目標 : 解耦

(解除 高階模組 (Caller 呼叫者) 與 低階模組 (Callee 被呼叫者)的 耦合關係  
使高階模組不再直接依賴低階模組。 』

實例優點 : 後續若需要添加第三種茶 可直接新增 不需用if else 判斷新增

Main

{

**BeverageStores** greenStore = new **BeverageStores(**new GreenTeaFactory());

Console.WriteLine("A 客人點了綠茶");

greenStore.BeverageOrders();

**BeverageStores** blackStore = new **BeverageStores**(new BlackTeaFactory());

Console.WriteLine("B 客人點了紅茶");

blackStore.BeverageOrders();

}

**public class BeverageStores**

**{**

**private** IBeverageFactory **\_factory;**

**public BeverageStores(**IBeverageFactory **pFactory)**

**{**

**\_factory = pFactory;**

**}**

**public** IBeverageProvide **BeverageOrders()**

**{**

IBeverageProvide **beverage;**

**// 用IBeverageProvide 產生 IBeverageProvide 實例**

**beverage = \_factory.CreateBeverage();**

**beverage.AddMaterial(); // 加料**

**beverage.Brew(); // 沖泡**

**beverage.PouredCup(); // 裝杯**

**return beverage;**

**}**

**}**

//////////////// 兩個 Interface ///////////////

public interface IBeverageFactory

{

IBeverageProvide CreateBeverage();

}

public interface IBeverageProvide

{

void AddMaterial();

void Brew();

void PouredCup();

}

////// Class 分別對應到的 Interface /////////////////////

class GreenTeaFactory : IBeverageFactory

{

public IBeverageProvide CreateBeverage()

{

return new GreenTea();

}

}

class BlackTeaFactory : IBeverageFactory

{

public IBeverageProvide CreateBeverage()

{

return new BlackTea();

}

}

public class GreenTea : IBeverageProvide

{

public void AddMaterial()

{

Console.WriteLine("GreenTea AddMaterial");

}

public void Brew()

{

Console.WriteLine("GreenTea Brew");

}

public void PouredCup()

{

Console.WriteLine("GreenTea PouredCup");

}

}

public class BlackTea : IBeverageProvide

{

public void AddMaterial()

{

Console.WriteLine("BlackTea AddMaterial");

}

public void Brew()

{

Console.WriteLine("BlackTea Brew");

}

public void PouredCup()

{

Console.WriteLine("BlackTea PouredCup");

}

}

