**Principio de Inversión de Dependencia (DIP) -> Los modulos de alto nivel no deben depender de los modulos de bajo nivel**

**Ejercicio1**

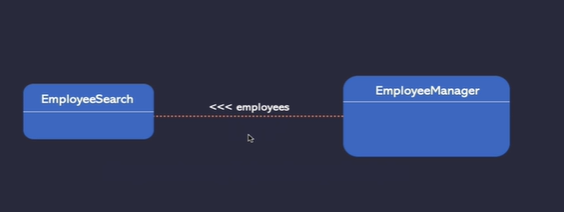
**BEFORE**

Vamos a realizar una aplicación donde se ingrese empleados de una compañía, para ello tendremos una clase **Employee** que tendrá un **Name** un **Género**(enum: Male, Female) y un **Role**(enum: Manager, Executive, Developer), a su vez tendremos una clase **EmployeeManager** la cual tendrá una propiedad como **lista de empleados** y el metodo **Save(para guardar empleados)**, luego tendremos una clase **EmployeeSearch(la cual se encargara de buscar empleados).**

En nuestra clase **Program** primero almacenaremos nuestros empleados creados en una lista de **Employee**, luego llamaremos a nuestra clase **EmployeeManager** para almacenar nuestros empleados creados. Luego utilizaremos nuestro **EmployeeSearch** para imprimir por consola.

En nuestro metodo **EmployeeSearch** implementaremos un método que nos devuelva los empleados de sexo masculino y de rol ejecutivo.

Debemos realizar esta dependencia:



**EmployeeSearch** -> un modulo de alto nivel

**EmployeeManager** -> una modulo de bajo nivel(solo realiza el guardado)

Según el principio, los modulos de alto nivel no deben depender de un módulo de bajo nivel a no ser que sea una **abstracción**.

Por lo que cualquier cambio que se realice en mi clase **EmployeeManager** afectara directamente a la clase **EmployeeSearch.**

**AFTER**

Refactorizaremos el código solucionando dicha dependencia entre las dos clases, **desacoplando su nivel de dependencia.**

Como vemos la clase **EmployeeSearch**  está dependiendo de una implementación directamente no de una abstracción, por ello crearemos una interfaz que haga la conexión entre estas dos clases.

Procedemos entonces a crear un interfaz **IEmployeeSearcheable** que tendrá el metodo: **GetEmployeesByGenderAndRole,** luego se realizara la modificación de la clase **EmployeeManager** que implementara dicho metodo de la interfaz, luego realizar la modificación de la clase **EmployeeSearch** en cuanto a la dependencia ya que ahora será porpiamente de la interfaz **IEmployeeSearcheable.**

**Ejercicio 2**

**BEFORE**

Implementar una clase **Employee** que recibe un parámetro de **correo** y **sms** en su constructor y tiene un metodo **Send** que enviara el email y el sms, para ello tenemos una clase **SMS**(con un **PhoneNumber** y un **Message** y el metodo **SendSMS**) y una clase **Email**(que tendrá **Subject**, **Content** y el metodo **SendEmail**), la idea es que la clase **Employee** non tenga una dependencia de la clase **Email** y la clase **SMS**

**AFTER**

Como vemos tenemos dependencias directas hacia implementaciones como **Email** y **SMS** al momento de generar mi clase **Employee**, por lo que debería depender de abstracciones no de implementaciones, como vemos las clases **SMS**  y **Email** tienen un metodo en común **Send**, por lo que se podría crear una interfaz en función de dicho metodo, la interfaz se llamara **IMessage** con dicho metodo y hacemos la implementación propia para cada clase **SMS**  y **Email** y hacemos la modificación propia en la clase **Employee** modificando la dependencia por una lista de **IMessage.**