C# Summery part-1 By Mahmoud Badawy

-----------------------------------------------------

الملخص يحتوى على

- Design Patterns

المستخدمه فى الشرح   
<https://refactoring.guru/design-patterns/singleton>

MB

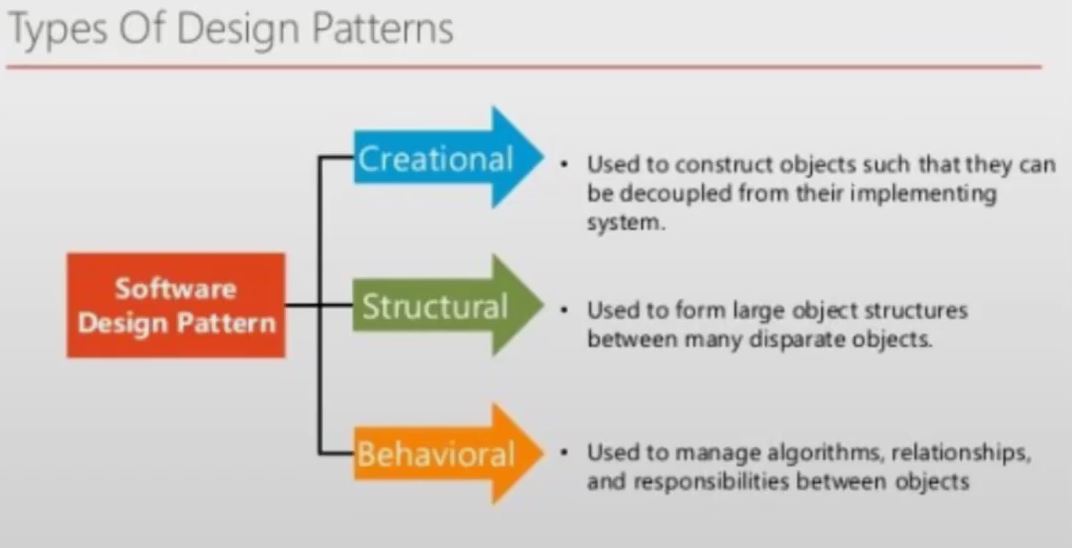
##################################################################

Design Pattern Importance

--------------------------------

الناس لما عملت ال OOP و ابتدوا يشتغلوا بيها و يعملوا مشاريع كبيره ابتدت تظهر مشاكل سواء فى ال Creation لل instance او سلوكه او طريقه تعامله مع الانستانس المختلفه عنه و ده الى خلاهم فكروا يعملوا حلول باستاندرد محترم يخلى الكود نضيف و فى نفس الوقت يحل المشكله الى بيوجهوها .و هى دى اهميه ال Design Pattern

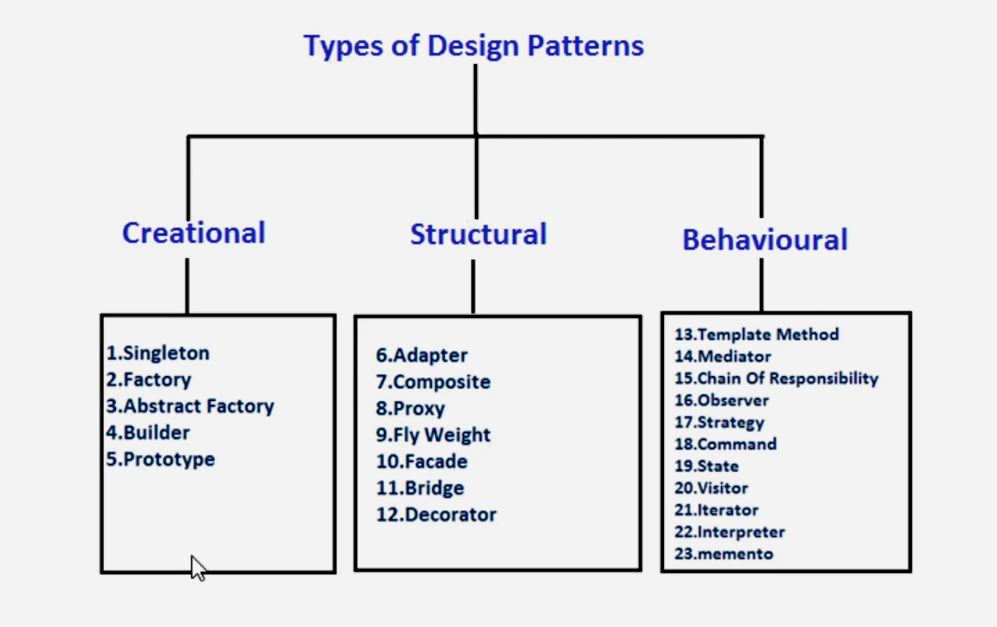
فى تلات انواع للديزاين باترن



ال creational : ده المسؤول عن انشاء ال objects .

ال Structural : ده المسؤول عن العلاقه بين ال objects المختلفه .

ال Behavioral : ده المسؤول عن السلوك الى بيحصل بين ال objects دى الى هو بيبقى سبب فى نقل البيانات بينهم ذى مثلا ال events , Algorithms , states



MB

##################################################################

Singleton DP

--------------------------------

قبل ما يعملوا ال singleton كان الواحد ممكن يعمل كلاس ما و يبتدى يستخدمه فيعمل منه new instance ثم فى مكان تانى يعمل new instance تانى و هكذا لحد ما ممكن يكون بيعمل ملايين ال instances و المشكله ان كل دول محفوظين فى الميمورى و واخدين مساحه كبيره منها و مش بيتمسحم و بالتالى الريسوريس بتاعت الجهاز بيتم استهلاكها بدون داعى و من هنا طلع فكره ال Singleton علشان تحل المشكله دى



ال Singleton بقه فكرتها ان انا هعمل only One Object من الكلاس و بالتالى هيبقى استهلاك الميمورى قلبل جدا جدا و بالتالى الكود هيبقى سريع جدا .

امتى استخدم ال Singleton بامثله من الحياه العمليه ؟  
1- لما اكون عاوز اضمن ان الكلاس ده محدش ينادى للكونستراكتور بتاعه مره واحده فقط على مستوى الابلكيشن

2- لما اكون بتعامل مع static Data بيانات مش بتتغير كتير او مش بتتغير اصلا و كل فتره كبير جدا على ما اعملها ريفريش او حتى انها تكون من البيانات الى لو غيرتها لازم اعمل ريستارت للابلكيشن علشان يحس بالتغيير ده ذى مثلا ال Configurations اهه دى بتبقى فى Json file او Xml File او DataBase و مش بحتاج اقراها على مستوى الابلكيشن الا مره واحده فقط و اسجلها على الميمورى كاش و استخدمها بمزاجى طول ما انا مشغل الابلكيشن لو غيرت فيها بحتاج اعمل ريسترت للابلكيشن علشان يحس بيها

3- اللوك ابس LookUps الى هى فى الغالب بتبقى static Records محتاجها للسيستم و تغييرها بيبقى قليل جدا بس بنحتاج نعملها لود كل مره بخش فيها على السيستم فبرضوا بدل ما كل مره اروح اجيبها فانا بجيبها مره واحده فى ال startup و اكيشها فى الميمورى برضوا و استخدمها جوه السيستم من السينجلتون بقه بدلا من انى كل شويه اكلم الداتا بيز اقولها هاتيلى اللوك ابس دى و ابقى كل شويه كونيكشن للداتا بيز على حاجه مش بتتغير كتير اساسا

**طل ما كل الكلام الجميل ده اقدر اعمله ب Static Class ايه بقه الى يخلينى اعمله ب singleton ؟**  
علشان ال Static Class بتفرض عليا شويه limitions ذى مثلا

فى ال **static Class** :

1. لا يمكن انى اطبق مبادئ ال OOP ذى ال inheritance مثلا
2. لا يمكن انى اعمل implemention لاى interface جواها
3. مقدرش استعملها ك method Parametars

كل ال Limitions دى ال Singleton بيحلها

تعالى بقه اوريك المشكله قبل ال SingleTon و الحل لما طبقناه عمل ايه

MB

##################################################################

Before Singleton DP

--------------------------------

المشكله قبل ال singleton هو انى عندى كلاس و كل اما اعمل new instance منه بيروح يخزن ال new instance ده فى الميمورى و للاسف مبيكونش مسح القدام و بالتالى كده انا بعمل instances كتير جدا جدا من class واحد و ممكن مبقاش محتاج كل ده و بالتالى ده استهلاك لموارد الكمبيوتر بشكل كبير و هيبطء البرنامج الى انا بعمله جدا و ده الكود الى بيوضح الفكره

using System;

namespace Before

{

internal class Program

{

static void Main(string[] args)

{

// with no Singleton it will create New 5 Instance in Memory

Console.WriteLine($"########################");

Console.WriteLine($"------------------------");

Counter ObjA = new Counter();

Counter ObjB = new Counter();

Counter ObjC = new Counter();

Counter ObjD = new Counter();

Counter ObjE = new Counter();

Console.WriteLine($"########################");

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

}

}

هنا انا بنادى على الكلاس 5 مرات و كل مره من ال 5 مرات دول بينشأ instance new من كلاس ال counter

using System;

using System.Collections.Generic;

using System.Text;

namespace Before

{

internal class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public Counter()

{

Count++;

NumberOfInstance++;

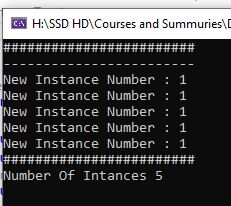
Console.WriteLine($"New Instance Number : " + Count);

}

}

}

و الدليل على انه بينشأ New instance كل مره بعمل منه object هى نتايج البرنامج و اهه علشان تبقى فاهم



**طيب انا بقه مش عاوز كده انا عاوز انى اعمل Creation لل instance مره واحده و فى اى وقت اناديله تانى يروح رايح واخد نفس ال instance و ميعملش غيرها و بالتالى انا كده هتعامل مع only One Instance و بالتالى استهلاك الميمورى هيكون اقل بكتير و بالتالى سرعه البرنامج هتبقى كبيره جدا خصوصا لما نستخدم الباترن ده مع الفانكشنز الى بتتعامل مع heavy Data**

MB

##################################################################

After Singleton DP

--------------------------------

اول حاجه هنحل المشكله الى فوق دى بال Singleton DP و ده بقه بيتعمل اذاى ؟؟؟

1- اول حاجه نخلى ال Constractor برايفيت Private و ده علشان فى ال Client side محدش يعرف يعمل new instance of class

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

2- ثم بعد كده هعرف جوه الكلاس بتاعى ال instance من نوع public static Counter ذى كده و هو ده هيبقى ال instance الوحيد الى بتعامل معاه علشان كده عملته Static يا ريس

public static Counter Instance = null;

3- المفروض بقه هعمل فانكشن كل شغلانتها انها بتشوف لو ال static instance ده ب null فتروح تملاه و تديله new instance اما لو مش ب null معنى كده انه فيه instance فكده ياباشا تروح قيلاله خد القديم استعمله يابا مش هعملك حاجه جديده

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

return Instance;

}

و الفانكشن دى اتعملت static علشان تبقى Shared و علشان اقدر اناديها باسم الكلاس و طبعا هى بترجع حاجه من نوع الكلاس Counter Class لان هى دى المسؤوله عن انشاء الاوبجيت اصلا او انها تديك الاوبجيكت القديم الى معمول اساسا بدلا من انها تعملك حاجه جديده فتهلك الميمورى

و بالتالى الكود بالكامل اهه

using System;

namespace After\_Simple\_Singleton

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########################");

Console.WriteLine($"------------------------");

Counter ObjA = Counter.GetInstance();

Counter ObjB = Counter.GetInstance();

Counter ObjC = Counter.GetInstance();

Counter ObjD = Counter.GetInstance();

Counter ObjE = Counter.GetInstance();

Console.WriteLine($"########################");

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

}

}

ثم الكلاس بقه

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Simple\_Singleton

{

internal class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

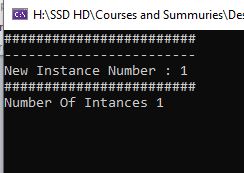
return Instance;

}

}

}

النتيجه اهه يا صديقى بقى مش بيعمل غير One Instance من الكلاس



MB

##################################################################

After Singleton DP But Using **Inhertance** (problem) solved By **Sealed** Key

----------------------------------------------------

بما انك عامل ال Constructor access modidier نوعها private فانت طول ما انت بره ال class و بتحاول تعمل inheritance مش هيرضى و ده المطلوب فعلا و لكن اول ما هتحط الكلاس دى جوه الكلاس بمعنى اول ما هتعمل علاقه وراثه بانك تحط كلاس ال child جوه كلاس ال parent ساعتها هيقبله عادى و هيعمل منه كمان مليون new instance عادى و ده بيضرب ال singleton pattern فى مقتل

و هنا هوريك بالكود ايه المقصود يا صديقى

using System;

using System.Collections.Generic;

using System.Text;

namespace Inhertance\_Problem\_Singleton

{

internal class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

return Instance;

}

//ChildCls inherite from Counter Class that will Make easy to create Million of instance by Child Class

public class ChildCls : Counter

{

}

}

}

و هنا هنادى للكلاس العادى الى معمول فيه singleton و هنادى لل Child هتلاحظ انى طول ما بنادى للعادى مفيش اى instance جديد تم انشاءه و لكن اول ما ابتديت انادى للكلاس ال child بقى يعمل new instance كتير جدا بقه

using System;

namespace Inhertance\_Problem\_Singleton

{

internal class Program

{

static void Main(string[] args)

{

// with no Singleton it will create New 5 Instance in Memory

Console.WriteLine($"########### Before using Child #############");

Console.WriteLine($"------------------------");

Counter ObjA = Counter.GetInstance();

Counter ObjB = Counter.GetInstance();

Counter ObjC = Counter.GetInstance();

Counter ObjD = Counter.GetInstance();

Counter ObjE = Counter.GetInstance();

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.WriteLine($"########### After usning Child #############");

Console.WriteLine($"------------------------");

Counter ChildObjA = new Counter.ChildCls();

Counter ChildObjB = new Counter.ChildCls();

Counter ChildObjC = new Counter.ChildCls();

Counter ChildObjD = new Counter.ChildCls();

Counter ChildObjE = new Counter.ChildCls();

Console.WriteLine($"########################");

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

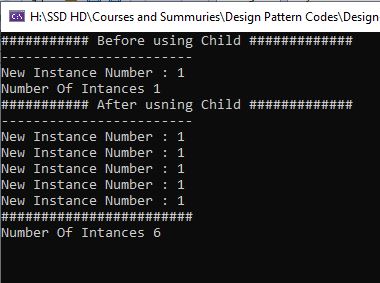
Console.ReadKey(true);

}

}

}

و ادى النتيجه علشان تفهم



**طب المشكله دى احلها اذاى ؟** احلها انى اخلى الكلاس الى بعمل عليه ال Singleton انه يبقى **Sealed** كلاس يعنى مبيورثش اى كلاس باى طريقه نهائى و بالتالى همنع الموضوع ده نهائيا و ابقى ضمنت انى هعمل only one instance تانى

using System;

using System.Collections.Generic;

using System.Text;

namespace Using\_Sealed\_ToSolve\_Inheretance\_Problem\_Singleton

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

}

return Instance;

}

//Sealed Will Stop any kind of Inhertance in Singleton Class

//public class ChildCls : Counter

//{

//}

}

}

كلاينت كلاس

using System;

namespace Using\_Sealed\_ToSolve\_Inheretance\_Problem\_Singleton

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########### With using Sealed class #############");

Console.WriteLine($"------------------------");

Counter ObjA = Counter.GetInstance();

Counter ObjB = Counter.GetInstance();

Counter ObjC = Counter.GetInstance();

Counter ObjD = Counter.GetInstance();

Counter ObjE = Counter.GetInstance();

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

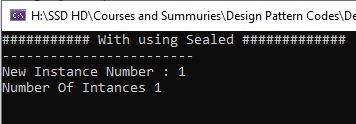
Console.ReadKey(true);

}

}

}

و بالتالى اصبح غير مسموح الوراثه فى الكلاس الى عامل فيه سنجلتون و بالتالى مبقاش ينفع يعمل اى new instance تانى يعنى اتقفلت تانى بالضبه و المفتاح و النتيجه هتكون بالشكل ده تانى



MB

##################################################################

After Singleton DP But Using **Multi Threads** (problem) solved By **Lock**

----------------------------------------------------

المشكله الحاله بقه انى لو استخدمت Multi Thread ساعتها الكود هيخش بال thread الاولى يلاقى ال instance ب null فهيروح عامل new instance و ال thread التانيه هتبقى شغاله مع ال thread الاولى فى نفس الوقت فبرضوا هتخش معاها علشان تشوف ال instance فهتلاقيها هى كمان ب null فهتروح عامله new instance و بالتالى الشغل ال parallel بوظ فكره ال Singleton تانى بس حل المشكله ان لما thread تخش تروح عامله Lock على الكود بحيث ان ال Thread التانى ميقدرش يخش و هى شغاله و بالتالى هيخشوا بعد بعض مش مع بعض و بالتالى كده الى هتخش ورا التانيه هتلاقى ان ال instance مش فاضى فمش هتعمل new instance و بالتالى كده رجعنا حلينا المشكله بتاعت ال multi threads هى كمان

اوريك بقه المشكله على كود

using System;

using System.Threading.Tasks;

namespace SingletonWithOutThreadSafety

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########### With using Parallel Or Threads #############");

Console.WriteLine($"------------------------");

Parallel.Invoke(

() => Method\_1(),

() => Method\_2(),

() => Method\_3()

);

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

public static void Method\_1()

{

Counter ObjA = Counter.GetInstance();

}

public static void Method\_2()

{

Counter ObjB = Counter.GetInstance();

}

public static void Method\_3()

{

Counter ObjC = Counter.GetInstance();

}

}

}

و كلاس السينجلتون اهه مغيرناش فيه اى حاجه خالص

using System;

using System.Collections.Generic;

using System.Text;

namespace SingletonWithOutThreadSafety

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

if (Instance == null)

{

Instance = new Counter();

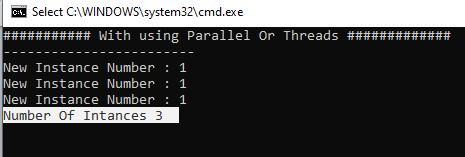
}

return Instance;

}

}

}



ولان ال parallel هيخلى الكود يشتغل على ال 3 فانكشن فى وقت واحد و بالتالى فى كل مره الكود هيروح يشوف هل فيه instance معمول ولا لأ فهيلاقيه فى الثلاثه لسه مش معمول فهيروح ينشأ واحد جديد لل 3 كلهم و النتيجه ان فى 3 new instance و بالتالى فكره ال singleton انضربنت تانى

حل المشكله اننا بقه جوه كلاس ال Singleton نعمل Lock على كود الى بيعمل فحص اذا كان فى instance و لا لأ بحيث مفيش Thread يخش المكان ده مع التانى و يخشوا بالدور الى يخش الاول يقفل الباب وراه لحد ما يخلص الى بعده يخش و يقفل وراه و هكذا و بالتالى لما التانى يجى بعد ما الاول خلص شغله هيلاقى ان فى instance و بالتالى مش هيعمل واحد جديد خلاص

العمليه دى يطلع عليها Thread Safty و كل الى بنعمله حاجتين

اول حاجه بنعرف حاجه من نوع Static ReadOnly Object

public static readonly object InstanceLock = new object();

الحاجه التانيه بنعمل lock على الكود الى هنعمل فيه ال Check عن اذا كان فى انستانس قديمه ولا لأ

public static Counter GetInstance()

{

lock (InstanceLock)

{

if (Instance == null)

{

Instance = new Counter();

}

}

return Instance;

}

الكود كامل اهه مفيش اى تغيير نهائى فى كلاس ال program كل الجديد فى كلاس ال Counter الى بعمله singleton

using System;

using System.Threading.Tasks;

namespace Singleton\_After\_ThreadSafety

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine($"########### With using Parallel and ThreadSafety #############");

Console.WriteLine($"------------------------");

Parallel.Invoke(

() => Method\_1(),

() => Method\_2(),

() => Method\_3()

);

Console.WriteLine($"Number Of Intances {Counter.NumberOfInstance}");

Console.ReadKey(true);

}

public static void Method\_1()

{

Counter ObjA = Counter.GetInstance();

}

public static void Method\_2()

{

Counter ObjB = Counter.GetInstance();

}

public static void Method\_3()

{

Counter ObjC = Counter.GetInstance();

}

}

}

**ده الكلاس الى بعمل عليه singleton و الى التغيير تم عليه**

using System;

using System.Collections.Generic;

using System.Text;

namespace Singleton\_After\_ThreadSafety

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static Counter Instance = null;

// Defination Of Instance Lock

public static readonly object InstanceLock = new object();

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

lock (InstanceLock)

{

if (Instance == null)

{

Instance = new Counter();

}

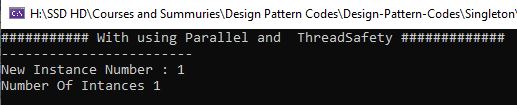
}

return Instance;

}

}

}



طيب دلوقتى فى مشكله كبيره هتواجهنا بقه ايه هى يا معلم ؟؟

كتر استخدام ال Lock بيستنذف الميمورى و بيبطء الابلكيشن و بالتالى انا كده لازم اعمل حاجه تخلى ان ال lock ميترنش الا لما اعوزه للشديد القوى فقط و بالتالى الابلكيشن يبقى سريع جدا جدا

طب حل المشكله ايه يا سيدى ؟؟

double check for thread safty

اعمل check فوق ال lock نفسه بال instance و اقوله فى حاله ان مفيش instance و ضرورى اعمل new instance خلاص ياعم خش على ال lock غير كده مش عاوز اشوف خلفتك اساسا

public static Counter GetInstance()

{

if (Instance == null) // Double thread Safty

{

lock (InstanceLock)

{

if (Instance == null)

{

Instance = new Counter();

}

}

}

return Instance;

}

دى كل التغييره الى هتتعمل فى الكود علشان يبقى double thread safty ياكبير فمش محتاجه كود كامل بقه

MB

##################################################################

Singleton DP Using Eager Loading Vs Lazy Loading

-------------------------------------------------------------------

اول حاجه الفرق بين ال lazy loading و ال Eager loading

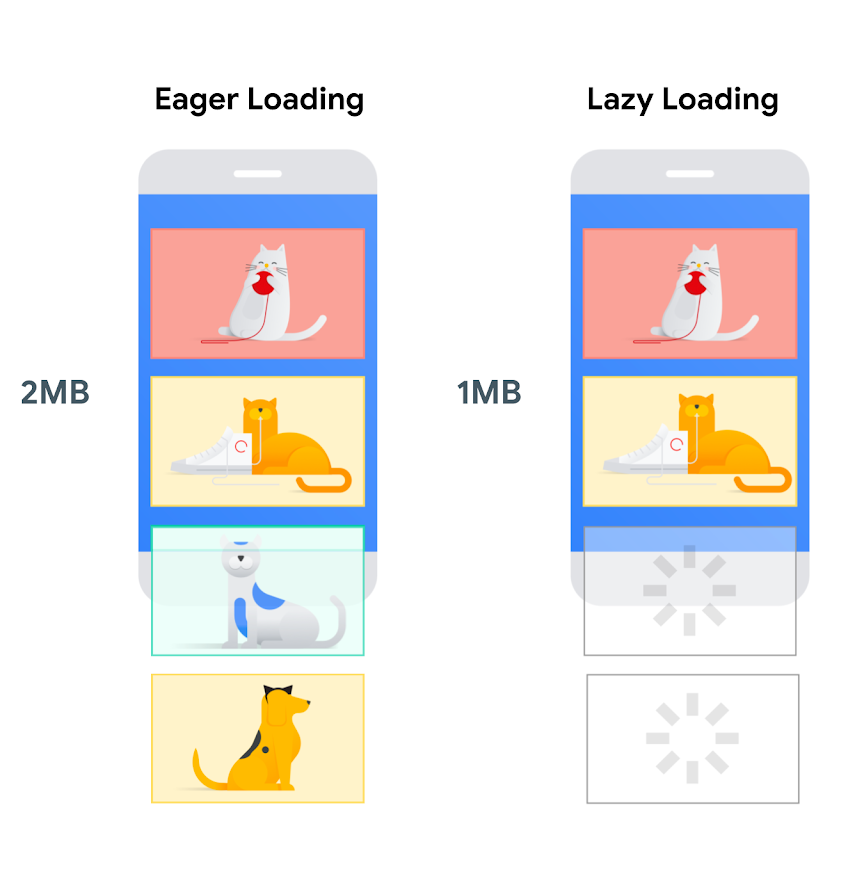
بص يا سيدى ال Lazy Loading ده بستخدمه لما اكون مش عاوز اجيب الداتا كلها و عاوز فقط اجيب الداتا الى محتاجها فقط بمعنى دلوقتى انا عندى كميه داتا كبيره جدا جدا فى البنك عن العملاء مثلا و انا دلوقتى عاوز اخش عليها اعرف بيانات اتنين او تلاته عملاء هيبقى ايه المفيد انى اجيب كل الداتا ؟؟ ولا اي حاجه بالعكس كده السيستم هيموت و يقولى حرام عليك انت كل مره تيجى تجيب بيانات عن واحد تروح رايح جايب كل البيانات الى فيا و تستهلك الميمورى بتعتى على الفاضى لا ياعم خد شوف الى انت محتاجه بس و متتعبنيش معاك

طيب ال Eager دى يا سيدى لما بنستخدمها لما تكون عاوز اجيب الداتا كلها مره واحده و طبعا بنستخدمها ففط مع الداتا القليله يعنى و ليكن انا عاوز اجيب بيانات عن محمود الى هى الاسم و السن و الصوره و الميلاد فهل يعقل انى اجيب كل واحد لما احتاجه بس و لا طالما الداتا صغيره كده اروح اجيبها كلها عادى ما هى تكلفه انى اجيبها كلها اخزنها فى الميمورى اقل بكتير من انى كل شويه اكلم السيرفر انى عاوز اسم محمود عاوز صوره محمود عاوز مقاس شورت محمود

و بالتالى الفرق بين الاتنين ان ال lazy بتجيب الداتا لما اطلبها فقط و بنستخدمها مع الداتا الضخمه

اما ال Eager بتجيب الداتا كلها الى انا عاوزها مره واحده و بنستخدمها مع الداتا الصغيره

و ادى صوره معبره توضح الفرق بين الاتنين



طيب احنا فى الاكواد السابقه دى كنا شغالين ايه ؟؟

Lazy loading و ده لان و انت بعمل Intailization لل instance قولتله ياباشا انت ب null و مش هملاك الا لما الكلاينت ينادينى و يعمل Check عليا لو انا لسه معملتش ال new instance هخليك تعمل غير كده مش هتعمل و بالتالى كده الكود ده LazyLoading

طب احوله اذاى الى Eager Loading ؟

تعديله هبله صغيره خالص

بدل من انى اقوله ال instance = null فى ال Defination هقوله بيساوى new instance و بكده حصل Creation لل instance بمجرد ما البرنامج اتفتح و راح حجز مساحه فى الميمورى حتى لو انا مش عاوز اكلم الكود من اى كلاينت سايد

و طبعا هعمله ReadOnly علشان محدش يقدر يكتب عليه تانى خلاص

و ادى الكود بالكامل بعد تحويله الى Eager Loading

مفيش تغيير فى الكلاينت سايد نهائى كل التغيير هيبقى فى كلاس ال Singleton

using System;

using System.Collections.Generic;

using System.Text;

namespace Singleton\_Eager\_Loading

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

// Eager loading add ReadOnly + New intance

//public static Counter Instance = null;

**public static readonly Counter Instance = new Counter();**

// Defination Of Instance Lock

public static readonly object InstanceLock = new object();

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

//if (Instance == null) // Double thread Safty

//{

// lock (InstanceLock)

// {

// if (Instance == null)

// {

// Instance = new Counter();

// }

// }

//}

return Instance;

}

}

}

كده بقى الكود Eager loading بمجرد ما خليت الديفينيشن فيه ReadOnly و اديته ال new instance فوق مع الديفينيشن و مسحت ال Double Safty Check لان خلاص الانستنس بعد كلمه ReadOnly اصبح مبيعملش غير انستانس واحد بس

طيب لو انا حابب احوله ل LazyLoading بس بشكل اسهل من الاكواد السابقه اعمل ايه ؟

بص يا سيدى سى شارب عملت حاجه اسمها Lazy و دى Key word جاهزه تقدر تستخدمها تسهل عليك الدنيا خالص و تخلى الكود اسهل بكتير و متلخص على الاخر و شكل الكود ساعتها هيبقى كده

كل الى هتعمله انك هتحط lazy فى ال Defination و هتنزل تقول لل instance فى الفانكشن انا عاوز ال Value بتعتك ذى ما انت شايف الدنيا سهله اذاى و بسطر واحد عمل كل شغل ال thread safty , double check و كل حاجه يا صديقى

using System;

using System.Collections.Generic;

using System.Text;

namespace Singleton\_Lazy\_Loading

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class Counter

{

public int Count = 0; //it is not shared so it is repeated in any new intance

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

// Eager loading add ReadOnly + New intance

//public static Counter Instance = null;

//public static readonly Counter Instance = new Counter();

public static readonly Lazy<Counter> Instance = new Lazy<Counter>(()=> new Counter());

// Defination Of Instance Lock

public static readonly object InstanceLock = new object();

private Counter()

{

Count++;

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + Count);

}

public static Counter GetInstance()

{

return Instance.Value;

}

}

}

دلوقتى بقه بعد ما عملنا ال Singleton Class كده يا معلم داخل كلاس ال Singleton ده هتعمل الفانكشن الى هى المفروض تجيب بيها الداتا علشان تبتدى تجرب بيها شغلك ياباشا و ده مثال بسيط ي simulate الاستخدام بقه

التطبيق العملى صرف بقه

MB

##################################################################

Before Singleton with Heavy GetData Method

-------------------------------------------------------------------

اولا عندنا كلاس بتعمل Convert للعملات من دولار لجنيه مصرى و من سعودى لجنيه و من كويتى لجنيه مصرى و هكذا

فاول حاجه بعملها انى بعمل Table فى الداتا بيز يكون فيه BaseCurrency و TargetCurrency و Rate يعبر عن الفلوس لما هتتحول من ال base لل target المفروض ينضربوا فى كام علشان يتحولوا

بعد ما بعمل الجدول بعمل ال Entity بتعها الى هو الكلاس الى هيعمل map مع الجدول ده

هنمسيه مثلا ExchaneRate

using System;

using System.Collections.Generic;

using System.Text;

namespace Before\_Singleton\_LazyLoading\_With\_GetDataMethod

{

internal class ExchangeRate

{

public string BaseCurrency { get;}

public string TargetCurrency { get; }

public decimal Rate { get; }

public ExchangeRate(string \_BaseCurrency , string \_TargetCurrency , decimal \_Rate)

{

BaseCurrency= \_BaseCurrency;

TargetCurrency= \_TargetCurrency;

Rate= \_Rate;

}

}

}

بعد كده المفروض عندى كلاس هيكون مسؤول عن التحويل من عمله لعمله و ده هيبقى اسمه CurrencyConverter

ده بقه هملى فيه ال lookup الى هيجيب التحويلات من ال Base لل Target و طبعا دى لو فانكشن بتجيب حاجات كتير فهتبقى تقيله و هتقعدلها مده فانا هعملها سيموليت ب 3 ثوانى

و بعد ما املى الانتيتى بتاع جدول التحويلات المفروض بقه هستخدمه فى فانكشن رئيسيه جدا هنا الا و هى ال Convert الى هتحول العمله لعمله اخرى

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace Before\_Singleton\_LazyLoading\_With\_GetDataMethod

{

internal class CourencyConverter

{

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public IEnumerable<ExchangeRate> \_ExchangeRate { get; set; }

public CourencyConverter()

{

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + NumberOfInstance);

getLookUpExchangeRate();

}

private void getLookUpExchangeRate()

{

Thread.Sleep(3000); // Wait 3 Sec Simulate Heavy Get Function

\_ExchangeRate = new[]

{

new ExchangeRate("DK","EGP",101.1m),

new ExchangeRate("USD","EGP",30.75m),

new ExchangeRate("SAR","EGP",8.30m),

};

}

public decimal Convert(string BaseCurrenct , decimal Amount)

{

var ExchangeRate = \_ExchangeRate.FirstOrDefault(\_ => \_.BaseCurrency == BaseCurrenct);

return ExchangeRate.Rate \* Amount ;

}

}

}

طيب كده بقه فى الكلاينت سايد انا عاوز اقعد بقه احول فى فلوس من هنا للصبح فهعمل لوب ميوقفش التحويل خالص الا لما اقفل الجهاز

using System;

namespace Before\_Singleton\_LazyLoading\_With\_GetDataMethod

{

internal class Program

{

static void Main(string[] args)

{

while (true)

{

Console.WriteLine("Enter BaseCurrency ( DK - USD - SAR) it will Convert to EGP :");

var BaseCurrency = Console.ReadLine();

Console.WriteLine("Enter Amount of BaseCurrency :");

var Amount = decimal.Parse(Console.ReadLine());

var ExchangeAmount = new CourencyConverter().Convert(BaseCurrency, Amount);

Console.WriteLine($"BaseCurrency : {Amount} {BaseCurrency} --> TargetCurrency : {ExchangeAmount} EGP ");

}

}

}

}

لاحظ بقه ان هنا كل مره هينشأ object جديد و بالتالى كل مره هيملى ال lookup من جديد و بالتالى كل عمليه هيعطل 3 ثوانى من جديد

طيب كان حل المشكله ايه ؟؟ استخدام ال Singleton سواء Eager مع الداتا الخفيفه او lazy مع الدتا التقيله جدا

و هنا هنستخدم بقه ال Lazy Singleton و بعديه جربه و شوف تاُثيره على سرعه الابلكيشن

ExchaneRate ذى ما هو

using System;

using System.Collections.Generic;

using System.Text;

namespace SingleTon\_LazyLoading\_With\_GetData\_Method

{

internal class ExchangeRate

{

public string BaseCurrency { get;}

public string TargetCurrency { get; }

public decimal Rate { get; }

public ExchangeRate(string \_BaseCurrency , string \_TargetCurrency , decimal \_Rate)

{

BaseCurrency= \_BaseCurrency;

TargetCurrency= \_TargetCurrency;

Rate= \_Rate;

}

}

}

تحويل ال CourencyConverter الى SingletonLazyLoading

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace SingleTon\_LazyLoading\_With\_GetData\_Method

{

//Sealed Will Stop any kind of Inhertance in Singleton Class

internal sealed class CourencyConverter

{

public static int NumberOfInstance = 0; //it is shared between all instances to Count Instrances

public static readonly Lazy<CourencyConverter> Instance = new Lazy<CourencyConverter>(()=> new CourencyConverter());

public static readonly object InstanceLock = new object();

public IEnumerable<ExchangeRate> \_ExchangeRate { get; set; }

private CourencyConverter()

{

NumberOfInstance++;

Console.WriteLine($"New Instance Number : " + NumberOfInstance);

getLookUpExchangeRate();

}

public static CourencyConverter GetInstance()

{

return Instance.Value;

}

private void getLookUpExchangeRate()

{

Thread.Sleep(3000); // Wait 3 Sec Simulate Heavy Get Function

\_ExchangeRate = new[]

{

new ExchangeRate("DK","EGP",101.1m),

new ExchangeRate("USD","EGP",30.75m),

new ExchangeRate("SAR","EGP",8.30m),

};

}

public decimal Convert(string BaseCurrenct , decimal Amount)

{

var ExchangeRate = \_ExchangeRate.FirstOrDefault(\_ => \_.BaseCurrency == BaseCurrenct);

return ExchangeRate.Rate \* Amount ;

}

}

}

ثم نداء الفانكشن بتاعت التحويل ملايين المرات و لكن المره ده انا انشأت الاوبجيكت مره واحده و نديت لل lookup مره واحده فقط و بالتالى اول مره هنعطل 3 ثوانى بعد كده خلاص الداتا فى الميمورى و مش هنعطل تانى خالص

using System;

namespace SingleTon\_LazyLoading\_With\_GetData\_Method

{

internal class Program

{

static void Main(string[] args)

{

while (true)

{

Console.WriteLine("Enter BaseCurrency ( DK - USD - SAR) it will Convert to EGP :");

var BaseCurrency = Console.ReadLine();

Console.WriteLine("Enter Amount of BaseCurrency :");

var Amount = decimal.Parse(Console.ReadLine());

var ExchangeAmount = CourencyConverter.GetInstance().Convert(BaseCurrency, Amount);

Console.WriteLine($"BaseCurrency : {Amount} {BaseCurrency} --> TargetCurrency : {ExchangeAmount} EGP ");

}

}

}

}

MB

##################################################################

Factory DP

-------------------------------------------------------------------

بص يا سيدى الديزاين باترن ده اهميته انه بيبقى عامل ذى المصنع و ليكن انت قدامك مصنع عربيات بتخش تقولهم بصوا انا جايبلكم مودل شكله حلو لعربيه سزوكى عاوزكوا تعملوهالى و هما بيخدوا المودل ده يعملولك العربيه و يسلموهالك تقولهم طب انا معايا مودل تانى لعربيه مرسيدس يقوموا عاملينها و هكذا بقه   
يعنى مصنع واحد لنوع معين من المنتجات بيطلعلك كل اشكالها الى انت عاوزها بدون معاناه انك تروح لكل منتج منهم تعمله بنفسك

طيب تعالى بقه بمثال كود   
بص يا سيدى دلوقتى انت عندك مصنع عربيات بينتج عربيات SUV & Sedan & Truck

3 انواع المفروض انك دلوقتى ك client هتخش تختار نوع من العربيات تشتريه فهتختار رقم معين بيعبر عن العربيه دى فتقوم طلعالك تفاصيل العربيه الى انت اخترتها   
  
فى الحاله العاديه من with out Factory هنخش نعمل كلاس لكل نوع عربيه TruckCar – SedanCar – SUVCar

و هنعمل فيهم الامبلمنتيشن الخاص بيهم و بعد كده جوه ال client Side الى هى واجهه البرنامج هنعمل if Condition فى حاله اختبار رقم 1 يروح ينادى على الكلاس Truck و 2 ينادى على SUV و تلاته ينادى على SEDAN

امتى استخدمه ؟

لما يكون قدامى مثلا منتج بيتعل ذى العربيات و مش عارف انا محتاج اعمل انهى نوع عربيه بناءا على قرار معين من الكلاينت و احتمال كبير عدد انواع العربيات هيكبر جدا و باتالى ال if Condition الى منها بعمل Create للعربيه بنوع معين هيزيد عند الكلاينت و بالتالى ساعتها لازم اعمل فاكتورى العميل يديلى اختياره فى ال Client side اخد الاختيار بتاعه ده على اساسه ابتدى اعمل Create ل object بنوع العربيه المطلوبه

**يعنى من الاخر بستخدم ال Simple factory لما اكون عندى اختيارات انى اعمل اوبجيكت من كلاسات من نفس العائله يعنى ايه كلاسات من نفس العائله ؟؟**

**يعنى كلاسات بتعمل implemention لنفس ال interface او كلهم بيورثوا من اب واحد يعنى كلاسات اخوات يعنى ياعم**

كما فى الكود الحالى بالظبط

##################################################################

Before Factory DP

-------------------------------------------------------------------

اول حاجه هنعمل كلاس parent نسميه Car و يكون اما abstract او interface و ده علشان نقدر نتحكم فى ان اى حد يورث منه يبقى مجبر يمشى على شكل معين احنا عاملينه للكود و ده لان احنا فى مصنع عربيات فلازم يبقى عندى شروط او standard معينه لصناعه اى نوع عربيات مينفع متتعملش و الشروط دى بتتحط فقط فى ال interface او الابستراكت كلاس

فأول حاجه هعملها interface ال ICar علشان احدد فيها Standard كل العربيات الى بتتصنع فى المصنع

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BeforeFactory.Interfaces

{

interface ICar

{

public void RunCar();

public void ShowCar();

}

}

ثم بعد كده المفروض هعمل الكلاس الى هيعمل Implemention لل interface ده و هو ده الى هيبقى الشكل الاولى لكل انواع العربيات الى فى المصنع و هيبقى ده كلاس Abstract علشان ده الى هيتورث منه فى كل انواع العربيات و هيبقى اسمه Car

using BeforeFactory.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BeforeFactory

{

abstract class Car : ICar

{

public string CarName { get; set; }

public int CarModelYear { get; set; }

public void RunCar()

{

Console.WriteLine($"{CarName} is Running .");

}

public void ShowCar()

{

Console.WriteLine($"Car Model Year : {CarModelYear} is Showing .");

}

}

}

ثم بعد كده خلاص هبتدى اعمل جميع الانواع الى ممكن المصنع يعملها على نفس الاستاندارد بتاع كلاس ال Car

فعندنا دلوقتى 3 انواع

النوع الاول TruckCar و ده طبعا بما انه هيبقى ماشى على نفس نمط الكلاس الاساسى بتاع العربيات فهيورث منه و فقط هيعدل جواه شويه تعديلات بسيطه تخصه علشان العربيه تطلع بالشكل الخاص بكل نوع

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BeforeFactory

{

class Truck\_Car : Car

{

public Truck\_Car()

{

CarName = "Mercides";

CarModelYear = 2010;

}

}

}

ثم بعد كده نعمل النوع التانى Sedan بنفس الطريقه

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BeforeFactory

{

class Sedan\_Car : Car

{

public Sedan\_Car()

{

CarName = "BMW";

CarModelYear = 2020;

}

}

}

ثم بعد كده نعمل النوع الثاث SUV بنفس الطريقه

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BeforeFactory

{

class SUV\_Car : Car

{

public SUV\_Car()

{

CarName = "Nissan Cashcai";

CarModelYear = 2015;

}

}

}

كده انا عملت كل الانواع الى انا محتاجها عندى فى المصنع و ظبطت حالى

الطريقه البدائيه بقه الى قبل ما يعملوا حكايه Factory Design pattern ان الى عاوز يطلب نوع عربيه مفيش مصنع هو يطلبها من ال Client Side الى هى ال main Function

using BeforeFactory.Interfaces;

using System;

namespace BeforeFactory

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Choose type of car ==> 1=Truck or 2=SUV or 3=Sedan : ");

string type = Console.ReadLine();

ICar car = null;

switch (type)

{

case "1":

car = new Truck\_Car();

break;

case "2":

car = new SUV\_Car();

break;

case "3":

car = new Sedan\_Car();

break;

default:

Console.WriteLine("Your Choise type of car was wrong . ");

break;

}

car.RunCar();

car.ShowCar();

Console.ReadKey(true);

}

}

}

و بالتالى كده صناعه ال object من مسؤوليه ال client فاى تعديل او زياده فى انواع العربيات لازم كده ارجع اعدل فى ال client Side و طبعا كده حاجه غلط جدا لان ده هيؤدى ان كل اما يحصل اى تعديله صغيره اجرى اعدل و اجرى اتيست المكان ده كويس جدا تانى على كل السيناريوهات من جديد لان التعديله ممكن تاثر على كل الانواع

طب كان ايه الحل فى الموضوع ده ؟؟  
الحل الاول كان حل بسيط جدا و يعتبر ده ال Simple Factory

##################################################################

After Classic Factory DP

-------------------------------------------------------------------

الحل ده اننا بنعمل كلاس جديده اسمها CarFactory تبقى كل مهمتها انها بتعمل فقط creation لنوع العربيه الى انت بتختاره و بترجع العربيه جاهزه لصاحبها فى ال Client side فكده اى زياده او تعديل فى الكود هيبقى فقط فى ال Factory و مش هنيجى جنب ال Client side او main Function تماما و يبقى شكل الكود بعد التعديلات دى كل حاجه ذى ما هى ما عدا اضافه CarFactory class و نشيل الكونديزشنز بقه من ال main function و نحط بدالها سطر نداء صناعه العربيه من المصنع فقط

فهيبقى اول حاجه كلاس ال Factory اهه CarFactory جواه فانكشن تديها نوع العربيه و هى هتخش تعمل صناعه للاوبجيكت بتاع العربيه صناعه للعربيه يعنى و ترجعهالك فى return جميله جاهزه على الاستخدام الفورى

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_Classic\_Factory.Factory

{

class CarFactory

{

public Car GetCar(string type)

{

switch (type)

{

case "1":

return new Truck\_Car();

case "2":

return new SUV\_Car();

case "3":

return new Sedan\_Car();

default:

Console.WriteLine("Your Choise type of car was wrong . ");

return null;

}

}

}

}

و التعديلات الى تمت فى ال client Side او main Function اهه يا صديقى

using After\_Classic\_Factory.Factory;

using After\_Classic\_Factory.Interfaces;

using System;

namespace After\_Classic\_Factory

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Choose type of car ==> 1=Truck or 2=SUV or 3=Sedan :");

string type = Console.ReadLine();

ICar car = new CarFactory().GetCar(type);

car.RunCar();

car.ShowCar();

Console.ReadKey(true);

}

}

}

خدت بالك ان خلاص مبقاش فى الكونديشنز هنا و بالتالى حميت الكلاينت سايد من كل خمس دقايق تعديل فى الكود هنا و يبقى كل التعديلات فى المصنع بس

طبعا باقى الكلاسات (SUV\_Car – Sedan\_Car – Truck\_Car – Car – Icar ) هتبقى ذى ما هى كلهم مفيش فيهم كلمه هتتغير عن الى معمولين فوق يا صديقى



طيب دلوقتى لو حابيين ان المصنع ده يبقى معزول خالص و محدش عارف هو بيصنع ايه بالظبط علشان مفيش حد يقدر يبوظه نعمل ايه ؟؟

نعمله Abstraction او Interface

بحيث كده المصنع كده هيبقى بيصنع الى بيصنعه محدش عارف هو بيصنع ايه الا لما المنتج بتاعه يخرج و يقول انا بقيت عربيه انا بقيت كوتش انا بقيت اكسسوارات عربيات كده يعنى انما طول ما المنتج محطوط جوه الكرتونه متفتحتش محدش هيبقى عارف ايه الى جوه الكرتونه دى و الكرتونه هنا يعنى نعمله abstraction يا صديقى

**مميزات ال Simple – Classic Factory ايه ؟**

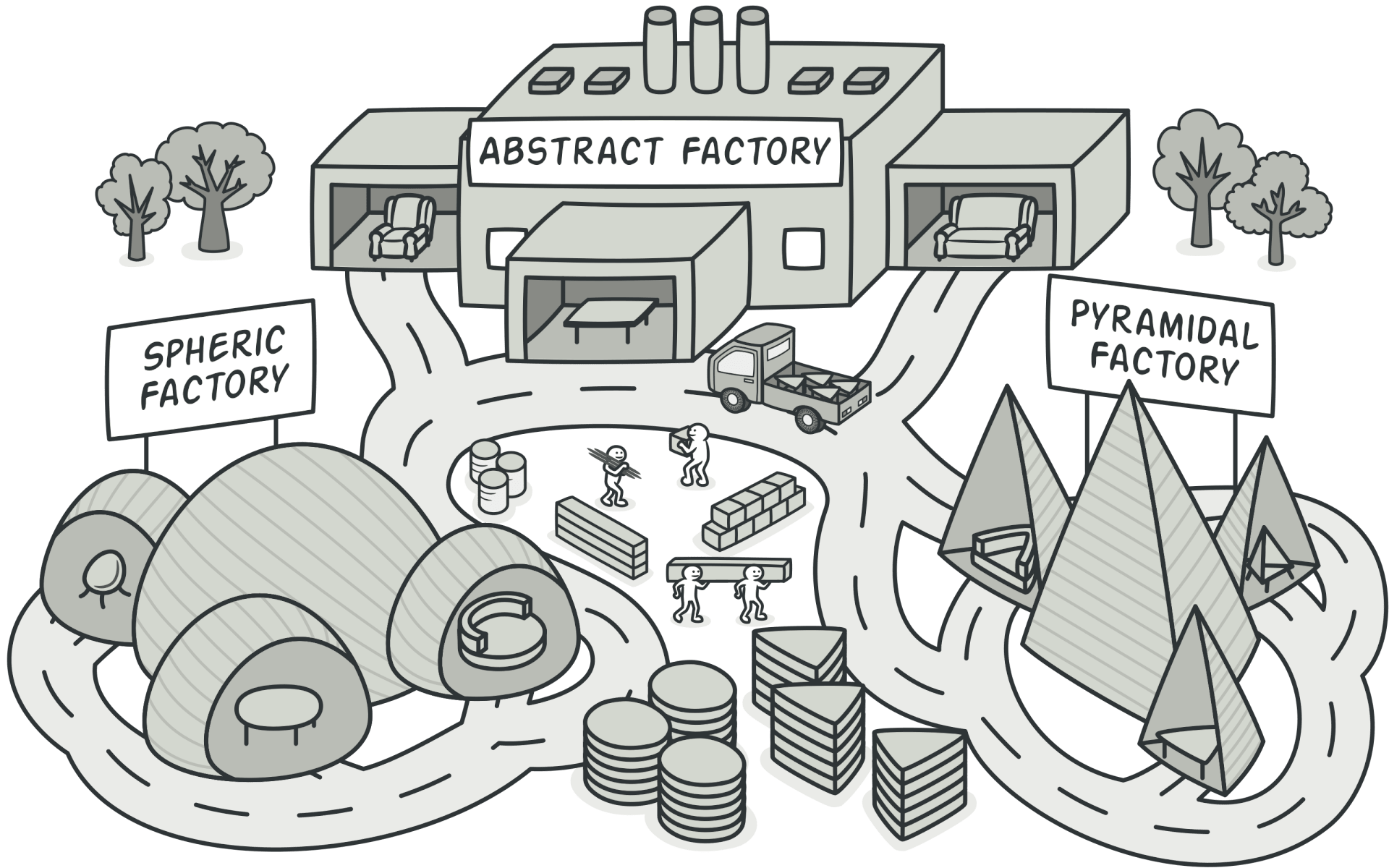
1. **الكود فى مكان واحد فاى تغيير هيبقى فى مكان واحد**

##################################################################

After Abstract Factory DP

-------------------------------------------------------------------

المصنع كبر منى و مبقاش مصنع عربيات بس ده بقى مصنع عربيات و اطارات عربيات و اكسسوارات عربيات فجه الوقت الى اغلب كل منتج خارج من المصنع بقه بحيث محدش يعرف هو ايه الا لما يفتحه و يشوف الى جواه و علشان كده لازم نعمل abstraction للمصنع



فاول حاجه هعملها انى هزود جوه المصنع منتج الاطارات و طبعا لكل عربيه نوع اطار خاص بيها نوع كبير و نوع صغير و نوع وسط

اول حاجه هزودها ال standard بتاعت صناعه الاطارات بحيث محدش يعملى مثلا اطار مربع لا يبقى كل الاطارات ليها شكل الدايره مثلا هههههههه و يبقى المقاس فقط الى بيختلف باختلاف النوع الى انا هعمله عند الاختيار

فاول حاجه interface ال IWheel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_abstract\_Factory.Interfaces

{

interface IWheels

{

void drawWheel();

}

}

ثم بعد كده هعمل ال Abstract كلاس الخاص بال Wheels علشان ده الى هيبقى بيورث منه انواع العربيات هعمل انواع ال Wheels الى عاوزه اصنعها

using After\_abstract\_Factory.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_abstract\_Factory.Wheels

{

class Wheel : IWheels

{

public string WheelSizeName { get; set; }

public virtual void drawWheel()

{

Console.WriteLine($"Wheel Size is {WheelSizeName} ");

}

}

}

ثم بعد كده انواع الاطارات الى انا هصنعها و اولها ال SmallWheel

using After\_abstract\_Factory.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_abstract\_Factory.Wheels

{

class SmallWheel : Wheel

{

public SmallWheel()

{

WheelSizeName = "Small";

}

}

}

ثم ال MediumWheel

using After\_abstract\_Factory.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_abstract\_Factory.Wheels

{

class MediumWheel : Wheel

{

public MediumWheel()

{

WheelSizeName = "Medium";

}

}

}

ثم ال Large

using After\_abstract\_Factory.Interfaces;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_abstract\_Factory.Wheels

{

class LargeWheel : Wheel

{

public LargeWheel()

{

WheelSizeName = "Large";

}

}

}

كده انا بقى عندى كل حاجه تبع الاطارت و اقدر ادخلها دلوقتى فى خط انتاج جوه المصنع فهروح رايح حاططها جوه المصنع تتعمل و يبقى اختيار المنتج داخل المصنع برضوا

فهيبقى شكل المصنع من جوه انه بقى فيه خط انتاج عربيات و خط انتاج اطارات

CarFactory

using After\_abstract\_Factory.Interfaces;

using After\_abstract\_Factory.Wheels;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_abstract\_Factory.Factory

{

class CarFactory : IAbstractCarFactory

{

public Car GetCar(string type)

{

switch (type)

{

case "1":

return new Truck\_Car();

case "2":

return new SUV\_Car();

case "3":

return new Sedan\_Car();

default:

Console.WriteLine("Your Choise type of car was wrong . ");

return null;

}

}

public void GetCarWheelSize(string type)

{

switch (type)

{

case "1":

new LargeWheel().drawWheel();

break;

case "2":

new MediumWheel().drawWheel();

break;

case "3":

new SmallWheel().drawWheel();

break;

default:

Console.WriteLine("Your Choise type of car was wrong . ");

break;

}

}

}

}

و طبعا علشان احنا قولنا فى الاول ان لازم يبقى معزول و لازم يبقى ماشى على ستاندر معينه فانا كان لازم اكون عامله ال Abstraction فهكون مخلي المصنع وارث من IAbstractFactory

دى الى بحدد فيها ايه اقل منتجات يقدر المصنع يعملها و لو زودنا اى منتج فيها لازم نروح نزوده فى المصنع على طول

using After\_abstract\_Factory;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace After\_abstract\_Factory.Interfaces

{

interface IAbstractCarFactory

{

Car GetCar(string type);

void GetCarWheelSize(string type);

}

}

ولان احنا عاوزين نعمل الابستركشن الصح فى ال Client Side فطبعا هيبقى دلوقتى انا بعمل اوبجيكت من المصنع و من جوه المصنع نفسه بحدد عاوز ايه فى العربيات و عاوز ايه فى الاطارات يا صديقى

و هيكون شكل ال main function دلوقتى كده

using After\_abstract\_Factory.Factory;

using After\_abstract\_Factory.Interfaces;

using System;

namespace After\_abstract\_Factory

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Choose type of car ==> 1=Truck or 2=SUV or 3=Sedan :");

string type = Console.ReadLine();

IAbstractCarFactory AF = new CarFactory();

var Car = AF.GetCar(type);

Car.RunCar();

Car.ShowCar();

AF.GetCarWheelSize(type);

Console.ReadKey(true);

}

}

}

طبعا باقى الكلاسات (SUV\_Car – Sedan\_Car – Truck\_Car – Car – Icar ) هتبقى ذى ما هى كلهم مفيش فيهم كلمه هتتغير عن الى معمولين فوق يا صديقى

و على فكره احنا نقدر نعدل تعديله كمان صغيره خالص نخلى الكلاينت ميعرفش اى حاجه تخص العربيه خالص يعنى مش صرورى اصلا نرجعله اوبجيكت من ال Car يتعامل معاه و ده احسن حاجه للابستراكشن اصلا فتعديله بسيطه بقه كده فى الحنين هتتعمل فى الكود

كده يا صديقى كل التغييرات الى هتحصل انى هعمل فانكشن بتعرفنى تفاصيل العربيه داخل المصنع برضوا

و طبعا لازم اكون عملها ديفينيشن فى ال Interface

فاول حاجه هعملها ديفينشن جنب اخوتها فى ال Interface بتاع ال IAbstractFactory

using AfterAbstractWithHidingCarFromClientSide;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AfterAbstractWithHidingCarFromClientSide.Interfaces

{

interface IAbstractCarFactory

{

Car GetCar(string type);

void GetCarWheelSize(string type);

void CarDetails(string type);

}

}

بعجد كده هعمل الامبلمنتيشن انه يجيب كل تفاصيل العربيه جواها و كان ده يا صديقى ال Menu بتاع العربيه هههههههه فهنطلعه هو كمان من المصنع CarFactory

using AfterAbstractWithHidingCarFromClientSide.Interfaces;

using AfterAbstractWithHidingCarFromClientSide.Wheels;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AfterAbstractWithHidingCarFromClientSide.Factory

{

class CarFactory : IAbstractCarFactory

{

public Car GetCar(string type)

{

switch (type)

{

case "1":

return new Truck\_Car();

case "2":

return new SUV\_Car();

case "3":

return new Sedan\_Car();

default:

Console.WriteLine("Your Choise type of car was wrong . ");

return null;

}

}

public void CarDetails(string type) {

Car car = GetCar(type);

car.RunCar();

car.ShowCar();

}

public void GetCarWheelSize(string type)

{

switch (type)

{

case "1":

new LargeWheel().drawWheel();

break;

case "2":

new MediumWheel().drawWheel();

break;

case "3":

new SmallWheel().drawWheel();

break;

default:

Console.WriteLine("Your Choise type of car was wrong . ");

break;

}

}

}

}

و بالتالى شكل ال Client Side مش هيبقى باين فيه اى ملامح لاى حاجه و لا معروف المصنع ده بينتج ايه اصلا هو مجرد اوامر باخدها من المصنع فقط لا غير

using AfterAbstractWithHidingCarFromClientSide.Factory;

using AfterAbstractWithHidingCarFromClientSide.Interfaces;

using System;

namespace AfterAbstractWithHidingCarFromClientSide

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Choose type of car ==> 1=Truck or 2=SUV or 3=Sedan : ");

string type = Console.ReadLine();

IAbstractCarFactory AF = new CarFactory();

AF.CarDetails(type);

AF.GetCarWheelSize(type);

Console.ReadKey(true);

}

}

}

كده انا عملت مصنع لمنتجات و خليت العميل يبقى بيطل بالحاجه تجيله لحد عنده و بس و ملوش دعوه باى تفاصيل خالص

IAbstractCarFactory AF = new CarFactory();

انا ليه برجع الاوبجيكت فى ال interface بتاعه ؟؟

علشان يا صاحبى محدش يروح يحطلى منتج جديد جوه المصنع بدون علم مراقبيين الجوده لازم بتوع الاستاندرد يقولى ماشى حطها و علشان مراقبين الجود يقولوا ماشى لازم الفانكشن او المنتج تبقى جواهم عندهم اصلا بمعنى ان مينفعش احط كده اى فانكشن جديده جوه الكلاس CarFactory قبل ما احط الفانكشن دى جوه ال IAbstractCarFactory و الا لو جيت اناديها عن طريق ال AF هتسوط و تقولك انا مش معمولى الديفينشن اصلا جوه ال interface



##################################################################

Builder DP

-------------------------------------------------------------------

اهميته انى لو عندى اوبجيكت بيتم عمل ليه Creation بخطوات كتيره او معقد انى عن طريقه افصل عمليه الكريشن دى فى مكان لوحده بعيد تماما عن ال client Side بحيث يبقى فى صندوق او كونتينر بيتم صناعه الاوبجيكت فيه خطوه بخطوه لحد ما يخلص تماما و بعد كده يدهونى عند الكلاينت سايد جاهز و خلصان على الاستخدام على طول

من الاخر اهميته Build Complex Object Step by Step  
يعنى ايه Complex object ؟

يعنى object فيه properties كتير بياخد برامتيارز كتير فى الكونستراكتور و احيانا بعض البراميتارز دى فيها منها optional و كمان اوقات بيبقى object بيعملى Heavy IO Operations و بيكلم فايلز و حوارات

او بيكلم داتا بيز فعمليه انشاء الاوبجيكت نفسه بتبقى مكلفه جدا للسيستيم و الميمورى و بالتالى لازم استخدم مع كل ده ال Builder DP و ده لانه هيأجل انشاء الاوبجيكت لاخر لحظه ممكنه بحيث انى فى اى لحظه اقدر اعمل cancel لانشاء الاوبجيكت

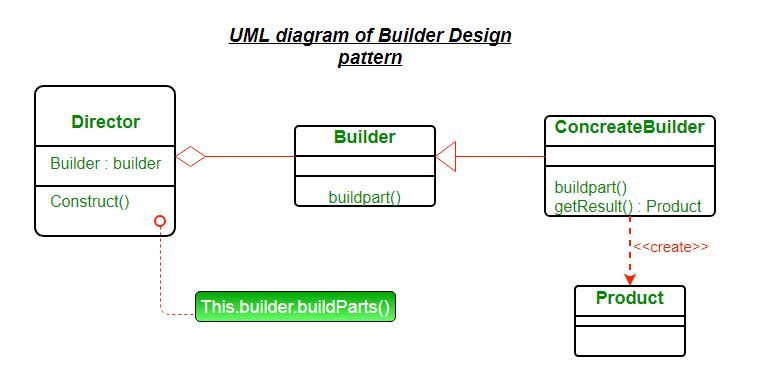
**يستحدم بشكل عملى و واضح جدا فى ال Building ال services فى ASP.net Core**

**يستخدم فى عمل ال Fluent API**

مثال عليه معروف String Builder

مثال من الحياه عندك عربيه و المفروض هتصنعها فالمفروض هتعمل العجل و هتعمل الابواب و الماتور و الكشافات فهل يعقل انى اروح اركب كل العربيه دى عند ال Client ولا بيبقى فى مكان لتجميع قطع العربيه دى قبل ما اديها للعميل ؟ بيبقى فى مكان بجمع فيه و العميل بيجى يستلم العربيه على الاستخدام بس

طيب اهه ده بقه الى ال Builder بيعمل ذيه بالظبط بيكون الاوبجيكت لوحده كده فى مكان منفضل و العميل بس بيقول كريتولى ال product فبيتعمله بدون ما العميل يخش فى تفاصيل صناعته او تركيبه او اى حاجه



ال Product ده يا صديقى الكلاس الى انا عاوز اعمل للاوبجيكت بتاعه تجميع مع بعض الى هو ممكن مثلا انسان ألى و عاوز اجمع قطعه

ال Concrete Builder دى يا سيدى الكلاس الى انا بعمل ال Build لل product جواها و منها برضوا برجع الاوجيكت ال product بعد ما ببنيه بشكل كامل يعنى بحط هنا خطوات بناء الانسان الالى

ال Builder ده ال Interface او ال Abstract class الى المفروض ال ConcreteBuilder بيعمله امبلكنتيشن بعد كده

ال Director ده ياسيدى الكلاس الى فيه تعليمات انشاء ال Object يعنى بيرتب الخطوات انهى خطوه تيجى قبل انهى خطوه و من هنا باخد الاوامر بانشاء المنتيج بخطوات مترتبه علشان يرجعلى بعد كده Builder اقدر استخدمه انى اجيب بيه product مكتمل بقه و خلصان بعيد تماما عن ال Client Side

##################################################################

Before Builder DP

-------------------------------------------------------------------

دلوقتى البيلدر عاوزينه يبقى بيعمل عربيات طب العربيات دى عباره عن ايه ؟ عباره عن قطع و هجيب قطعه جنب قطعه لحد ما يكونوا العربيه كامله

قبل ما يخترعوا ال Builder كان الاول الواحد يروح عامل كلاس Car يحط فيه فانكشنز اضافه و تركيب قطع العربيه على هيئه خطوات و يبقى المسؤول الرئيسى عن الخطوات دى و عن طريقه تركيب العربيه قطعه قطعه هو ال Client Side

فالاول كده ال Car Class اهه

using System;

using System.Collections.Generic;

using System.Text;

namespace Before\_Builder\_DP

{

internal class Car

{

private string BrandName;

public Car(string \_BrandName)

{

this.BrandName = \_BrandName;

}

public void startOperations()

{

Console.WriteLine($"Car Model Name : {this.BrandName}");

}

public void BuildBody()

{

Console.WriteLine($"Build Body of {this.BrandName}");

}

public void InsertWheels()

{

Console.WriteLine($"Insert Wheels of {this.BrandName}");

}

public void InsertHeadLights()

{

Console.WriteLine($"Insert Head Lights of {this.BrandName}");

}

public void EndtOperations()

{

Console.WriteLine($"Endt Operations of {this.BrandName}");

}

}

}

و ال Client Side اهه

using System;

namespace Before\_Builder\_DP

{

internal class Program

{

static void Main(string[] args)

{

Car car = new Car("BMW");

car.startOperations();

car.InsertWheels();

car.InsertHeadLights();

car.BuildBody();

car.EndtOperations();

}

}

}

واضح وضوح الشمس مسؤوليه ال Client Side عن تركيب العربيه و طبعا ده هيخلى فى مشاكل كتير جدا جدا على المدى البعيد و القريب كمان وحياتك مع اول تعديله فى الحته دى

و من هنا قالك لأ احنا عاوزين نفصل عمليه تركيب قطع الاوبجيكت دى عن ال Client Side دى تماما و يبقى الموضوع و ما فيه ان كل حاجه تحصل فى الخفاء و الكلاينت سايد ميشوفش غير المنتج الاخير فقط

طب ننفذ الكلام ده اذاى ؟

عن طريق ال Builder Design Patten لانه هو الى بيجمع ال object المعقد فى خطوات سهله و بسيطه و بتفصل الموضوع ده لوحده فى حاجه اسمها Builder بحيث ان ال Client Side يبقى ملهمش دعوه بالتعقيدات بتاعت ال Object تماما

##################################################################

After Builder DP

-------------------------------------------------------------------

دلوقتى ال Builder عاوزينه يبقى بيعمل عربيات طب العربيات دى عباره عن ايه ؟

عباره عن منتج product يبقى هنعمل كلاس اسمه Product هيكون فيه اجزاء العربيات بعد ما جمعتها مع بعض و بقت منتج نهائى

وال product Class ده هيبقى معمول بشكل General ليه ؟

لانه هيخش فى تجميع كذا نوع من العربيات مش هيبقى نوع واحد فقط و كمان ممكن اجمع فيه موتوسيكلات كمان فلازم يبقى معمول بما يناسب الكل مش حاجه واحده بس و علشان كده اول حاجه هنعملها ال Product class لان هو ده المكان الى بجمع فيه قطع المنتجات الى هعوز اعرضها بعد كده كمنتج نهائى

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_DP

{

internal class Product

{

private LinkedList<string> Parts;

public Product()

{

Parts = new LinkedList<string>();

}

//Collect Parts of Product

public void Add(string Part)

{

Parts.AddLast(Part);

}

// Show All parts of Products

public string Show()

{

StringBuilder sb = new StringBuilder();

sb.AppendLine("Product Components are : ");

foreach (string Part in Parts)

{

sb.AppendLine(Part);

}

return sb.ToString();

}

}

}

طيب و بما انى عاوز اعمل عربيات فالمفروض هعمل كلاس ال Car و كلاس ال Car ده هيبقى فيه فانكشنز بخطوات تجميع قطع العربيه الى هبعتها لل product على انها منتج قطع يجمعها مع بعض علشان يدينى الشكل النهائى للعربيه فى مكان واحد و بما اننا هنبعت قطع العربيه دى لل product اذاى المفروض يبقى فى اوبجيكت من ال product كلاس جوه ال Car كلاس و ده علشان اقدر اوصل قطع العربيات المتفككه لل product يجمعها جواه

طيب دلوقتى بما انى هعمل جوه كلاس ال Car الفانكشنز الى المفروض هى دى خطوات تجميع العربيه فالمفروض انى ابقى عامله Standard يمشى عليه بحيث مينقصش عن الخطوات المطلوبه فعشان كده اول حاجه هفكر فيها انى اعمل حاجه اسمها IBuilder يكون فيها الخطوات الخاصه ببناء العربيه و كذلك الفانكشن الى هترجعلى العربيه بعد تجميعها كمنتج Producr يعنى بالظبط طده انا بجمع العربيه قطع قطعه خطوه خطوه و اول ما تكمل اروح واخدها بقه كمنتج كامل علشان كده بحتاج اعمل فانكشن بترجع اوبجيكت من نوع product يبقى ده الى هيبقى فيه المنتج او العربيه بع تجميعا مع بعض

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_DP

{

internal interface IBuilder

{

void startOperations();

void BuildBody();

void InsertWheels();

void InsertHeadLights();

void EndtOperations();

Product GetVehicle();

}

}

و بما ان دى فيها خطوات بناء العربيه او التوكتوك او اى بتنجان فكده يا صديقى المفروض هعملها Implemention فى كلاس ال Car علشان اقدر انادى على الى هيصيف كل الى ناقص للعربيه بخطوات محدده و سليمه

ده بقه كلاس ال Car الى هيبقى فيه خطوات التركيب الخاصه بالعربيه قبل ما تبقى product

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_DP

{

internal class Car:IBuilder

{

private string BrandName;

private Product Product = new Product();

public Car(string \_BrandName)

{

this.BrandName = \_BrandName;

}

public void startOperations()

{

this.Product.Add($"Model Name {this.BrandName}");

}

public void BuildBody()

{

this.Product.Add($"Build Body");

}

public void InsertWheels()

{

this.Product.Add($"Insert Wheels");

}

public void InsertHeadLights()

{

this.Product.Add($"Insert Head Lights");

}

public void EndtOperations()

{

this.Product.Add("End Of Opertaions");

}

public Product GetVehicle()

{

return this.Product;

}

}

}

طيب احنا كده عملنا الخطوات و عملنا الكلاس الى هيدينى العربيه كمنتج نهائى تم تجميعه على هئيه عربيه حلوه كده المفروض بقه الى بقيلى انى اعمل حاجه يتحط فيها الخطوات بالترتيب الصحيح و الى يبقى عامل ذى كونتينر للخطوات بحيث منساش او الغبط خطوات صناعه العربيه فهعمل كلاس اسمه المدير Director ده الى هيدير عمليه تركيب القطع مع بعض قبل ما تبقى عربيه و علشان كده هيبقى فيه اوبجيكت من Builder علشان هعرف من ال Builder خطوات بناء العربيه و جوه ال Director هبقى مجرد بعملها ترتيب كويس و بعزل طريقه التركيب عن العالم كله و اخلي التركيب من المكان ده فقط اول ما بقوله Construct من ال Director فهينشئ البلدر الى هيرجع المنتج للعميل من غير تعب

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_DP

{

internal class Director

{

IBuilder Builder;

//steps to create Complex Object

public void Construct\_Prodcut(IBuilder \_Builder)

{

this.Builder = \_Builder;

Builder.startOperations();

Builder.BuildBody();

Builder.InsertHeadLights();

Builder.InsertWheels();

Builder.EndtOperations();

}

}

}

بعد كده خلاص انا اقدر انادى للبلدر من عند ال Client Side ذى ما انت شايف بالترتيب الاتى علشان كل حاجه تبقى واخده الترتيب الصح و التكوين الصح

using System;

namespace After\_Builder\_DP

{

internal class Program

{

static void Main(string[] args)

{

Director director = new Director();

IBuilder CarBuilder = new Car("BMW");

director.Construct\_Prodcut(CarBuilder);

Product product = CarBuilder.GetVehicle();

Console.WriteLine( product.Show());

}

}

}

طبعا لو هنتج حاجه غير العربيات ذى ال Motocycle مثلا كل هعمله انى هعمل Class لل Motocycle ارزع فيها امبلمنتيشن شبيه جدا بالعربيات ده هو فقط التغيير هيبقى فى التفاصيل الخاصه بالموتوسيكلات و لما اجى اناديله هنا للبليدر بما انه وارث منه هو كمان فكل الى هعمله انى هغير فى السطر التانى ليصيح شكله كده

IBuilder CarBuilder = new Motocycle ("Fisba");

##################################################################

Builder DP With Fluent API

-------------------------------------------------------------------

اولا كل الى هعمله هنا انى هخلى شكل النداء للبناء فى ال Director متسلسل عباره عن chain

طب ده يتعمل اذاى علشان يبقى شبه ال Build بتاع ال StartUps

هيتعمل كده اولا البروداكت مفيهوش اى تغيير اصلا

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_FluentAPI

{

internal class Product

{

private LinkedList<string> Parts;

public Product()

{

Parts = new LinkedList<string>();

}

//Collect Parts of Product

public void Add(string Part)

{

Parts.AddLast(Part);

}

// Show All parts of Products

public string Show()

{

StringBuilder sb = new StringBuilder();

sb.AppendLine("Product Components are : ");

foreach (string Part in Parts)

{

sb.AppendLine(Part);

}

return sb.ToString();

}

}

}

ثانى حاجه فى ال IBuilder هخلى كل ال Functions ترجع حاجه من نوع IBuilder لان الحوار ده الى هيخلينى اقدر اعمل ال Chain او سلسله الفانكشنز الى بتنادى على بعض

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_FluentAPI

{

internal interface IBuilder

{

IBuilder startOperations();

IBuilder BuildBody();

IBuilder InsertWheels();

IBuilder InsertHeadLights();

IBuilder EndtOperations();

Product GetVehicle();

}

}

اروح اعمل Implemetion جديد ليها فى ال Car كل الاختلاف انى هرجع الريتيرن بتاع الفانكشنز ل IBuilder برضوا

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_FluentAPI

{

internal class Car:IBuilder

{

private string BrandName;

private Product Product = new Product();

public Car(string \_BrandName)

{

this.BrandName = \_BrandName;

}

public IBuilder startOperations()

{

this.Product.Add($"Model Name {this.BrandName}");

return this;

}

public IBuilder BuildBody()

{

this.Product.Add($"Build Body");

return this;

}

public IBuilder InsertWheels()

{

this.Product.Add($"Insert Wheels");

return this;

}

public IBuilder InsertHeadLights()

{

this.Product.Add($"Insert Head Lights");

return this;

}

public IBuilder EndtOperations()

{

this.Product.Add("End Of Opertaions");

return this;

}

public Product GetVehicle()

{

return this.Product;

}

}

}

بعد كده فى ال Director انا اقدر اتعامل مع الفانكشنز على انها Chain متسلسله

using System;

using System.Collections.Generic;

using System.Text;

namespace After\_Builder\_FluentAPI

{

internal class Director

{

IBuilder Builder;

//steps to create Complex Object

public void Construct\_Prodcut(IBuilder \_Builder)

{

this.Builder = \_Builder;

Builder.startOperations().BuildBody().InsertHeadLights()

.InsertWheels().EndtOperations();

}

}

}

اما فى ال Client Side مش هيحصل اى تغيير فى الكود برضوا

using System;

namespace After\_Builder\_FluentAPI

{

internal class Program

{

static void Main(string[] args)

{

Director director = new Director();

IBuilder CarBuilder = new Car("BMW");

director.Construct\_Prodcut(CarBuilder);

Product product = CarBuilder.GetVehicle();

Console.WriteLine(product.Show());

}

}

}

طيب دلوقتى ده مثال كتيان مش موضح هستخدمه اذاى لان مستحيل انى ابقى عامل LinkedList و اقعد املاه استرينجات و خلاص اكيد هيبقى فى مجموعه من البرويرتى للعربيه الى بيها اقدر اتعامل مع الموضوع بشكل اسهل من الهبد ده

أول حاجه علشان ابنلك انى بستخدم object معقد و فيها برويرتيس كتير فقولت ايه انا هعمل كل الى كنت بعمله فى ال linkedlist على هيئه Properties و بالتالى كده انا ضمنت انى هملى البروبرتيس دى واحده واحده و بما انى عاوز املاها فطبعا خير مكان انى اعملها الامتلاء بتعها يحصل مره واحده فقط لما اجى انشأ ال object من الكونستراكتور يا باشا فهيبقى الكود بتاع كلاس ال Product كده

using System;

using System.Collections.Generic;

using System.Reflection;

using System.Text;

namespace Builder\_FluentAPI\_With\_RealExample

{

internal class Product

{

public Product(string productSerial = null, string model =null, string bodyPart = null,

string wheelsPart = null,

string headLightPart = null)

{

ProductSerial = productSerial;

Model = model;

BodyPart = bodyPart;

WheelsPart = wheelsPart;

HeadLightPart = headLightPart;

}

// Must be Ordered

public string Model { get; set; }

public string BodyPart { get; set; }

public string WheelsPart { get; set; }

public string HeadLightPart { get; set; }

public string ProductSerial { get; set; }

// Show All parts of Products

public string Show()

{

StringBuilder sb = new StringBuilder();

sb.AppendLine("Product Components are : ");

foreach (PropertyInfo propertyInfo in this.GetType().GetProperties())

{

if (propertyInfo.GetValue(this, null) != null)

sb.AppendLine($"{propertyInfo.Name} --> {propertyInfo.GetValue(this, null)}");

}

//if (Model != null)

// sb.AppendLine($"Model --> {Model}");

//if (BodyPart != null)

// sb.AppendLine($"BodyPart --> {BodyPart}");

//if (WheelsPart != null)

// sb.AppendLine($"WheelsPart --> {WheelsPart}");

//if (HeadLightPart != null)

// sb.AppendLine($"HeadLightPart --> {HeadLightPart}");

//if (ProductSerial != null)

// sb.AppendLine($"ProductSerial --> {ProductSerial}");

return sb.ToString();

}

}

}

كده انا عندى شويه properties ببعتلها داتا فى ال Builder عن طريق الفانكشنز بتاعت ال Build قطعه قطعه و بعد كده بناديها و اقولها اتملى كلك يا حبيبتى بقه فى ال Constructor بتاعك و هظهرك انا بقه فى الكلاس بتاعك الى هيبقى بيتعامل مع العميل بشكل مباشر و بالتالى ال Builder هيبقى مكتوب كده ال interface بتاعه

using System;

using System.Collections.Generic;

using System.Text;

namespace Builder\_FluentAPI\_With\_RealExample

{

internal interface IBuilder

{

IBuilder startOperations();

IBuilder BuildBody();

IBuilder InsertWheels();

IBuilder InsertHeadLights();

IBuilder EndtOperations();

Product GetVehicle();

}

}

انما البلدر نفسه الى هو فى الاخر car / Car Builder هيبقى كده

using System;

using System.Collections.Generic;

using System.Text;

namespace Builder\_FluentAPI\_With\_RealExample

{

internal class Car:IBuilder

{

private string BrandName;

private Product Product = new Product();

public Car(string \_BrandName)

{

this.BrandName = \_BrandName;

}

public IBuilder startOperations()

{

this.Product.Model= $"Model Name {this.BrandName}";

return this;

}

public IBuilder BuildBody()

{

this.Product.BodyPart = $"Build Body";

return this;

}

public IBuilder InsertWheels()

{

this.Product.WheelsPart = $"Insert Wheels";

return this;

}

public IBuilder InsertHeadLights()

{

this.Product.HeadLightPart= $"Insert Head Lights";

return this;

}

public IBuilder EndtOperations()

{

this.Product.ProductSerial = ($"End Of Opertaions With product Serial = {Guid.NewGuid()}");

return this;

}

public Product GetVehicle()

{

return this.Product;

}

}

}

بملأ فى كل فانكشن الجزئيه الخاصه بالبروبرتى بتعها كانها بعمل Set للبروبرتى هنا و برجع بعد كده ال Builder class علشان ابقى عامل Chain للفانكشن و كده كده هو ملأ ال product properties واحده بواحده و لما بحتاج انشأ او ارجع البيانات الخاصه بالمنتج نفسه بعمله فى ال GetVehicle

كده نقدر نعمل ال Director الى هيبقى فيه اختيارات البناء ثواء كل حاجه او بعضها و احنا مطمنين

using System;

using System.Collections.Generic;

using System.Text;

namespace Builder\_FluentAPI\_With\_RealExample

{

internal class Director

{

IBuilder Builder;

//steps to create Complex Object

public void Construct\_Prodcut(IBuilder \_Builder)

{

this.Builder = \_Builder;

Builder.startOperations().BuildBody().InsertHeadLights()

.InsertWheels().EndtOperations();

}

}

}

كده برضوا نقدر بقه نخلى ال client يختار العربيه الى عاوزها و هتبنى كلها او جزء منها على حسب ما هو عاوز

using System;

namespace Builder\_FluentAPI\_With\_RealExample

{

internal class Program

{

static void Main(string[] args)

{

Director director = new Director();

IBuilder CarBuilder = new Car("BMW");

director.Construct\_Prodcut(CarBuilder);

Product product = CarBuilder.GetVehicle();

Console.WriteLine(product.Show());

// can Build Some Parts Only

IBuilder SomePartsCarBuilder = new Car("BMW Some Parts");

SomePartsCarBuilder.startOperations().BuildBody().EndtOperations();

product = SomePartsCarBuilder.GetVehicle();

Console.WriteLine(product.Show());

}

}

}

##################################################################

Prototype DP

---------------------------

البروتوتايب ديزاين باترن بيتعامل مع نوعيه الاوبجيكتس الى بتاخد وفت كبير على ما بتتنشئ و ده لانى لو بتعامل مثلا مع داتا بيز و بملى بيها الاوبجيكت فكده الاوبجيكت ده هياخد وقت كبير على ما يتعمل .

فالبروتوتايب بيحل المشكله دى و ده عن طريق انه بياخد نسخه من الاوبجيكت الى جاى من الداتا بيز ده مع بعض التغييرات الصغيره عليه لو عاوز و بالتالى لما اجى اتعامل مع البيانات تانى فبتعامل مع داتا موجوده متخزنه من المره الاولى مش لسه هروح للداتا بيز و استنى لما الداتا بيز ترجع و تاخد وقت على ما تملى الابوجيكت تانى و حوار كبير بقه لأ هو ذى ما تقول كده بيعمل نسخه من الداتا و يخزنها عنده فلما تحتاج الداتا دى ترحلها على طول من غير ما تروح للداتا بيز.

Copy heavy intialization object instead of creating it again.

البروتوتايب بينقسم ل   
Shallow Clone & Deep Clone

Shallow Clone : طريقه الكوبى دى بينسخ الاوبجيكت ذى ما هو بالظبط من الاصلى و بعد كده لو انت عملت اى تغييرات على اى اوبجيكت منهم فى بعض الحاجات مش بتتاثر و بتبقى مختلفه فى الاوبجيكت الاول عن التانى و دى ال Value types : ذى ال int , double , bool, enum , struct, datetime

و فى نوع تانى اسمه Reference Types ده اى تغيير فى الاوبجيكت الاول بيحصل نفس التغيير فى الاوبجيكت التانى و اى تغيير فى الاوبجيكت التانى بيغير على طول برضوا فى الابويجكت الاول بمعنى اصح الحاجات ال Reference type بتبص فى الميمورى على نفس ال value فلما بيحصل تغيير لل value فالاتنين بما انهم بيبصوا على نفس الريفرينس فبيشوفوا القيمه الى اتغييرت و بالتالى الاتنين قيمتها بتتغير ذى بعض ال Reference types ذى   
Complex class , interface , delegate

Deep Clone : ده برنس بيعمل كوبى من الاوبجيكت كله سواء reference or value type و كانه اوبجيكت منفصل تماما اى تغيير فى اى اوبجيكت منهم مش بياثر على التانى خالص ملوش دعوه بيه يعنى من الاخر .

UML Of Prototype DP

A screenshot of a computer

Description automatically generated with medium confidence

و ده كود ال Shallow copy بسيط جدا

أول حاجه هنعمل ال Abstract Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShallowCopy

{

    public abstract class PrototypeEmployee

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public EmpAddresss protEmpAddresss { get; set; }

        public abstract PrototypeEmployee ShallowClone();

        public override string ToString()

        {

            return $"Emp Id : {this.Id}  - Name : {this.Name} - City : {protEmpAddresss.City} - Street : {protEmpAddresss.Street} - BuildingNum : {protEmpAddresss.BuildingNum} " ;

        }

    }

}

ثم بعد بعمل ال Abstract Class المفروض هو ده الى هعمله implementation  
فعمله inherite ثم هعمله implementation

و اصلا ال implementation هيتعمل لفانكشن واحده الى هى بقوله انسخلى الاوبجيكت ك Shallow copy

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShallowCopy

{

    public class Employee : PrototypeEmployee

    {

        public override PrototypeEmployee ShallowClone()

        {

            return (Employee)this.MemberwiseClone();

        }

    }

}

بما ان انا عندى reference type من نوع EmpAddress فى ال properties بتاعت ال abstract class اذا لازم اكون عامل للكلاس الجديده دى فايل لوحده فيه properties الكلاس

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShallowCopy

{

   public class EmpAddresss

    {

        public string City { get; set; }

        public string Street { get; set; }

        public int BuildingNum { get; set; }

    }

}

كده كل حاجه جهزه انه يشتغل مش ناقص غير ان ال Client يكلم فانكشن ال shallow Clone علشان يبتدى ينفذها و نقارن بين نتايج النسخ قبل و بعد تغيير ال reference and value types وده هنعرفه لما نعمل كود ال program و نقارن بينهم

using System;

namespace ShallowCopy

{

    class Program

    {

        static void Main(string[] args)

        {

            PrototypeEmployee Original\_Emp = new Employee();

            Original\_Emp.Id = 1;

            Original\_Emp.Name = "Mahmoud Badawy";

            Original\_Emp.protEmpAddresss = new EmpAddresss

            {

                City = "Cairo",

                Street = "Badawy St",

                BuildingNum = 20

            };

            PrototypeEmployee ShallowCopy\_Emp = Original\_Emp.ShallowClone();

            Console.WriteLine($" Original => " + Original\_Emp.ToString());

            Console.WriteLine($" Shallow => " + ShallowCopy\_Emp.ToString());

            Console.WriteLine("");

            Console.WriteLine(" ####################################### ");

            Console.WriteLine(" Change Original Object and you will find that shallow clone make" +

                " Refrence types values changed in Shallow Object as orginal except string type");

            Console.WriteLine(" ####################################### ");

            Console.WriteLine("");

            Original\_Emp.Id = 2;

            Original\_Emp.Name = "Shallow M Badawy";

            Original\_Emp.protEmpAddresss.City = "Shallow New Cairo";

            Original\_Emp.protEmpAddresss.Street = "Shallow Badawy Street";

            Original\_Emp.protEmpAddresss.BuildingNum = 25;

            Console.WriteLine($" Original after Change => " + Original\_Emp.ToString());

            Console.WriteLine($" Shallow  after Change  => " + ShallowCopy\_Emp.ToString());

            Console.WriteLine(" ####################################### ");

        }

    }

}

من النتايج تم ملاحظه ان فى ال Shallow copy الاوبجيكت الاصلى اذا تم تغيير فيه Value type لا يغيير من الاوبجيكت المنسوخ اما لو كان التغيير فى ال reference Type فتختلف فيمه الاوبجيكت الاصلى اذا تم تغيير الاوبجيكت المنسوخ و العكس صحيح برضوا يعنى الى هيتغير فيهم هيسمع تغييره فى التانى و جه لان الاتنين بيبصوا على نفس الريفرنس و بالتالى بيبصوا على نفس القيمه

اما فى ال Deep Clone يختلف الوضع تماما حيث ان الاوبجيكت المنسوخ ليس له اى علاقه بالاوبجيكت الاصلى بعد نسخه فا اى تغيير فى اى حاجه فى الاوبجيكت الاصلى لا يؤثر ابدا على الاوبجيكت المنسوخ و كذلك العكس .

و ده كود ال Deep Clone التغيير كله فطريقه النسخ فقط و الباقى مفيهوش اى تغييرات

ده ال abstract Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeepCopy

{

    public abstract class PrototypeEmployee

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public EmpAddresss protEmpAddresss { get; set; }

        public abstract PrototypeEmployee DeepClone();

        public override string ToString()

        {

            return $"Emp Id : {this.Id}  - Name : {this.Name} - City : {protEmpAddresss.City} - Street : {protEmpAddresss.Street} - BuildingNum : {protEmpAddresss.BuildingNum} " ;

        }

    }

}

ده كلاس ال EmpAddress

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeepCopy

{

   public class EmpAddresss

    {

        public string City { get; set; }

        public string Street { get; set; }

        public int BuildingNum { get; set; }

    }

}

و ده الكلاس الى هنعمل جزاه الامبلمنتيشن خلاص

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DeepCopy

{

    public class Employee : PrototypeEmployee

    {

        public override PrototypeEmployee DeepClone()

        {

            Employee Emp = new Employee();

            Emp = (Employee)this.MemberwiseClone();

            // Changes in Refrence Types.

            EmpAddresss EmpAddr = new EmpAddresss();

            EmpAddr.City = protEmpAddresss.City;

            EmpAddr.Street = protEmpAddresss.Street;

            EmpAddr.BuildingNum = protEmpAddresss.BuildingNum;

            Emp.protEmpAddresss = EmpAddr;

            Emp.Id = this.Id;

            Emp.Name = this.Name;

            return Emp;

        }

    }

}

ده ال Client

using System;

namespace DeepCopy

{

    class Program

    {

        static void Main(string[] args)

        {

            PrototypeEmployee Original\_Emp = new Employee();

            Original\_Emp.Id = 1;

            Original\_Emp.Name = "Mahmoud Badawy";

            Original\_Emp.protEmpAddresss = new EmpAddresss

            {

                City = "Cairo",

                Street = "Badawy St",

                BuildingNum = 20

            };

            PrototypeEmployee DeepCopy\_Emp = Original\_Emp.DeepClone();

            Console.WriteLine($" Original => " + Original\_Emp.ToString());

            Console.WriteLine($" DeepCopy => " + DeepCopy\_Emp.ToString());

            Console.WriteLine("");

            Console.WriteLine(" ####################################### ");

            Console.WriteLine(" Change Orginal Will not Affect Deep Object Ever .");

            Console.WriteLine(" ####################################### ");

            Console.WriteLine("");

            Original\_Emp.Id = 3;

            Original\_Emp.Name = "Deep M Badawy";

            Original\_Emp.protEmpAddresss.City = "Deep New Cairo";

            Original\_Emp.protEmpAddresss.Street = "Deep Badawy Street";

            Original\_Emp.protEmpAddresss.BuildingNum = 50;

            Console.WriteLine($" Original after Change => " + Original\_Emp.ToString());

            Console.WriteLine($" Deep obj after Change => " + DeepCopy\_Emp.ToString());

            Console.WriteLine(" ####################################### ");

        }

    }

}

من النتايج هنلاحظ ان اى تغيير فى النسخه لا يؤثر بتاتا على الاصل و العكس صحيح ايضا .

##################################################################

Proxy DP

---------------------------