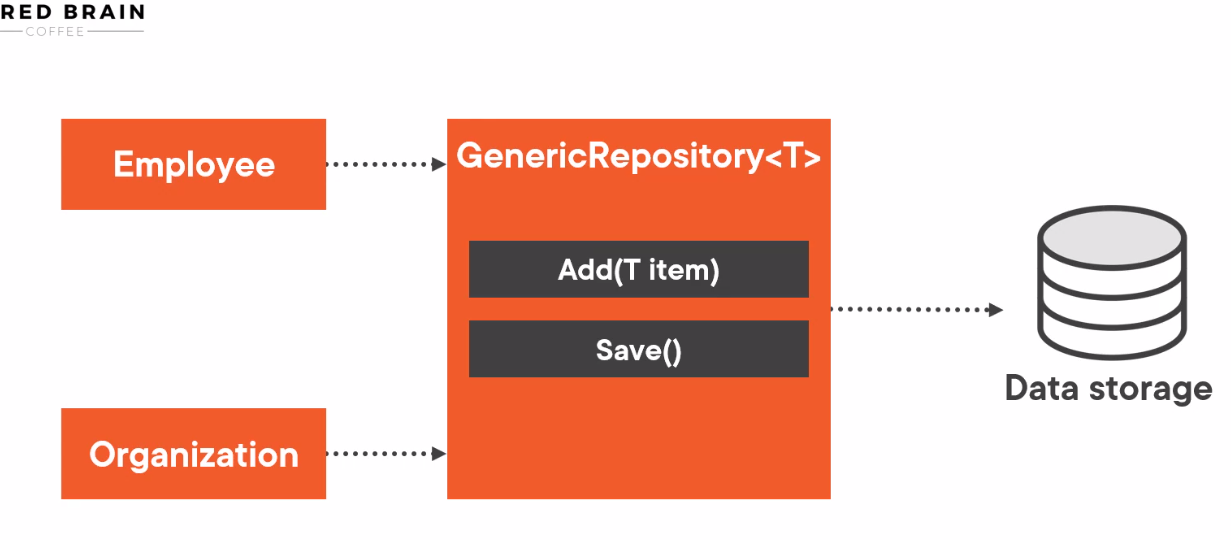
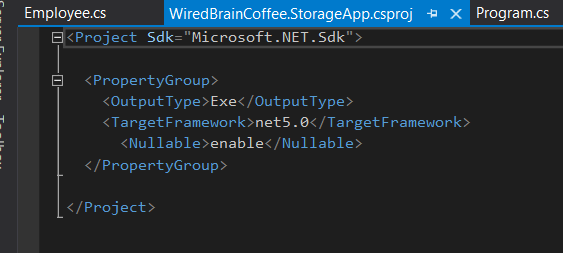
## Understanding the scenario



## Create a .NET console application

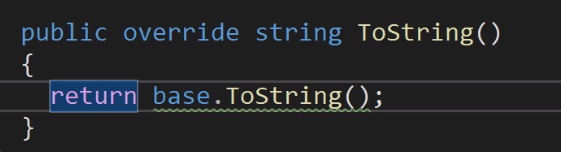
This enables Nullable reference types, you will get additional compiler warning that help you to avoid null reference exceptions



Where we use this first name property the C# compiler know now that this property can be null. And give u some hints to avoid null reference Exceptions



Next we override the to String Method. But instead of this we use



This



For C#9 we can omite the second part of the initialization

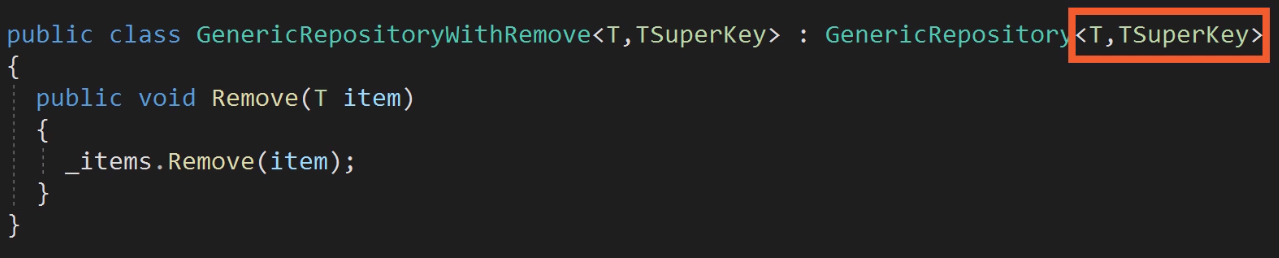


## Use Multiple Type Parameters

To declare a second type parameter, you edit it like this

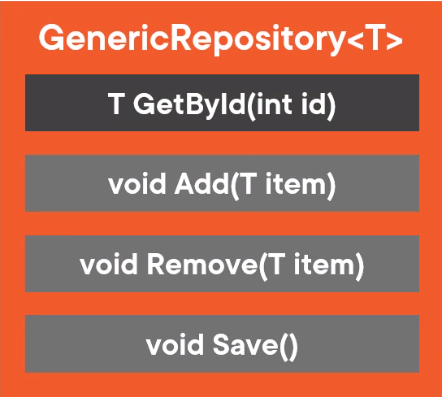


This means now when Generic Repository with remove class is created the two types are passed to the base class

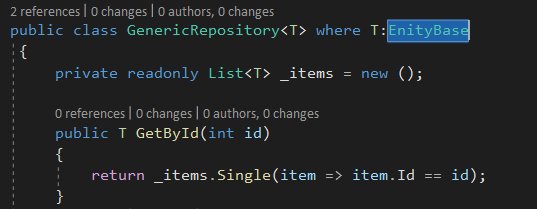


## Add a Generic Type Constraint \*

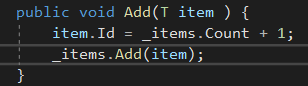
Let’s Implement a GetById method



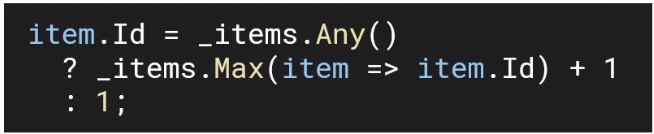
T is now of type EntityBase therefore we can now use its properties . because item is now of type entity base we can now use its id property



Now we set the id of an item when it is set in a repository



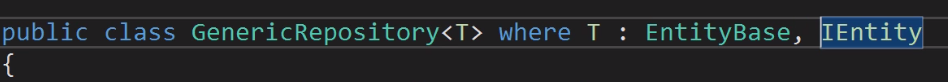
When you add and remove item this can result to duplicate id, I better approach would be



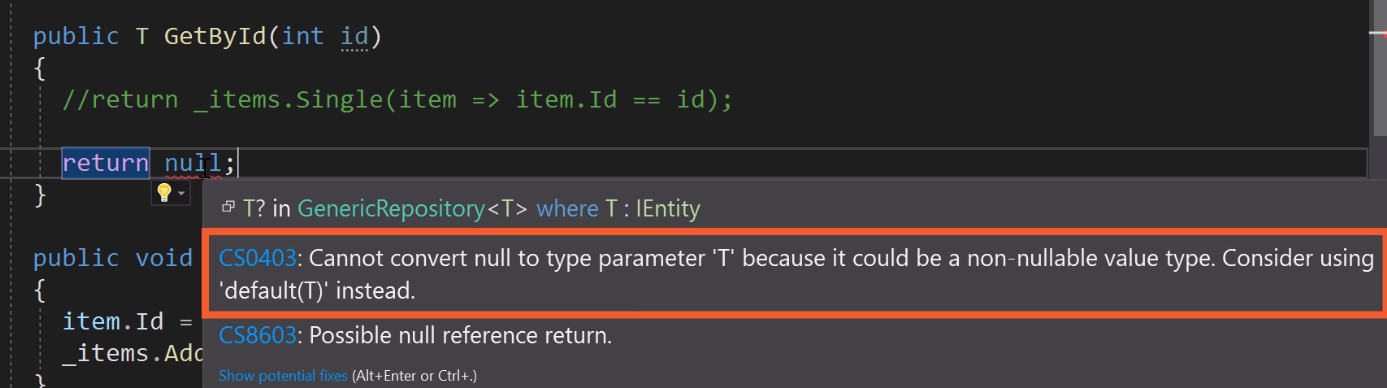
Type Constrain, because Employee and Organization inherit from Entity base only those two type can be define as generic type or type parameter T

## Work with the Class Constraint

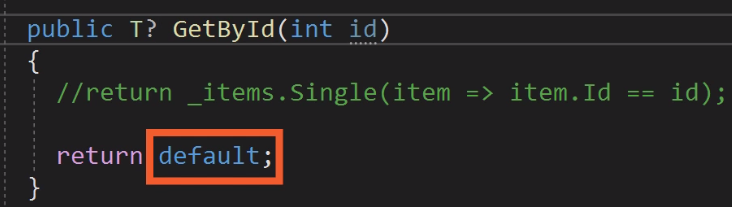
If only EntityBase is defined this means that it can only be a reference type , but if IEntity is defined this can be an IEntity type of a reference or value type

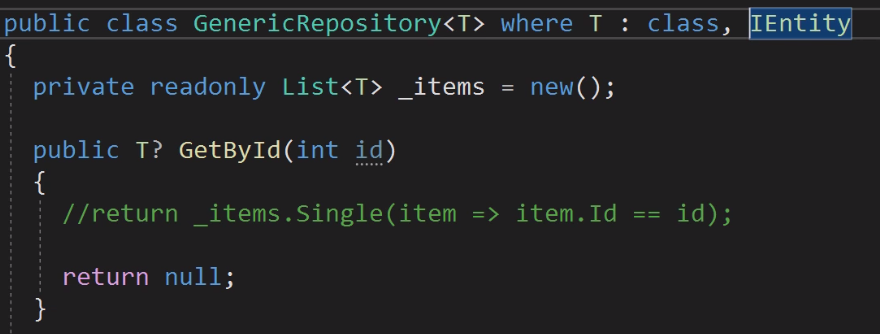


Now for example we cant return null it must be a type T



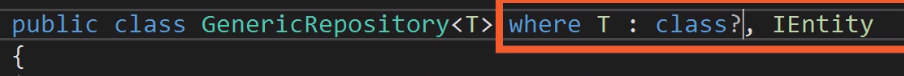
So, we can return a default (T) like shown below and at T? since reference type can be null



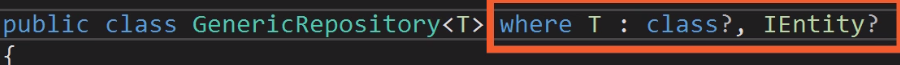
To specifically use null , you can specify class constrain that says that T is a reference type so it can actually be null

Because we enable reference type this means now that t is , by default, not nullable

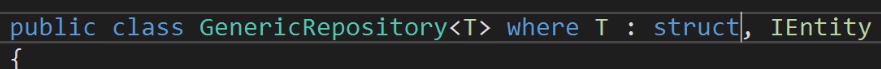
We can add? like below to say that T can be a nullable reference type



But also add ? to the interface to say that it can be a nullable IEntity



Struct to say that T must be a value type



System.Enum to say that T is an Enum

Class says T should be a class

## Use the New () Constraint \*

TKey needs to be a sub type of T

