**Lesson05 Dependency Inversion Principle**

**Notes:-**

**1-Dependency inversion principle say that the upper part of system should not depend on the lower parts of the system directly but with some kind of abstraction**

**(By pass the interface IRelationshipBrowserRepo as parameter on the ResearchManager we apply the DI by create an instance of the IRelationshipBrowserRepo which is RelationshipsRepo)**

**Steps:-**

**1-create Enums > Relationship.cs**

**public enum Relationship{Parent,Child,Sibling}**

**2-create Models > Person.cs**

**public class Person{public string Name;}**

**3-create Interfaces > IRelationshipBrowserRepo.cs**

**public interface IRelationshipBrowserRepo{**

**IEnumerable<Person> FindAllChildrenOf(string name);}**

**4-create Repositories > RelationshipsRepo.cs**

**public class RelationshipsRepo : IRelationshipBrowserRepo // low-level{**

**private List<(Person, Relationship, Person)> relations**

**= new List<(Person, Relationship, Person)>();**

**public void AddParentAndChild(Person parent, Person child){**

**relations.Add((parent, Relationship.Parent, child));**

**relations.Add((child, Relationship.Child, parent));}**

**public List<(Person, Relationship, Person)> Relations {**

**get{return relations;}}**

**public IEnumerable<Person> FindAllChildrenOf(string name){**

**return relations.Where(x => x.Item1.Name == name**

**&& x.Item2 == Relationship.Parent).Select(r => r.Item3);}}**

**5-create Manager > ResearchManager.cs**

**public class ResearchManager{**

**public ResearchManager(IRelationshipBrowserRepo browser){**

**foreach (var p in browser.FindAllChildrenOf("John")){**

**WriteLine($"John has a child called {p.Name}");}}}**

**6-on the main Program.cs we set the following code by create instance of the RelationshipsRepo and pass it into the ResearchManager which is accept any type that inherirt from the IRelationshipBrowserRepo**

**static void Main(string[] args){**

**var parent = new Person { Name = "John" };**

**var child1 = new Person { Name = "Chris" };**

**var child2 = new Person { Name = "Matt" };**

**// low-level module**

**var relationships = new RelationshipsRepo();**

**relationships.AddParentAndChild(parent, child1);**

**relationships.AddParentAndChild(parent, child2);**

**new ResearchManager(relationships);**

**ReadLine();}**