C# Summery part-1 By Mahmoud Badawy

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

الملخص يحتوى على

- Design Patterns

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

MB

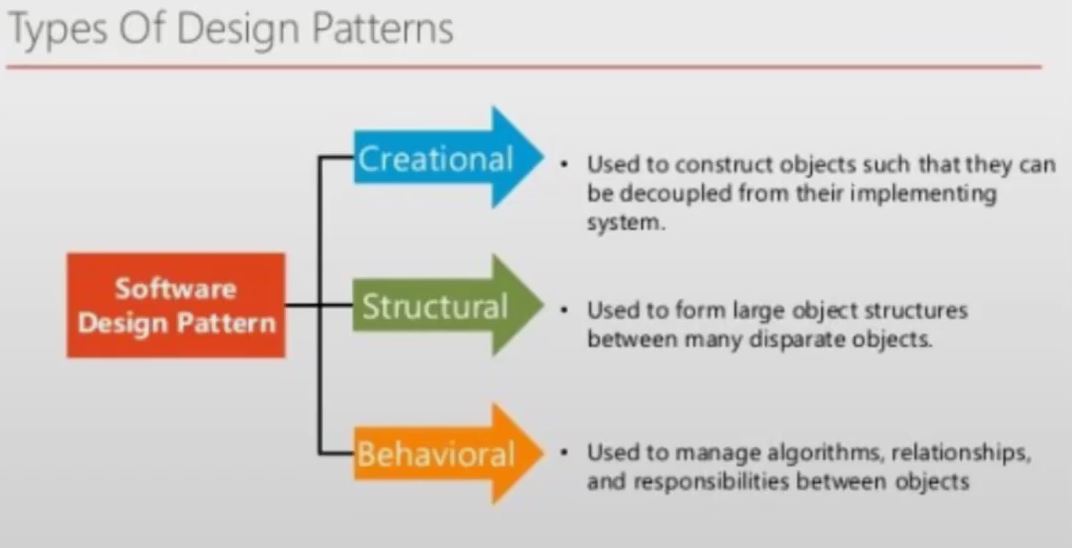
##################################################################

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

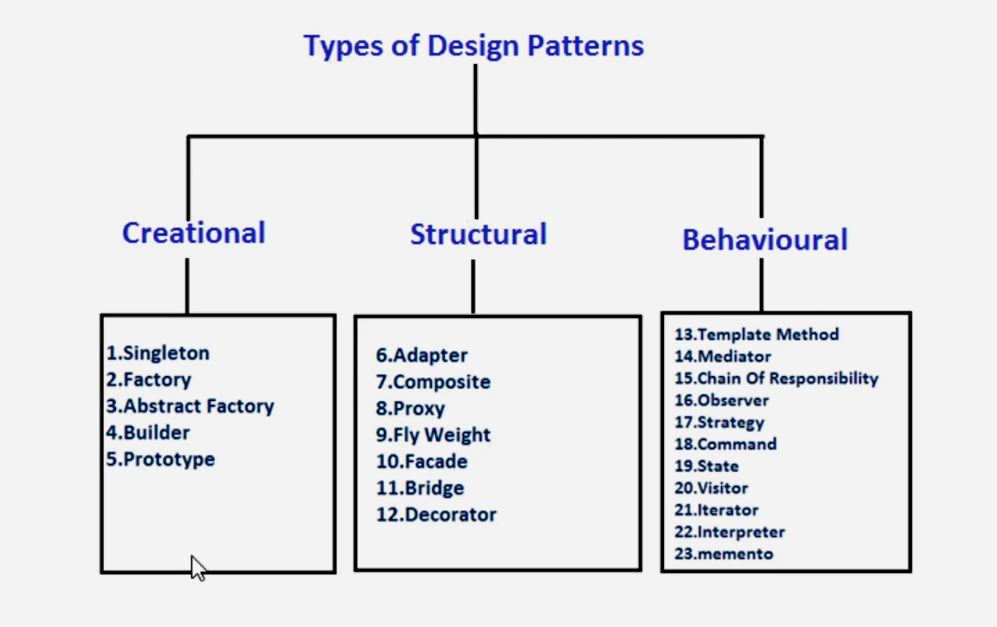
فى تلات انواع للديزاين باترن



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 .

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++;

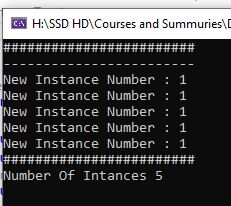
Console.WriteLine($"New Instance Number : " + Count);

}

}

}

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();

}

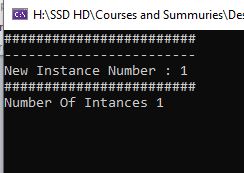
return Instance;

}

}

}

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}");

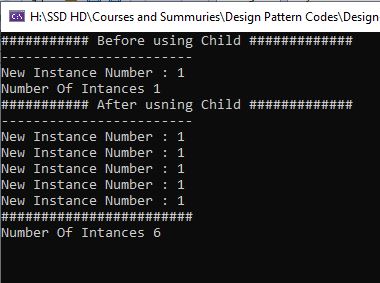
Console.ReadKey(true);

}

}

}

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}");

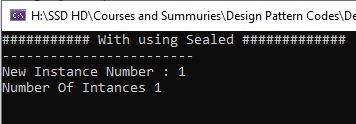
Console.ReadKey(true);

}

}

}

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();

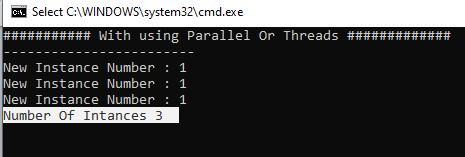
}

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();

}

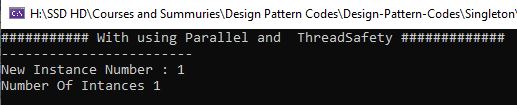
}

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

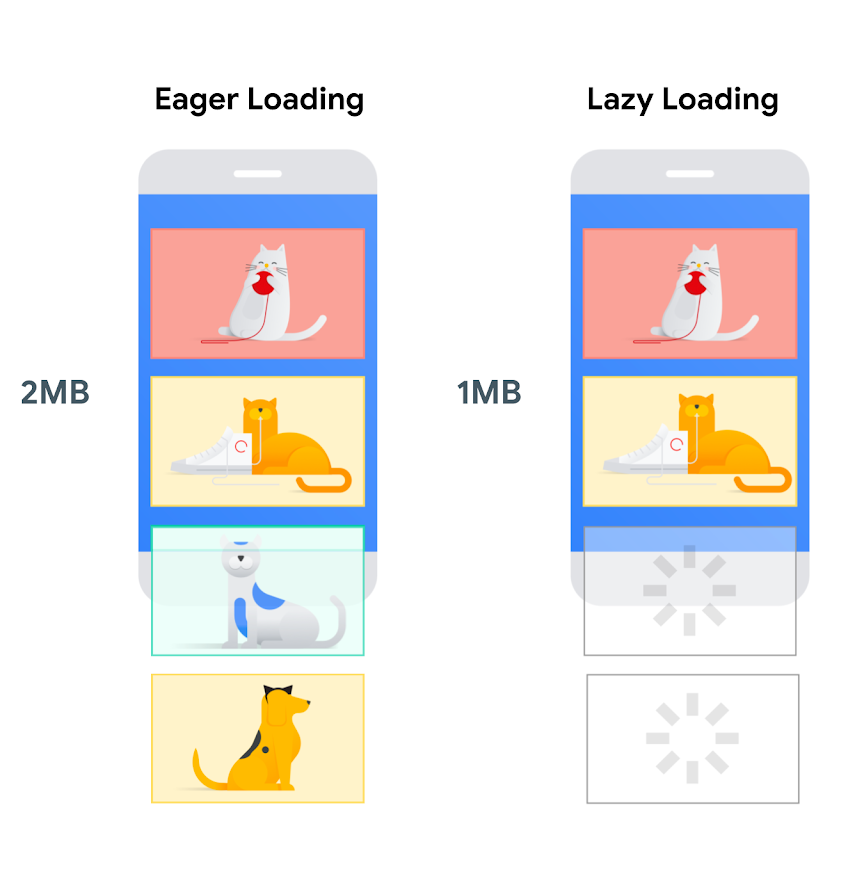
بص يا سيدى ال Lazy Loading ده بستخدمه لما اكون مش عاوز اجيب الداتا كلها و عاوز فقط اجيب الداتا الى محتاجها فقط بمعنى دلوقتى انا عندى كميه داتا كبيره جدا جدا فى البنك عن العملاء مثلا و انا دلوقتى عاوز اخش عليها اعرف بيانات اتنين او تلاته عملاء هيبقى ايه المفيد انى اجيب كل الداتا ؟؟ ولا اي حاجه بالعكس 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;

}

}

}

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

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

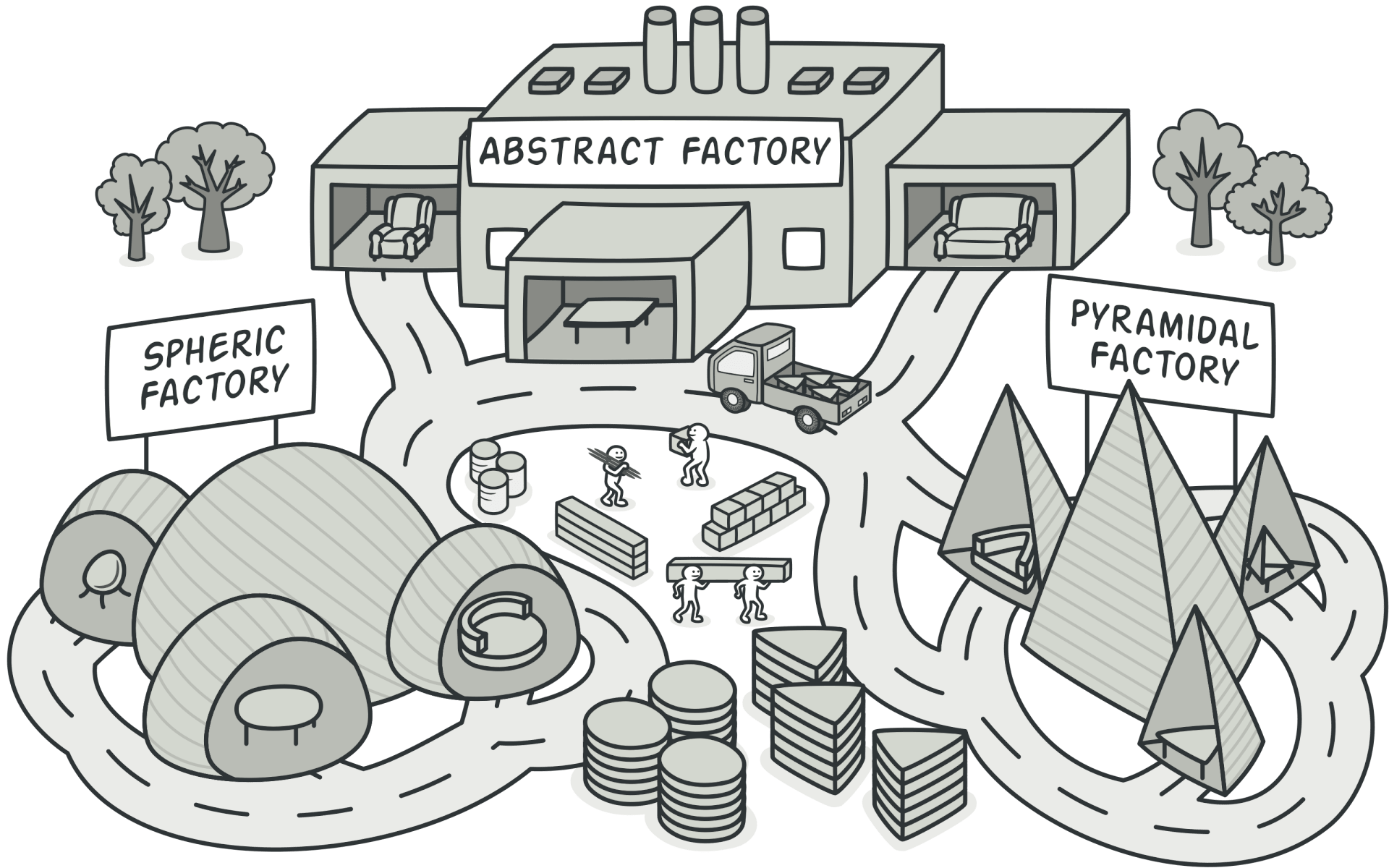
بحيث كده المصنع كده هيبقى بيصنع الى بيصنعه محدش عارف هو بيصنع ايه الا لما المنتج بتاعه يخرج و يقول انا بقيت عربيه انا بقيت كوتش انا بقيت اكسسوارات عربيات كده يعنى انما طول ما المنتج محطوط جوه الكرتونه متفتحتش محدش هيبقى عارف ايه الى جوه الكرتونه دى و الكرتونه هنا يعنى نعمله abstraction يا صديقى

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

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

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