Kareem Montaser

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 الـ 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 اختلفت فده بقي باب و دى بقي ترابيزة الفكرة دي غلط
الصح مينفعش احنا الاتنين نطلع من نفس ال class و انت عندك صفات مختلفة عني يعني مثلا الترابيزة ليها صفات غير الباب و actions 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()

طب لو انا عندي private variable و عايز access عليه اقدر اعمل كده ان احط ال public ده جوا private variable
و بكده 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 setter, 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 هنفذ الأول الدالة Main بالتالي هيكون جوه ال varibale name, stack اسمه num و قيمته 5

بعدين جوه ال stack ال variable اللي اسمه y

بعدين هينده للدالة test و اللي فيها variable x قيمته 3 و بكده 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❎**
- 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 InstruactorTurnLightOn()
      {
         // 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** كأنها reference

يعني كده اقدر accesses ال protected **اللي جوا** class Creature جوا Class human

بس الكلام ده مش هيعدي كده و خلاص ده فيه حوار ليه ؟ باختصار علشان احنا معتمدين علي ال default Constructors اللي عملها ال Compiler لكن لو انا كاتب ال Constructor بنفسي الـ compiler مش هيعملى constructor

فعلشان اعمل create object من Creature لازم اعمل call ل واحد من Constructors

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

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 function 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 (Runt-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 للـ function في المرتين هيظهر Animal make sound انا عايز اخلي كل animal يصدر صوت مختلف في الحالة دي هحط function لكل class و جوا ال function هعمل Console.WriteLine() بعدين اعمل **overriding** فهاجى عند ال class ال parent و اخلي ال **method virtual** وال class 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 function Main لما اعمل calling للـ method 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 الي 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();
```

c1. Add() جوا الأقواس ctrl + shift + space هحدد ال method اللي انا عايز اعلمها execute وبكده اكون عملت overloading

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

Console.WriteLine(result); هينفذ الميثود الاخيرة و بكده بقيت compile

search for polymorphism with array

Abstract Classes :

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

Creature c = new Creature (); هل كده الكلام ده صح ؟ لا

ليه لما عملت object من Creature مش صح ؟

لان النوع ده نوع ناقص لبعض الصفات انا عارف هو بيعمل ايه بس مقدرش احط قيمة مقدرش احدد ال Creature ده بيمشي 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**

public **abstract** void move(); يعني انا عارف ان هو بيتحرك بس معرفش بيتحرك الزاي

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

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

Another example

What class are the dog and cat derived from? الكلب و القطة طالعين من كلاس ايه

ايوة برافو عليك **Animal** ؟ غلط ليييه ؟ ارجع للسلايد رقم (3) مينفعش احنا الاتنين نطلع من نفس ال class و انت عندك صفات مختلفة عني. الكلب و القطة مش نفس الصفات إذا كلب من class كلب و القطة من class قطة بالتالي ال class اللي اسمه animal هيكون abstract هو كمان ليه ؟ انت عمرك شوفت animal ماشي في الشارع قولت اه ده animal اهو شوفته ؟ طيب تقدر توصفهولى 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 منين functions هلاقي مجموعة 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

فكرته اني محتاج احط standardization الناس اللي شغالة معايا تمشي عليه باننا نحط كل الحاجات ال common في class واحد و خلي الناس ت inherit منه إذا class object هو الـ parent class لأي data type وبالتالي أي class عندي لازم ي inherit من class object

Some conclusions:

These functions that appeared must have a **public access modifier**.دول functions وبالتالي ال access لاني انا قادر اعمل عليهم abstract مستحيل يكون معمول ليهم

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**

هلاقي 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 فايدته ايه

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

```
Class Employee : Human
{
}
```

كده Human عامل inherit من class object ف انا ممكن اخليه كده

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

h. // 4 كده هشوف بس ال functions اللي في class object و مش هشوف ولا ال 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());
         }
}
```

ليه اقدر اعمل call بأي value لاني انا عارف ان اي data type عامل inherit من class object

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

public virtual bool Equals (Object obj); هيبقي عندي object هينده Equals() وعندي object يتبعت ك obj

علي المثال بتاع Human بما ان Human عامل inherit من object ممكن اندها عن طريق object من Human و اعمل passing من نوع human

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

```
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 default implementation هيقارن بين reference و reference لان ال code مكتوب جوا ال function Equals هو مكتوب object لكن object ايه هو ميعرفش ف اكيد مش هيعرف ال data بتاعته

بس ال function معمول ليها virtual معني كده ان class human اللي بيعمل inherit من object يقدر يعمل override علي ال function Equals و يغير ال default implementation بدل ما هو reference ب reference يخليه data ب data

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

```
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 ؟ لان انا لو غيرت هنا

if (h.Equals(“kareem”)); خليته string مثلا او int ال compiler هيعديها تمام لكن هيضرب في ال run time

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

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

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

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

Another method (is) : condition احط ال casting قبل ما اعمل

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

Second Function

public virtual string ToString(); جاي منين object هدفها انها بترجعلي جملة تعبر عن الـ

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 execute لل function اللي جوا object هيديني اسم ال namespace + اسم ال class ده ال default implementation

بس بما ان ال function دي 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(); هيعمل execute ل Tostring اللي جوا class Human

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

Console.WriteLine (h1); / compiler هنا مش هيعترض لان دي overload عندها function بتاخد من نوع object

كده اقدر ابعتلها أي object من اي data type وبالتالي أي object لازم يكون جواه function Tostring فهنده عليها

third Function

public virtual int GetHashCode();

فيه علاقة بين ال function دى و ال function bool equals ال GetHast بتعمل return الي integer مثال لو انا عملت لل function equals override و خليت ال HashCode من غير override ب ال default بس هيبقي فيه warning

دور ال GetHash ني اقدر اوصل لاي object من خلال ال hash code بتاعه بدل ما انا اقعد اقارن بين ال object وبعض , لو ال objects مختلفة المفروض كل object يكون ليه ال hash code مختلف بس ال hash code بيتعمله generate من خلال ال Hashing Algorithms و (search about it) بس دي يعني طرق علشان generate hash code لكل object , في ال default implementation بتاع ال function GetHashCode ال generate بتاع ال hash code بيحصل automatic . الحوار كله لو خليته override فعلشان generate كود بنفسي لازم اعتمد علي ال data اللي جوا الـ class هعمل عليها processing و بعدين اطلع الرقم علشان اعمل generate ل hash code ممكن اعتمد على ال ID لو مش متكرر لكن لو متكرر هعمل generate ب ال Algorithms

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

الخلاصة : انا دلوقتى رحت عملت override لل function equals و لل function GetHashCode اعتمدت على الـ default implementation ده كده هيسبب مشكلة لية ؟ لاني كده ممكن اديك 2 objects و اسالك هما متساويين تقولي اه بس لما اجي ابص علي ال hash code الاقيهم مختلفين فده مش صح . الصح ان لو ال objects بتاعتي هتستخدم ال hash code function زي ما اعمل override لل equals function اعمل كمان لل hash code و generate ال hash code بالطريقة اللي انا عايزها

fourth function

Public type GetType();

مفيهاش override و بتعمل return من class اسمه type .ال object من class type هيكون محتوي علي شوية معلومات تخص ال data type يعني ممكن نقول انها بتعمل return لل meta data الخاصة للاوبجكت مش بس اعرف ال namespace و اسم ال class لا كمان اعرف ال variables اللي جوا و ال functions و ال access modifiers و ده هيفيدنى فى حاجه اسمها reflection و ده عبارة 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();

طب ال function دى لازمتها ايه و ليه معمولة protected ؟ دي بتعمل copy من ال object اللي معايا بكل حاجه فيه و ترجعلي object جديد معمولة protected علشان مش دايما هشتغل بيها بنفس الشكل ده طريقة ال implementation

انا مثلا عايز h1 انقله في object جديد بكل ال data اللي فيه

هعمل function جوا class Human الآتي public Human Clone وبعدين هعمل return

```
    public Human Clone() // function
    {
        return(Human)this.MemberWiseClone();
    }
main void
Human h3 = h1.Clone();  // كده اقدر اشاور على ال reference الجديد ب h3
```

طب ليه من الاول ال function دى مش public ؟ لان انا من جوا ال class اللي بعمل control علي ال object انا اللي بسمح لحد انه هو يعمل clone لل object بتاعي سبب تاني ان ال Member Wise Clone هينتج منه مشاكل استخدامها مش هيكون الأمثل لانها بتعمل Shallow Copying انا عندي نوعين من ال Copying Shallow : Copying و deep copy 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 ولا female مثلا ممكن اعبر عنه باكتر من طريقة string و Int و char تعالي نفترض اني عندي class Employee

```
class Employee

{

    public int ID;

    public string Name;

    public string Gender; // made it a string.

}
```

احنا كده 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 و ال validation صعب عليه

لكن مثلا الملاحظات زي عنوان الشقة و تفاصيل الـ location انا ممكن اخليها string عادي لان مش هحتاج اعمل عليها processing ولا validation لكن لو حاجه تتطلب processing 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** بقي 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 ;
```

هنا الجملة تمام و زي الفل ولو حد كتبتها غلط ال compiler هيعترض لانها متعرف هنا على اساس انه هو variable بس عيبه ان هو متعرف علي مستوى ال function و local variable مقدرش accesses عليهم من اي مكان تاني يعني لو فيه بشمهندس شغال معايا تاني هيبقى لازم يعرف نفس القيم دي تاني فيهعمل Code Duplication و ده مش صح ف الحل اني اخليه enum

enum : set of constant values

declaration inside **namespace**

```
namespace EmployeeData
{
    enum Gender    // عبارة 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 gender ده data type يعني اقدر اعمل منه variabel 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 الـ enum ب by default بتكون **integer** وينفع يكون اي حاجه من مشتقات ال integer زي ال short و ال long مثلا 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 ينفع اكتب values و ينفع مكتبش خالص لما مكتبش هيعتبر ال red بزيرو و يزود 1 و بعدين ال green ب 1 و الـ blue ب 2 كلام جميل . لكن لو انا مدي value لل **red بس ؟** هيزود هو **واحد** علي القيمة اللي انا مديها لل red مثلا انا عامل كده Red =4 هو هيخلى ال green ب 5 و الـ blue ب 6 . طيب لو مدي قيمة لل في النص بس اللي هو الـ green ب 20 مثلا كده ال Red هيكون ب 0 و ال blue ب 21 .

main void

color c = color.Red; خليت ال c ب red

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

ex

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

Console.WriteLine(c); // 3 هيديني

ايه اللي بيحصل هنا : ال red قيمته ب 1 و الـ green قيمته ب 2 دور الـ bitwise or انها بتعمل bitwiseing و بتحول الرقمين الي Binary و تبدا تعمل بينهم bitwise or معنى bitwiseing يعني عملية هتم علي مستوي 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 هيبقي 3 اللي هي قيمة ال int الناتج ده لو حولته الى **0011** = 0 or 0 , 0 or 0 = 0 , 1 or 0 = 1

طب لو كنت سايب القيمة بتاعت ال Red ب 0 ال binary بتاع ال 0 = 0000 هيجي يعمل bitwise مع ال green هيطلعلي green بردو و ده طبعا مش صح

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

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

main

color c = color.Red;

color c = color.Red |color.Green;

if(c == Color.Red) // او green مش هتفرق

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 attribute **[Flag]** 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

—----------

0111 // الرقم ده معناه 7

بس ال red **الواحد** بتاعه موجود في الاول و الناتج اللي طلع **الواحد** بتاعه في الاول بردو كده اكيد ال red موجود لان مفيش غير ال red بس هو فيه واحد بس في الاول لو كان **الناتج** اداني الرقم للي في الاول ب zero كده هيبقي ال red مش موجود

طب خلاص انا بقى معايا الناتج اللي هو 0111 هعمله **anding** بقي مش oring مع ال red اللي هو 0001 هيطلع red

لو مش فاهم النقطة دي ذاكر binary

Struct :

عبارة عن data type بعبر بيه عن logical representation زي ال class

علي مستوى ال logic مفيش فرق بين ال class و ال struct لكن فيه فرق على مستوى ال implementation

```
// declaration

struct Complex

{

    public int Real;

    public int Img;

}

void main

Complex c1;
```

ايه الفرق بين ال class وال struct ؟

ف ال class ده Complex c1; بيكون reference لكن في ال struct بيكون variable وبيتعرف في الـ stack

ال c1 هيكون جواه ال values علطول مش جواه عنوان زي ال class

علشان استخدم ال variable لازم اعمله initialization

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

لو الجملة دي اتعلمت وانا بستخدم class هيبقي معناها اني عملت object من class Complex لكن علي مستوي ال struct يبقي عملت initialization لل variable c بدلالة ال default constructor ومعملتش اي حاجة في ال heap

دور الـ default constructor هيعمل initialize لل variables اللي جوا ال struct ب ال values بتاعتهم سوا كان value types أو reference types ب null

```
Comlex C1 = new Complex ();

Console.WriteLine(C1);
```

class و اسم ال namespace هيديني اسم ال

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

```
Complex C1;

C1.Real = 9;

C1.Img = 5;
```

بس مينفعش اروح جوا struct نفسها و احط values لل variables علي عكس ال class

طيب بما ان ال variable ده ليه constructor لو عملت constructor جوا ال struct 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 initialize لل img هنا ال compiler هيعترض لاني مش هينفع اعمل initialization غير وانا عامل لكل ال variables initialize

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

في ال class كنت بقدر اعمل overloading لل constructor default و احط ال parameterized هنا مش هينفع حتي بعد ما عملت ال parameterized constructor ال default هيفضل موجود

ال constructor في ال class فايدته اني اتحكم الزاي اعمل object و وجود الـ constructor في ال class بقول للناس اللي عايز يعمل object يعمله بالطريقة دي لكن هنا في ال struct استدعائي لل constructor ولا لا مش هيمنعني اني اعمل variable لاني ممكن أعلمه بكذا طريقة غير طريقة ال new

مينفعش struct ي inherit من struct ولا حتى من class

قاعدة : فيه حاجات هيبقي مسموح لل compiler يعملها و ممنوع عليا انا اعلمها

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

طب ليه اساسا ال inheritance ممنوع لاني ببساطه مقدرش اعمل polymorphism علشان اعمل polymorphism لازم ابص للحاجه انها حاجه تانية زي مثلا ال Human ان هو Creature عن طريق ال reference لكن هنا مفيش reference

امتى اعمل الحاجه دي class ولا struct ؟

هل الداتا محتاج ابعتها من مكان إلى مكان لو اه يبقى class لو لا يبقي struct

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

هل محتاج اعمل inheritance ولا لا ؟

لو عندي data type بيمثل نوع من انواع الارقام زي ال fractions مثلا بعمله struct لاني بتعامل مع رقم لو حبيت ابعته في حته ببعته by value مش by reference

فيه data type اسمه DateTime ده بقدر represent تاريخ و ساعات و ده جزء من library في ال .NET Framework بيكون معمول struct لاني لو حبيت ابعته بيكون by value و مش محتاج inherit منه

****

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

C2 بتاع Real مش هياثر علي ال // ;C1.Real = 5

بس لو ال C2 جواه حاجه reference هيعمل Shallow copy وابقي انا محتاج اعمل deep Copy

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

- ال value types زي ما عاملين Inheit من class object عامين كمان inherit من value object

Boxing & Unboxing

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

{

}

Test(5); // جيت انا بعتله value
```

الكلام ده اسمه **boxing** و هو تحويل ال value type الي reference type

اللي بيحصل علي مستوي ال memory ان هو هيعمل object جديد فى ال heap ال value بتاعته 5 و هيخلى ال obj يشاور عليه .

لو هعمل override لل function Equals ف انا كده هعمل boxing و unboxing ولو عملته كتير هياثر علي ال performance

```
public override bool Equals(object obj)  // يستقبل object
{
   return   // casting unboxing
}
main void
if (C1.Equals(C2) ) // ببعتله value  هيبقي نفس الحوار بتاع test و يبقي boxing
{


}
```