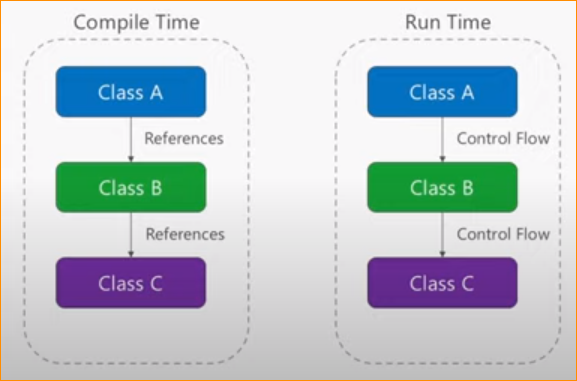
**Dependency injection-**

1. Tightly Coupled
2. Loosely Coupled

Compile time – This is the phase where the source code is converted into machine code or bytecode by a compiler.

Run time - This is the phase when the compiled code is executed by the computer's processor.

**Direct Dependency Graph**



* In Compile time -

Class A – Reference by Class B

Class B – Reference by Class C

If ones missing mean show error and compile time error. Why because of Directly depend on each other.

Can not survive each classes without each other.

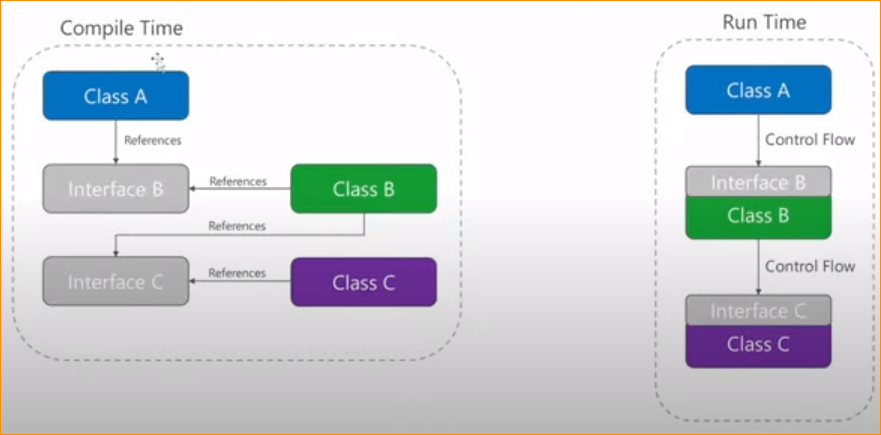
* In run time

If there missing reference mean show error. Because flow connect each other.

So in this both each other have connect and have reference.

**Tightly couple** – without Class A or Class B or Class C. Program not work. Class bind each other.

**Inverted Dependency Graph**



* Compile Time -

In here that Class A reference by Interface B.

Interface implement by class B

Advantage - Class A only depend on Interface B , Class B only depend on Interface C.

If Class B or Class C missing?

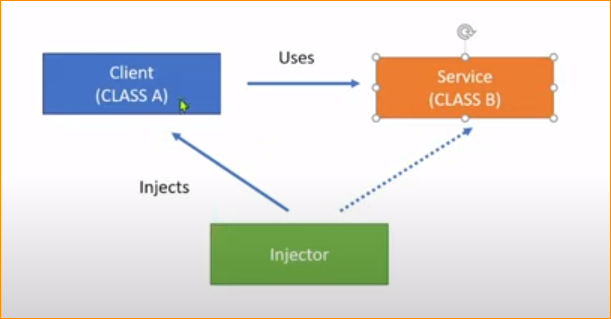
No worry – program work. Because class connect with the interface.

* Run Time -

If we call Interface B from Class A , Class B inject methods to the interface. Because Interface b is implement in Class B.

So in here to put Interfaces, We use Container - call IOC Container, DI container (Dependency injection)

* **How working Dependency injection**



In here we should have 3 Classes For dependency injection.

1. Client Class (Class A)
2. Service Class(Class B)
3. Injector Class

* Client (Class A ) using from server(Class B)
* Injector class – making an instance on Class B inject them Class A
* That means Class A not Directly depend on Class B
* So every time the injector class, creates an instance and injects to Class A.
* That is what happens dependency injection.

Client – Dependent

Type of Dependency injection

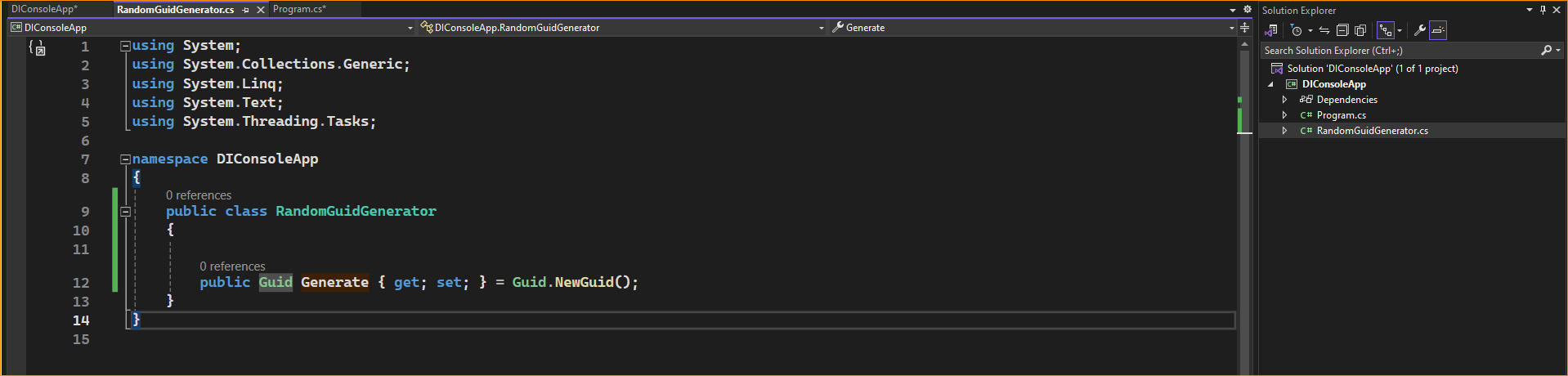
* Constructor Injection (Inject to Class A contractor)
* Property Injection (Inject to Class A Property)
* Method Injection (Inject to Class A Method )

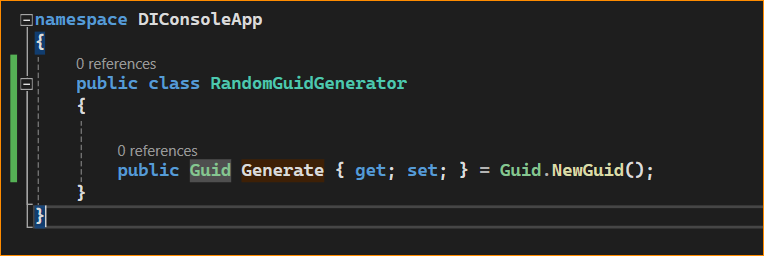
**IOC Containers**- Third-party libraries to help Dependency injection.

* Unity
* StrctureMap
* Castle Windsor
* Ninject
* Autofac
* Dryloc
* Simple Injector’
* Light inject

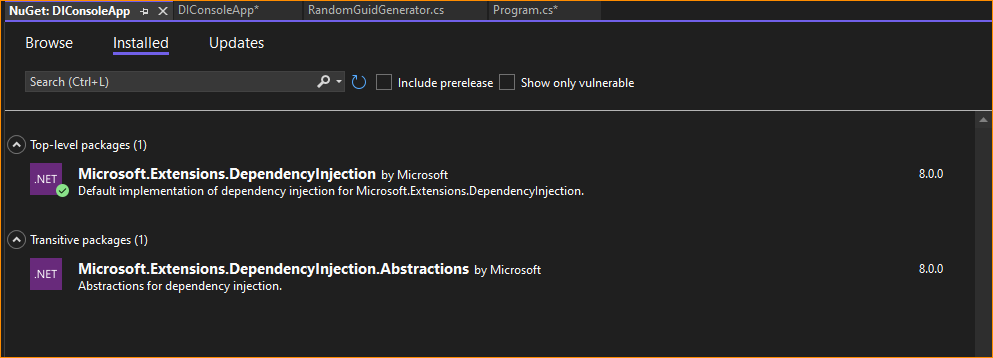
**How to do Dependency Injection**

I have created Console Application. Then Create New Class and make property.





Add Dependency injection nugget packages –

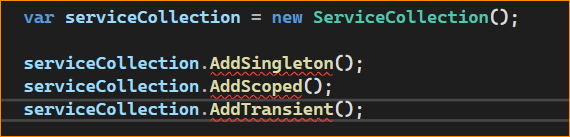


Now add the container –

Service-Collection Container

All collect in to that container , So on we can inject one by one.

There are 3 type can Add the Class..



1. AddSingleton()

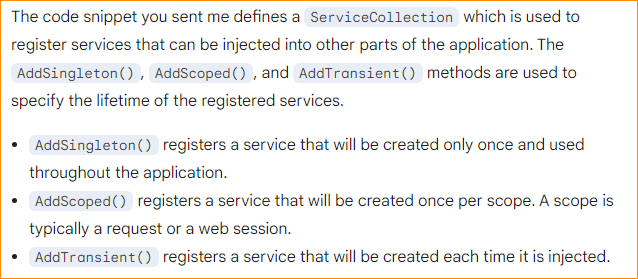
If we put class in to this, there is a One instance in the all application. There is one implementation.

1. AddScoped()

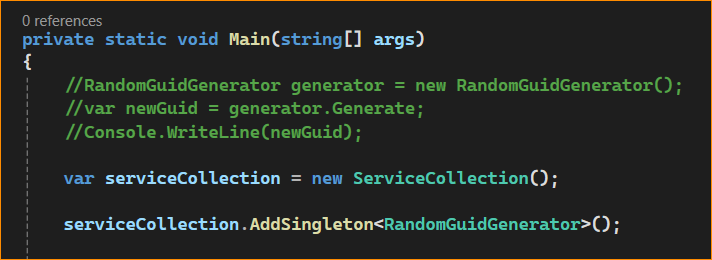
If we put class into this, there are created instances per request.

1. AddTransient()

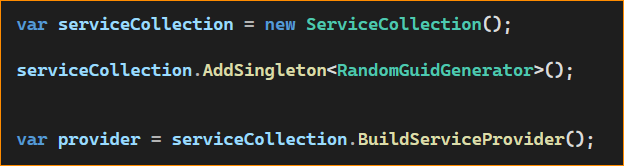
There is a new class create every time per inject.



* **AddSingleton()**
* Add the “RandomGuildGenerator “ Class

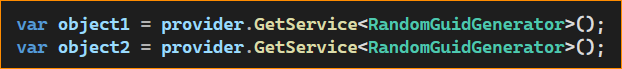


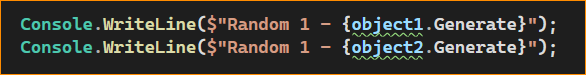
Then is you want to get that class – you should get **BuildServiceProvider();**



Give me the type of “RandomGuidGenerator” instance.

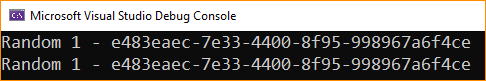
Give me another one.



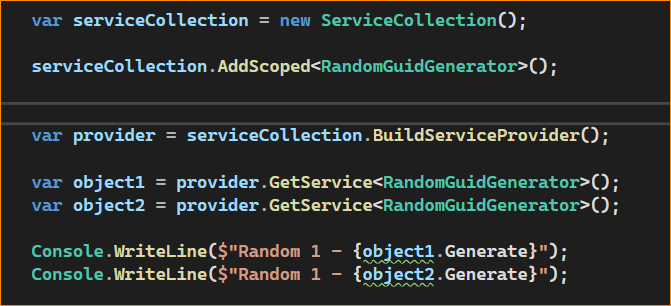


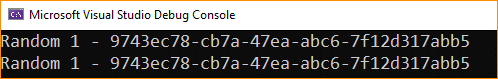
There is a two instance –

But Result –



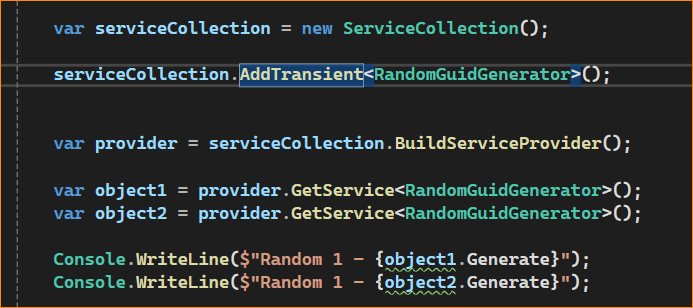
* **AddScoped()**

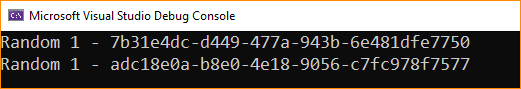




If you consider API, that mean it provide multiple guid

* **AddTransient**





In here , consider as a new object.