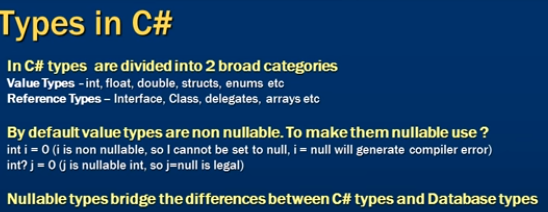
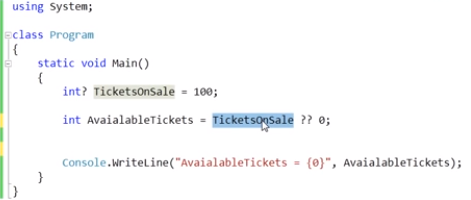


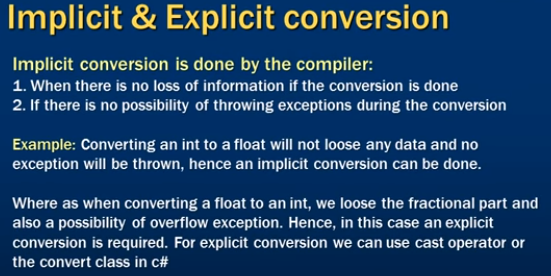
1. Ternary operator returns Boolean type.



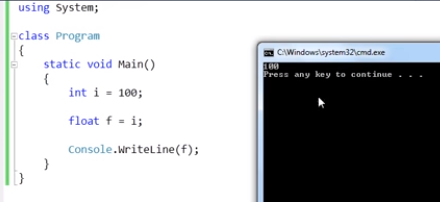




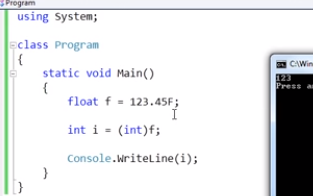
If TicketsOnSale is null return 0.

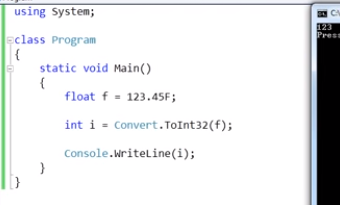


1. Implicit

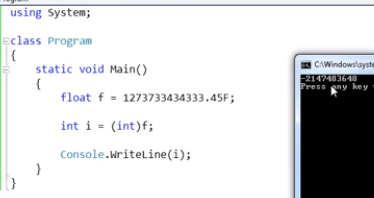


1. explicit
2. Type cast operator



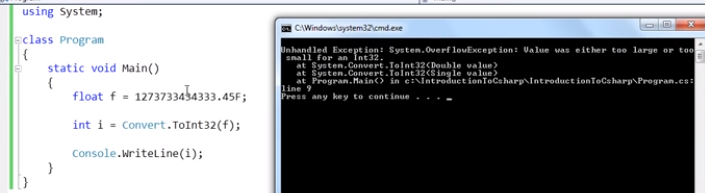


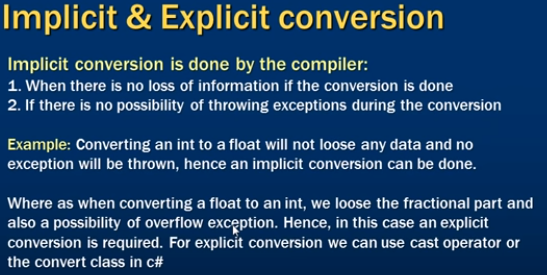
Difference:

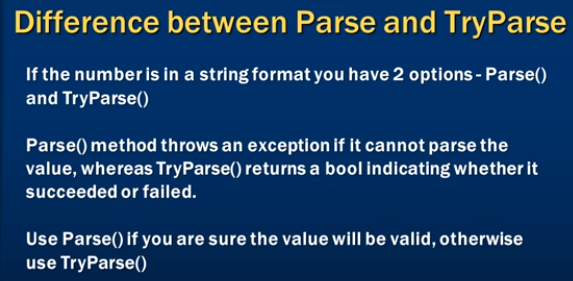


If float number is bigger, typecast operator doesn’t throw exception instaed it will hold the minimum value.

Where as convert.ToInt will throw an exception.

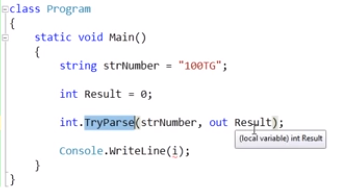




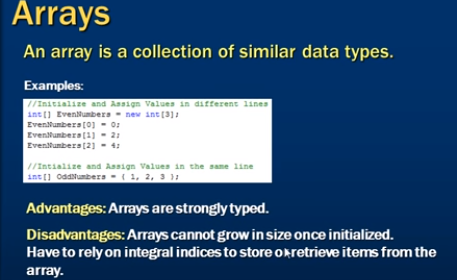


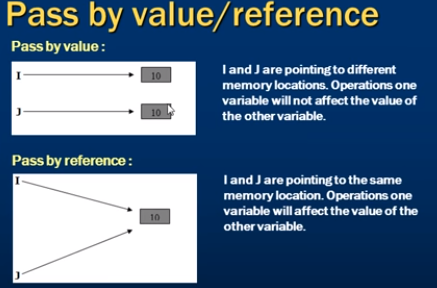
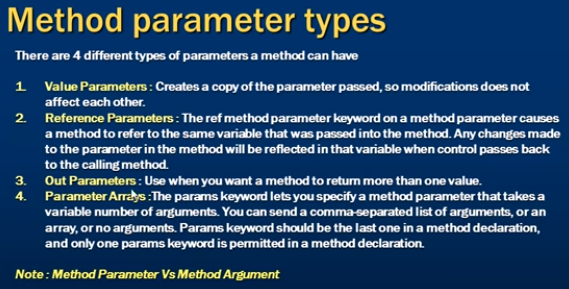


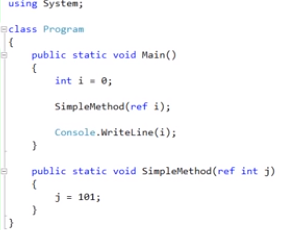
We will get format exception



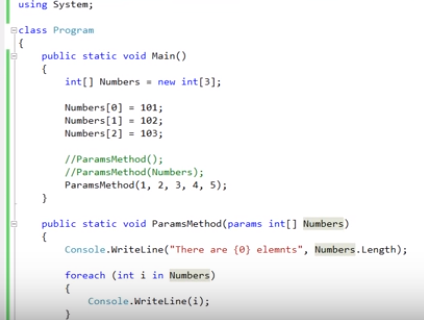
When it will not be able to convert to int it will return zero.









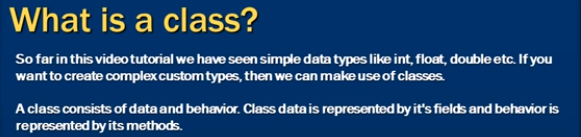


Difference between arguments and parameters

When we declare a method and pass parameters that is – parameters

While when we invoke value to method parameters – arguments

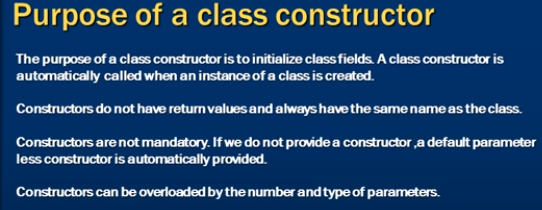


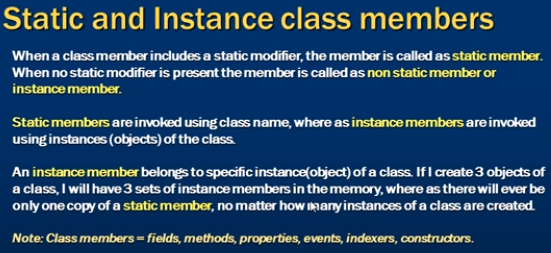


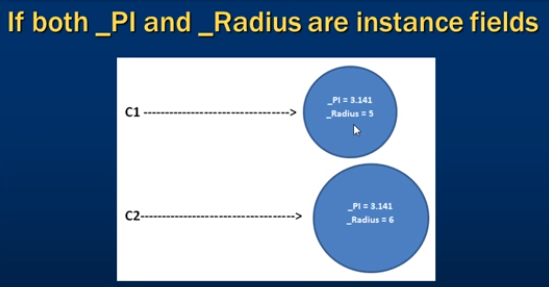
1. Constructor used to initialize
2. Don’t gave return type
3. Have parameters
4. Destructors are used to clean up all the code resources
5. Don’t have parameters or return type
6. Destructors are automatically called by garbage collectors

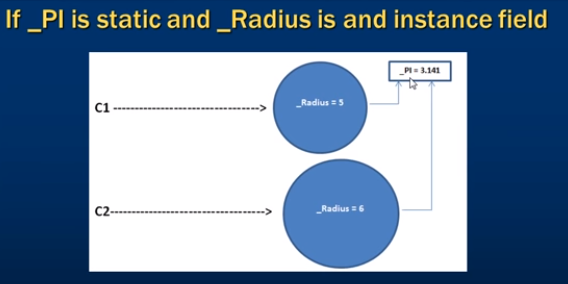


This keyword is used to refer to fields of a class

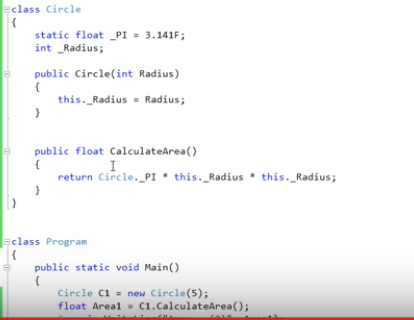


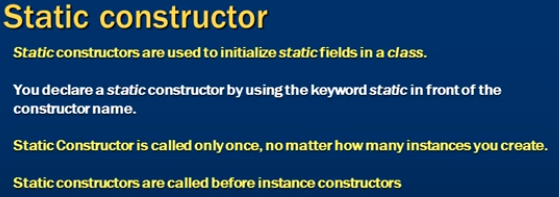


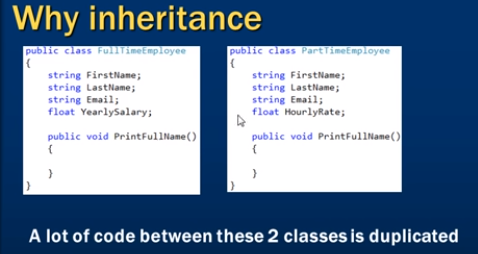


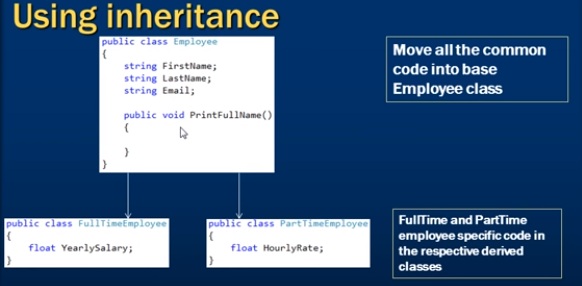


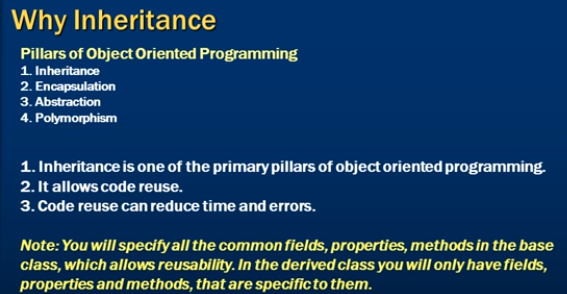
If we make pi as static it will be shared among all the object of the class and will occupy one place in memory instead of create multiple pi objects

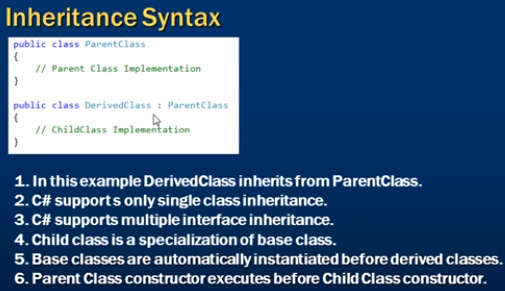


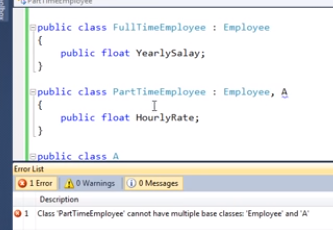




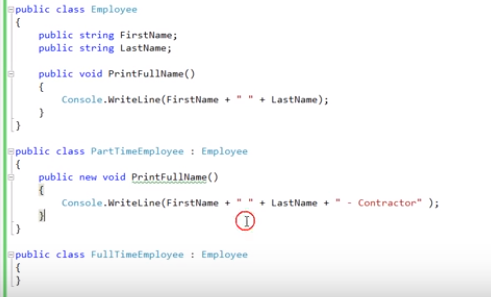




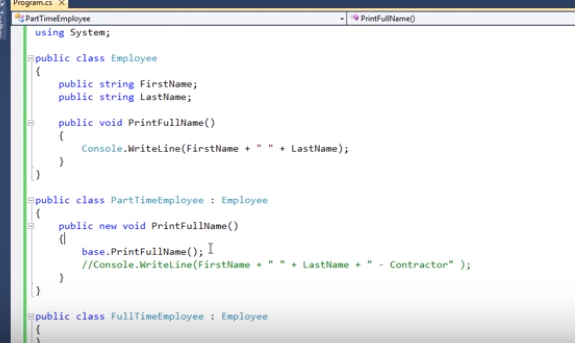




1. When we want to hide base class method

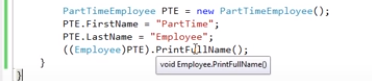


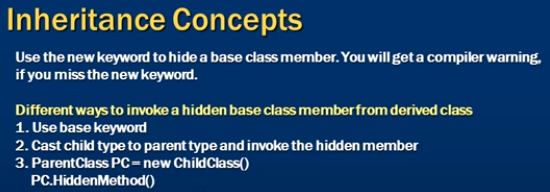
1. When we want to show base class method

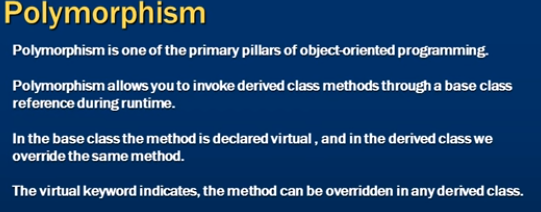


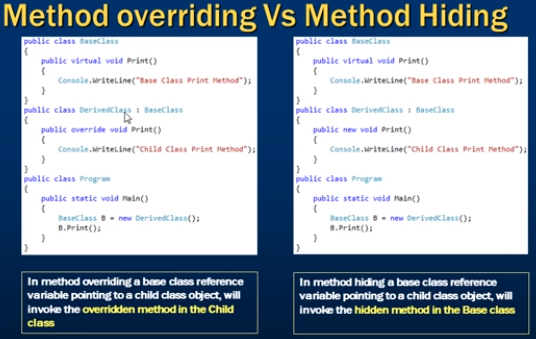
Or

Use type caste object







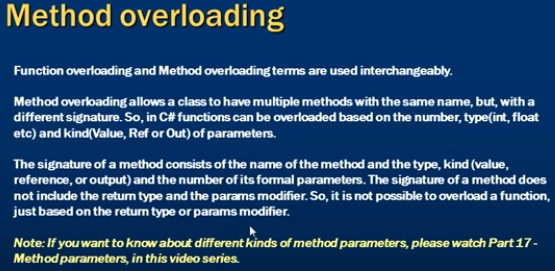


Difference:

A base class reference variable pointing to child class object, in method overriding will call the child class implementation

Where as

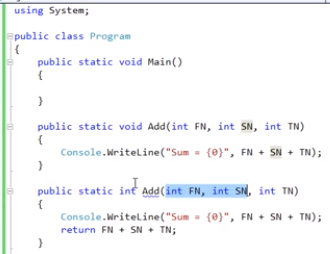
A base class reference variable pointing to child class object , in method hinding will call the base class method.



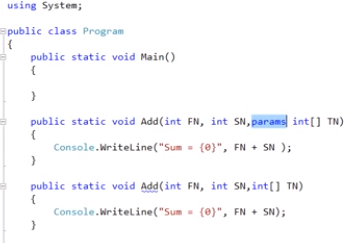
Method Overload:

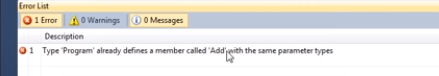
1. Name,
2. Type,
3. Kind
4. Number

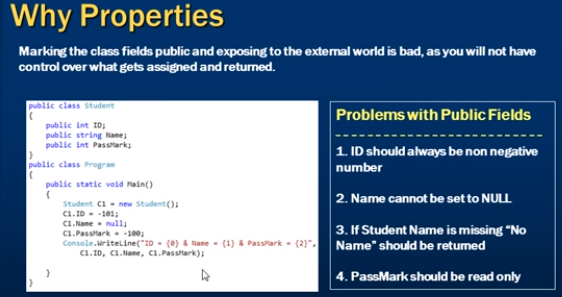
**We cannot overload a method with return type and params kind of parameter.**



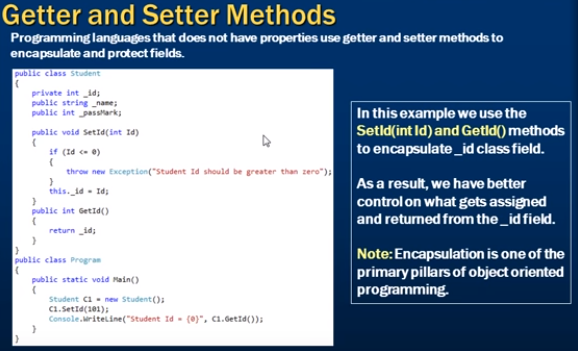
**Will get compile time error**







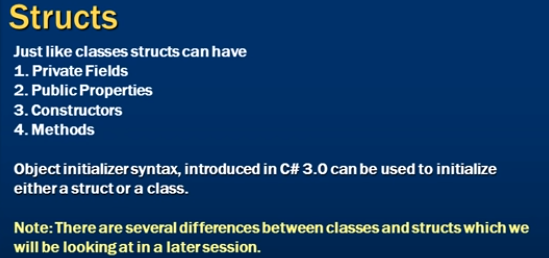
**For business point this is not correct.**

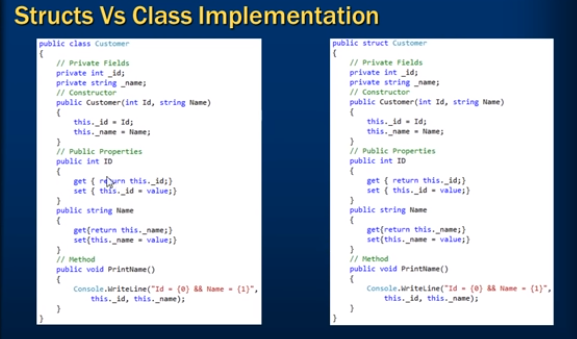


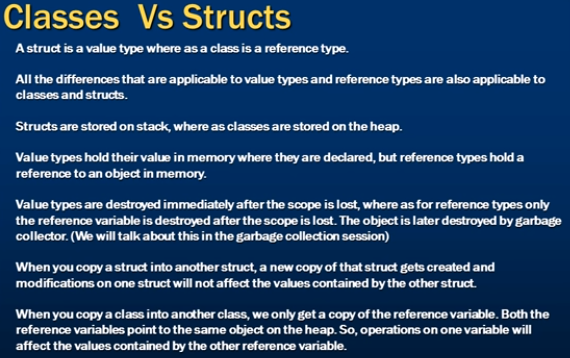
**Getter and setters are protecting fields.**

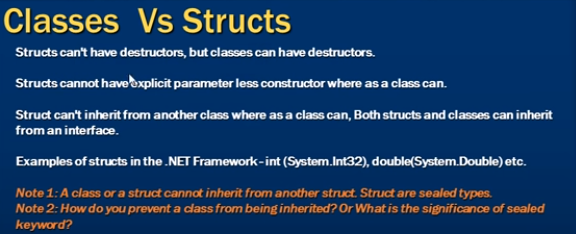




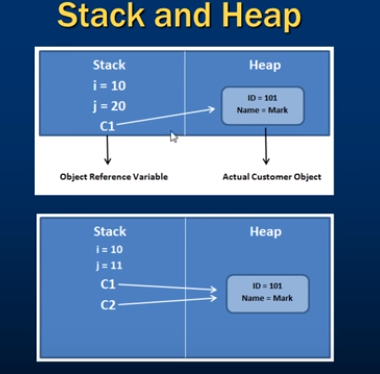








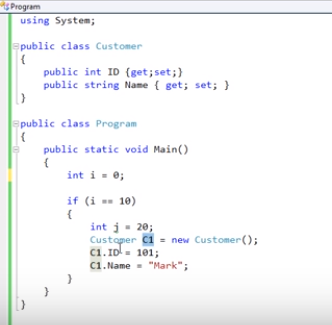
**Whatever variable we create those values are stored into computer memory i.e value type and reference type.**

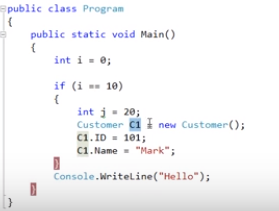


**C1 reference get created on stack and actual value is created on heap.**

**Value types are stored on stack whereas reference are stored on heap,**

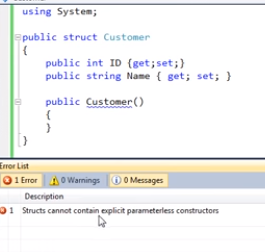
**Here, c1 is reference stored on stack which points to object which is stored on heap**





**j will be destroyed just after if block as well as C1 also.**

**When we create copy of reference it just create another reference but point to same object value.**

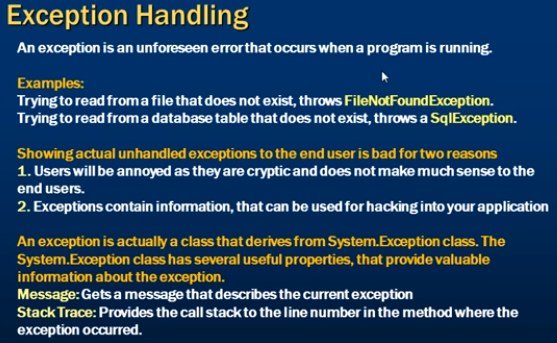


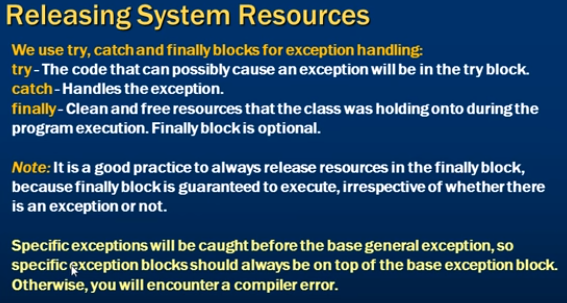
**A struct cannot have explicit parameterless constructor.**

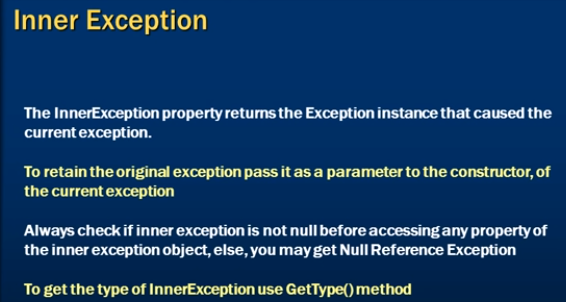
**A struct doect have concept of object that why don’t have destructor and patrmeterless.**

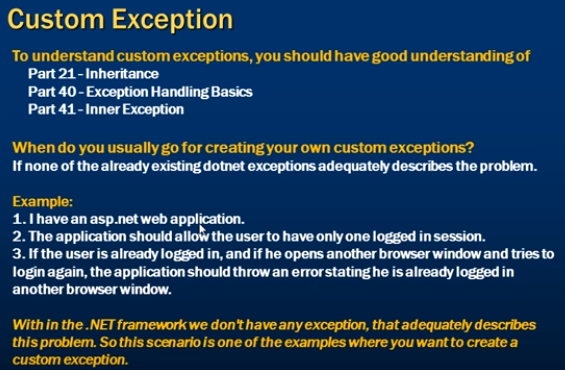
**Why we use STRUCT**

<https://stackoverflow.com/questions/1216993/why-do-we-need-struct-c>









**Allow us to track original exceptions.**

**When we have to move objects from one application domain to another application domain over network, those objects have to be serializable.**

