C# Summery OOP By Mahmoud Badawy

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

ملخص سى شارب الجزء الاول

الملخص يحتوى على

- OOP

الفيديوهات المستخدمه فى الشرح   
<https://www.youtube.com/channel/UCgRFf3_D5H1Qi8pvw2Czyzg/playlists>

**MB**

**################################################################**

**NameSpace**

**--------------------------**

بص يا سيدى دى يعتبر معموله للتنظيم عشان احط فيها كلاسات و انموريشنز و ديليجيتس براحتى و تبقى تابعه للنيم سبيس الفلانى و يعتبر ال dll بيتعمل باسمها برضوا و بيبقى لامم كميه كبيره من الكلاسات و الانترفيسيس و اى حاجه ينفع تتحط جوه نيم سبيس عمتا

لاحظ ان ينفع تحط نيم سبيس جوه نيم سبيس و عشان انادى اى حاجه من جوه نيم سبيس طبعا بعتبرها مكتبه هجيب منها حاجه و بالتالى بقوله using و اكتبله اسم النيم سبيس و افضل اتتبع الحاجه لحد ما اجيبها بقه لان ممكن يبقى الحاجه جوه كلاس و الكلاس جوه نيم سبيس و النيم سبيس جوه نيم سبيس و كمان ممكن يبقى في نيم سبيسيس باسامى متشابهه فساعتها بعمل Aliases عشان افرق بينهم و هيصه كده ربنا الى عالم بيها و عمتا عشان تفهم ادينى عملت مثال

using PATA = ProgectA.TeamA; // namespace have Alias PATA

using PATB = ProgectA.TeamB; // namespace have Alias PATB

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

ProgectA.TeamA.MyCls.Myfunc();

ProgectA.TeamB.MyCls.Myfunc();

PATA.MyCls.Myfunc();

PATB.MyCls.Myfunc();

}

}

}

namespace ProgectA

{

namespace TeamA

{

class MyCls

{

public static void Myfunc()

{

Console.WriteLine("Hello From ProgectA > TeamA > MyCls ");

}

}

}

}

namespace ProgectA

{

namespace TeamB

{

class MyCls

{

public static void Myfunc()

{

Console.WriteLine("Hello From ProgectA > TeamB > MyCls ");

}

}

}

}

MB

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

NameSpace

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

يمكن كتابه الكود الى فات بطريقتين الطريقه الأولى

النيم سبيس أحيانا لما بيبقى تحت السوليوشن على طول بيبقى عباره عن بروجيكت الى جوه السوليوشن و طبعا انت عارف ان السوليوشن ممكن يشيل 5000000 بروجيكت عادى

using PATA = ProgectA.TeamA; // namespace have Alias PATA

using PATB = ProgectA.TeamB; // namespace have Alias PATB

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

ProgectA.TeamA.MyCls.Myfunc();

ProgectA.TeamB.MyCls.Myfunc();

PATA.MyCls.Myfunc();

PATB.MyCls.Myfunc();

}

}

}

namespace ProgectA.TeamA

{

class MyCls

{

public static void Myfunc()

{

Console.WriteLine("Hello From ProgectA > TeamA > MyCls ");

}

}

}

namespace ProgectA.TeamB

{

class MyCls

{

public static void Myfunc()

{

Console.WriteLine("Hello From ProgectA > TeamB > MyCls ");

}

}

}

الطريقه التانيه ان النيم سبيس الى متشابه في الاسم اجمع فيه كل النيم سبيسيس و الكلاسات الخاصه بيه بدل ما انطور اسمه في كل حته على الفاضى و خلاص

using PATA = ProgectA.TeamA; // namespace have Alias PATA

using PATB = ProgectA.TeamB; // namespace have Alias PATB

namespace LearnProg

{

class Program

{

static void Main(string[] args)

{

ProgectA.TeamA.MyCls.Myfunc();

ProgectA.TeamB.MyCls.Myfunc();

PATA.MyCls.Myfunc();

PATB.MyCls.Myfunc();

}

}

}

namespace ProgectA

{

namespace TeamA

{

class MyCls

{

public static void Myfunc()

{

Console.WriteLine("Hello From ProgectA > TeamA > MyCls ");

}

}

}

namespace TeamB

{

class MyCls

{

public static void Myfunc()

{

Console.WriteLine("Hello From ProgectA > TeamB > MyCls ");

}

}

}

}

MB

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

Why OOP

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

بص يا سيدى الحكايه ابتدت ان احنا عندنا برنامج ذى ده بديله الاسم و اجر الساعه و ساعات الشغل الى اشتغلتها و الضرايب و يحسبلى صافى المرتب ليا

using System;

class Program

{

static void Main(string[] args)

{

// const لان الضريبه مش هتتغير

const double Tax = 0.03;

Console.WriteLine("First Name : ");

var FName = Console.ReadLine();

Console.WriteLine("Last Name : ");

var LName = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

var Wage = Convert.ToDouble( Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

var LoggedHours = Convert.ToDouble(Console.ReadLine());

var Salary = Wage \* LoggedHours;

var Taxs = Salary \* Tax;

var NetSalary = Salary - Taxs;

Console.WriteLine($"First Name : {FName}");

Console.WriteLine($"Last Name : {LName}");

Console.WriteLine($"Wage per hour : {Wage}" );

Console.WriteLine($"Logged Hours : {LoggedHours}" );

Console.WriteLine($"Salary :{Salary}");

Console.WriteLine($"Taxs :{Taxs}");

Console.WriteLine($"Net Salary :{NetSalary}" );

}

}

ده بيحسبها مره واحده و بعد كده شكرا البرنامج يقفل و افتحه تانى طب لو عاوز احسبها ليا و لصاحبى فى نفس السيشن مثلا ؟؟

الطبيعى هعمل كده و اكرر الكود مرتين

using System;

class Program

{

static void Main(string[] args)

{

const double Tax = 0.03;

Console.WriteLine("First Name : ");

var FName = Console.ReadLine();

Console.WriteLine("Last Name : ");

var LName = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

var Wage = Convert.ToDouble( Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

var LoggedHours = Convert.ToDouble(Console.ReadLine());

var Salary = Wage \* LoggedHours;

var Taxs = Salary \* Tax;

var NetSalary = Salary - Taxs;

Console.WriteLine($"First Name : {FName}");

Console.WriteLine($"Last Name : {LName}");

Console.WriteLine($"Wage per hour : {Wage}" );

Console.WriteLine($"Logged Hours : {LoggedHours}" );

Console.WriteLine($"Salary :{Salary}");

Console.WriteLine($"Taxs :{Taxs}");

Console.WriteLine($"Net Salary :{NetSalary}" );

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

const double Tax2 = 0.03;

Console.WriteLine("First Name : ");

var FName2 = Console.ReadLine();

Console.WriteLine("Last Name : ");

var LName2 = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

var Wage2 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

var LoggedHours2 = Convert.ToDouble(Console.ReadLine());

var Salary2 = Wage \* LoggedHours;

var Taxs2 = Salary \* Tax;

var NetSalary2 = Salary - Taxs;

Console.WriteLine($"First Name : {FName2}");

Console.WriteLine($"Last Name : {LName2}");

Console.WriteLine($"Wage per hour : {Wage2}");

Console.WriteLine($"Logged Hours : {LoggedHours2}");

Console.WriteLine($"Salary :{Salary2}");

Console.WriteLine($"Taxs :{Taxs2}");

Console.WriteLine($"Net Salary :{NetSalary2}");

}

}

كده هيعمل الشغل ده لشخصين فى الفاتحه الواحده للبرنامج

طب لو عاوز اعمله مليون مره بقه ؟؟

هل ينفع انى اقعد اكرر الكود مليون مره ؟؟

اكيد لأ اكيد هيجى مره و اتلغبط فى كتابه الفاريبولز الى هبقى عاملها دى ده هيبقى مليون مره بعمل الفاريبولز انت متخيل ههههههههههههههههههه اكيد الاسامى هتخش فى بعض فى اوقات كتير

طب و ايه الحل ؟

الحل هو انى اشتغل OOP و اعمل كلاس و يبقى الكلاس ده نموذج اقدر اعمل منه نسخ كتير من غير ما يحصل اى لغبطه

اكتب الكود ده مره واحده فى الكلاس و لما اعوز اشغله اعمل انستانس من الكلاس يشغلى الكود ده فيه و بالتالى انا كده اقدر اعمل مليون نسخه على هيئه object او instance من الكلاس منه وفرت على نفسى تعديل كود كل شويه و منى وفرت ميمورى لانى مش هقعد اكريت مليون فاريبول لا ده انستانس بيتكريت من الكلاس و لما بيخلص شغله بيروح خارج من الميمورى على طول و طبعا مش هيحصل لغبطه و انا شغال فى حته الفاريبولز دى لانى هبقى بعمل اوبجيكت و الابجيكت جواه خصائصه الى نسخها من الكلاس

طب الكلام ده يتعمل OOP ااذاى ؟

اعمل كلاس يا اسطى الاول فيه Fields و Method لنفس الحاجه الى انت شغال عليها و ناديله ذى ما هوريك فى الكود كده بالظبط

using System;

class Program

{

static void Main(string[] args)

{

Employee Emp = new Employee();

Console.WriteLine("First Name : ");

Emp.FName = Console.ReadLine();

Console.WriteLine("Last Name : ");

Emp.LName = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

Emp.Wage = Convert.ToDouble( Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

Emp.LoggedHours = Convert.ToDouble(Console.ReadLine());

Emp.Salary = Emp.Wage \* Emp.LoggedHours;

Emp.Taxs = Emp.Salary \* Employee.Tax;

Emp.NetSalary = Emp.Salary - Emp.Taxs;

Console.WriteLine($"First Name : {Emp.FName}");

Console.WriteLine($"Last Name : {Emp.LName}");

Console.WriteLine($"Wage per hour : {Emp.Wage}" );

Console.WriteLine($"Logged Hours : {Emp.LoggedHours}" );

Console.WriteLine($"Salary :{Emp.Salary}");

Console.WriteLine($"Taxs :{Emp.Taxs}");

Console.WriteLine($"Net Salary :{Emp.NetSalary}" );

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

Employee Emp2 = new Employee();

Console.WriteLine("First Name : ");

Emp2.FName = Console.ReadLine();

Console.WriteLine("Last Name : ");

Emp2.LName = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

Emp2.Wage = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

Emp2.LoggedHours = Convert.ToDouble(Console.ReadLine());

Emp2.Salary = Emp2.Wage \* Emp2.LoggedHours;

Emp2.Taxs = Emp2.Salary \* Employee.Tax;

Emp2.NetSalary = Emp2.Salary - Emp2.Taxs;

Console.WriteLine($"First Name : {Emp2.FName}");

Console.WriteLine($"Last Name : {Emp2.LName}");

Console.WriteLine($"Wage per hour : {Emp2.Wage}");

Console.WriteLine($"Logged Hours : {Emp2.LoggedHours}");

Console.WriteLine($"Salary :{Emp2.Salary}");

Console.WriteLine($"Taxs :{Emp2.Taxs}");

Console.WriteLine($"Net Salary :{Emp2.NetSalary}");

}

}

public class Employee

{

public const double Tax = 0.03;

public string FName;

public string LName;

public double Wage;

public double LoggedHours;

public double Salary;

public double Taxs;

public double NetSalary;

}

طيب هل ممكن اصغر البرنامج و مقعدش اقرر كده كتير ؟؟

ايوه ممكن اعمل ال Emp عباره عن Array شايله كل الموظفين و الف على موظف موظف بالفور ايتش اعمله شغله ذى كده

using System;

class Program

{

static void Main(string[] args)

{

Employee[] Emp = new Employee[2];

Emp[0] = new Employee();

Emp[1] = new Employee();

foreach (var item in Emp)

{

Console.WriteLine("First Name : ");

item.FName = Console.ReadLine();

Console.WriteLine("Last Name : ");

item.LName = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

item.Wage = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

item.LoggedHours = Convert.ToDouble(Console.ReadLine());

item.Salary = item.Wage \* item.LoggedHours;

item.Taxs = item.Salary \* Employee.Tax;

item.NetSalary = item.Salary - item.Taxs;

Console.WriteLine($"First Name : {item.FName}");

Console.WriteLine($"Last Name : {item.LName}");

Console.WriteLine($"Wage per hour : {item.Wage}");

Console.WriteLine($"Logged Hours : {item.LoggedHours}");

Console.WriteLine($"Salary :{item.Salary}");

Console.WriteLine($"Taxs :{item.Taxs}");

Console.WriteLine($"Net Salary :{item.NetSalary}");

}

}

}

public class Employee

{

public const double Tax = 0.03;

public string FName;

public string LName;

public double Wage;

public double LoggedHours;

public double Salary;

public double Taxs;

public double NetSalary;

}

كده بكود واحد عملت اتنين موظفين و اقدر اعمل مليون موظف كمان

اما هنا بقه فلو عاوز اعمل مليون موظف هعمله براحتى و الكود اصغر من الاول كمان

using System;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Num of Employees in System : ");

var Num = Convert.ToInt32( Console.ReadLine());

List<Employee> Emp = new List<Employee>();

for (int i = 0; i < Num; i++)

{

Emp.Add(new Employee());

}

foreach (var item in Emp)

{

Console.WriteLine("First Name : ");

item.FName = Console.ReadLine();

Console.WriteLine("Last Name : ");

item.LName = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

item.Wage = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

item.LoggedHours = Convert.ToDouble(Console.ReadLine());

item.Salary = item.Wage \* item.LoggedHours;

item.Taxs = item.Salary \* Employee.Tax;

item.NetSalary = item.Salary - item.Taxs;

Console.WriteLine($"First Name : {item.FName}");

Console.WriteLine($"Last Name : {item.LName}");

Console.WriteLine($"Wage per hour : {item.Wage}");

Console.WriteLine($"Logged Hours : {item.LoggedHours}");

Console.WriteLine($"Salary :{item.Salary}");

Console.WriteLine($"Taxs :{item.Taxs}");

Console.WriteLine($"Net Salary :{item.NetSalary}");

}

}

}

public class Employee

{

public const double Tax = 0.03;

public string FName;

public string LName;

public double Wage;

public double LoggedHours;

public double Salary;

public double Taxs;

public double NetSalary;

}

مع كل الى فوق ده احنا لسه محسناش البرنامج اصلا و لسه مطبقناش فيه حوار ال Method فى ال OOP بشكل كلى فلما هتشوفه فى الاخر هتلاقى ان فعلا ال OOP دى فله شمعه منوره حرفيا

MB

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

Const Vs Static field Vs Normal Fields

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

لاحظ فى المثال السابق بقه ان ال Fields كلها ظهرت او جت و قدرت اتحكم فيها عن طريق الاوبجيكت و ده لان لكل موظف اسمه الخاص و راتبه و ساعاته الخاصه

بينما الوحيد الى مجاش عن طريق الاوبجيكت و جه عن طريق ال Class Name هو ال Const

Emp.Taxs = Emp.Salary \* Employee.Tax;

و ده لان ال Const هو عباره عن field مشترك للكل بيتكريت مره واحده داخل الكلاس و اى نسخه من الكلاس بتاخد نفس القيمه لانه مشترك بين كل ال instance الى بتتعمل من الكلاس و مبيتغيرش نهائى طالما اتعمله initialization

الحاله الوحيده الى بقدر اغيره فيها جوه ال constructor و الحاله دى هنشرحها بعدين عموما

طب لو عاوز اغير عليه اعمل ايه ؟؟

استخدم Static بدلا من ال Const و ده لان ال Static بيبقى قيمته مشتركه بين كل ال instances ذيه ذى ال const و لكن الفرق بينهم ان ال static بينفع اغيره والتغيير ده بيتشاف فى كل ال instances برضوا

و بالتالى اقدر اتحكم فيها كده ذى ما انت شايف فى التعديله الجديده دى

using System;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Num of Employees in System : ");

var Num = Convert.ToInt32( Console.ReadLine());

List<Employee> Emp = new List<Employee>();

for (int i = 0; i < Num; i++)

{

Emp.Add(new Employee());

}

foreach (var item in Emp)

{

Console.WriteLine("Tax from 0 to 100 % : ");

Employee.Tax = Convert.ToDouble( Console.ReadLine()) \* 0.01;

Console.WriteLine("First Name : ");

item.FName = Console.ReadLine();

Console.WriteLine("Last Name : ");

item.LName = Console.ReadLine();

Console.WriteLine("Your Wage (Rateb per hour) : ");

item.Wage = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Your Logged hours (Working hours) : ");

item.LoggedHours = Convert.ToDouble(Console.ReadLine());

item.Salary = item.Wage \* item.LoggedHours;

item.Taxs = item.Salary \* Employee.Tax;

item.NetSalary = item.Salary - item.Taxs;

Console.WriteLine($"First Name : {item.FName}");

Console.WriteLine($"Last Name : {item.LName}");

Console.WriteLine($"Wage per hour : {item.Wage}");

Console.WriteLine($"Logged Hours : {item.LoggedHours}");

Console.WriteLine($"Salary :{item.Salary}");

Console.WriteLine($"Taxs :{item.Taxs}");

Console.WriteLine($"Net Salary :{item.NetSalary}");

}

}

}

public class Employee

{

public static double Tax = 0.03;

public string FName;

public string LName;

public double Wage;

public double LoggedHours;

public double Salary;

public double Taxs;

public double NetSalary;

}

MB

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

Classes Methods

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

انا هنا هديك مثال بسيط عن ال Method فى الكلاس و انت طبقه بقه على المثال السابق الى كان فيه الموظفين

بص يا سيدى دلوقتى انا عندى شغلانه عاوز اكررها مع كل اوبجيكت فهعمل ايه ؟

اعملها ميثود ياباشا فى الكلاس علشان مش كل شويه اكتبها و حتى تبقى طالما حاجه تخص الكلاس ده تبقى موجوده فيه هو و تبقى بتبتشغل مع ال instance بتعها بس و غير كده يبقى مش موجوده و ملهاش تلاتين لازمه ههههههه

طيب ورينى مثال كده

using System;

class Program

{

static void Main(string[] args)

{

Employee employee1 = new Employee();

employee1.Print();

Employee employee2 = new Employee();

employee2.Print();

Employee employee3 = new Employee();

employee3.Print();

}

}

public class Employee

{

public void Print()

{

Console.WriteLine("Hello From Employee Cls");

}

}

طب لو عاوز ابعتلها فاريبول برقم يعبر مثلا عن ان ده الموظف رقم كام اعملها اذاى ؟

عادى هنعمل فانكشن ببعت ليها براميتار عادى و استخدمه جواها

using System;

class Program

{

static void Main(string[] args)

{

Employee employee1 = new Employee();

employee1.Print(1);

Employee employee2 = new Employee();

employee2.Print(2);

Employee employee3 = new Employee();

employee3.Print(3);

}

}

public class Employee

{

public void Print(int Num)

{

Console.WriteLine($"Hello From Employee {Num} ");

}

}

MB

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

Param By Value and Ref in Methods

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

البراميتار الى بتتبعت By value لو حصل ليها اى تغيير فى اى وقت خارج الاسكوب الخاص بالفاريبول الاصلى فده لا يؤثر على قيمه الفاريبول الاصلى

بينما By Ref اى تغيير فى اى مكان يأثربشكل مباشر على قيمه الفاريبول الاصلى سواء داخل او خارج الاسكوب الخاص بالفاريبول الاصلى

ده مثال لل By value لاحظ ان ال age قبل البرينت تساوى الى بعد ال print هنا هما الاتنين ب 18

using System;

class Program

{

static void Main(string[] args)

{

var Age = 18;

Console.WriteLine($"Age Before Print {Age} "); // 18

Employee employee1 = new Employee();

employee1.Print(Age); // 25

Console.WriteLine($"Age After Print {Age} "); // 18

}

}

public class Employee

{

public void Print(int AgeNum)

{

AgeNum = AgeNum + 5;

Console.WriteLine($"Age in Print {AgeNum} ");

}

}

ده مثال لل By Ref

لاحظ ان ال age قبل البرينت اختلفت عن بعد ال print هنا الى قبل ب 18 و الى بعد ب 25 علشان تم تغيير الفاريبول الاصلى بالريفرينس و اصبح ب 25 داخل اسكوب البرينت

using System;

class Program

{

static void Main(string[] args)

{

var Age = 18;

Console.WriteLine($"Age Before Print {Age} "); // 18

Employee employee1 = new Employee();

employee1.Print(ref Age); // 25

Console.WriteLine($"Age After Print {Age} "); // 23

}

}

public class Employee

{

public void Print(ref int AgeNum)

{

AgeNum = AgeNum + 5;

Console.WriteLine($"Age in Print {AgeNum} ");

}

}

طيب ايه شروط ال Ref دى ؟

شروطها ان لازم قبل ما ابعت القيمه كبراميتار انى ابقى عاملها intial Value يعنى لو كتبت الكود كده هيضرب

var Age ; // syntax Error must be initialize value as it is Ref Variable

employee1.Print(ref Age);

لازم كده و المثال كله موضح فوق طبعا

var Age = 18;

employee1.Print(ref Age);

طيب لو حابب مبعتوش و تشتغل اعمل ايه ؟؟

هنا بقه ساعتها استخدم Out لان دى هى هياها يعتبر بس الفرق ان ال out مش بتحتاج intial value قبل ارسالها للفانكشن ولكن تحتاج الى ال intial Value داخل الفانكشن نفسها

يعنى من الاخر ال out ملهاش قيمه قبل هى ليها قيمه داخل و بعد فقط

انما ال ref قبل و داخل و بعد

using System;

class Program

{

static void Main(string[] args)

{

int Age ;

Employee employee1 = new Employee();

employee1.Print(out Age); // 25

Console.WriteLine($"Age After Print {Age} "); // 23

}

}

public class Employee

{

public void Print(out int AgeNum)

{

AgeNum = 18;

AgeNum = AgeNum + 5;

Console.WriteLine($"Age in Print {AgeNum} ");

}

}

طبعا ال Out احسن تطبيق عليها TryParse احنا شارحينها فى ال C# الاساسيات ارجعلها

MB

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

Method Signature

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

حوده

MB

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

Classes

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

بص يا سيدى دى يعتبر اهم حاجه في ال OOP و دى يعتبر كده الاستراكتشار بتاع اى حاجه هتعملها جديده كان معاك اوضه و بتفسمها براحتك او كان هي دى اللوحه الى بتحط فيها تصميمك و تمشى الناس عليه من الاخر كده الكلاسات هي الى بتنظم المشاريع كلها

كل كلاس بيحتوى على مجموعه بروبرتيس الى هي عباره عن مدخلات و مخرجات و كذلك مجموعه من الميثودز و دى الى هي وظايف الكلاس ده بيعمل ايه و كذلك يحتوى على كونستراكتور و ده ياعم الحاجه الى بتربط اى كلاس باى كلاس اخر و بتنقل بيه الداتا من كلاس لكلاس اخر و ده دايما ملازم الاوبجيكت الى بتعمله من الكلاس و يمكن عمل اكتر من كونستراكتور للكلاس الواحد عادى المهم تغيير عدد البراميتر الى دخلاله عشان نعرف نفرق بين كل واحد (اوفرلودينج يعنى ) و وانت بتعمل الاوجيكت بتختار بقه تعمل الاوبجيكت بانهى كونستراكتور ساعتها

و فى الديستراكتور و ده بقه لما يكون الاوبجيكتات الى انت عاملها في الكلاس كلها خلصت و خلاص هيطلع من الكلاس و مبقاش ليه شغل فيه ساعتها بيشوف انت انشات كام اوبجيكت من الكلاس و على أساسه بيعمل ديستراكت لكل الاوبجيكتات دى (لاحظ ان الكونستراكتورو الديستراكتور بيخدوا نفس اسم الكلاس بس الديستراطتور بيبقى قبل اسمه ~)

namespace ConsoleApplication1

{

public class Customer

{

// Class Properties

public string \_FirstName;

public string \_LastName;

// Class Constracor

public Customer(string FirstName, string LastName)

{

this.\_FirstName = FirstName;

this.\_LastName = LastName;

}

// Class Methods

public void FullName()

{

Console.WriteLine("My name is : {0} {1}", this.\_FirstName, this.\_LastName);

}

~Customer()

{

Console.WriteLine("Destracor Works Successfully");

}

}

class Program

{

public static void Main(string[] args)

{

// Class Cusomer Obj (instance)

Customer Obj1 = new Customer("Mahmoud", "Badawy");

Customer Obj2 = new Customer("Mostafa", "Badawy");

// Call Method from Cusomer Class By Obj (instance)

Obj1.FullName();

Obj1.FullName();

Obj1.FullName();

Obj2.FullName();

}

}

}

MB

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

Static = shared and Normal instance

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

الفرق بين ستاتيك و انستانس في الكلاسات   
الانتستانس كلاس هو انك تنادى الكلاس بأوبجيكت او انستانس منه و على أساس الاوبجيكت بتقدر تستدعى بيه الى جوه الكلاس و فى كل مره بتعمل فيها اوبجيكت كانك يا اسطى بتاخد نسخه من الكلاس كلها فلوس عندى مثلا عدد 100 تلفزيون فى الكلاس فهتلاقى ان فى كل اوبجيكت عدد التلفزيونات 100 تلفزيون و لو نقصت فى اوبجيكت ما 50 تلفزيون مثلا فهتلاقى ان الاوبجيكتات التانيه متأثرتش يعنى كانك يا اسطى عندك شركه فيها 3 فروع و كل فرع شغال على سيستم لوحده و كل سيستم عليه 100 تلفزيون فلو فرع رقم 1 خرج 50 تلفزيون هيبقى هو الوحيد الى هينقص عنده 50 تلفزيون و باقى الفروع عندهم ال 100 عادى و بالتالى الموضوع الموضوع مش سنتراليزد او مفيهوش مشاركه

الاستاتيك كلاس بقه انت مش بتنشأ من الكلاس اى اوبجيكت خالص لا بتناديه عن طريق انك تكتب اسمه دوت اى حاجه جواه بقه

ملاحظات هامه

الاستاتيك كلاس بقه هو بيتم استخدام هو بس و بالتالى مهما استخدمت فيه اى حاجه جواه فهو بيتخزن في الميمورى مره واحده بس و اى تغيير بيبقى في المكان ده بس في الميمومورى

**اذا لما يكون حاجه انت متأكد انها مش هتحتاج تغيرها كل شويه فريح دماغك و اعملها ستاتيك احسن**

**طبعا ينفع تعمل الكلاس ستاتيتك و ينفع تعمل الفيلدس او المدخلات او الفاريبولز او البروبرتيس او الفانكشنز او الميسودز الى جواه من نوع ستاتيك**

**ستتيك معاناه شيرابول يعنى من الاخر كده هو موجود في كل الاوبجيكتات بنفس العنوان في الميمورى و بالتالى مش محتاج تعرفه جوه اى اوبجيكت لانك معلمه بالكلاس يا صديقى يعنى مفيش new و بالتالى انا لما بغير قيمته فى اى مكان فى المشروع بشوف التغيير ده فى كل الامكان على طول و ده لانه shared مثال بضاعه المخزن بفرض عندى فى الشركه 100 كمبيوتر لما بخرج من فرع القاهره 20 كمبيوتر ابيعم بيبان فى السيستم كله ان الباقى فى كل الفروع 80**

**انما لو انا بعمل اوبجيكت من كل فرع فهتلاقى ان كل فرع فيه 100 كمبيوتر و لما بخرج من الفرع الاول مش بيبان التأثير الا فى الفرع الى انا فيه بس**

**من الاخر الاستاتيك بيوحد المكان فى الميمورى للقيمه و بالتالى اى تغيير فى القيمه من اى مكان خلاص شكرا بتتغير فى كل الاماكن مع بعض**

أول ملحوظه الكلاس الى فات في الشرح الى فات ده كل شغل انستانس كلاس و ميثودز و بروبرتيس

تعالى بقه اوريك اذاى تعمل الكلاس ستاتيك و اذاى تعمل الكلاس عادى و الفانكشنز او البروبرتيس الى جواه ستاتيك

ده كود فيه كلاس ستاتيك و من الواضح ان اى كلاس ستاتيك لا يقبل بداخله الا بروبرتيس او ميثودز و تكون من نوع ستاتيك هي كمان

اما الانستانس كلاس العادى بيقبل ان جواه يتحط ستاتيك ميثود او ستاتيك بروبرتيس عادى

الاستاتيك كونستراكتور لازم يبقى من غير اى اكسيس موديفاير لان مينفعش يتقرأ بره الكلاس بتاعه لان الاستاتيك أصلا المفروض انه مش بيتنادى باى اوبجيكت أساسا فسواء عملته جوه ستاتيك كلاس او انستانس كلاس يبقى متدلوش اى اكسيس موديفاير بقه

الانستانس كلاس تقدر تعمل منه ألاف الاوبجيكتات و بالتالى كل مره بتعمل فيه اوبجيكت و تنادى بيه اى حاجه بتستخدم مساحه من الميمورى للاوبجيكت ده و كل ابويجيكت ذي ما انت عارف بيشور على مكان في الميمورى و بياخد القيمه عن طريق الادرس و كده فلفه سوده يعنى ههههههههههه

أدى الكود بص عليه بقه و هتفهم الفرق بين الانستانس و الاستاتيك كلاس كويس ان شاء الله

لاحظ ان اول حاجه الكومبايلر بيقراها في الكلاس هو الاستاتيك كونستراكتور عشان يروح يحط قيم الفيلدز على طول هناك الأول و يملاها و بعد كده يروح يقرأ اى حاجه بقه ستاتيك ثم يقرأ الحاجات العاديه بقه مهما كان ترتيب سطور الكود يا اسطى

namespace ConsoleApplication1

{

// instance Class

public class Customer

{

// instance Class Properties

public string \_FirstName;

public string \_LastName;

// static Class Field (Propert)

static Double PI ;

// Class Constracor

public Customer(string FirstName, string LastName)

{

this.\_FirstName = FirstName;

this.\_LastName = LastName;

}

static Customer()

{

Customer.PI = 3.41;

Console.WriteLine("Static Customer Constracor");

}

// Class Methods

public void FullName()

{

Console.WriteLine("My name is : {0} {1}", this.\_FirstName, this.\_LastName);

// Calling static Method

Customer.GetPI();

}

// static Class Method

static void GetPI()

{

// Calling static Field (Propert)

Console.WriteLine("PI = {0}",Customer.PI);

}

~Customer()

{

Console.WriteLine("Destracor Works Successfully");

}

}

public static class PersonalInf

{

// Static Class Can not Accept any prop or Methods if it is not static

public static string Name;

// Static Constracor

static PersonalInf()

{

Console.WriteLine("Static PersonalInf Constracor");

}

public static void MyName() {

Console.WriteLine("My name is : {0} ", PersonalInf.Name);

}

}

class Program

{

public static void Main(string[] args)

{

// Class Cusomer Obj (instance)

Customer Obj1 = new Customer("Mahmoud", "Badawy");

Customer Obj2 = new Customer("Mostafa", "Badawy");

// Call Method from Cusomer Class By Obj (instance)

Obj1.FullName();

Obj1.FullName();

Obj1.FullName();

Obj2.FullName();

// Call Methods and Properties from Static Class PersonalInf

PersonalInf.Name = "Mesho MeBO";

PersonalInf.MyName();

}

}

}

MB

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

Inheritance

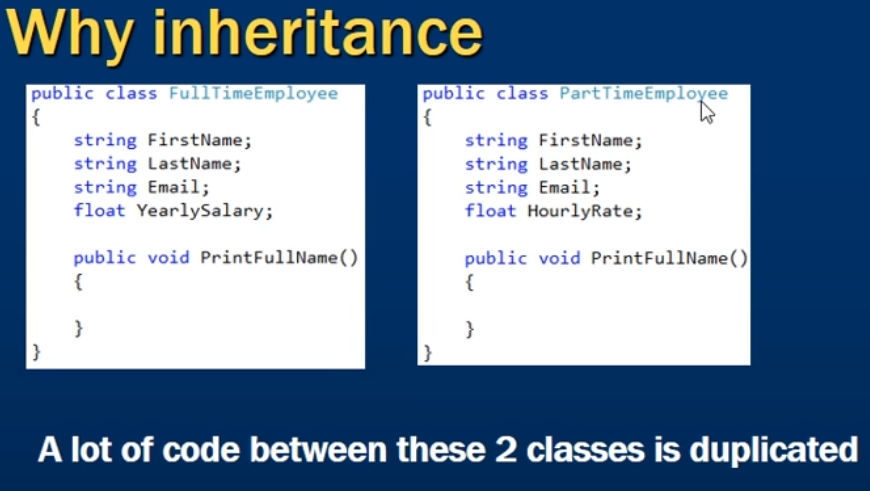
----------------------------

الوراثه و دى كونسيت مهم جدا في الاو او بى و هو ان اذاى كلاس يورث من كلاس كل خواصه من اول البروبرتيس للميثودز و الحاجات الى فيه عمتا كان ابن بيورث ابوه فبياخد منه كل صفاته

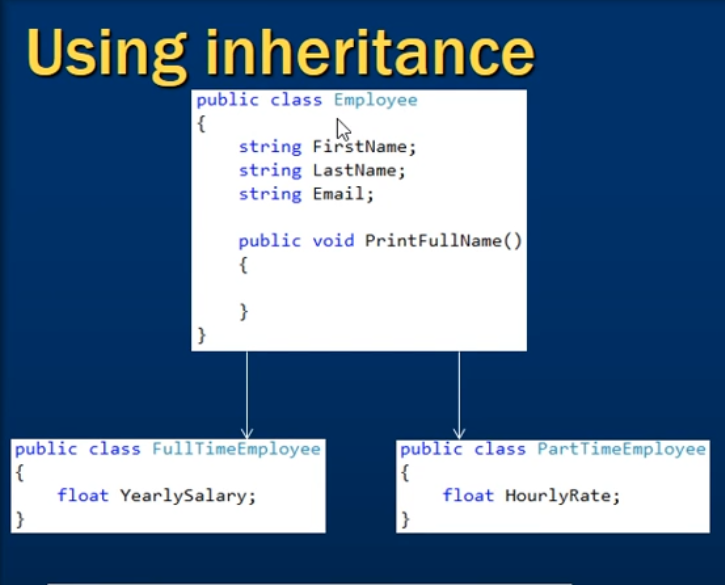
بفرض ان كلاس PersonINFO فيه بروبرتى الاسم و النوع و العمر و الشغل و فيه ميثود بيحسب المرتب و بيحسب السن باليوم مثلا

و جبت كلاس تانى اسمه Person و قولتله اورث يا با من الكلاس PersonINFO ساعتها هيورث منه الحاجات الى فيه و هي بروبرتى الاسم و النوع و العمر و الشغل و فيه ميثود بيحسب المرتب و بيحسب السن باليوم كانهم عنده بالظبط بس هو مش كاتبهم هو ورثهم من صاحبنا التانى الى هو اب يعتبر و الوريث هو الكلاس الابن

دى صور بتوضح اهميه الوراثه



بدل العك الى فوق ده المفروض نعمله كده شوف توفير الكود عامل اذاى



تعالى اوريك مثال بالكود بقه و عشان تعرف الوراقه بتتكتب اذاى بالكود كمان

namespace ConsoleApplication1

{

public class EmployeeParent\_CLS

{

public string FirstName;

public string LastName;

public void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName , this.LastName);

}

}

// HourlyEmp inherits from EmployeeParent\_CLS

public class HourlyEmp : EmployeeParent\_CLS

{

public float HourlyEmpSalary = 30;

public void Salary()

{

Console.WriteLine("EMP Salary is per month : {0} ", this.HourlyEmpSalary \* 8 \* 30);

}

}

// YearlyEmp inherits from EmployeeParent\_CLS

public class YearlyEmp : EmployeeParent\_CLS

{

public float YearlyEmpSalary = 60000;

public void Salary()

{

Console.WriteLine("EMP Salary is per month : {0} ", this.YearlyEmpSalary / 12);

}

}

class Program

{

public static void Main(string[] args)

{

HourlyEmp objHourlyEmp = new HourlyEmp();

objHourlyEmp.FirstName = "Mahmoud";

objHourlyEmp.LastName = "Badawy";

objHourlyEmp.printFullName();

objHourlyEmp.Salary();

YearlyEmp objYearlyEmp = new YearlyEmp();

objYearlyEmp.FirstName = "Mostafa";

objYearlyEmp.LastName = "Badawy";

objYearlyEmp.printFullName();

objYearlyEmp.Salary();

}

}

}

لاحظ ان الكلاسات الى ورست ورست كل حاجه من الاب طالما الحاجه الى في الاب بابليك او بروتيكتيد انما لو برايفيت الكلاسات الى بتورث مش هتورثها أصلا + الابن يقدر يورث الحاجه و يعدل عليها طالما هو بيتعامل مع اوبجيكت بتاعه من نفسه يعنى لو الاوبجيكت ده بتاع الابن الى بيورث فليه حريه انه يورث الحاجه و يعدل عليها عادى يعنى ما هي بتعته خلاص + لو لقيت فانكشن جواها مختلف عن الى بيورث منه اشطه اعمل اوفر رايتينج جوه كلاس الابن بحيث الابن ينادى على حاجته و ملوش دعوه بالى وارثه + الوراثه انك بتكتب الكلاس الابن ثم نقتطين فوق بعض ثم كلاس الاب + في السى شارب الكلاس بينفع يورث من كلاس واحد انما بينفعش يورث من كذا كلاس في سطر واحد و انما ينفع الابن يورث من الاب ثم ابن صغنن حفيد يعنى يورث من الابن الوسط و بكده يبقى الحفيد ورث صفات ابوه و جده

MB

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

Member or Method Hiding

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

دلوقتى لما يكون في كلاس بيورث من كلاس تانى و الكلاس الاب فيه فانكشن و الابن فيه نفس الفانكشن بالحته الى بيحصل بقه ان الاوبجيكت بتاع الابن بيجى ياخد الفانكشن دى فبيستقرب فبيروح واخد بتاعته و مبيورثسش بتاعه ابوه و ده بقه بيتسمى في البرمجه ان الفانكشن بتاعت الابن عملت اوفارايت او هايدينج للفانكشن الى وارثها من الاب و طبعا متنساش كلمه new في فانكشن الابن

namespace ConsoleApplication1

{

Public class EmployeeParent\_CLS

{

public string FirstName;

public string LastName;

public void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName , this.LastName + " - From EmployeeParent\_CLS ");

}

}

public class HourlyEmp : EmployeeParent\_CLS

{

// Method Hiding

public new void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName, this.LastName + " - From HourlyEmp ");

}

}

class Program

{

public static void Main(string[] args)

{

HourlyEmp HourlyEmpObj = new HourlyEmp();

HourlyEmpObj.FirstName = "Mahmoud";

HourlyEmpObj.LastName = "Badawy";

HourlyEmpObj.printFullName();

//######################################

HourlyEmp HourlyEmpObj1 = new HourlyEmp();

HourlyEmpObj1.FirstName = "Mahmoud";

HourlyEmpObj1.LastName = "Badawy";

// Convert Hourly Obj to EmployeeParent\_CLS to Cancel OverWritting

((EmployeeParent\_CLS)HourlyEmpObj1).printFullName();

//######################################

// EmployeeParent\_CLS Point to HourlyEmp

EmployeeParent\_CLS HourlyEmpObj2 = new HourlyEmp();

HourlyEmpObj2.FirstName = "Mahmoud";

HourlyEmpObj2.LastName = "Badawy";

HourlyEmpObj2.printFullName();

}

}

}

طيب لو عندى نفس الاتنين فانكشن و مش عاوز الابن يجيب الفانكشن بتاعته و عاوزه يجيب فانكشن الاب بتاعه اعمل ايه ؟؟

* الى بيحصل ان بعمل اثناء و انا بعمل كول للفانكشن بروح عامل كاستينج للاوبجيكت من نوعه لنوع ابوه عشان يشغل فانكشن ابوه يا حج
* الحل التانى انى اعمل اوبجيكت من الاب عامل ريفرنس او بوينتينج للكلاس الابن و ده هيتيح انى اتعامل مع حاجات الاب و الابن و الاب يبقوا مشتركين في نفس مكان التخزين في الميمورى عادى لانه اوبجيكت الاب بيبقى عامل بوينتج لحاجات الابن و ده مفهوم البوليمرفيزم بتبلمر الحاجه بحيث تزاول الهاكر ميبقاش عارف انهى اوبجيكت بيبص على انهى حاجه هههههههههههههههههههه :D
* ده من الاخر بتعمل اوبجيكت من الاب و بتخزن في أماكن الابن و بتبص على الى انت خزنته بقه من اوبجيت الاب عادى مع انه متخزن في ميمورى الابن ههههههههه حد فاهم حاجه يا جدعان

مثال اخر للتوضيح اكثر عن جزء ال **Polyemarphism**

في الكود ده هتلاحظ ان فكره الهايدينج ملغيه تماما و كل الاوبجيكت بتقرأ من الاب و سايبه الأبناء عشان انا عامل بوليمرفيزم

namespace ConsoleApplication1

{

public class EmployeeParent\_CLS

{

public string FirstName;

public string LastName;

public void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName , this.LastName + " - From EmployeeParent\_CLS ");

}

}

public class HourlyEmp : EmployeeParent\_CLS

{

public new void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName, this.LastName + " - From HourlyEmp ");

}

}

public class yearlyEmp : EmployeeParent\_CLS

{

public new void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName, this.LastName + " - From yearlyEmp ");

}

}

class Program

{

public static void Main(string[] args)

{

EmployeeParent\_CLS [] Obj = new EmployeeParent\_CLS[3];

Obj[0] = new EmployeeParent\_CLS();

// start Polyimervism

Obj[1] = new HourlyEmp();

Obj[2] = new yearlyEmp();

// End Polyimervism

foreach (EmployeeParent\_CLS I in Obj)

{

I.FirstName = "Mahmoud";

I.LastName = "Badawy";

I.printFullName();

}

}

}

}

طيب في الحاله دى لو عاوز اعمل Override هعمل ايه يا اسطى بقه ساعتها ؟؟

هكتب قبل الحاجه الى عاوز اعملها اوفررايد في الاب كلمه virtual

و قبلها في الابن كلمه Override كده

**Polyemarphism** **دى معناها تعدد الاشكال و و هو انى لما الابن يورث فانكشن معينه من الاب و عاوز ياخدهم بنفس اسمهم و يغيير فى اللوجيكت بتعها او البراميتار يبقى كده هو غير شكلها و بالتالى نفذنا عليه كده مفهوم البوليمارفيزم**

**ملخص الموضوع Overriding , Overloading**

MB

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

Member or Method Override

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

namespace ConsoleApplication1

{

public class EmployeeParent\_CLS

{

public string FirstName;

public string LastName;

public virtual void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName , this.LastName + " - From EmployeeParent\_CLS ");

}

}

public class HourlyEmp : EmployeeParent\_CLS

{

public override void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName, this.LastName + " - From HourlyEmp ");

}

}

public class yearlyEmp : EmployeeParent\_CLS

{

public override void printFullName()

{

Console.WriteLine("EMP name is : {0} {1}", this.FirstName, this.LastName + " - From yearlyEmp ");

}

}

class Program

{

public static void Main(string[] args)

{

EmployeeParent\_CLS [] Obj = new EmployeeParent\_CLS[3];

Obj[0] = new EmployeeParent\_CLS();

// start Polyimervism

Obj[1] = new HourlyEmp();

Obj[2] = new yearlyEmp();

// End Polyimervism

foreach (EmployeeParent\_CLS I in Obj)

{

I.FirstName = "Mahmoud";

I.LastName = "Badawy";

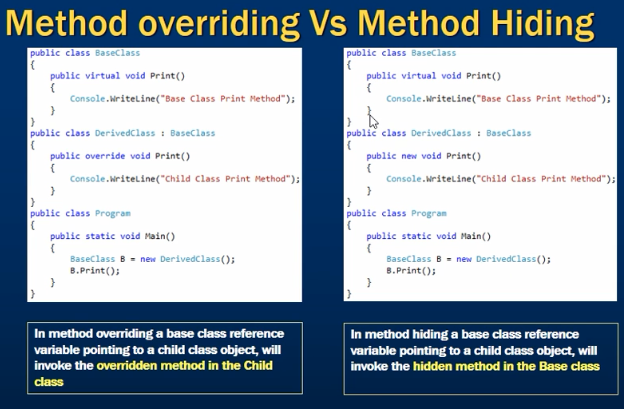
I.printFullName();

}

}

}

}



الميثود هايدينج لو انت عامل بوليمرفيزم او كاسيتنج للاوبجيكت مش هينادى للميثود الى جوه الابن لا هينادى للى جوه الاب و يبقى كده فكره الهايدينج ادمرت ذى الصوره الى ناحيه اليمين   
بينما فى الصوره الى فى ناحيه الشمال بيحصل فيها اوفر رايد مهما كان عامل بقه كاستينج عامل بوليمرفيزم عامل بتنجان هو الميثود الى جوه الابن هى الى بيتم منادتها

خد بالك كويس ان على طول الاوبجيكت بتاع الاب هو الى بيعمل بوينتينج للكلاس بتاع الابناء مش العكس

MB

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

Method Overloading

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

هى انك تعمل نفس الميثود بنفس الاسم ولكن مختلفه اما فى عدد البراميترز او نوع الداتا تايب انتيجر ولا فلوت بتاعت البراميتر او نوع الفاريبول نفسه انبوت ولا اوت بوت ولا فاليو ولا ريفرنس   
مثال

namespace ConsoleApplication1

{

class Program

{

public void MyFunc()

{

Console.WriteLine("Hello From My Function With No Param");

}

public void MyFunc(int Num)

{

Console.WriteLine("Hello From My Function With one Integar Param : " + Num);

}

public void MyFunc(string Num)

{

Console.WriteLine("Hello From My Function With one string Param : " + Num);

}

public void MyFunc(int Num , int Num2)

{

Console.WriteLine("Hello From My Function With Two Integar Input Param : " + Num + " & " + Num2);

}

public void MyFunc(int Num, out int Num2)

{

Num2 = Num;

Console.WriteLine("Hello From My Function With Two Integar input and output Param output Num2 = " + Num2);

}

static void Main(string[] args)

{

Program Obj = new Program();

int x;

Obj.MyFunc();

Obj.MyFunc(2);

Obj.MyFunc("3");

Obj.MyFunc(2,3);

Obj.MyFunc(2,out x);

}

}

}

MB

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

Why Properties ??

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

بص يا صاحبى فى الغالب انت لما تيجى تعمل كود من غير بروبرتيس بيبقى الكود سمك لبن تمر هندى الى هو ممكن تدخل درجات طالب بالسالب مع ان المفروض ان مفيش درجات بالسالب اصلا و ممكن متدخلش فيها حاجه اصلا و تسيبها ب Null و برضوا الكلام ده غلط ما هو مستحيل واحد هيبقى درجته مفيش هههههههه و فى اوقات بتبقى عاوز تعمل الحاجه مينفعش حد يغيرها يعنى و ليكن مثلا المفروض ان درجه النجاح 50 و انت عاوز ان مينفعش حد يعدلها فالمفروض انك بتعملها Read Only و طبعا ده مش متوفر فى الفاريبولز العاديه   
و عشان كده اتعمل ال Properties   
لأن هى ذى ما تقول كده بتعمل قوانين و فالديشنز للفاريبولز   
يعنى عن طريق البروبرتيس تقدر تخلى الفاريبول لا تقبل الا الموجب بس و مينفعش تبقى null و كمان تخليها Read only او حتى تخليها Write Only من الاخر كده بتمسك الفاريبول تظبه على كيفك

ده كود بيبين مساوئ عدم استحدام البروبرتيس

namespace ConsoleApplication1

{

class Program

{

public int ID;

public int PassMark;

public string StudentName;

static void Main(string[] args)

{

Program Obj = new Program();

Obj.ID = -20;

Obj.PassMark = -50;

Obj.StudentName = null;

Console.WriteLine("ID : {0} - Name : {1} - PassMark: {2}", Obj.ID, Obj.StudentName, Obj.PassMark);

}

}

}

تعالى نظبط الاى دى مثلا بفانكشنز عاديين يشبهوا السيتر و الجيتر

namespace ConsoleApplication1

{

class Program

{

public int ID;

public string StudentName;

public int PassMark = 50;

public void SetID(int id)

{

if (id < 0)

{

throw new Exception("ID should be greater zero");

}

this.ID = id;

}

public int GetID()

{

return this.ID;

}

public void SetSN(string SN)

{

if (string.IsNullOrWhiteSpace(SN))

{

throw new Exception("Student name should not be null");

}

this.StudentName = SN;

}

public string GetSN()

{

return this.StudentName;

}

// Make PassMark Read Only

public int GetPassMark()

{

return this.PassMark;

}

static void Main(string[] args)

{

Program Obj = new Program();

// -101 for ID or null For SN will Make Exception So I Comment them

//Obj.SetID(-101);

//Obj.SetSN(null);

Obj.SetID(101);

Obj.SetSN("Mahmoud Badawy");

Console.WriteLine("ID : {0} - Name : {1} - PassMark : {2} " , Obj.GetID(), Obj.GetSN(),Obj.GetPassMark());

}

}

}

فى الحاله ان الاى دى بالسالب او اسم الطالب فاضى الكود هيضرب لان انا عامل اكسيبشن فى الحالات الى انا عاملها الى هى الاى دى بالسالب او اسم الطالب بال Null و طبعا الباس مارك كده خلاص بقى ريد اونلى

طبعا الى فات ده بلح الى فات ده مجرد فانكشنز عاديه بستخدمها لو اللغه مفيهاش البروبرتيس فبطر استخدم الفانكشن سيت و جيت

انما لغه ذى السى شارب فيها اى ابن كلب يبقى لازم نشرح البروبرتيس بقه و نشوف بقه حته ال Encapsualtion

و طريقه عمل البروبرتى بسرعه حاجه من اتنين يا اما بال insert snip ctrl + k , Ctrl + C  
او اعمل الفاريبول برايفيت و اروح اقف عليه و ادوس Ctrl + . هيعمل سيت و جيت جوه بروبرتى بابليك هروح حاطط اللوجيك جوه السيت و اخده بقه بالجيت ذى ما هتشوف فى الكود ده بالملى

namespace ConsoleApplication1

{

class Program

{

private int \_ID;

private string \_StudentName;

private int \_PassMark = 50;

public int ID

{

// Read And Write

set

{

if (value < 0)

{

throw new Exception("ID should be greater zero");

}

this.\_ID = value;

}

get

{

return this.\_ID;

}

}

public string StudentName

{

// Write Only

set

{

if (string.IsNullOrWhiteSpace(value))

{

throw new Exception("Student name should not be null");

}

\_StudentName = value;

}

}

public int PassMark

{

// Read Only

get

{

return \_PassMark;

}

}

static void Main(string[] args)

{

Program Obj = new Program();

Obj.ID = 101;

Obj.StudentName = "Mahmoud Badawy"; // Write only so i can not read it in console

// I can not write in PassMark So it will Shown only in console

Console.WriteLine("ID : {0} - PassMark : {1} " , Obj.ID, Obj.PassMark);

}

}

}

ملحوظه مهمه جدا فى شغلك لازم اى فاريبول هتعمله تعمله بروبرتى عشان ده هيخلى شغلك معموله انكبسوليشن و مفيهوش ثغرات و هيبقى شغلك نضيف حتى لو البروبتى مفيهاش اى لوجيك استخدمها رخامه

**Encapsualtion** : دى من الاخر يعنى data hidden بمعنى ان الواحد بيخش يعمل كل الفاريولز و الميثودز جوه كلاس و بالتالى محدش يقدر يوصل للحاجات دى الا عن طريق الاوبجيكت بتاعه يعنى كانى حطيت كل الميمبرز فى كبسوله باسم الكلام اسمها اوبجيكت و لو فى ميمبر بقه private ممكن اوصله بحاجه اسمها property و ال setter and Getter

MB

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

Structs

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

الاستراكت فى طريقه الكتابه تعتبر متطابقه للكلاس و لكن هناك اختلافات بين الاستراكت و الكلاسيس هنقول عليها بس الاول تعالى اوريك اد ايه الاستراكت شبه الكلاس فى الكتابه يعنى مثلا هتلاقى فيها فيلدس و بروبرتيس و ميثودز و و كونستراكتور و كان هتلاقى ان بيتعمل منها اوبجيكت عادى و فل الفل

الكود

namespace ConsoleApplication1

{

struct Person

{

// fields and Properties

private string name;

private int age;

public string Name

{

get{return name;}

set{name = value;}

}

public int Age

{

get{return age;}

set{age = value;}

}

//Constructor

public Person(String \_Name , int \_Age)

{

this.name= \_Name;

this.age = \_Age;

}

//Methods

public void MyFunc()

{

Console.WriteLine("My Name is : {0} , My Age is : {1} ", this.name, this.age);

}

}

class Program

{

static void Main(string[] args)

{

Person Obj1 = new Person("Mahmoud Badawy", 28);

Obj1.MyFunc();

Person Obj2 = new Person();

Obj2.Name = "Mosatafa Badawy";

Obj2.Age = 2;

Obj2.MyFunc();

}

}

}

MB

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

Struct Vs Class

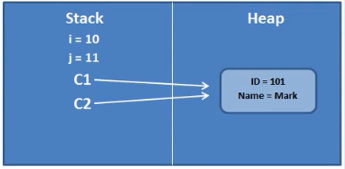
--------------------------------

هناك اختلافات كثيره بين الاستراكت و الكلاس و اهمهم ان ان مفيش ديستراكتور فى الاستراكت و كمان لو حبيت اعمل فيه كونستراكتور لازم الكونستراكتور يبقى بيخشله بارميتر و الا مش هيرضى يتعمل على عكس الكلاس بيتعمل عادى

الاستراكتشار لا يرث الا من الانترفيس بينما الكلاس يرث من الكلاسيز و الانترفيسيز عادى

الاستراكتشار value Type بينما الكلاس Reference type  
و بالتالى الاستراكت بيتخدن فى الاستاك بينما الكلاس بتخزن الاوبجيكت فى الاستاك و بتشاور على القيم بتعتهم فى ال heap و دول اماكن فى الميمورى

و بالتالى الاستراكت بيتخزن فى الميمورى بشكل مباشر فى نفس المكان الى هو اتعرف فيه بينما الكلاس بيخزن الداتا بتعته فى الميمورى جوه الاوبجيكت و بالتالى بيبقى متخزن فى مكان يقدر هو يشاور عليه يجيبه بمعنى اصح و انت بتعرف الاوبجيكت كده التعريفه دى فى الاستاك و لما تيجى تدى الاوبجيكت قيمه بيخدها فى مكان تانى فى الهيب بس بيبقى فى حلقه وصل يعنى الاوبجيكت فى الاستاك بيشاوى على مكان فى الهيب



و بالتالى الفرق بين الفاليو و الريفرنس فاريبول هو ان الفاليو مجرد ما تطلع من الاسكوب بتاعه بيتمسح من الميمورى   
بينما الريفرنس بقه الريفرنس فاريبول نفسه بيتمسح من الاستاك بمجرد الخروج من الاسكوب بينما الحاجات الى جوه الهيب الى هى الا بجيكت و الحاجات الى جواه بتتمسح لما الجاربيدج سيشن يخلص

الفرق بقه بين الفاليو و الريفرنس فاريبول فى مثال بالكود

namespace ConsoleApplication1

{

class Numbers\_CLS

{

public int Num\_1 , Num\_2;

}

class Program

{

static void Main(string[] args)

{

int num\_1, num\_2;

num\_1 = 10;

num\_2 = num\_1;

num\_2 = num\_2 + 1;

Console.WriteLine("Value type : num1 = {0} , num2 = {1} ",num\_1 , num\_2);

Numbers\_CLS Obj1 = new Numbers\_CLS();

Obj1.Num\_1 = 10;

Obj1.Num\_2 = 11;

//Refrence Values Obj 2 Refer to Obj 1

Numbers\_CLS Obj2 = Obj1;

Obj2.Num\_1 = 100;

Obj2.Num\_2 = 110;

Console.WriteLine("Refrence type : Obj1.Num\_1 = {0} , Obj1.Num\_1 = {1} ", Obj1.Num\_1 , Obj1.Num\_2);

Console.WriteLine("Refrence type : Obj2.Num\_1 = {0} , Obj2.Num\_1 = {1} ", Obj2.Num\_1, Obj2.Num\_2);

}

}

}

لاحظ ان الفاليو تايب اى تغيير فى اى حته مش بيأثر على التانيه مع انى واخد من نامبر وان حاطط فى نامبر تو

اما فى الريفرنس فاريبول انا شاورت على الاوبجيت 1 و انا بعرف اوبجيكت 2 فبقى اى حاجه بتحصل فى اوبجيكت 1 او اتنين يروحوا الاتنين مأثرين على بعض و ده الى بيسموه pointing و الريفرنس فاريبولز

MB

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

Sealed Class

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

ده بيتعمل لما تكون مش عاوز حد يورث من الكلاس ده خالص

اذا نقدر نعتبر الاستراكتشار بقه انه سيلد كلاس ههههههههههه لانه بورسش حد حاجه

MB

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

interfaces

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

الانترفيسيز دى من الاخر معموله عشان السينيور يحط فيها الديفينيشنز و الرولز الى الجينيور يمشى عليها و اجبره انه يحطها ذى ما انا حاططها من غير ميجود و لذلك هى يمكن وضع فيها ميثودز و ايقينتس و ديليحيتس بس مجرد ديفينيشنز فقط ولا يمكن وضع اى امبليمينتيشن بداخله و مبيتحطش فيها اى اكسيس موديفايرز هههههه :D

الامبلمينتيشن بقه بيبقى فى الكلاس او الاستراكت الى وارث من الانترفيس ده على فكره الكلاس مينفعش يورث من اتنين كلاس بس ينفع يورث من اتنين انترفيس :D و يعملهم امبليمينيشن و كذلك الاستراكت مينفعش يورث من اى حاجه اصلا الا من الانترفيس و ينفع يورث كذا انترفيس فى نفس الوقت

ملحوظه مهمه برضوا الانترفيس ينفع يورث من انترفيس و الاتنترفيس التالث يورث من التانى و هكذا بس الى هيعمل امبليمينتيشن للحفيد بقه لازم يعمل امبليمينتيشن لكل حاجه فيهم كلهم بقه يعنى بفرض انترفيس 3 ورث من 2 و اتنين ورث من 1 و انا جيت عملت كلاس بيورث من الانترفيس 3 ساعتها لازم اكتب جوه الكلاس كل فانكشنز انترفيس 1 و 2 و 3

namespace ConsoleApplication1

{

interface I\_one

{

void Print();

}

interface I\_Two

{

void Display();

}

class Employee\_Cls : I\_one , I\_Two

{

public void Print()

{

Console.WriteLine("Hello From InterFace 1 in Cls");

}

public void Display()

{

Console.WriteLine("Hello From InterFace 2 in Cls");

}

}

struct Employee\_Struct : I\_one, I\_Two

{

public void Print()

{

Console.WriteLine("Hello From InterFace 1 in Struct");

}

public void Display()

{

Console.WriteLine("Hello From InterFace 2 in Struct");

}

}

class Program

{

static void Main(string[] args)

{

Employee\_Cls Cls\_Obj = new Employee\_Cls();

Cls\_Obj.Print();

Cls\_Obj.Display();

Employee\_Struct Struct\_Obj = new Employee\_Struct();

Struct\_Obj.Print();

Struct\_Obj.Display();

}

}

}

MB

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

Explicit interfaces

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

بص يا سيدى فى مشكله دلوقتى ان انا ممكن اعمل اتنين انترفيس و الاتنين انترفيس ممكن يبقى فيهم نفس المثود بنفس الاسم بنفس البراميترز من الاخر نسخه من بعض طب دلوقتى انا كمحمود افرق قى الامبليمينتيشن بينهم اذاى ؟؟؟  
هنا بقه يجى دور الاكسبليسيت انترفيس و ده كود عشان تفهم بيتعمل اذاى

namespace ConsoleApplication1

{

interface I\_one

{

void Print();

}

interface I\_Two

{

void Print();

}

class Program : I\_one , I\_Two

{

void I\_one.Print()

{ Console.WriteLine("Print() From InterFace 1 "); }

void I\_Two.Print()

{ Console.WriteLine("Print() From InterFace 2 "); }

static void Main(string[] args)

{

Program obj = new Program();

((I\_one)obj).Print(); // Obj Casting to Interface 1 Type

((I\_Two)obj).Print(); // Obj Casting to Interface 2 Type

// Another Solution By Pointing

I\_one I\_one\_Obj = new Program();

I\_one\_Obj.Print();

I\_Two I\_Two\_Obj = new Program();

I\_Two\_Obj.Print();

}

}

}

MB

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

Abstract Class

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

الابستراكت كلاس ده من الاخر Base Class ده معمول عشان يبقى اب للكلاسات العاديه و مش بيتعمل منه اوبجيكت خالص و ينفع ابستراكت يورث من ابستراكت بس مينفعش يورث من كلاس عادى او انترفيس بس المؤكد انه كلاس بيورث الكلاسات العاديه

من مميزات الابستراكت كلاس انه بيجمع بين الانترفيس فى جزئيه انك تعمل فيه الديفينيشن بس و كمان ممكن تحط فيه ميثودز كامله عادى

و ده بيتم بانك لو عاوز تعمل ميثود اجباريه فى الكلاسات الى هتورث منه بتعمل الكلاس دى من نوع Abstract  
و لما تيجى تعمل الامبليمينتيشن بتعمها بتعملها انها Override على الى فى الابستراكت كلاس   
بينما لو عاوزها عاديه اشطه يبقى اكتبها بشكل عادى و قشطه و فل الفل و عنب تعالى اوريك بقه فى الكود الكلام ده

و هنا هتلاحظ ان الفيلدس الى انت عاملها ديكليريشن فى الابستراكت كلاس مش بتحتاج تكتبهم تانى فى الكلاس الابن الى بيورث و كذلك الفانكشنز العاديه بيتم وراثتها اوتوماتيك

اما بقه الفانكشنز الى معموله انها ابستراكت فى دى مجبر انك تعملها امبليمنتيشن فى الكلاس الوريث (الابن)

namespace ConsoleApplication1

{

public abstract class Abstract\_Cls

{

public string Name;

public abstract void Print();

public void Display()

{

Console.WriteLine("Hello In Normal Func in Abstract Class");

}

}

class Program : Abstract\_Cls

{

public override void Print() {

Console.WriteLine("Hello In Override Func inherted from Abstract Class");

}

static void Main(string[] args)

{

Program Obj = new Program();

Obj.Display();

Obj.Print();

Obj.Name = "Mahmoud";

Console.WriteLine("My Name From Abstract Field : {0}", Obj.Name);

}

}

}

MB

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

Abstract Class VS Interfaces

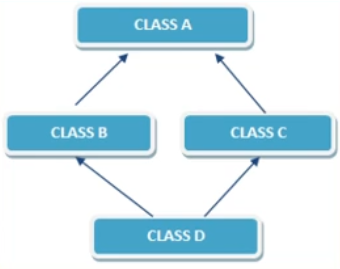
------------------------------------------

|  |  |  |
| --- | --- | --- |
| م | Abstract | Interfaces |
| 1 | ينفع اعمل امبلمينتيشن داخل الكلاس الابستراكت يعنى ينفع احط جواه فانكشن كامله عادى | مينفعش |
| 2 | ينفع احط جواها فيلدس و بروبرتيس | مينفعش |
| 3 | يرث من كلاس عادى او ابستراكت اخر و يرث من انترفيس دون اى مشكله | يرث من انترفيس فقط |
| 4 | بيتكتب جواها اكسيس موديفاير | بيتكتبش فيها اكسيس موديفاير |
| 5 | بنعتبره ذي اى كلاس بس الى بيميزه انه  1- مش بيتعمل منه اوبجيكت لانه معمول عشان يتورث منه بس  2- بيقبل انى اعمل فيه امبلمنتيش لميثودز عاديه او انى اعمل جواه ميثودز ابستراكت و احط فيها فقط الدفنيشن و ابقى اعملها امبلمينتشن في الوريث | دى بنعتبرها تمبلت بيتورث منه فقط عشان يربط الجينيورز انهم ميهبدوش بمزاجهم  و طبعا مينفعش اعمل منه اوبجيكت برضوا |
| 6 | يمكن لاى كلاس وراقه كلاس واحد ابستراكت في اللاين الواحد لان الابستراكت في الاخر كلاس و السى شارب مينفعش اورث كلاسين في لاين واحد | يمكن وراثه 10000000 انترفيس في لاين واحد و اكتر عادى يعنى |

MB

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

Problems of multiple class inheritance in Arabic

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

دلوقتى انا عندى كلاس A فيه فانكشن و جيت في كلاس C عملت وراثه من A و عملت اوفرايد للفانكشن الى في ال A داخل ال C و نفس الكلام عملته مع كلاس B   
يعنى دلوقتى B , C وارثين من كلاس A و عاملين اوفرايد للفانكشن بتاعته الاتنين   
جينا بقه عملنا كلاس جديد خالص اسمه D و قولنا نعمل وراثه من C بس مش هيقول لأ   
نعمل وراثه من B بس برضوا مش هيقول لأ   
اما بقه لو قولناله اورث من الاتنين ساعتها هيقول لأ   
ليه بقه لانه هيبقى مش فاهم ياخد انهى فانكشن الى في C ولا الى في ال B

طب حل المشكل هدى ايه بقه

1. استخدم انترفيس اكيد و لما تيجى تعمل اوبجيكت روح قوله من انهى انترفيس بالظبط بالبوينتينج

و هوريك مثال الصفحه الى جايه

MB

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

Multiple class inheritance using interfaces

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

أولا كود المشكله بتاعت الوراثه من اكتر من كلاس

namespace LearningCSharp

{

public class A

{

protected void Method()

{

Console.WriteLine("Hello from CLS A");

}

}

public class B : A { }

public class C : A { }

public class D : B,C { } // this is the Error Line Test it ..

class Program

{

static void Main(string[] args){ }

}

}

حل المشكله

namespace LearningCSharp

{

interface IA

{

void Method();

void AMethod();

}

interface IB

{

void Method();

}

public class AB : IA, IB {

public void AMethod()

{

Console.WriteLine("Hello AMethod from CLS AB");

}

public void Method()

{

Console.WriteLine("Hello Method from CLS AB");

}

}

class Program

{

static void Main(string[] args)

{

AB Obj = new AB();

Obj.AMethod();

Obj.Method();

}

}

}

هو الكود ده فعلا حل المشكله بس انا برضوا لسه معرفتش هو جاب انهى Method(); و كده حصل Ambigus بس مش مأثر على الكود و هيأثر على المدى البعيد و ككود نضيف مينفعش

طب اذاى احل المشكله دى بحيث ابقى كاتب كود نضيف جدا   
يبقى امشى ذي الكود ده بالظبط

namespace LearningCSharp

{

interface IA {void AMethod();}

public class A : IA { public void AMethod(){

Console.WriteLine("Hello AMethod from CLS A");} }

interface IB {void BMethod();}

public class B: IB { public void BMethod(){

Console.WriteLine("Hello BMethod from CLS B");} }

public class AB : IA, IB {

A objA = new A();

B objB = new B();

public void AMethod() { objA.AMethod();}

public void BMethod() { objB.BMethod();}

}

class Program

{

static void Main(string[] args)

{

AB Obj = new AB();

Obj.AMethod();

Obj.BMethod();

}

}

}

MB

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

access modifiers

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

|  |  |  |  |
| --- | --- | --- | --- |
| Access Modifier | وصفه | types | Type Members |
| Public | ده يا سيدى انت بتعمله و المشروع كله بيقدر يشوفه يعنى الكلاس بتاعه و الى بيورث منه و الى بيورسش كمان و غريب عنه و اى كلاس معدى في الشارع بيقدر يشوفه و كمان لو بتكلم مشروع تانى المشروع التانى بيشوفه برضوا | yes | yes |
| private | ده يا كبير بيتعمل في الكلاس بتاعه و محدش بيشوفه غير الكلاس بتاعه بس | NO | Yes and Default |
| protected | الكلاس بتاعه بيشوفه و الكلاس الى وارث منه | NO | yes |
| internal | ده ذي البابليك بس الفرق ان ده لو عندك مشروعين المشروع التانى مش هيشوفه ابدا فقط مشروعه و الكلاس الى فيه هي الى بتشوفه اما اى مشروع تانى مبيشوفوش | Yes and Default | yes |
| Protected Internal | ده بيجمع بين صفات الانترنال و البروتيكتيد و بالتالى هو بيتشاف في اى مكان في مشروعه اما لو مشروع تانى ف في الحاله دى المشروع التانى مش بيشوفه الا في حاله واحده بس ان يبقى كلاس من المشروع التانى بيورث من الكلاس بتاعه الى في المشروع الأول بس كده | NO | yes |