام دي احطها في ال values دي (1,2,4,8,16,32) ليه ؟ لأن الارقام دي ال binary بتعالها فيه 1 بس و الباقى اصفار و الواحد ده مكانه بيتغير في كل رقم فيهم ف لما اجي اعمل bitwise هيطبع رقم جديد و بالتالي لون جديد

main

```
color c = color.Red;  
color c = color.Red | color.Green;  
if(c == Color.Red) // او green مش هنفرق او  
console.WriteLine(c) // هيطبع false
```

```
if(c == (Color.Red | Color.Green))  
console.WriteLine(c) // هيطبع true
```

هيطبع 3 لو عايزه يطبع Red,Green مثلا هستخدم حاجه اسمها [Flag] attribute بستخدمه فوق Enum دي بتخليه بدل ما يطبع integer يطبع الالوان

لو انا عايز اعمل check لل c جواه لون ال red من الالوان دي ولا لا

```
c == color.Red | Color.Green | Color.Blue;  
if ((c & color.Red) == Color.Red)  
Console.WriteLine("Contains Red");
```

هعمل oring من خلال ال binary بناء على الأرقام الموجودة في ال values

0001

0010

0100

-----

الرقم ده معناه 7 // 0111

بس ال red الواحد بتعاله موجود في الاول و الناتج اللي طبع الواحد بتعاله في الاول بردو كده اكيد ال red موجود لأن مفيش غير ال red بس هو فيه واحد بس في الاول لو كان الناتج اداني الرقم اللي في الاول ب zero كده هيبقى ال red مش موجود

طب خلاص انا بقى معايا الناتج اللي هو 0111 هعمله oring مع ال red اللي هو 0001 هيطبع red

لو مش فاهم النقطة دي ذاكر binary

## Struct :

عبارة عن data type يعبر عنه عن logical representation زى ال implementation على مستوى ال logic مفيش فرق بين ال class و ال struct لكن فيه فرق على مستوى ال class // declaration

```
struct Complex
{
    public int Real;
    public int Img;
}
```

void main

```
Complex c1;
```

ايه الفرق بين ال struct وال class ؟

ف ال class ده reference يكون في ال struct variable لكن في ال class يكون عنوان زى ال stack

ال c1 هيكون جواه ال class values علطول مش جواه عنوان زى ال initialization علشان استخدم ال variable لازم اعمله

```
Complex C1 = new Complex(); // initialization
```

لو الجملة دي اتعلمت وانا استخدم class object هيقي معناها انه عملت class Complex لكن على مستوى ال heap بدلالة ال variable عملت default constructor ومعلمتش اي حاجة في ال

دور ال default constructor يعمل بـ variables اللي جوا ال struct بـ initialization بـ null أو reference types

```
Complex C1 = new Complex();
```

Console.WriteLine(C1);

يمكن اعمل initialization بطريقة تانية بس لازم كل ال variables اللي جوا struct اعملهم

```
Complex C1;
```

```
C1.Real = 9;
```

```
C1.Img = 5;
```

بس مينفعش اروح جوا struct نفسها و احط values على عكس ال

طيب بما ان ال variable ده ليه constructor لو عملت constructor جوا ال ايه اللي هيحصل

```

struct Complex

{

    public int Real;

    public int Img;

    public Complex (int real , int img)

    {

        Real = real;

        Img = img;

    }

}

void Main

    Comlex C1 = new Complex (9,5);

```

طب افرض جوا ال constructor معلمتش img هنا ال compiler هيتعرض لاني مش هينفع اعمل variables initialize غير وانا عامل لكل ال initialization

طب ليه الكلام ده مش بيحصل في ال class لأني لما بعمل object بيكون ال variables معنول ليهم initialize من خلال constructor و بعدين بنده لل new keyword

في ال class كنت بقدر اعمل اعمل ال constructor default و احط ال parameterized هنا مش هينفع حتى بعد ما عملت ال constructor default موجود

ال class في ال constructor في ال class فايدته اني اتحكم اللي اعمل object وجود ال constructor في ال class بقول للناس اللي عايز يعمل object يعمله بالطريقة دي لكن هنا في ال struct استدعائي ال constructor ولا لا مش هيمعني اني اعمل variable لأنني ممكن أعلمه بكل طريقة غير طريقة ال new

مینفعش class ي inherit من struct ولا حتى من struct

قاعدة : فيه حاجات هيقي مسموح لل compiler يعملها و من نوع عليا انا اعلمه

زي مثلما ال struct هو ب inherit من object لكن انا مقدرش اخليه ي inherit من object

طب ليه اساسا ال inheritance من نوع اللي ببساطه مقدرش اعمل polymorphism علشان اعمل polymorphism لازم ابص للحاجه انها حاجه تانية زي مثلما ال Creature عن طريق ال Human لكن هنا مفيش reference

امتنى اعمل الحاجه دي struct ولا class ؟

هل الداتا تحتاج بعتها من مكان إلى مكان لو اه يبقى class لو لا يبقى struct

ال variables اللي من نوع struct هي value types لما اجي اعرفها على مستوى ال main function هتتخزن في ال stack فهياخذ مساحة كبيرة في ال stack لو حجم ال data عندي كبير الافضل اخليه class لأن ال reference هيتخزن في ال heap و ال object في ال stack

هل تحتاج اعمل inheritance ولا لا ؟

لو عندي data type بيمثل نوع من انواع الارقام زي ال fractions لاني بنعامل مع رقم لو حبيت بعثته في by reference مش by value

فيه اسمه DateTime data type represent تاريخ و ساعات و ده جزء من library في ال .NET. بيكون معمول struct لاني لو حبيت بعثته بيكون inherit و مش تحتاج منه Framework

\*\*\*\*\*

بعد السطر ده مفيش اي علاقة بينهم C2 هحطها في C1 اللي في values ال

C1.Real = 5; بناء Real مش هياثر على ال

بس لو ال C2 جواه حاجه reference Shallow copy وابقى انا تحتاج اعمل deep Copy

جوا if(C1==C2) لو عايز استخدمه اعمله references overloading لانهم مش struct مينفعش اعمل كده بين حاجتين // لـ overloading و بالباقي compex not equal

• ال value object زي ما عاملين class object من inherit Inheit

### Boxing & Unboxing

static void Test (object obj) // reference

```
{  
}  
Test(5);
```

حيث انا بعثته value

الكلام ده اسمه boxing و هو تحويل ال value type الى reference type

اللـي بيحصل على مستوى ال memory ان هو هيعمل object جديد في ال heap value بـناعته 5 و هيلـي ال obj يـشاور عليه .

لو هعمل function Equals override لـ boxing و unboxing ولو عملته كـثير هيـاثر على ال performance

```
public override bool Equals(object obj) // يسبق object
{
    return // casting unboxing
}

main void
if (C1.Equals(C2) ) // يبقى نفس الحوار بناء value بيعتلہ boxing
{
}
```