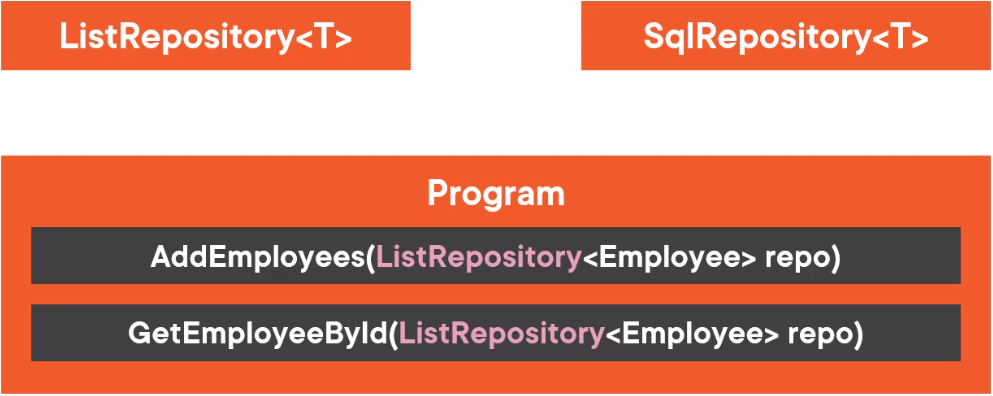
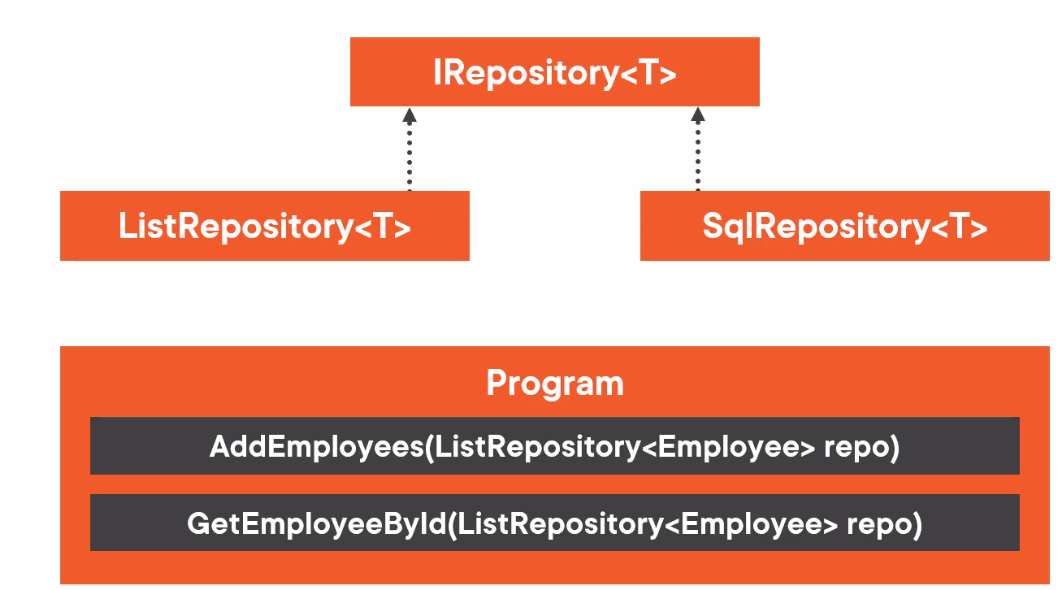
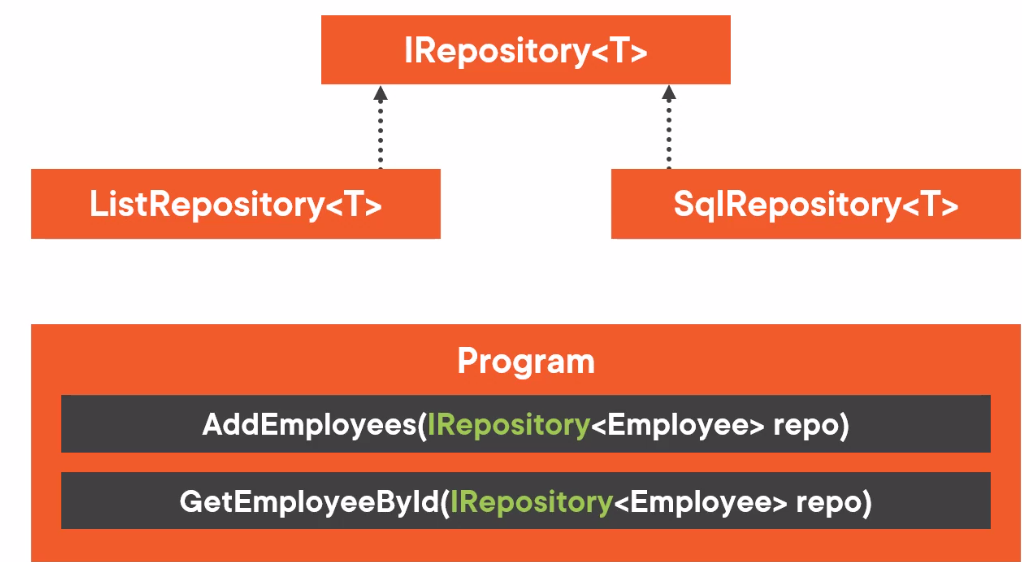
## Why use a generic Interface?

AddEmployee and GeEmployeeById are methods that have a ListRepository parameter. This means that you cannot pass a SqlRepository instance to these methods



There for we need to implement IRepositry<T> interface



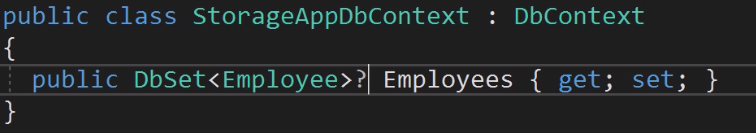
When implemented it can work with both, this is a software principle that we used here and it called Dependency inversion principle.

Dependency inversion principle.



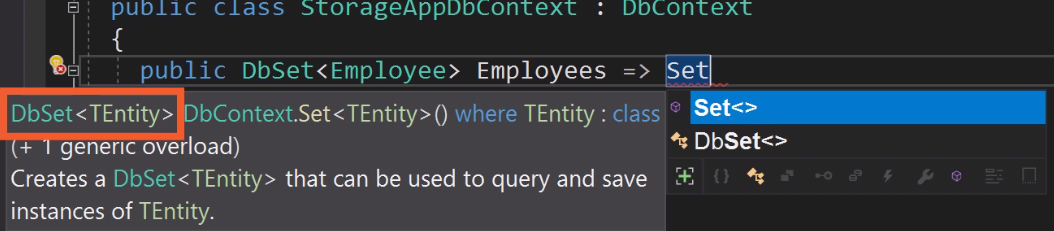
## Build a SqlRepository<T> Class \*

On the DbContext the DbSet is usually not null as it is initialized in the constructor of the DbContext base class.

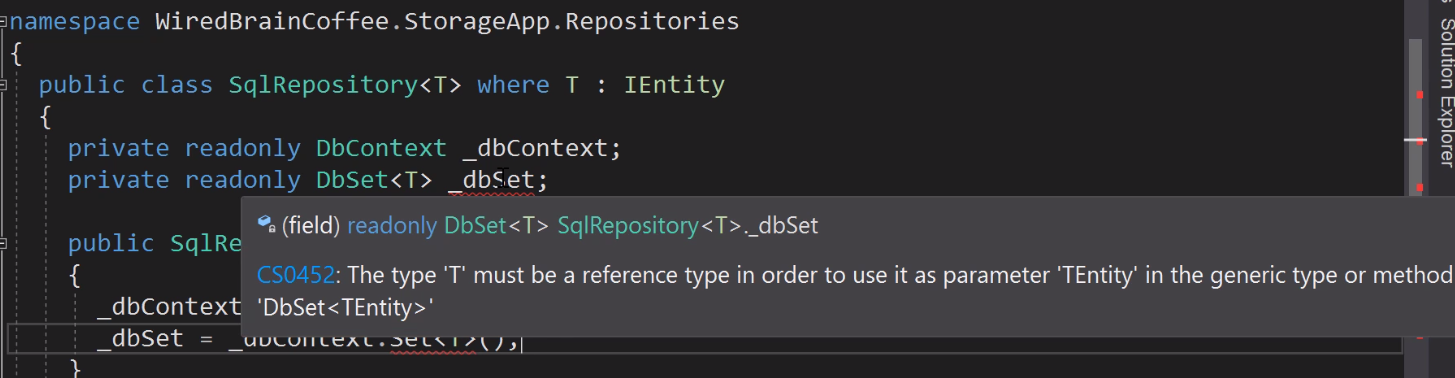


Instead of using? we can use an expression body and use the Generic set Method that is defined in the DbContext base class

And this method returns DbSet of <TEntity>



Since type T must be a reference type we must define the generic type constrain as class



## Create a Generic Interface

When you define type constraints like here

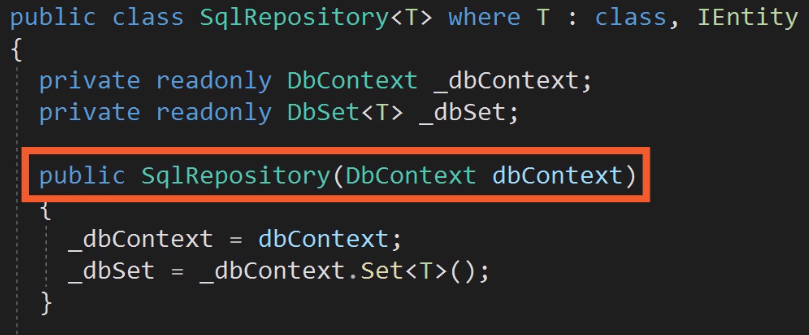


It means that generic classes that implement this interface also must define these types constrains for their generic type parameter



## Conclusion

Now I have implemented a generic SqlRepository<T> class that uses an Entity framework DbContext to store the items.



## Definitions

Entity: a thing with distinct and independent existence.