Introduction

لازم تكون مخلص كل دول

What is OOP and Why??

هنا كنا بنقول اننا عندنا functional programming و object oriented programming
ال functional programming اشتغلنا عليها وعملنا بيها مشاريع قبل كده وكانت بتتلخص في
انك بتلخص المشروع في functions صغيره وتقعد تركبهم شبه البازل عشان تعمل مشروع
بس لو عندك مشروع كبير ممكن تضطر تعمل الاف من ال functions عشان تعمل المشروع ده
وبعد كده ممكن تنسي وتكتب نفس ال function مرتين لانك ناسي انك كتبتهم قبل كده ده غير التعقيد
بتاع البرنامج
المشكلة هنا مش ان يكون عندك الف function
المشكله انهم مش منظمين وان انا مااقدرش بسهولة اتعامل معاهم
فواحدة من فوائد ال oop هيا انها بتنظملك الكود بتاعك وبتخليك تفكر فيه بطريقه اقرب للواقع

لو اخدنا مثال اننا عايزين نعمل برنامج للجامعه
لو هنعمل المشروع باستخدام ال FP هنعمل الاف ال functions واحده تضيف طالب وواحده
تضيف دكتور وواحده تحذف طالب وهكذا
انما ال OOP هنفكر في الأشياء اللي عايز ابرمجها فهلاقي نفسي بقول ان الأشياء او ال objects
دي هيا دكتور وطالب وكورس وهكذا فاصبحت بفكر في البرمجه زي مابفكر في الحياة الواقعيه
وبعدين هاخد كل ال functions الخاصه بالدكتور واحطها في ال object اللي اسمه دكتور
يعني ال oop بتخليك تتعامل مع الكود من فوق لتحت مش زي ماكنا بنتعامل من تحت لفوق
كمان بتديك ميزات عديده لسه هنجيلها بعدين تخليك تقدر تتحكم بالكود بشكل افضل
زي مثلا قبل كده ال functions كنا حاطينها في الشارع أي حد ييجي ياخدها
انما هنا انت تقدر تمنع حد معين من انه يوصلها

Functional (FP) vs Object Oriented (OOP)

University System Example (FP)

- AddStudent
- UpdateStudent
- DeleteStudent
- CalculateAverage
- AddCourse
- UpdateCourse
- DeleteCouse
- EnrollStudentInCourse
- UnEnrollStudentFromCoruse
- HowManyStudentsInCourse
- Doctor (Add, Edit, Delete)
- AssignCourseToDoctor
- SendEmailToStudent
- SendTextMessageToStudent
- SendEmailToDoctor
- SendTextMessageToDoctor
- CallStudent
- CallDoctor
- AddEmployee
- UpdateEmployee
- DeleteEmployee
- CalculateSalary
- PaySalary
- .....

Simply

You can have thousands of functions in your System!!!!

The problem is not with number of functions at all ☺

The problem is with organizing them and the way you look at them ☺

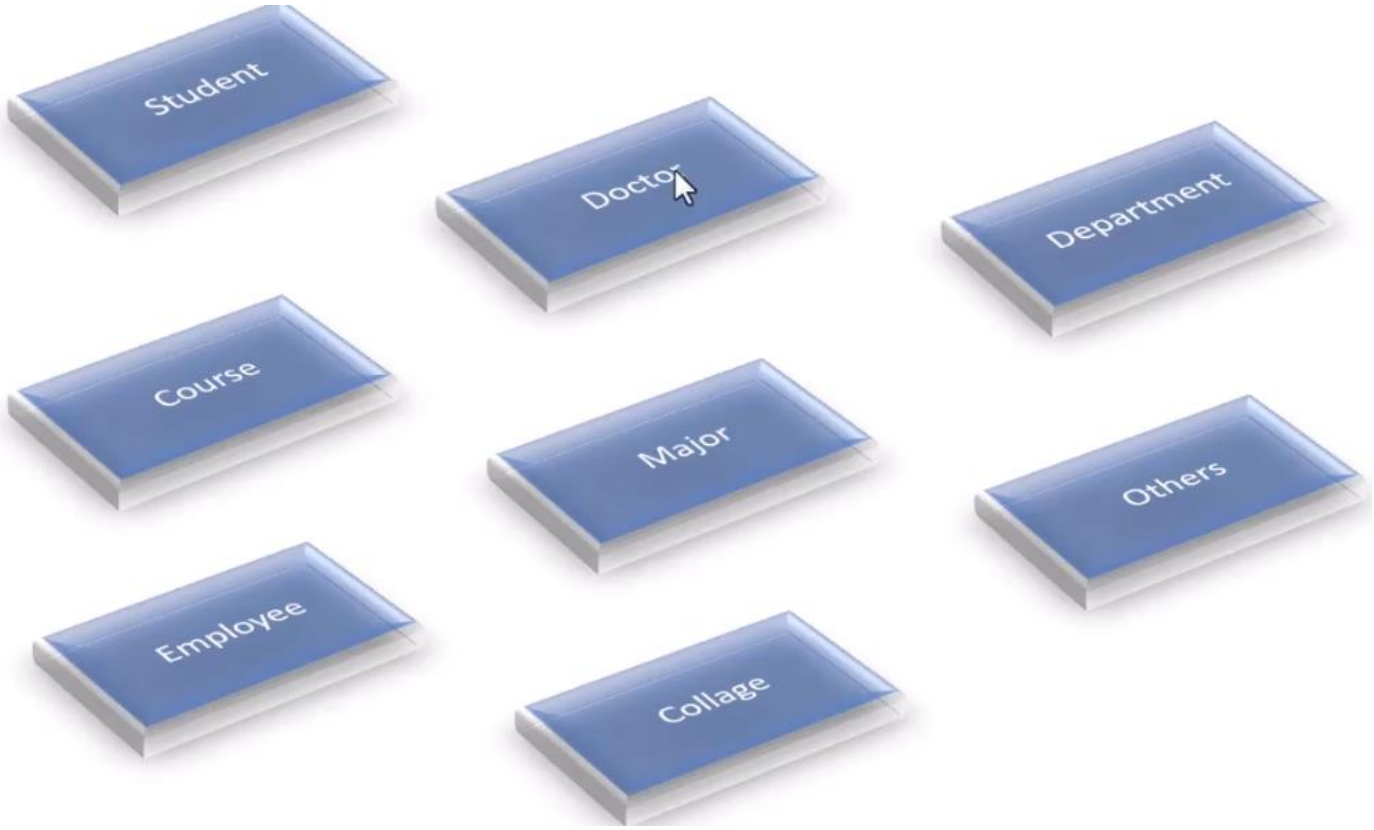
To make it simple for you ☺

What if you have thousands of contacts and phone numbers and all these contacts and numbers are not organized!

You will have a nightmare ☺

Functional (FP) vs Object Oriented (OOP)

University System Example (OOP)



Simply

Group Members and Functions
Inside Related Objects

لما كنا عايزين نعرف طالب قبل كده ليه firstname و last name وage كنا بنحطه في structure بس ماكناش بنقدر نحط functions جوه ال structure ده
هنا بقي ال oop تقدر تعرف functions و variables جوه ال object وبيبقي اسمه كلاس class

Functional (FP) vs Object Oriented (OOP)

Student Object

StudentObject
Student1.Name
Student1.Email
Student1.GetEnrolledCourses()
Student1.CalcuateAverage()
Student1.SendEmail(Subject, Body)
Student1.EnrollInCourse(10)
Student1.PayFees
Student1.UnEnrolFromCourse(10)

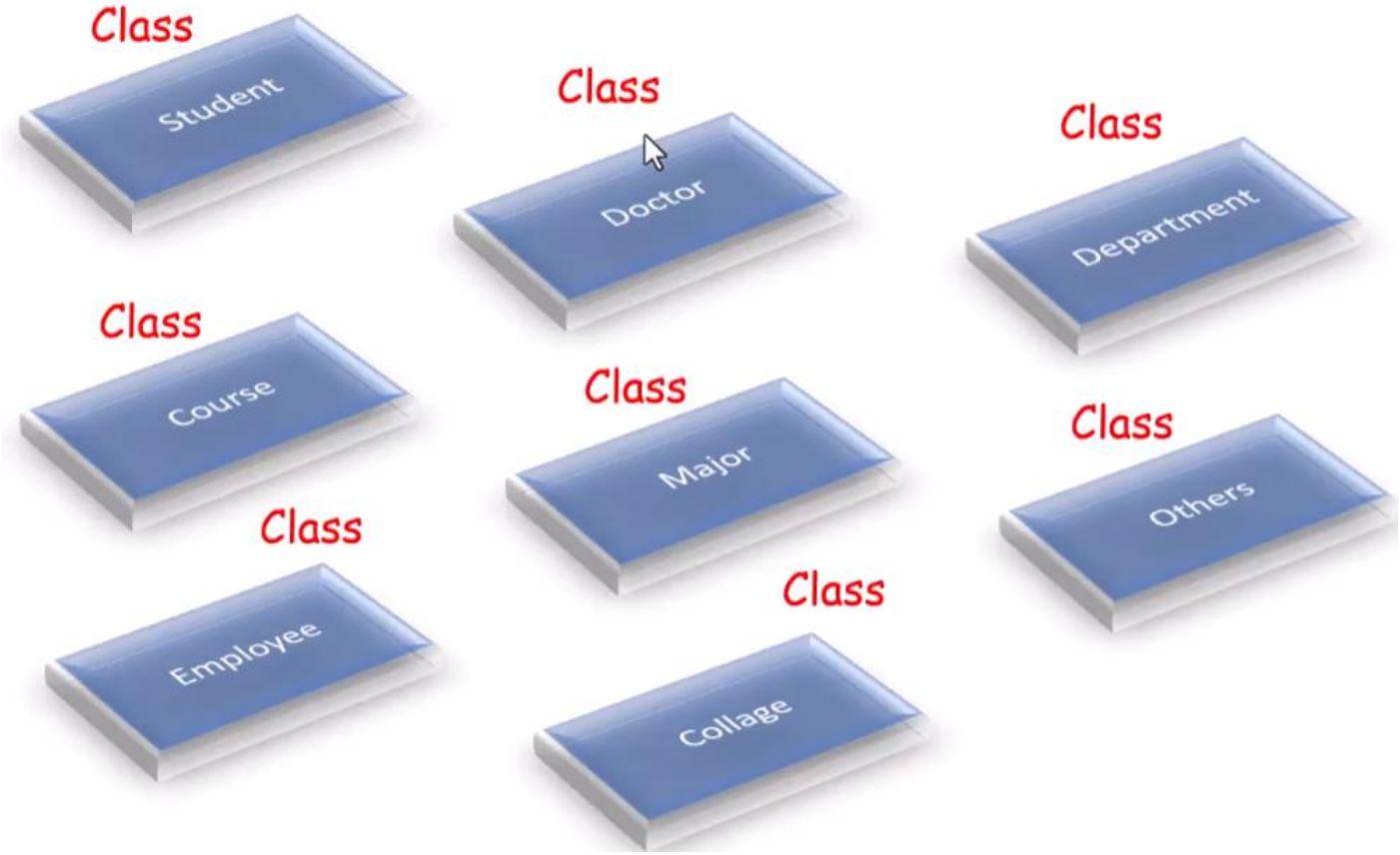
You can do whatever you want with a certain student through its object

Classes and Objects

اتكلمنا المره اللي فاتت ان من فوايد ال oop انك بتقدر تفكر في البرنمج بتاعك علي انه أشياء فمثلا لو برنامج جامعه هتفكر في انك تعمل دكتور وطالب وكورسات وجوه كل object من دول هتحط ال variables وال functions بتاعته
فاانت كده يعتبر عملت تصنيف للكود بتاعك انك جبت كل ال viarables وال functions بتاعت الدكتور وحطيتها في مع بعضها
التصنيف بالانجليزي اسمه classification وعشان كده كل عنصر من دول ب مايكون object لازم يكون class وهيا اختصار ل classification

What is Class? Why we call it Class?

University System Example (OOP)



Simply

Class came from Classification

```
#include<iostream>

using namespace std;

struct stPerson{
	string FirstName;
	string LasttName;
};

int main() {

	stPerson Person1;

	return 0;
}
```

بص كده عالكود ده
عرفنا structure اسمه stPerson
ال structure ده عبارة عن data type
زي ال enum وال string وال int

هنا جينا عملنا variable من النوع
stPerson اسمه person1

اصبح هنا ال stPerson هوا datatype وال Person1 عباره عن variable وال stPerson بتاعه هوا data type	

طيب انا دلوقتي مش عايز اعمل structure وعايز اعمل class عشان استفيد من موضوع اني اقدر اضيف function فيه
فكل اللي هعمله اني بدل مااكتب كلمة struct هكتب كلمة class
بص الكود اللي جاي

هنا نفس الكود السابق بس بدل كلمة struct كتبنا class وجوه ال class ضيفنا function

هنا ال clsPerson اصبح اسمه class وال person1 اصبح اسمه object

يعني ال variable اسمه variable عادي لكن لو كان ال variable ده من نوع class اصبح اسمه object او instance

```cpp
#include<iostream>

using namespace std;

class clsPerson{
	string FirstName;
	string LasttName;

	string FullName() {
		return FirstName + " " + LasttName;
	}

};

int main() {

	clsPerson Person1;

	return 0;
}
```

في الكود اللي فات لو جينا قولنا .Person1 يعني كتبنا اسم الكلاس وحطينا بعده نقطه عشان نستدعي ال first name او ال last name او ال full name مش هنقدر
طب ليه؟
قالك لان أي حاجه جوه الكلاس بيكون ال default بتاعه private يعني بيقتصدر استخدامه من جوه الكلاس ماتقدرش تستخدمه بره الكلاس
طيب عشان نتغلب علي الحوار ده نعمل ايه ؟
ببساطه قالك انك تيجي قبل متعرف أي حاجه تكتب كلمة public وبعدها نقطتين فوق بعض بس كده وبعدها تقدر تستخدم كل حاجه جوه الكلاس عادي

خلينا الالمتغيرات وال function من النوع public

```cpp
#include<iostream>

using namespace std;

class clsPerson{
public:
	string FirstName;
	string LasttName;

	string FullName() {
		return FirstName + " " + LasttName;
	}

};

int main() {
```

وهنا قدرنا نستخدم كل حاجه جوه ال class

```
        clsPerson Person1;

        Person1.FirstName = "Mohammed";
        Person1.LasttName = "Abu_Hadhoud";

        cout << Person1.FullName() << endl;
        return 0;
}
```

هنا بيقولك انك تقدر تخلي بعض المتغيرات او ال functions من النوع private وبعضها public
عن طريق انك تخلي الحاجه اللي عايزها private تكتبها قبل ال public ياما تكتب قبلها كلمه private
في الحاله دي لما تيجي تستدعيه مش هتظهر معاك الا جوه ال class

وبيقولك ان ال string في حد ذاته هوا class

```
class clsPerson{
private:
        int x;

public:
        string FirstName;
        string LasttName;

        string FullName() {
                return FirstName + " " +
LasttName;
        }

};
```

الواجب

What is Class?

- Class is the blue-print of object
- Class is a Datatype
- Class is the same as object
- None of the Above

Can you access class members and methods directly?

Yes

No, You have to declare an object of the class first, and access all members and methods through the object not class.

How to access member function of class using Object ?

FunctionName();

Class.FunctionName();

ObjectName.FunctionName();

None of the above.

If you have a private member or method in class, can you access this member or method through Object?

Yes, it can be accessed

No, only public members and methods can be accessed through the object, all private members and methods are for internal use inside the class

Any Function or Procedure inside class is called "Method".

True

Flase

Object is Instance of class.

True

Flase

What is the difference between C & C++?

They are the same no differences.

C is procedural/Functional programming language.

C does not support OOP, While C++ Supports OOP.

C++ Supports Procedural/Functional Programming and OOP as well

Class members means any variable or function inside the class is called "Member".

True

False

You scored 8 / 8 (100%)

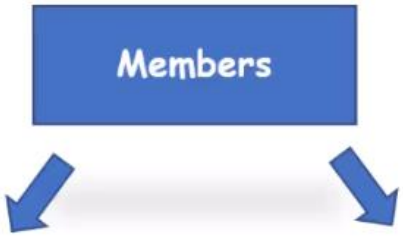
Retake Quiz

Class Members

هنا بيقولك ان أي حاجه موجوده جوه الكلاس اسمها member
وال class member فيه منه نوعين
Data member وده المتغيرات اللي بخزن فيها بيانات
Member method ودي ال functions اللي جوه الكلاس

Class Members

```cpp
#include <iostream>
using namespace std;

class clsPerson
{
public:

string FirstName;
string LastName;

string FullName()
{
   return FirstName + " " + LastName;
}
};

int main()

{
clsPerson Person1;

Person1.FirstName = "Mohammed";
Person1.LastName = "Abu-Hadhoud";

cout <<"Person1: " << Person1.FullName()
<< endl;

}
```



Data Members: Any variable declared inside the class that holds Data.

In Our Case:
FirstName, LastName
Are Data Members

Member Methods(Functions): Any Function Or Procedure declared inside the class.

In Our Case:
FullName()
Is a Member Function/Method

الواجب

Data Member is any variable inside the class that holds data.

True

False

Function Member is any function or procedure inside a class.

True

False

Class Members are Data Members and Function Members

True

False

Objects In Memory

زي ماقولنا ان ال class هوا عباره عن data type وقولنا ان العناصر اللي جواه بتنقسم ل data member و member function

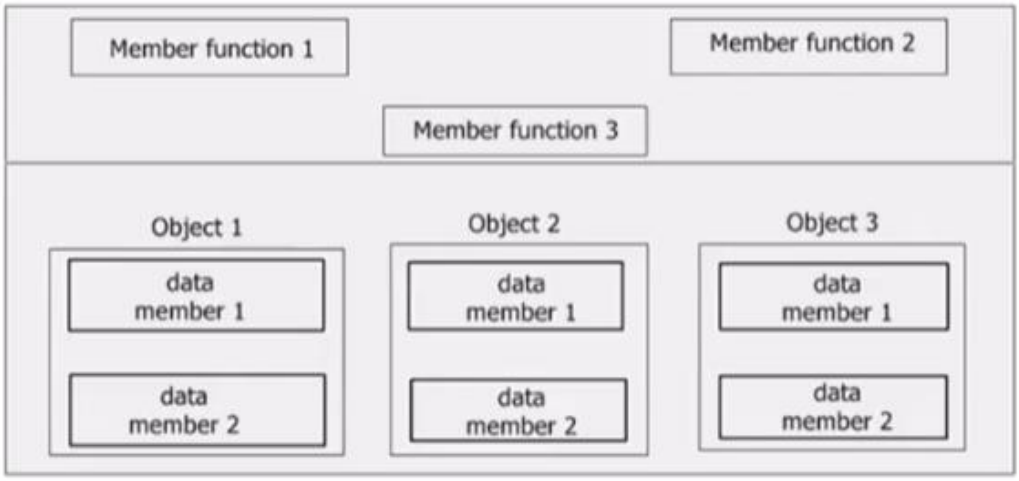
وان كل variable متاخد من كلاس اسمه object

طيب ازاي بتتخزن ال objects في الذاكره؟

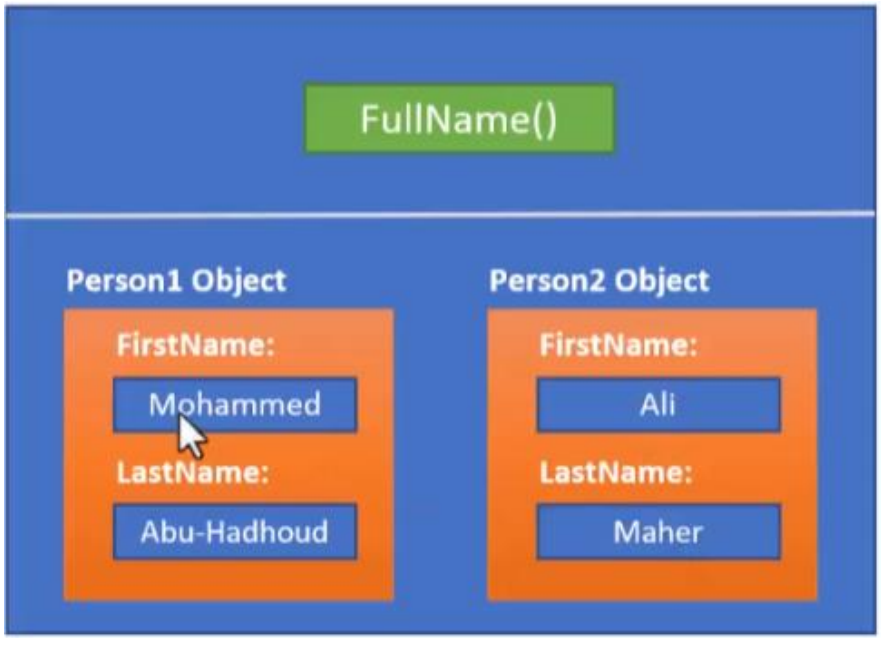
قالك كل ان ال كل object بيتاخد ال data members بتاعته وبتتخزن في مكان معين في الذاكرة

لكن ال member functions بتتخزن في مكان تاني وبتكون مشتركة بين كل ال objects

Each instance has its own space in memory, Only Member functions are shared among all objects




```cpp
#include <iostream>
using namespace std;

class clsPerson
{
public:

string FirstName;
string LastName;

string FullName()
{
   return FirstName + " " + LastName;
}
};

int main()

{
clsPerson Person1,Person2;

Person1.FirstName = "Mohammed";
Person1.LastName = "Abu-Hadhoud";

Person2.FirstName = "Ali";
Person2.LastName = "Maher";

cout <<"Person1: " << Person1.FullName() << endl;
cout << "Person2: " << Person2.FullName() << endl;

}
```



الواجب

Every Object has it's own space in memory that hold both Data / Function Members.

True

False

Every Object has it's own space in memory that holds only Data Members.

True

False

Function Members are shared to all objects in memory and has one space for them.

True

Flase

Access Specifiers/Modifiers

ال access modifiers هيا من الحاجات اللي بتخليك تتحكم في الكود وتقول مين اللي تقدر تشوفه من بره ومين لا الموضوع ده بيديك security علي الكود

طيب مين اللي يقدر يستفيد من ال members ؟

1- الناس اللي بره الكلاس وحابه تاخد object منه
2- الناس اللي شغاله جوه الكلاس نفسه
3- كل ال classes اللي بتورث من الclass ده (هنيجي للموضوع ده بعدين)

طيب ال access modifiers اللي عندي هما 3 أنواع

1- Private :- ماحدش بيقدر يوصله الا من جوه الكلاس بس
2- Protected :- اللي بيقدر يوصله الناس اللي جوه الكلاس واي كلاس بيورث منه بس
3- Public :- ده أي حد يقدر يوصله


```cpp
#include<iostream>
using namespace std;

class clsPerson{
private:
	//only accessable inside this calass
int Variabl1=5;
int Function1(){ return 40;}

protected:
	//only accessable inside this calass and all classes inhirets this calss
int Variabl2=100;
int Function2(){return 50;}

public:
	//accessable for everyone outside/inside/and classes inherits this class
	string FirstName;
	string LastName;

	string FullName(){
		return FirstName + " " + LastName;}

	float Function3(){
		return Function1() * Variabl1 * Variabl2;}
};

int main() {
	clsPerson Person1;
	Person1.FirstName = "Mohammed";
	Person1.LastName = "Abu-Hadhoud";
	cout << "Person1: " << Person1.FullName() << endl;
	cout << Person1.Function3();

	return 0;
}
```

الواجب

Access modifiers (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members.

- True
- False

Which of the following is Access Specifiers/Modifiers:

- Public
- Private
- Protected
- Constant

Public Members can be accessed from inside and outside the class.

- True
- False

Private Members can be accessed from outside the class through object.

- True
- Flase

Private Members can be accessed by any class inherits the current class.

- Ture
- False

Private Members can be accessed only from inside the class, it cannot be accessed from outside the class nor from the classes inherits the current class..

- True
- Flase

If you want to have a member that is private to outside class and public to classes inherits the current class, which access specifier/modifier you use?

- Public
- Private
- Protected

Protected Members can be accessed from outside class through objects.

- True
- False

Protected Members can be accessed from inside class and from all classes inherits the current class.

Tue

False

OOP is more secured because you can hide members from developers.

True

False

Inside the class I can access everything including Public, Private , and Protected Members.

True

False

Properties Set and Get

بيقولك اننا كنا قبل كده بنعين قيم للمتغيرات اللي جوه الكلاس بشكل مباشر
قالك الكلام ده ممنوع في ال oop وانك لازم تعدل القيم او تعينها عن طريق function ولما تيجي تستدعي القيم تستدعيها عن طريق function
فاانت عشان تعدل عليه بتعمل function اسمها property set وعشان ترجعها بتستدعي function اسمها property get
من المميزات بتاعت الحوار ده انك مثلا عينت قيمه جديده للمتغير وعايز تحتفظ بالقيمه القديمه ليه

ده الكلاس اللي عملناه عملنا المتغيرات private وعملنا دالتين لكل متغير منهم من النوع public واحده set والتانيه get	`#include<iostream>` `using namespace std;` `class clsPerson{` `private:` `string _FirstName;` `string _LastName;` `public:` `//Property Set` `void setFirstName(string FirstName){` `_FirstName = FirstName;` `}` `//Property Get`

```
    string FirstName(){
        return  _FirstName;
    }
    //Property Set
    void setLastName(string LastName){
        _LastName = LastName;
    }
    //Property Get
    string LastName(){
        return  _LastName;
    }


    string FullName() {
        return  _FirstName + " " + _LastName;
    }
};


int main() {
    clsPerson Person1;
    Person1.setFirstName("Mohammed");
    Person1.setLastName("Abu-Hadhoud");
    cout << "First Name:" << Person1.FirstName() << endl;
    cout << "Last Name:" << Person1.LastName() << endl;
    cout << "Full Name:" << Person1.FullName() << endl;


    system("pause>0");
    return  0;
}
```

وهنا بنستخدم ال functions
في تعيين القيم وطباعتها

الواجب

Properties are Functions allow you to Update Private Members inside the class

True

False

Properties are two functions one for Setting Data and One for getting Data.

True

False

If you want to update data inside class you should write a property function to set them.

True

False

If you want to retrieve Data Member from a class you should write a property function to get that data.

True

False

Both property functions set and get they use a private data member to store and get data from it.

True

False

Read Only Property

دلوقتي احنا عاوزين نعمل read only property بحيث ان محدش يقدر يعدل علي ال data member

هنا الفكره انك تعمل property get وماتعملش property set

ولو عايز تعملها write only property اعمل set وماتعملش get

```
#include<iostream>
using namespace std;

class clsPerson{

private:
int _ID = 10;
string _FirstName;
string _LastName;

public:
	//Property Get, this is a read only property because we don't have a set function
	int ID(){ return _ID;}

	//Property Set
	void setFirstName(string FirstName){
		_FirstName = FirstName;}
```

```
        //Property Get
        string FirstName(){
                return _FirstName;}

        //Property Set
        void setLastName(string LastName){
                _LastName = LastName;}

        //Property Get
        string LastName(){
                return _LastName;}

        string FullName(){
                return _FirstName + " " + _LastName;}
};

int main() {
        clsPerson Person1;
        Person1.setFirstName("Mohammed");
        Person1.setLastName("Abu-Hadhoud");
        cout << "ID:" << Person1.ID() << endl;
        cout << "First Name:" << Person1.FirstName() << endl;
        cout << "Last Name:" << Person1.LastName() << endl;
        cout << "Full Name:" << Person1.FullName() << endl;

        system("pause>0");
        return 0;
}
```

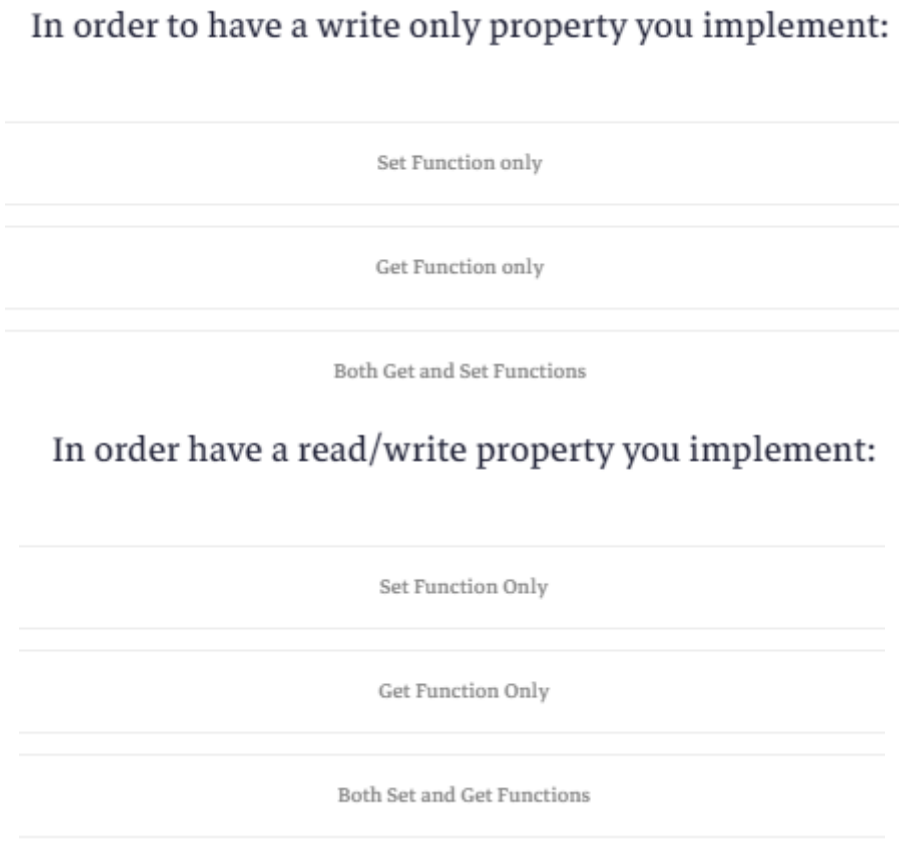
الواجب

In order have a read only property you only implement the set function and you don't implement the get function

- True
- False

In order have a read only property you implement:

- Set Function Only.
- Get Function Only
- Both Get and Set Functions



Properties Set and Get through "="

احنا قبل كده كنا عشان نغير قيمة ال first name كنا بنستدعي ال function اللي اسمها set
وعشان نستدعي القيمه المخزنه فيه كنا بنستدعي ال function اللي اسمها get

طيب هل فيه طريقه تخليني بدل ماستدعي الدالتين دول اعمل كده؟

```
Person1.FirstName = "Mohammed";
   cout << Person1.FirstName;
```

ومن غير ماخلي المتغير اللي اسمه firstName يكون public ؟

قالك اه ينفع
طب ازاي ؟
قالك انه فيه كلاس اسمه __declspec اختصار ل Declaration Specification
الكلاس ده بكتب اسمه وافتح قوسين اكتب جواهم property وافتح قوسين وجوه القوسين بكتب كلمة get وبعدها علامة يساوي وبعدين اسم ال property get اللي انا عاملها
وبعدين فاصله واكتب put وبعدها يساوي وبعدها اكتب اسم ال property set اللي انا عاملها
وبعد الاقواس دي بعرف متغير بنفس نوع المتغير اللي عايز اعدل عليه او استدعيه(المتغير اللي كتبته جديد ده هوا اللي هستخدم اسمه لما اجي اعدل القيمه او استدعيها) زي كده

```
__declspec(property(get = GetFirstName, put = SetFirstName)) string FirstName;
```

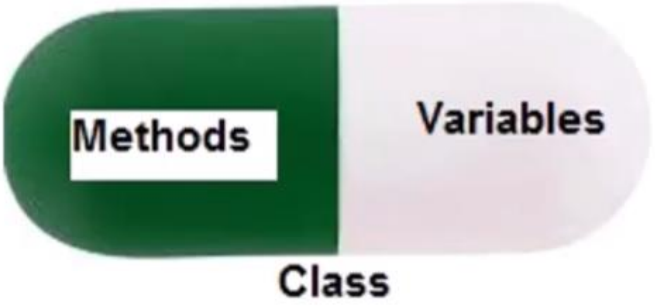
``#include <iostream> using namespace std; class clsPerson {```	ده الكلاس بتاعنا

`` private: string _FirstName; public: void SetFirstName(string FirstName) { _FirstName = FirstName; } string GetFirstName() { return _FirstName; } __declspec(property(get = GetFirstName, put = SetFirstName)) string FirstName; }; int main() { clsPerson Person1; Person1.SetFirstName("Mohammed"); cout << Person1.GetFirstName() << endl; //instead of the above we only write this Person1.FirstName = "Mohammed"; cout << Person1.FirstName; system("pause>0"); return 0; }; ```	ده متغير private دي ال property set ودي ال property get وده السطر اللي قولنا عليه وهنا ده اسم المتغير اللي هستخدمه بعدين وهنا دي الطريقه القديمه وهنا دي الطريقه الجديده

First Principle/Concept of OOP: Encapsulation

كنا فاكرين قبل كده لما قولنا ان من فوائد ال oop انه بدل مال functions مرمية في الشارع كنا بنعمل class نحط فيه المتغيرات وال functions الخاصه بشئ معين
هوا ده بالضبط معني ال encapsulation وهوا ان الكلاس عندك اصبح زي الكبسولة بتاعت الدواء اللي مقسومة نصفين نص موجود فيه ال functions والنص التاني محطوط فيه المتغيرات واصبح لايمكن تتعامل مع member بدون ماتستخدم object

Encapsulation in Object Oriented (OOP)



StudentObject:

Student1.Name

Student1.Email

Student1.GetEnrolledCourses()

Student1.CalcuateAverage()

Student1.SendEmail(Subject, Body)

Student1.EnrollInCourse(10)

Student1.PayFees

Student1.UnEnrolFromCourse(10)

You can do whatever you want with a certain student only through its object

الواجب

In normal terms Encapsulation is defined as wrapping up of data and information under a single unit. In Object Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulates them.

True

False

Second Principle/Concept of OOP: Abstraction

هنا بيقولك تخيل انك عايز تصور حد بالموبايل
كل اللي بتعمله انك بتفتح التطبيق بتاع الكاميرا وبتدوس علي زر التصوير
طيب في كود البرنامج بتاع الكاميرا عندك مثلا function اسمها take picture وال function دي بتستدعي 10 functions تانيين
ال functions دي انت ملكش علاقه بيهم ولا يهموك فتخيل لو اظهروهم ليك في شكل ازرار او حاجه
بالنسبالك الموضوع هيعملك distraction وهتحس ان الدنيا معقدة
هما عشان يسهلو عليك عملوا لل 10 functions دول abstraction
الفكرة من ال abstraction انك تسهل عاللي هيستخدم الكود بعدك سواء كان مطور تاني او كنت انت نفسك اللي هتكمل في التطوير
فالفكرة هنا انك بتختصر عمليات كتير في خطوة واحدة
ال abstraction مش بيقصد بيه ال abstract class ومالهومش علاقه ببعض

الواجب

In simple terms, abstraction “displays” only the relevant attributes of objects and “hides” the unnecessary details.

- True
- False

You Achieve Abstraction?

- Through Private Members only
- Through Public Members only
- Through Public and Private Members

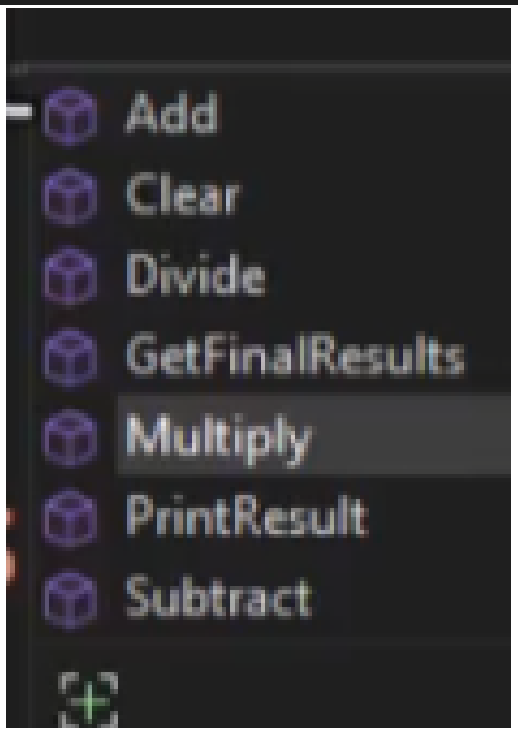
Project 1: Calculator - Requirements

بتعمل كلاس اسمه calculator وجوه ال main بتاخد منه object وتبدأ تستدعي الي العمليات منه

```
class clsCalculator { ... };

int main()

{
    clsCalculator Calculator1;

    Calculator1.
    system("paus
    return 0;
}
```

Add
Clear
Divide
GetFinalResults
Multiply
PrintResult
Subtract

public : inline void clsCalculator::Multiply(float Number)
File: Main2.cpp



```
clsCalculator Calculator1;

Calculator1.Clear();

Calculator1.Add(10);
Calculator1.PrintResult();

Calculator1.Add(100);
Calculator1.PrintResult();
```

```
C:\Users\USER\source\repos\Main2\x64\Debug\Main2.exe
Result After Adding 10 is: 10
Result After Adding 100 is: 110
```

```
Calculator1.Clear();

Calculator1.Add(10);
Calculator1.PrintResult();

Calculator1.Add(100);
Calculator1.PrintResult();

Calculator1.Subtract(20);
Calculator1.PrintResult();
```

```
C:\Users\USER\source\repos\Main2\x64\Debug\Main2.exe
Result After Adding 10 is: 10
Result After Adding 100 is: 110
Result After Subtracting 20 is: 90
```

```
Calculator1.Divide(0);
Calculator1.PrintResult();

Calculator1.Divide(2);
Calculator1.PrintResult();
```

```
C:\Users\USER\source\repos\Main2\x64\Debug\Main2.exe
Result After Adding 10 is: 10
Result After Adding 100 is: 110
Result After Subtracting 20 is: 90
Result After Dividing 1 is: 90
Result After Dividing 2 is: 45
```

```
Calculator1.Multiply(3);
Calculator1.PrintResult();
```

```
Result After Multiplying 3 is: 135
```

```
Calculator1.Clear();
Calculator1.PrintResult();
```

```
Result After Clear 0 is: 0
```

Project 1: Calculator - Solution

```
#include<iostream>
using namespace std;

 class clsCalculator{
 private:
     float _Result = 0;
     float _LastNumber = 0;
     string _LastOperation = "Clear";
     float _PreviousResult = 0;

     bool _IsZero(float Number) {
      return  (Number == 0);}


public:

    void Add(float Number) {
        _LastNumber = Number;
        _PreviousResult = _Result;
        _LastOperation = "Adding";
        _Result += Number;
    }

    void Subtract(float Number) {
        _LastNumber = Number;
        _PreviousResult = _Result;
        _LastOperation = "Subtracting";
        _Result -= Number;
    }

    void Divide(float Number) {
         _LastNumber = Number;
       if (_IsZero(Number)) {
         Number = 1;
       }
       _PreviousResult = _Result;
       _LastOperation = "Dividing";
       _Result /= Number;
    }

    void Multiply(float Number) {
        _LastNumber = Number;
        _LastOperation = "Multiplying";
        _PreviousResult = _Result;
        _Result *= Number;
    }
```

```
    float GetFinalResults() {
     return  _Result;
    }

    void Clear() {
        _LastNumber = 0;
        _PreviousResult = 0;
        _LastOperation = "Clear";
        _Result = 0;
    }

    void CancelLastOperation() {
         _LastNumber = 0;
         _LastOperation = "Cancelling Last Operation";
         _Result = _PreviousResult;
    }

    void PrintResult() {
        cout << "Result ";
        cout << "After " << _LastOperation << " " << _LastNumber << " is: " << _Result <<
"\n";
    }
 }
                                                    ;

 int main() {
     clsCalculator Calculator1;

     Calculator1.Clear();

     Calculator1.Add(10);
     Calculator1.PrintResult();

     Calculator1.Add(100);
     Calculator1.PrintResult();

     Calculator1.Subtract(20);
     Calculator1.PrintResult();

     Calculator1.Divide(0);
     Calculator1.PrintResult();

     Calculator1.Divide(2);
     Calculator1.PrintResult();

     Calculator1.Multiply(3);
     Calculator1.PrintResult();

     Calculator1.CancelLastOperation();
     Calculator1.PrintResult();

     Calculator1.Clear();
     Calculator1.PrintResult();

     system("pause>0");
     return  0;
 }
```

Constructors

اكبر غلط انك يكون عندك object فاضي مافيهوش بيانات في المتغيرات بتاعته علي الأقل حط فيه initial values يعني قيم مبدأية

للغرض ده عملوه حاجه اسمها constructor

كل object انت بتعرفه لازم يكون ليه constructor ولو انت ماعملتهوش ال compiler بيعرفه عنك

ال constructor هوا function بتاخد نفس اسم الكلاس بتاعها بقدر اعمل جواها أي حاجه
بمجرد مابعرفه ال default constructor بيتلغي
والاكواد اللي فيه لازم تتنفذ بمجرد ماتاخد object من الكلاس ده
سواء انك تعمل عمليات معينه او انك تجبر اللي هيستخدم الكلاس بتاعك انه يدخلك قيم المتغيرات اللي
انت حاططها في الكلاس ده
وطبعا تقدر تسيبه فاضي

```cpp
#include<iostream>
using namespace std;

 class clsAddress{
 private:
 string _AddressLine1;
 string _AddressLine2;
 string _POBox;
 string _ZipCode;

 public :
     clsAddress(string AddressLine1, string AddressLine2, string POBox, string ZipCode){
         _AddressLine1 = AddressLine1;
         _AddressLine2 = AddressLine2;
         _POBox = POBox;
         _ZipCode = ZipCode;
     }

     void SetAddressLine1(string AddressLine1){
         _AddressLine1 = AddressLine1;}

     string AddressLine1(){
         return _AddressLine1;}

     void SetAddressLine2(string AddressLine2){
         _AddressLine2 = AddressLine2;}

     string AddressLine2(){
         return _AddressLine2;}

     void SetPOBox(string POBox) {
         _POBox = POBox;
     }

     string POBox() {
         return _POBox;
     }

     void SetZipCode(string ZipCode) {
         _ZipCode = ZipCode; }

     string ZipCode() {
         return _ZipCode; }

     void Print() {
         cout << "\nAddress Details:\n";
         cout << "-----------------------";
         cout << "\nAddressLine1: " << _AddressLine1 << endl;
         cout << "AddressLine2: " << _AddressLine2 << endl;
         cout << "POBox       : " << _POBox << endl;
         cout << "ZipCode     : " << _ZipCode << endl; }
 };

 int main() {
     clsAddress Address1("Queen Alia Street", "B 303 ", "11192", "5555");
     Address1.Print();

     system("pause>0");
```

```
    return 0; }
```

الواجب

A constructor is a special type of member function that is called automatically when an object is created.

- True
- False

In C++, a constructor has the same name as that of the class and it does not have a return type.

- True
- False

Constructor should:

- Have the same name of the class.
- Should not return type.
- Should be Public.
- None of the above.

Default Constructor: A constructor with no parameters is known as a default constructor.

- True
- False

You should always write Default Constructor.

No, because if you don't write it the compiler will write it for you.

Yes, you should write it always.

Parameterized Constructor: a constructor with parameters is known as a parameterized constructor. This is the preferred method to initialize member data.

True

False

When you have a parameterized constructor it will override the default constructor.

True

False

A constructor is primarily used to initialize objects. They are also used to run a default code when an object is created.

True

False

Copy Constructors

لو انا عندي object من كلاس معين اقدر اعمل object تاني من نفس الكلاس واقوله ان ال object الجديد هوا هوا ال object القديم زي ماكنت بعمل في المتغيرات

طيب هنا حابب يقولك حاجتين
اول حاجه هيا انك زي ماكنت بتقدر تعمل دالتين بنفس الاسم (overloading) قالك تقدر تعمل اكتر من constructor وكل واحد بياخد parameters غير التاني

تاني حاجه هيا ال copy constructor وهوا عباره عن انك بتعمل constructor بتديله ال parameter عباره عن object من نفس الكلاس اللي فيه ال constructor ده
وتبدا تاخد الداتا اللي في القديم تحطها عندك
ال copy constructor مش ضروري يكون موجود لان ال compiler بيكتبه عنك لما تقوله ان ال object الجديد بيساوي القديم

```
#include<iostream>
using namespace std;

 class clsAddress{
 private:
 string _AddressLine1;
 string _AddressLine2;
 string _POBox;
 string _ZipCode;

 public :
     clsAddress(string AddressLine1, string AddressLine2, string POBox, string ZipCode){
         _AddressLine1 = AddressLine1;
         _AddressLine2 = AddressLine2;
         _POBox = POBox;
         _ZipCode = ZipCode;
     }

     //Copy Constructor
     clsAddress(clsAddress & old_obj){
         _AddressLine1 = old_obj.AddressLine1();
         _AddressLine2 = old_obj.AddressLine2();
         _POBox = old_obj.POBox();
         _ZipCode = old_obj.ZipCode();}

     void SetAddressLine1(string AddressLine1){
         _AddressLine1 = AddressLine1;}

     string AddressLine1(){
         return _AddressLine1;}

     void SetAddressLine2(string AddressLine2){
         _AddressLine2 = AddressLine2;}

     string AddressLine2(){
         return _AddressLine2;}

     void SetPOBox(string POBox) {
         _POBox = POBox;
     }

     string POBox() {
         return _POBox;
     }

     void SetZipCode(string ZipCode) {
         _ZipCode = ZipCode; }

     string ZipCode() {
         return _ZipCode; }

     void Print() {
         cout << "\nAddress Details:\n";
         cout << "-----------------------";
         cout << "\nAddressLine1: " << _AddressLine1 << endl;
         cout << "AddressLine2: " << _AddressLine2 << endl;
         cout << "POBox       : " << _POBox << endl;
         cout << "ZipCode     : " << _ZipCode << endl; }
 };
```

```
int main() {
    clsAddress Address1("Queen Alia Street", "B 303 ", "11192", "5555");
    Address1.Print();

  clsAddress Address2 = Address1;
  Address2.Print();

    system("pause>0");
    return 0; }
```

الواجب

The copy constructor is used to initialize the members of a newly created object by copying the members of an already existing object.

True

False

The process of initializing members of an object through a copy constructor is known as copy initialization.

True

False

It is also called member-wise initialization because the copy constructor initializes one object with the existing object, both belonging to the same class on a member-by-member copy basis.

True

False

The copy constructor can be defined explicitly by the programmer. If the programmer does not define the copy constructor, the compiler does it for us.

True

False

You should always implement a copy constructor in your code.

Yes , you should.

No, because the compiler will do it for you.

What are the types of constructors:

Default Constructor

Parameterized Constructor

Copy Constructor

None of the above

Can you have more than one constructor in a class?

No, You cannot

Yes, You can using function overloading, and this is called "Constructor Overloading"

Destructors

هوا عكس ال constructor وبيتم مناداته قبل مايتم تدمير ال object بيتعرف زي ال constructor بالضبط بس بتحط قبله العلامه ~ ودي بتعملها بحرف ال ذ مع زرار ال shift
ال object بيتم تدميره لما بتخرج من ال function اللي انت فيها سواء كانت ال main او أي حته تانيه
بتستفيد منها لو عندك بيانات عاوز تستخدمها بعد كده او انك تنفذ كود معين

لوجيت عملت pointer وعملت فيه object لما بتطلع من الكود ال object ده مش بيتدمر وبيفضل في ال runtime ولازم تدمره بايدك في ال++c
عشان كده كان بيقولك لما تستخدم ال pointer لازم تعمله delete

```
#include<iostream>
using namespace std;

class clsPerson{
public:
	string FullName;
	//This is Instructor will be called when object is built.
	clsPerson(){
		FullName = "Mohammed Abu-Hadhoud";
		cout << "\nHi, I'm Constructor";}

	//This is destructor will be called when object is destroyed.
	~clsPerson(){
		cout << "\nHi, I'm Destructor";} };

void Fun1() {
	clsPerson Person1;
	//after exiting from function, person1 will be
	//destroyed and destructor will be called.
}

void Fun2() {
	clsPerson* Person2 = new clsPerson;
	//always use delete whenever you use new,
	//otherwise object will remain in memory
	delete Person2;
}

int main() {
	Fun1();
	Fun2();

	system("pause>0");
	return 0; }
```

الواجب

Destructor is an instance member function which is invoked automatically whenever an object is going to be destroyed. Meaning, a destructor is the last function that is going to be called before an object is destroyed.

True

False

Destructor has the same name as their class name preceded by a tilde (~) symbol.

True

False

It is not possible to define more than one destructor.

True

False

The destructor is only one way to destroy the object create by constructor. Hence destructor can-not be overloaded.

True

False

Destructor neither requires any argument nor returns any value.

True

False

It is automatically called when object goes out of scope.

True

False

Static Members

لما كنا بنعرف المتغير علي انه static كان المتغير ده بيفضل معاك لحد نهاية المشروع
لكن هنا المقصود بال static member انك لو عملت كلاس وعملت member جواه من النوع static هنا ال member ده هيكون مشترك بين كل ال objects اللي هتاخدها من الكلاس ده

زي ال functions اللي جوه الكلاس
في ال c++ اسمه static وفي ال visual basic اسمه shared
تقدر تستخدمه في انك مثلا تقدر تعرف انت اخدت كام object من الكلاس ده
ال initialization بتاعه لازم يكون بره الكلاس وبره ال main
أي object يقدر يعدل عليه

```cpp
#include <iostream>
#include <string>
using namespace std;


class clsA
{

public:

    int var;
    static int counter;

    clsA()
    {
        counter++;
    }

    void Print()
    {
        cout << "\nvar     = " << var << endl;
        cout << "counter = " << counter << endl;
    }

};

int clsA::counter = 0; //static variable initialisation outside the class

int main()
{


    clsA A1, A2, A3;

    A1.var = 10;
    A2.var = 20;
    A3.var = 30;

    A1.Print();
    A2.Print();
    A3.Print();


    A1.counter = 500;

    cout << "\nafter chaning the static member counter in one object:\n";

    A1.Print();
    A2.Print();
    A3.Print();

}
```

الواجب

Static Member is a variable that is shared for all objects, any object modifies it it get modified for all other objects.

True

Flase

Each Object has it's own static members.

True

False

Static members are on the class level not for each object.

True

False

Static Members are accessible from all objects.

True

False

Static Methods (Functions)

ال static variable لما كنت بتعمله جوه كلاس كان عشان تستخدمه لازم تعمل object
انما ال static method عشان تستخدمها مش لازم تعمل object عشان تستخدمها يعني بيكون shared علي مستوي ال classes
عشان تستدعيه يدوب بتكتب اسم الكلاس وبعدين اسم ال method وبتقدر تستدعيها من ال object عادي

```
#include <iostream>
using namespace std;

class clsA
{

public:

    static int Function1()
    {
```

```
        return 10;
    }

    int Function2()
    {
        return 20;
    }

};

int main()
{

    //The following line calls static function directly via class no throught the object
    //At class level you can call only static methods and static members
    cout << clsA::Function1() << endl;

    //static methods can also be called throught the object.
    clsA A1, A2;
    cout << A1.Function1() << endl;
    cout << A1.Function2() << endl;
    cout << A2.Function1() << endl;


}
```

الواجب

Static Functions can be called at class level without a need to have an object.

True

Flase

Static Functions can not be called through object.

True

False

Static Functions can be called through any object as well as through the class itself.

True

False

If you have a static function, can you access a non-static members of the class from inside that function?

Yes

No, Static methods can only access static members , because static methods can be called at class level without objects, and non static members you cannot access them without having object first.

Person Exercise - Requirements

محتاج تعمل كلاس person فيه كل المتغيرات اللي في الصورة اللي فوق وتعملهم get وset
ماعدا ال id تعمله get بس
وتعمل اتنين method واحده send email والتانيه send sms

```
using namespace std;

class clsPerson { ... };

int main()

{
    clsPerson Person1(10,"Mohammed","Abu-Hadhoud","my@gmail.com","0098387727");
    Person1.Print();

    Person1.SendEmail("Hi", "How are you?");
    Person1.SendSMS("How are you?");
```

ADVICES
ProgrammingAdvices.com
Course #10
OOP as it Should Be
Concepts

بتمنعه انه يعمل object من الكلاس الا عن طريق constructor
الداله send email بتاخد body و subject
الداله send sms بتاخد message

Person Exercise - Solution

```
#include<iostream>
using namespace std;

 class clsPerson{
 private:
 int _ID;
 string _FirstName;
 string _LastName;
 string _Email;
 string _Phone;

 public:
     clsPerson(int ID, string FirstName, string LastName,string Email, string Phone){
         _ID = ID;
         _FirstName = FirstName;
         _LastName = LastName;
         _Email = Email;
         _Phone = Phone;
     }

     //Read Only Property
     int ID(){
         return _ID;}

     //Property Set
      void setFirstName(string FirstName){
          _FirstName = FirstName;}

      //Property Get
      string FirstName(){
          return _FirstName;}

      //Property Set
      void setLastName(string LastName){
          _LastName = LastName;}

      //Property Get
      string LastName(){
          return _LastName;}

      //Property Set
      void setEmail(string Email){
          _Email = Email;}

      //Property Get
      string Email(){
          return _Email;}

      //Property Set
      void setPhone(string Phone){
          _Phone = Phone;}

      //Property Get
      string Phone(){
          return _Phone;}

      string FullName(){
          return _FirstName + " " + _LastName;}

      void Print() {
          cout << "\nInfo:";
          cout << "\n____________________";
```

```
        cout << "\nID       : " << _ID;
        cout << "\nFirstName: " << _FirstName;
        cout << "\nLastName : " << _LastName;
        cout << "\nFull Name: " << FullName();
        cout << "\nEmail    : " << _Email;
        cout << "\nPhone    : " << _Phone;
        cout << "\n___________________\n";
    }

    void SendEmail(string Subject, string Body) {
        cout << "\nThe following message sent successfully to email: "<< _Email;
        cout << "\nSubject: " << Subject;
        cout << "\nBody: " << Body << endl;
    }

    void SendSMS(string TextMessage){
        cout << "\nThe following SMS sent successfully to phone: " << _Phone;
        cout << "\n" << TextMessage << endl;
    }
};


int main() {
    clsPerson Person1(10, "Mohammed", "Abu-Hadhoud", "my@gmail.com", "0098387727");
    Person1.Print();
    Person1.SendEmail("Hi", "How are you?");
    Person1.SendSMS("How are you?");

    system("pause>0");
    return 0;
}
```

Homework - Employee Exercise

اعملو كلاس للموظف فيها التالي:

ID

First name

Last name

FullName()

Title

Email

Phone

Salary

Department

SendEmail(..)

SendSMS(...)

Print()

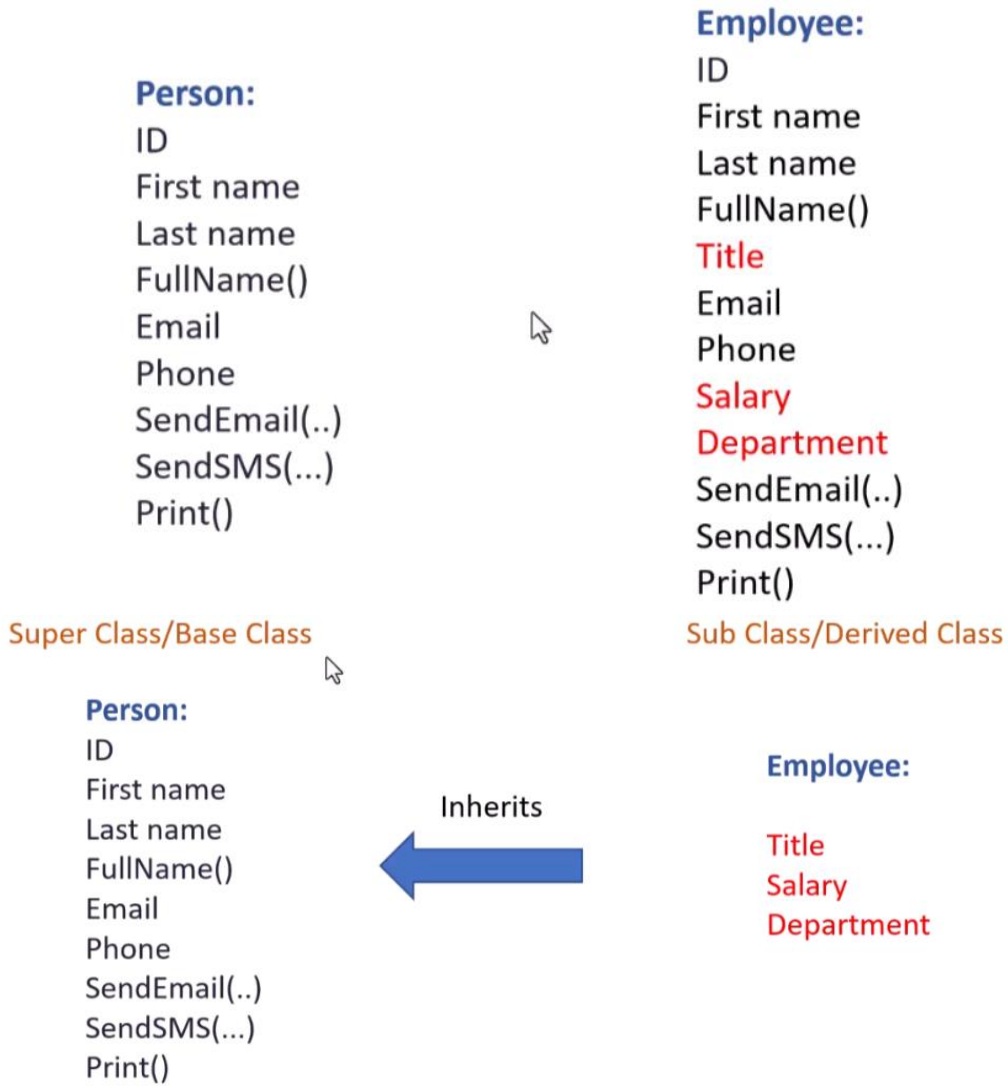
including all properties set and get, except for the ID readonly

Use constructor to iniate all values

Third Principle/Concept of OOP: Inheritance

في مسألة الواجب طلب منك تعمل كلاس للموظف
لو بصيت للمتغيرات اللي انت عايز تعملها هتلاقيها هيا هيا نفس المتغيرات اللي في الكلاس Person وعليه تغييرات بسيطه
فليه تقعد تكتب كل ده من اول وجديد في حين انك عندك في ال oop حاجه اسمها وراثة؟
حتي لو اخدت الكود copy paste لو عايز بعد كده تعدل علي حاجه هتعدل عليها مرتين مرة في الكلاس employee ومرة في الكلاس person
طب ايه الحل؟
قالك الحل في انك تعمل كلاس employee عادي بس يورث من كلاس ال person وبكده الكلاس employee بياخد كل خصائص الكلاس person بالإضافة للخصائص بتاعه هوا كمان
في الحالة دي الكلاس person بيطلق عليه subclass يعني كلاس فرعي او derived class والكلاس employee بيطلق عليه super class او base class

Last Homework - Employee



طريقة الوراثة انك بعدماتعرف ال sub class بتكتبجنبه نقطتين فوق بعض وبعدها public وبعدها اسم ال super class

لو ال super class بيطلب constructor يبقي لما لما تاخد object من ال subclass تديله ال parameters اللي عايزها منك وهنيجي للكلام ده بعدين بس عشان نتغلب عالموضوع ده ممكن نعمل constructor فاضي في ال super class
طيب لو انا مثلا جيت في الكلاس employee وحبيت لم استدعي داله الطباعه اللي موجوجه في ال person يطبع معاه ال salary هل ينفع؟
ده هنحله بعدين

ده الكلاس person

```cpp
#include <iostream>

using namespace std;

class clsPerson
{

private:
    int _ID;
    string _FirstName;
    string _LastName;
    string _Email;
    string _Phone;

public:

    //we put the default constructor here temperorly becasue
inheritance will use it,
    //in the comming lectures we will solve the prametarized
constructor with inheritance.
    clsPerson()
    {

    }

    clsPerson(int ID, string FirstName, string LastName, string
Email, string Phone)
    {
        _ID = ID;
        _FirstName = FirstName;
        _LastName = LastName;
        _Email = Email;
        _Phone = Phone;
    }

    //Read Only Property
    int ID()
    {
        return _ID;
    }

    //Property Set
    void setFirstName(string FirstName)
    {
        _FirstName = FirstName;
    }

    //Property Get
    string FirstName()
    {
        return _FirstName;
    }

    //Property Set
    void setLastName(string LastName)
    {
        _LastName = LastName;
    }
```

```
    //Property Get
    string LastName()
    {
        return _LastName;
    }

    //Property Set
    void setEmail(string Email)
    {
        _Email = Email;
    }

    //Property Get
    string Email()
    {
        return _Email;
    }

    //Property Set
    void setPhone(string Phone)
    {
        _Phone = Phone;
    }

    //Property Get
    string Phone()
    {
        return _Phone;
    }

    string FullName()
    {
        return _FirstName + " " + _LastName;
    }

    void Print()
    {
        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID       : " << _ID;
        cout << "\nFirstName: " << _FirstName;
        cout << "\nLastName : " << _LastName;
        cout << "\nFull Name: " << FullName();
        cout << "\nEmail    : " << _Email;
        cout << "\nPhone    : " << _Phone;
        cout << "\n____________________\n";

    }

    void SendEmail(string Subject, string Body)
    {

        cout << "\nThe following message sent successfully to email:
" << _Email;
        cout << "\nSubject: " << Subject;
        cout << "\nBody: " << Body << endl;

    }

    void SendSMS(string TextMessage)
    {
        cout << "\nThe following SMS sent successfully to phone: " <<
_Phone;
        cout << "\n" << TextMessage << endl;
    }

};
```

ده الكلاس
employee
بيورث من الكلاس
person

```
class clsEmployee : public clsPerson
{

private:
    string _Title;
    string _Department;
    float _Salary;

public:

    //Property Set
    void setTitle(string Title)
    {
        _Title = Title;
    }

    //Property Get
    string Title()
    {
        return _Title;
    }

    //Property Set
    void setDepartment(string Department)
    {
        _Department = Department;
    }

    //Property Get
    string Department()
    {
        return _Department;
    }

    //Property Set
    void setSalary(float Salary)
    {
        _Salary = Salary;
    }

    //Property Get
    float Salary()
    {
        return _Salary;
    }


};


int main()

{

    clsEmployee Employee1;

    Employee1.setFirstName("Mohammed");
    Employee1.setLastName("Abu-Hadhoud");
    Employee1.setEmail("a@a.com");
    Employee1.Print();
    Employee1.SendEmail("Hi", "How are you?");

    Employee1.setSalary(5000);

    cout << "Salary is: " << Employee1.Salary();

    //Calling the print will not print anything from derived class,
only base class
```

باخد object من ال
employee
وبعبي فيه داتا
وبطبعه

```
    //therfore the print method will not serve me here, this is a
problem will be solved in the next lecture.
    Employee1.Print();


    system("pause>0");
    return 0;
}
```

هنا لو حابب اطبع ال salary مش هتظهر من ضمن الداله print هنحلها بعدين

الواجب

Inheritance: Inheritance is one in which a new class is created that inherits the properties of the already exist class. It supports the concept of code reusability and reduces the length of the code in object-oriented programming.

True

False

The class that inherits properties from another class is called Subclass or Derived Class

True

False

The class whose properties are inherited by a subclass is called Base Class or Superclass.

True

Flase

Derived Class and Sub Class are the same.

True

False

Base Class and Super Class are the same

True

False

You can inherit only public and protected members, private members are not inherited.

True

False

Parameterized Constructor of the Base Class

دلوقتي انا عايز ادخل بيانات ال employee بحيث ان البيانات اللي بيورثها تنتقل للكلاس person والبيانات الخاصه بالكلاس employee تفضل زي ماهي في الكلاس employee

عشان اعمل كده هحتاج اجي جوه الكلاس employee واعمل constructor ياخد نفس ال parameters اللي بياخدها ال constructor بتاع ال person وبعدين اكتب نقطتين فوق بعض واكتب اسم الكلاس ال super وافتح قوسين اكتب فيهم المتغيرات اللي عاوز ابعتهاله واللي جايه من الconstructor بتاع ال employee

ده كلاس ال person	#include <iostream> using namespace std; class clsPerson { private: int _ID; string _FirstName; string _LastName; string _Email; string _Phone; public: clsPerson(int ID, string FirstName, string LastName, string Email, string Phone) { _ID = ID; _FirstName = FirstName; _LastName = LastName; _Email = Email; _Phone = Phone; } //Read Only Property int ID() { return _ID;

```
    }

    //Property Set
    void setFirstName(string FirstName)
    {
        _FirstName = FirstName;
    }

    //Property Get
    string FirstName()
    {
        return _FirstName;
    }

    //Property Set
    void setLastName(string LastName)
    {
        _LastName = LastName;
    }

    //Property Get
    string LastName()
    {
        return _LastName;
    }

    //Property Set
    void setEmail(string Email)
    {
        _Email = Email;
    }

    //Property Get
    string Email()
    {
        return _Email;
    }

    //Property Set
    void setPhone(string Phone)
    {
        _Phone = Phone;
    }

    //Property Get
    string Phone()
    {
        return _Phone;
    }

    string FullName()
    {
        return _FirstName + " " + _LastName;
    }

    void Print()
    {
        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID       : " << _ID;
        cout << "\nFirstName: " << _FirstName;
        cout << "\nLastName : " << _LastName;
        cout << "\nFull Name: " << FullName();
        cout << "\nEmail    : " << _Email;
        cout << "\nPhone    : " << _Phone;
        cout << "\n____________________\n";

    }
```

```
    void SendEmail(string Subject, string Body)
    {

        cout << "\nThe following message sent successfully to
email: " << _Email;
        cout << "\nSubject: " << Subject;
        cout << "\nBody: " << Body << endl;

    }

    void SendSMS(string TextMessage)
    {
        cout << "\nThe following SMS sent successfully to phone:
" << _Phone;
        cout << "\n" << TextMessage << endl;
    }


};

class clsEmployee : public clsPerson
{

private:
    string _Title;
    string _Department;
    float _Salary;

public:

    clsEmployee(int ID, string FirstName, string LastName, string
Email, string Phone, string Title, string Department, float
Salary)
        : clsPerson(ID, FirstName, LastName, Email, Phone)

    {

        _Title = Title;
        _Department = Department;
        _Salary = Salary;

    }

    //Property Set
    void setTitle(string Title)
    {
        _Title = Title;
    }

    //Property Get
    string Title()
    {
        return _Title;
    }

    //Property Set
    void setDepartment(string Department)
    {
        _Department = Department;
    }

    //Property Get
    string Department()
    {
        return _Department;
    }

    //Property Set
    void setSalary(float Salary)
```

ده كلاس ال employee بيورث من person

هنا ده constructor بياخد المتغيرات اللي ال person عاوزها ويديهاله وبياخد المتغيرات اللي هوا نفسه عاوزها بيخزنها عنده

```
    {
        _Salary = Salary;
    }

    //Property Get
    float Salary()
    {
        return _Salary;
    }

};


int main()

{

    clsEmployee Employee1(10, "Mohammed", "Abu-Hadhoud",
"A@a.com", "8298982", "CEO", "ProgrammingAdvices", 5000);

    Employee1.Print();
    cout << "\n" << Employee1.Title() << endl;
    cout << "\n" << Employee1.Department() << endl;
    cout << "\n" << Employee1.Salary() << endl;


    system("pause>0");
    return 0;
}
```

هنا لما باجي اعمل object بديله كل المتغيرات اللي هوا عاوزها

Function Overriding

ال overriding هوا عباره عن function موجوده في الsuper class بكتبها بنفس الاسم واخليها ترجع نفس النوع في ال sub class بس بعدل في الكود بتاعها بحيث انها تلائم اكتر ال subclass

زي قبل كده لما كنا عاوزين نطبع بيانات الموظف كان فيه المتغيرات المتعلقه بكلاس ال employee مش عارف اطبعها فبكتب اسم ال function عادي في ال employee وابدا اعدل عالكود زي كده

```
void Print()
{
```

فيه معلومه هنا انك لو عايز ال function دي تستدعي ال function اللي موجوده في ال super class بتعملها كده

```
void Print()
{

    clsPerson::Print();

    cout << "\nTitle: " << _Title;
    cout << "\nDepartment    : " << _Department;
    cout << "\nSalary    : " << _Salary;


}
```

طيب ماشي انا دلوقتي عاوز اكتب الكود الجديد اللي هيتنفذ فكتبته كده وطلع خطأ

```
void Print()
{

    cout << "\nInfo:";
    cout << "\n_____________________";
    cout << "\nID        : " << _ID;
    cout << "\nFirstName: " << _FirstName;
    cout << "\nLastName  : " << _LastName;
    cout << "\nFull Name: " << FullName();
    cout << "\nEmail     : " << _Email;
    cout << "\nPhone     : " << _Phone;
    cout << "\n_____________________\n";


    cout << "\nTitle: " << _Title;
    cout << "\nDepartment     : " << _Department;
    cout << "\nSalary     : " << _Salary;
```

قالك هنا الخطأ سببه ان المتغيرات الموجوده في كلاس ال person معمولة private فعشان توصلها هتستدعي ال property get يأما لو كنت جايبها أصلا من ال constructor اللي موجود في ال employee تقدر تاخدها منه عادي

```
#include <iostream>

using namespace std;

class clsPerson
{

private:
    int _ID;
    string _FirstName;
    string _LastName;
    string _Email;
    string _Phone;

public:

    clsPerson(int ID, string FirstName, string LastName, string Email, string Phone)
    {
        _ID = ID;
        _FirstName = FirstName;
        _LastName = LastName;
        _Email = Email;
        _Phone = Phone;
```

```
    }

    //Read Only Property
    int ID()
    {
        return _ID;
    }

    //Property Set
    void setFirstName(string FirstName)
    {
        _FirstName = FirstName;
    }

    //Property Get
    string FirstName()
    {
        return _FirstName;
    }

    //Property Set
    void setLastName(string LastName)
    {
        _LastName = LastName;
    }

    //Property Get
    string LastName()
    {
        return _LastName;
    }

    //Property Set
    void setEmail(string Email)
    {
        _Email = Email;
    }

    //Property Get
    string Email()
    {
        return _Email;
    }

    //Property Set
    void setPhone(string Phone)
    {
        _Phone = Phone;
    }

    //Property Get
    string Phone()
    {
        return _Phone;
    }

    string FullName()
    {
        return _FirstName + " " + _LastName;
    }

    void Print()
    {
        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID       : " << _ID;
        cout << "\nFirstName: " << _FirstName;
        cout << "\nLastName : " << _LastName;
        cout << "\nFull Name: " << FullName();
```

```
        cout << "\nEmail    : " << _Email;
        cout << "\nPhone    : " << _Phone;
        cout << "\n____________________\n";

    }

    void SendEmail(string Subject, string Body)
    {

        cout << "\nThe following message sent successfully to email: " << _Email;
        cout << "\nSubject: " << Subject;
        cout << "\nBody: " << Body << endl;

    }

    void SendSMS(string TextMessage)
    {
        cout << "\nThe following SMS sent successfully to phone: " << _Phone;
        cout << "\n" << TextMessage << endl;
    }


};

class clsEmployee : public clsPerson
{

private:
    string _Title;
    string _Department;
    float _Salary;

public:

    clsEmployee(int ID, string FirstName, string LastName, string Email, string Phone,
string Title, string Department, float Salary)
        : clsPerson(ID, FirstName, LastName, Email, Phone)

    {

        _Title = Title;
        _Department = Department;
        _Salary = Salary;

    }

    //Property Set
    void setTitle(string Title)
    {
        _Title = Title;
    }

    //Property Get
    string Title()
    {
        return _Title;
    }

    //Property Set
    void setDepartment(string Department)
    {
        _Department = Department;
    }

    //Property Get
    string Department()
    {
        return _Department;
    }
```

```
    //Property Set
    void setSalary(float Salary)
    {
        _Salary = Salary;
    }

    //Property Get
    float Salary()
    {
        return _Salary;
    }

    void Print()
    {

        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID         : " << ID();
        cout << "\nFirstName  : " << FirstName();
        cout << "\nLastName   : " << LastName();
        cout << "\nFull Name  : " << FullName();
        cout << "\nEmail      : " << Email();
        cout << "\nPhone      : " << Phone();

        cout << "\nTitle      : " << _Title;
        cout << "\nDepartment: " << _Department;
        cout << "\nSalary     : " << _Salary;

        cout << "\n____________________\n";

    }

};


int main()

{

    clsEmployee Employee1(10, "Mohammed", "Abu-Hadhoud", "A@a.com", "8298982", "CEO",
"ProgrammingAdvices", 5000);

    Employee1.Print();

    system("pause>0");
    return 0;
}
```

الواجب

Function Overriding : The function in derived class overrides the function in base class.

True

False

If you override a function in base class will you be able to access this function from the object of derived class

True

False

If you override a function in base class will you be able to access this function inside derived class?

True

False

If you override a function in base how to access it from within the derived class?

BaseClassName.FunctionName()

BaseClassName::FunctionName()

Homework - Developer Exercise

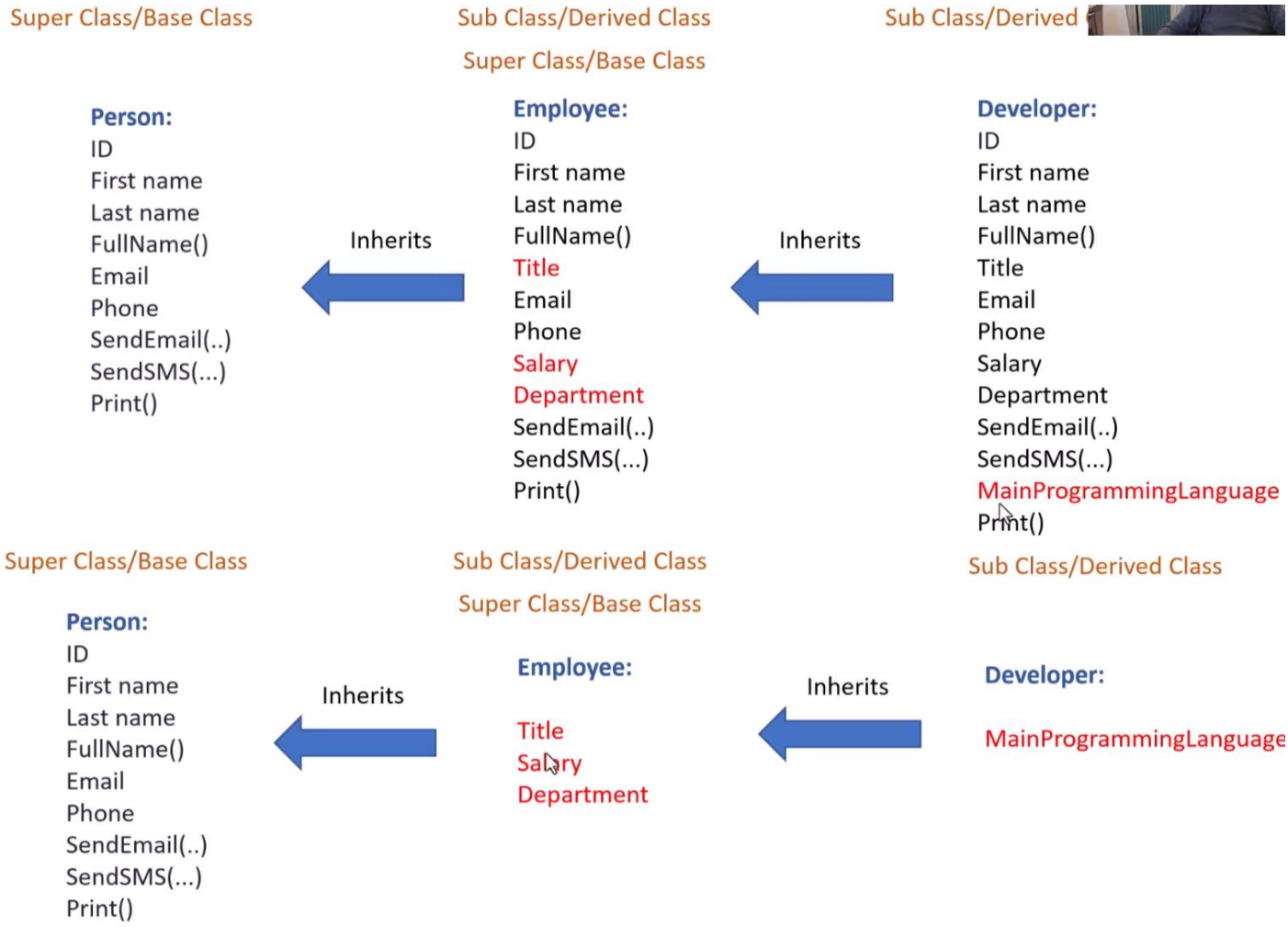
اعملو كلاس للمبرمج فيها التالي:

ID
First name
Last name
FullName()
Title
Email
Phone
Salary
Department
SendEmail(..)
SendSMS(...)
Title
Department

Salary
MainProgrammingLanguage
Print()
including all properties set and get, except for the ID readonly
Use constructor to initiate all values

Multi Level Inheritance

Last Homework - Developer




```
#include <iostream>

using namespace std;

class clsPerson
{

private:
    int _ID;
    string _FirstName;
    string _LastName;
    string _Email;
    string _Phone;

public:

    clsPerson(int ID, string FirstName, string LastName, string Email, string Phone)
```

```
    {
        _ID = ID;
        _FirstName = FirstName;
        _LastName = LastName;
        _Email = Email;
        _Phone = Phone;
    }

    //Read Only Property
    int ID()
    {
        return _ID;
    }

    //Property Set
    void setFirstName(string FirstName)
    {
        _FirstName = FirstName;
    }

    //Property Get
    string FirstName()
    {
        return _FirstName;
    }

    //Property Set
    void setLastName(string LastName)
    {
        _LastName = LastName;
    }

    //Property Get
    string LastName()
    {
        return _LastName;
    }

    //Property Set
    void setEmail(string Email)
    {
        _Email = Email;
    }

    //Property Get
    string Email()
    {
        return _Email;
    }

    //Property Set
    void setPhone(string Phone)
    {
        _Phone = Phone;
    }

    //Property Get
    string Phone()
    {
        return _Phone;
    }

    string FullName()
    {
        return _FirstName + " " + _LastName;
    }

    void Print()
    {
```

```
        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID       : " << _ID;
        cout << "\nFirstName: " << _FirstName;
        cout << "\nLastName : " << _LastName;
        cout << "\nFull Name: " << FullName();
        cout << "\nEmail    : " << _Email;
        cout << "\nPhone    : " << _Phone;
        cout << "\n____________________\n";

    }

    void SendEmail(string Subject, string Body)
    {

        cout << "\nThe following message sent successfully to email: " << _Email;
        cout << "\nSubject: " << Subject;
        cout << "\nBody: " << Body << endl;

    }

    void SendSMS(string TextMessage)
    {
        cout << "\nThe following SMS sent successfully to phone: " << _Phone;
        cout << "\n" << TextMessage << endl;
    }


};

class clsEmployee : public clsPerson
{

private:
    string _Title;
    string _Department;
    float _Salary;

public:

    clsEmployee(int ID, string FirstName, string LastName, string Email, string Phone,
string Title, string Department, float Salary)
        : clsPerson(ID, FirstName, LastName, Email, Phone)

    {

        _Title = Title;
        _Department = Department;
        _Salary = Salary;

    }

    //Property Set
    void setTitle(string Title)
    {
        _Title = Title;
    }

    //Property Get
    string Title()
    {
        return _Title;
    }

    //Property Set
    void setDepartment(string Department)
    {
        _Department = Department;
    }
```

```
    //Property Get
    string Department()
    {
        return _Department;
    }

    //Property Set
    void setSalary(float Salary)
    {
        _Salary = Salary;
    }

    //Property Get
    float Salary()
    {
        return _Salary;
    }

    void Print()
    {

        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID         : " << ID();
        cout << "\nFirstName  : " << FirstName();
        cout << "\nLastName   : " << LastName();
        cout << "\nFull Name  : " << FullName();
        cout << "\nEmail      : " << Email();
        cout << "\nPhone      : " << Phone();

        cout << "\nTitle      : " << _Title;
        cout << "\nDepartment: " << _Department;
        cout << "\nSalary     : " << _Salary;

        cout << "\n____________________\n";

    }

};

class clsDeveloper : public clsEmployee

{

private:
    string _MainProgrammingLanguage;

public:

    clsDeveloper(int ID, string FirstName, string LastName, string Email, string Phone,
string Title,
        string Department, float Salary, string MainProgrammingLanguage)
        : clsEmployee(ID, FirstName, LastName, Email, Phone, Title, Department, Salary)

    {
        _MainProgrammingLanguage = MainProgrammingLanguage;
    }

    //Property Set
    void setMainProgrammingLanguage(string MainProgrammingLanguage)
    {
        _MainProgrammingLanguage = MainProgrammingLanguage;
    }

    //Property Get
    string MainProgrammingLanguage()
    {
        return _MainProgrammingLanguage;
```

```
    }

    void Print()
    {

        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID         : " << ID();
        cout << "\nFirstName  : " << FirstName();
        cout << "\nLastName   : " << LastName();
        cout << "\nFull Name  : " << FullName();
        cout << "\nEmail      : " << Email();
        cout << "\nPhone      : " << Phone();

        cout << "\nTitle      : " << Title();
        cout << "\nDepartment: " << Department();
        cout << "\nSalary     : " << Salary();
        cout << "\nPLanguage  : " << MainProgrammingLanguage();
        cout << "\n____________________\n";

    }

};

int main()

{

    clsDeveloper Developer1(10, "Mohammed", "Abu-Hadhoud", "A@a.com", "8298982", "Web
Developer",
        "ProgrammingAdvices", 5000, "C++");

    Developer1.Print();

    Developer1.SendSMS("Hi mr Developer :-)");


    system("pause>0");
    return 0;
}
```

Access Specifiers/Modifiers Review

```
#include <iostream>

using namespace std;

class clsA
{
private:
    //only accessible   inside this class, neither derived classes nor outside class.
    int _Var1;
    void _Fun1()
    {
        cout << "Function 1";
    }

protected:
    //only accessible  inside this class and all derived classes, but not outside class
    int Var2;
    void Fun2()
    {
        cout << "Function 1";
    }

public:
    // Accessable inside this class, all derived classes, and outside class
    int Var3;
    void Fun3()
    {
```

```
        cout << "Function 1";
    }

};

class clsB : public clsA
{

public:

    void Func1()
    {
        cout << clsA::Var2;
    }
};

int main()

{


    system("pause>0");
    return 0;
}
```

الواجب

Access modifiers (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members.

True

False

Which of the following is Access Specifiers/Modifiers:

Public

Private

Protected

Constant

Public Members can be accessed from inside and outside the class.

- True
- False

Private Members can be accessed from outside the class through object.

- True
- Flase

Private Members can be accessed by any class inherits the current class.

- Ture
- False

Private Members can be accessed only from inside the class, it cannot be accessed from outside the class nor from the classes inherits the current class..

- True
- Flase

If you want to have a member that is private to outside class and public to classes inherits the current class, which access specifier/modifier you use?

- Public
- Private
- Protected

Protected Members can be accessed from outside class through objects.

True

False

Protected Members can be accessed from inside class and from all classes inherits the current class.

Tue

False

OOP is more secured because you can hide members from developers.

True

False

Inside the class I can access everything including Public, Private , and Protected Members.

True

False

Inheritance Visibility Modes

لما كنا بنعمل وراثه كنا بنكتب قبل اسم الكلاس اللي عاوزين نورث منه كلمة public
ال public دي اسمها visibility mode
الvisibility mode فايدتها في انك بتحدد استخدام العناصر اللي موجوده في الكلاس اللي هتورث منه
يعني انت مثلا عملت كلاس a وعملت كلاس b بيورث من كلاس a
لما ييجي حد ياخد object من الكلاس b هل انت عايز العناصر اللي في كلاس a تظهرله ولا لا
يعني العناصر اللي بتظهرلك عن طريق الوراثه عايزها تظهرللي هيستخدم الكلاس بتاعك؟ ولا عايز الموضوع يقف لحد عندك انت بس؟

بيقولك فيه 3 أنواع لل visibility mode :-

1- Public inheritance :- وده بيخلي كل شيء كماهوا ال public بيفضل public وال private بيفضل private

2- Private inheritance :- وده بيخلي العناصر المرئية مرئية ليك انت بس واللي هيستخدم الكلاس بتاعك مش هيشوفها

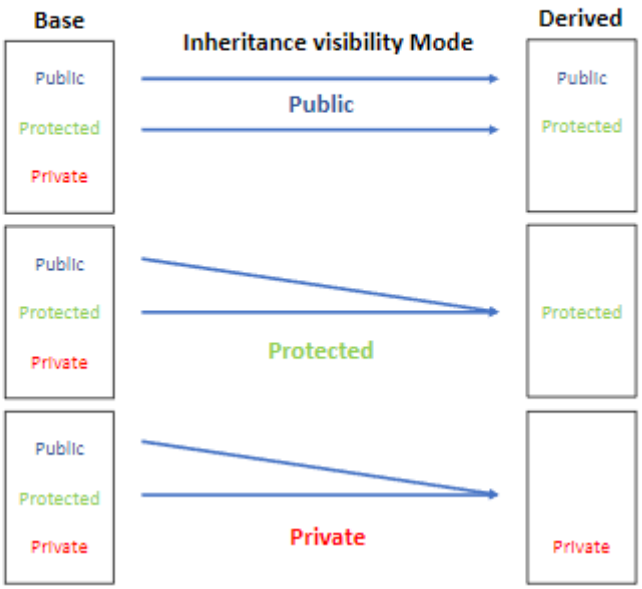
3- Protected inheritance :- وده بيخلي العناصر المرئية مرئية ليك وللي هييجي يورث من الكلاس بتاعك انما لو هياخد من الكلاس بتاعك object مش هتظهرله

Syntax:

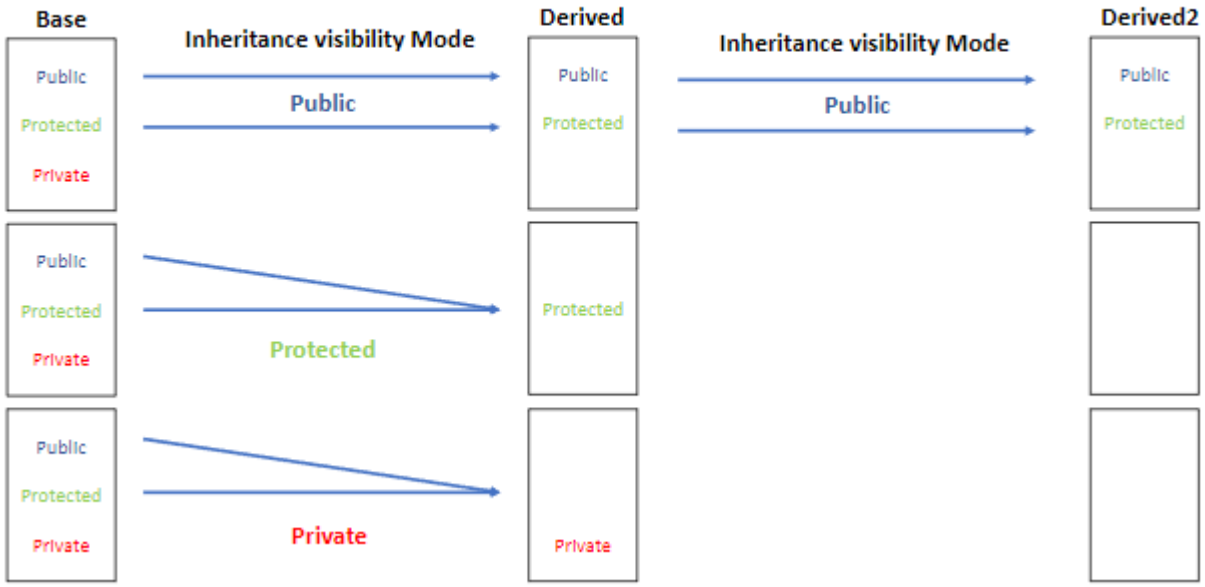
```
class DerivedClassName : <Visibility Mode> BaseClassName
{

};
```

	Private Members	Protected Members	Public Members
Public Inheritance	Inaccessible	Protected	Public
Private Inheritance	Inaccessible	Private	Private
Protected Inheritance	Inaccessible	Protected	Protected

Visibility Modes



Visibility Modes




```
#include <iostream>
using namespace std;

class clsA
{

private:
    int V1;
    int Fun1()
    {
        return 1;
    }

protected:
    int V2;
    int Fun2()
    {
        return 2;
    }

public:
    int V3;
    int Fun3()
    {
        return 3;
    }

};

//try to change visibility mode public/private/protected
//and see in the main what will happen iside objects.
class clsB : private clsA
{

public:

    int Fun4()
    {

        return 4;
    }

};
```

```
class clsC : public clsB
{

public:

    int Fun5()
    {

        return 5;
    }
};


int main()
{


    clsB B1;
    //Try B1 after you change visibility mode in clsB.   and see what you can see.


    clsC C1;
    //Try C1 after you change visibility mode in clsB.   and see what you can see.

}
```

الواجب

Visibility Mode "Private" will make everything private in the derived class, so it can make use of it and no one will make use of it neither derived classes nor objects.

True

False

Visibility Mode "Protected" will make everything protected in the derived class, so it can make use of it and all other derived classes will make use of it, but no objects will make use of it.

True

False

Visibility Mode "Public" will inherit the class publicly so every public members and protected members are useful for others.

True

False

Private Members in the Base Class are not accessible from outside the class nor the derived classes.

True

False

Inheritance Types

ال inheritance ليها انوع :-

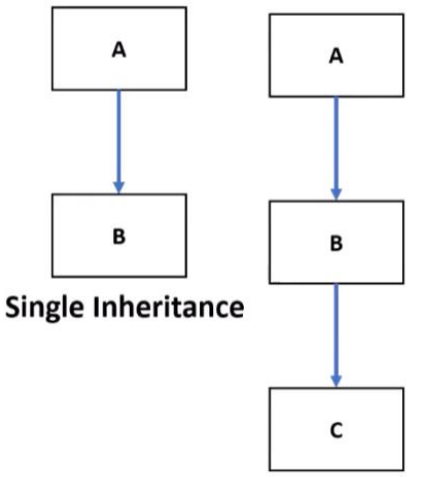
1- Single inheritance :- وفيها كلاس بيورث من كلاس

2- Multi level inheritance :- وفيها كلاس بيورث من كلاس بيورث من كلاس زي جد وابن وحفيد

3- Hierarchal inheritance :- ودول عبارة عن اكتر من كلاس بيورثوا من كلاس واحد زي اب له عدة أبناء

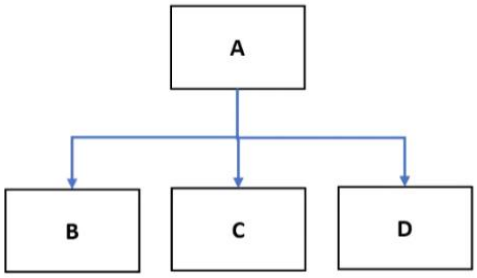
4- Multiple inheritance :-وده كلاس بيورث من اكتر من كلاس والتعامل معاها مش سهل ولغات البرمجه الحديثه لغوها

5- Hybrid inheritance وده كلاس بيورث من اكتر من كلاس والاكتر من كلاس دول بيورثوا من كلاس واحد

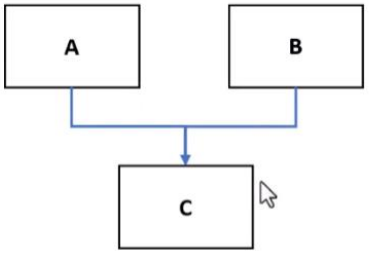


Single Inheritance

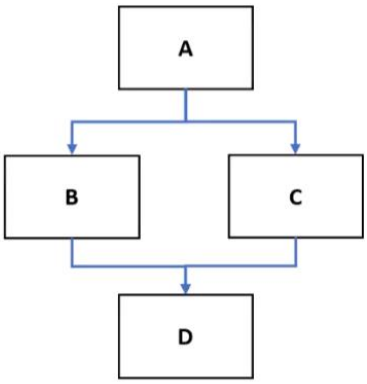
Multi Level Inheritance



Hierarchal Inheritance



Multiple Inheritance



Hybrid Inheritance

الواجب

What are types of Inheritance?

- Single inheritance
- Multi-level inheritance
- Multiple inheritance
- Hybrid inheritance
- Hierarchical inheritance
- All of the above

Multiple inheritance are not supported by modern languages such as JAVA and C#

True

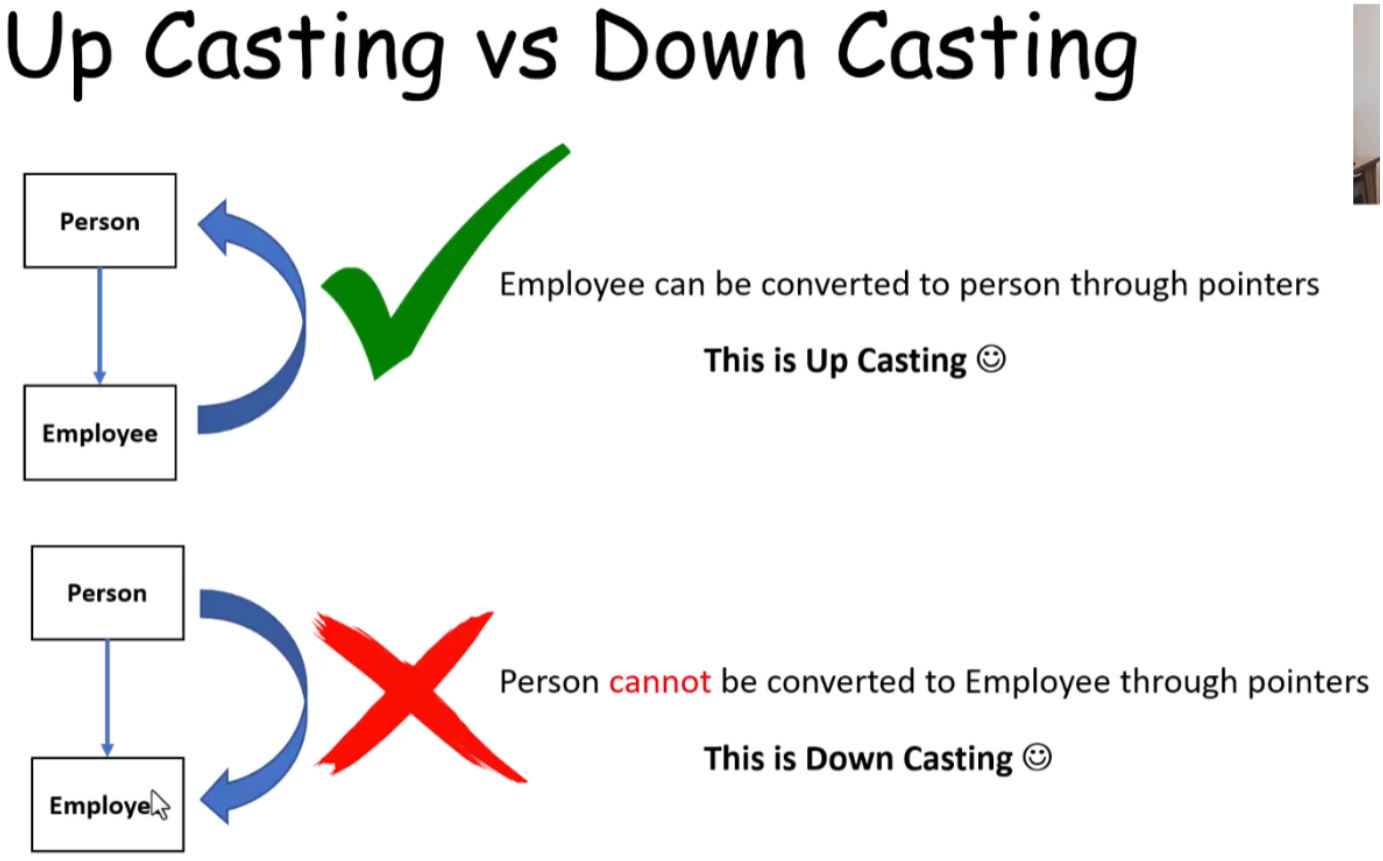
False

Up Casting vs Down Casting

هنا بيقولك انك ينفع تحول من كلاس للتاني وفيه نوعين من التحويل :-

1- Up casting :- وفيه الكلاس الابن يقدر يتحول للاب وده بيتم عن طريق ال pointer وانك بتخلي ال object اللي من النوع person يؤشر علي ال object اللي من النوع employee

2- Down casting :- وهنا لو عايز احول من الاب للابن وده ماينفعش لان الابن اكبر من الاب




```cpp
#include <iostream>

using namespace std;

class clsPerson
{

public:

    string FullName = "Mohammed Abu-Hadhoud";

};
```

```cpp
class clsEmployee : public clsPerson
{

public:
    string Title = "CEO";

};

int main()

{

    clsEmployee Employee1;

    cout << Employee1.FullName << endl;

    ////upcasting
    //this will convert employee to person.
    clsPerson* Person1 = &Employee1;
    cout << Person1->FullName << endl;


    //clsPerson Person2;
    //cout << Person2.FullName << endl;

    ////downcasting : you cannot convert person to employee
    //clsEmployee* Employee2 = &Person2;



    system("pause>0");
    return 0;
}
```

الواجب

Up Casting is converting derived object to it's base object

True

False

Down Casting is Converting Base object to Derived object

True

False

A pointer of type parent can point to an object of child class.

True, because all the members in which the pointer can access are exist in memory when the object of child class.

False

A pointer of child class cannot point to an object of parent class.

True, because the child class members the pointer can access do not exist in memory when the object is of parent class.

False

Virtual Functions

دلوقتي انا عندي كلاس اسمه person وفيه function اسمها print
قومت عملت كلاس اسمه employee بيورث من الكلاس person وبيعمل override لل function اللي اسمها print
بعد كده اخدت object من الكلاس employee واستدعيت الfunction اللي اسمها print
كده هينفذ الكود اللي موجود في ال employee
فيها حاجه دي؟
لا

طيب انا جيت دلوقتي عملت مؤشر من الكلاس person وخليته يؤشر علي كلاس ال employee
تقدر تقولي انا كده عملت ايه؟
اه انت كده حولت الemployee ل person عادي

قولتلك طب لو جيت دلوقتي تستدعي ال function اللي اسمها print انهي كود اللي هيتنفذ ؟
الكود اللي في ال person ولا الكود اللي في ال employee ؟

هنا بما انك عملت عملية تحويل من ال employee لل person فاانت تنسي خالص انك موجود في الكلاس employee وتنفذلي الكود اللي موجود في ال person

طيب انا لو عايزه ينفذلي الكود اللي في ال employee اعمل ايه؟
قالك سهلة خالص كل اللي هتعمله انك تكتب كلمة virtual قبل ال function اللي في ال person
وبكده هيتخزن جدول في ال memory فيه العمليات بتاعت ال override عشان لما تيجي تعمل عملية التحويل ينفذ الكود اللي في الابن ويفكه خالص من الكود اللي في الاب
وبكده ال function اللي اسمها print يبقي اسمها virtual function

```
#include <iostream>

using namespace std;

class clsPerson
{

public:

    virtual  void Print()
```

```
    {
        cout << "Hi, i'm a person!\n ";

    }

};

class clsEmployee : public clsPerson
{
public:
    void Print()
    {
        cout << "Hi, I'm an Employee\n";
    }
};

class clsStudent : public clsPerson
{
public:
    void Print()
    {
        cout << "Hi, I'm a student\n";
    }
};


int main()

{

    clsEmployee Employee1;
    clsStudent  Student1;

    Employee1.Print();
    Student1.Print();



    clsPerson* Person1 = &Employee1;
    clsPerson* Person2 = &Student1;

    Person1->Print();
    Person2->Print();


    system("pause>0");
    return 0;
}
```

الواجب

A virtual function is a member function in the base class that we expect to redefine in derived classes.

True

False

Basically, a virtual function is used in the base class in order to ensure that the function is overridden. This especially applies to cases where a pointer of base class points to an object of a derived class.

True

False

Static/Early Binding vs Dynamic/Late Binding

ال static binding او ال early binding معناها ان ال compiler عارف عنوان الfunction اللي هيروح يستدعيها وده بيحصل لما بتيجي تستخدم ال function بالطريقه العاديه بتاعتنا

ال dynamic binding او ال late binding معناها ان ال compiler مش عارف العنوان وبيعرفه بس في مرحلة ال runtime وده بيحصل في حالات زي الدرس اللي فات اننا خلينا مؤشر من النوع person يقف علي الكلاس بتاع ال employee فاصبح العنوان هنا متغير

ال early binding الكود فيه بيكون اسرع شوية من ال late binding

```cpp
#include <iostream>

using namespace std;

class clsPerson
{

public:

    virtual  void Print()

    {
        cout << "Hi, i'm a person!\n ";

    }

};

class clsEmployee : public clsPerson
{
public:
    void Print()
    {
        cout << "Hi, I'm an Employee\n";
    }
};

class clsStudent : public clsPerson
{
public:
    void Print()
    {
        cout << "Hi, I'm a student\n";
    }
};
```

```cpp
int main()

{

    clsEmployee Employee1;
    clsStudent  Student1;
    //Early-Static Binding: at compilation time
    Employee1.Print();
    Student1.Print();

    clsPerson* Person1 = &Employee1;
    clsPerson* Person2 = &Student1;

    //Late-Dynamic Binding: at runtime
    Person1->Print();
    Person2->Print();


    system("pause>0");
    return 0;
}
```

الواجب

Static Binding: The binding which can be resolved at compile time by the compiler is known as static or early binding. The binding of all the static, private methods is done at compile-time.

True

False

Dynamic Binding: In Dynamic binding compiler doesn't decide the method to be called. Overriding is a perfect example of dynamic binding. In overriding both parent and child classes have the same method.

True

False

Early Binding and Static Binding are the same.

True

False

Late Binding and Dynamic Binding are the same.

True

False

Early Binding / Static Binding are done at the compilation time.

True

False

Late Binding/Dynamic Binding are done at run time.

True

False

Fourth Principle/Concept of OOP: Polymorphism

ال polymorphism معناها تعدد الاشكال ومعمول عشان يخلي الكود بتاعك مرتب وعلي نسق واحد

تقدر تحقق ال polymorphism عن طريق اربع حاجات :-

1- Function overloading :- اكتر من function ليها نفس الاسم ومختلفه في عدد ال parameters او نوعها

2- Operator overloading :- لما يكون مثلا عندك متغيرين من النوع string وعايز تخزنهم في متغير ثالث زي ال firstname وال last name كنت برتبط بينه بعلامة ال + علامة ال + دي اشتتغلت شغلتين انها لو كانت بين رقمين جمعتهم ولو كانت بين نصين رصتهم جنب بعض

3- Function overriding :- لما كلاس بيورث من كلاس وانت عايز تعدل علي function معينه

4- Virtual functions :- لما تعمل مؤشر بيؤشر عالكلاس الابن وعايز لما تستدعي function تنفذ الكود اللي في الابن

Polymorphism

Polymorphism means **"many forms"**

Polymorphism allows us to create consistent code.

Polymorphism can be done through:

- Function overloading
- Operator overloading
- Function overriding
- Virtual functions

الواجب

Why Polymorphism? Polymorphism allows us to create consistent code.

- True
- False

Polymorphism is one of the important features/principles/concepts of OOP, word Ploy means "Many" and word Morphism means "Form" so it means "Many Forms", the ability to take more than one form.

- True
- False

Interfaces: Pure Virtual Functions and Abstract Classes

هنا هنعرف ازاي نجبر المستخدم اللي هيستخدم الكود بتاعنا انه يعمل functions معينه بال parameters اللي احنا نحددها

بص كده عالكود ده

```
class clsMobile
{
    virtual void Dial(string PhoneNumber) = 0;
    virtual void SendSMS(string PhoneNumber, string Text) = 0;
    virtual void TakePicture() = 0;

};
```

هنا احنا عاملين كلاس عادي بس جينا عند ال functions وكتبنا جنبها virtual وقولنا انها بتساوي صفر

ده معناه ايه؟

ده معناه ان ال functions دي أصبحت virtual functions يعني لما ييجي حد يورث من الكلاس الكود اللي هيتنفذ هوا الكود اللي في الكلاس اللي ورث منه

معني اننا عملناها بتساوي صفر اننا معملناش implementation لل function يعني مكتبناش اكواد جواها

كده أصبحت ال function دي اسمها pure virtual function والكلاس اللي هيا فيه اصبح اسمه abstract class ومعني ال abstract class او ال interface انك ماتقدرش تاخد object من الكلاس ده

تقدر تورث منه بس

وكمان لو هتورث منه فانت لازم تعمل override لل virtual functions اللي موجوده فيه وبال parameters اللي متحدده فيه

النوع ده من ال classes وال functions بيعبر عن ما يشبه العقد

لانك اللي عمل الكلاس ده قالك عشان تستخدم الكلاس بتاعي لازم تعمل implementation للfunctions دي وبال parameters دي الأول وبعدين ابقي اعمل اللي انت عايزه في الكلاس الجديد الابن بتاعك

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud
#include <iostream>
using namespace std;


//Abstract Class / Interface / Contract.
class clsMobile
{
    virtual void Dial(string PhoneNumber) = 0;
    virtual void SendSMS(string PhoneNumber, string Text) = 0;
    virtual void TakePicture() = 0;

};

class clsiPhone : public clsMobile
{

    //This class signed a contract with clsMobile abstract class   therefore it should
implement everything in the abstract class

public:

    void Dial(string PhoneNumber)
    {

    };

    void SendSMS(string PhoneNumber, string Text)
    {

    };

    void TakePicture()
    {


    };

    void MyOwnMethod()
    {

    }

};


class clsSamsungNote10 : public clsMobile
{
    //This class signed a contract with clsMobile abstract class   therefore it should
implement everything in the abstract class

public:
    void Dial(string PhoneNumber)
    {

    };

    void SendSMS(string PhoneNumber, string Text)
    {

    };
```

```
    void TakePicture()
    {


    };

};


int main()

{
    clsiPhone iPhone1;
    clsSamsungNote10 Note10;

    system("pause>0");
    return 0;

}
```

الواجب

A pure virtual function doesn't have the function body and it must end with = 0.

True

False

If you have one pure virtual function in a class then it will be converted to abstract class.

True

False

Abstract Class is the same concept of Interface Class and they are both contracts.

Ture

False

Abstract Class/Interface Class is a class with pure virtual functions.

True

False

You can have an object of abstract class.

Yes, you can.

No, you can only inherit it.

An abstract class in C++ has at least one pure virtual function by definition. In other words, a function that has no definition.

True

False

The abstract class's descendants (derived classes) must define the pure virtual function; otherwise, it is not allowed and you will get error.

True

False

Derived Classes from abstract class can have extra methods other than the methods in the abstract class.

Yes, it can have extra methods

No

The C++ interfaces are implemented using abstract classes and these abstract classes should not be confused with data abstraction which is a concept of keeping implementation details separate from associated data.

True

False

Abstract Class is the Same as Abstraction in OOP

Yes they are the same concepts.

No, they are two different things, these abstract classes should not be confused with data abstraction which is a concept of keeping implementation details separate from associated data.

Friend Classes

دلوقتي احنا عارفين انك لو عملت class member من النوع private ماحدش بيقدر يوصه الا من داخل الكلاس نفسه
ولو عملته protected هتقدر توصله من داخل الكلاس نفسه والكلاس اللي وارث منه لكن ما تقدرش توصله من بره

طيب هنا هوا عاوز يقولك انك ممكن تعمل class وتسميه a مثلا وتعمل فيه members من أي نوع حتي لو كان private وتعمل كلاس b ومن داخل الكلاس b ده تقدر توصل لاي عنصر من عناصر الكلاس a
طيب ازاي اعمل ده؟
قالك ببساطه لو جيت في الكلاس a وكتبت friend class وبعدها اسم الكلاس b
من داخل الكلاس b هتقدر توصل لاي حاجه موجوده في الكلاس a
يعني الكلاس b يكون عنده واسطه انه يدخل عالكلاس a

```
#include<iostream>
using namespace std;

class clsA
{
private:
    int _Var1;

protected:
    int _Var3;

public:
```

```
    int Var2;

    clsA()
    {
        _Var1 = 10;
        Var2 = 20;
        _Var3 = 30;

    }
    //this will grant access for everything to class B
    friend class clsB;    //friend class

};

class clsB
{
public:
    void display(clsA A1)
    {
        cout << endl << "The value of Var1=" << A1._Var1;
        cout << endl << "The value of Var2=" << A1.Var2;
        cout << endl << "The value of Var3=" << A1._Var3;
    }
};

int main()
{
    clsA A1;
    clsB B1;

    B1.display(A1);

    system("pause>0");

    return 0;
};
```

الواجب

We can use a friend Class in C++ using the "friend" keyword.

True

False

A friend class can access both private and protected members of the class in which it has been declared as friend.

True

False

Since ClassB is a friend class, we can access all members of ClassA from inside ClassB. However, we cannot access members of ClassB from inside ClassA. It is because friend relation in C++ is only granted, not taken.

True

False

If ClassB is declared as a friend Class inside ClassA , ClassB can access all private and protected members of ClassA, and also ClassA can Access all members of ClassB.

True.

False, only ClassB can access all members of ClassB but ClassA cannot.

Friend Function

مفهوم ال friend function زيه زي ال friend class بس بدل ماكانت الواسطه في ايد كلاس أصبحت في ايد function مرمية في الشارع بره الكلاس

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud
#include<iostream>
using namespace std;

class clsA
{
private:
    int _Var1;

protected:
    int _Var3;

public:
    int Var2;

    clsA()
    {
        _Var1 = 10;
        Var2 = 20;
        _Var3 = 30;

    }

    friend int MySum(clsA A1);    //friend function

};

//this function is normal function and not a member of any class
int MySum(clsA A1)
{
```

```
    return A1._Var1 + A1.Var2 + A1._Var3;
}

//int Fun2(clsA A1)
//{
//    return A1._Var1 + A1.Var2 + A1._Var3;
//}

int main()
{
    clsA A1;

    cout << MySum(A1);

    system("pause>0");

    return 0;
};
```

الواجب

A friend function in C++ is defined as a function that can access private, protected and public members of a class.

True

False

The friend function is declared using the friend keyword inside the body of the class.

True

Flase

By using the keyword, the 'friend' compiler understands that the given function is a friend function.

True

False

We declare friend function inside the body of a class, whose private and protective data needs to be accessed, starting with the keyword friend to access the data. We use them when we need to operate between two different classes at the same time.

True

False

Friend functions of the class are granted permission to access private and protected members of the class in C++. They are defined globally outside the class scope. Friend functions are not member functions of the class.

True

False

A friend function in C++ is a function that is declared outside a class but is capable of accessing the private and protected members of the class. There could be situations in programming wherein we want two classes to share their members. These members may be data members, class functions or function templates. In such cases, we make the desired function, a friend to both these classes which will allow accessing private and protected data of members of the class.

True

False

Generally, non-member functions cannot access the private members of a particular class. Once declared as a friend function, the function is able to access the private and the protected members of these classes.

True

False

Structure Inside Class

بتقدر تعرف structure جوه كلاس عادي

```
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#include<iostream>
using namespace std;

class clsPerson {

    struct stAddress
    {
        string AddressLine1;
        string AddressLine2;
        string City;
        string Country;
    };

public:
    string FullName;
    stAddress Address;

    clsPerson()
    {
        FullName = "Mohammed Abu-Hadhoud";
        Address.AddressLine1 = "Building 10";
        Address.AddressLine2 = "Queen Rania Street";
        Address.City = "Amman";
        Address.Country = "Jordan";
    }

    void PrintAddress()
    {
        cout << "\nAddress:\n";
        cout << Address.AddressLine1 << endl;
        cout << Address.AddressLine2 << endl;
        cout << Address.City << endl;
        cout << Address.Country << endl;
    }

};

int main()

{

    clsPerson Person1;
    Person1.PrintAddress();

    system("pause>0");
    return 0;
}
```

Nested Classes

بتقدر تعمل كلاس جوه كلاس
مثلا عمنا كلاس اسمه person وجواه كلاس تاني جواه اسمه address
الكلاس person بيكون اسمه enclosure class او الكلاس المحتوي والكلاس اللي اسمه address اسمه inner class او الكلاس الداخلي

```cpp
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#include<iostream>
using namespace std;

class clsPerson {

    class clsAddress
    {
    public:
        string AddressLine1;
        string AddressLine2;
        string City;
        string Country;

        void Print()
        {
            cout << "\nAddress:\n";
            cout << AddressLine1 << endl;
            cout << AddressLine2 << endl;
            cout << City << endl;
            cout << Country << endl;
        }
    };

public:
    string FullName;
    clsAddress Address;

    clsPerson()
    {
        FullName = "Mohammed Abu-Hadhoud";
        Address.AddressLine1 = "Building 10";
        Address.AddressLine2 = "Queen Rania Street";
        Address.City = "Amman";
        Address.Country = "Jordan";
    }


};

int main()

{

    clsPerson Person1;

    Person1.Address.Print();

    system("pause>0");
    return 0;
}
```

الواجب

خد الكود اللي فات واعمل constructor لل inner class واملي البيانات عن طريق ال constructor الخرجي

Nested or Inner Classes : A class can also contain another class definition inside itself, which is called “Inner Class” in C++.

Ture

False

In the case of nested of inner classes, the containing class is referred to as the “Enclosing Class”. The Inner Class definition is considered to be a member of the Enclosing Class.

True

False

An Inner class is a member and as such has the same access rights as any other member of the enclosure class.

True

False

The members of an enclosing class have no special access to members of a nested class; the usual access rules shall be obeyed.

True

False

Nested Classes - Constructor Homework Solution

The following code will show you how to initiate an inner-class using constructor:

```
#include<iostream>
using namespace std;

class clsPerson {

    string _FullName;
```

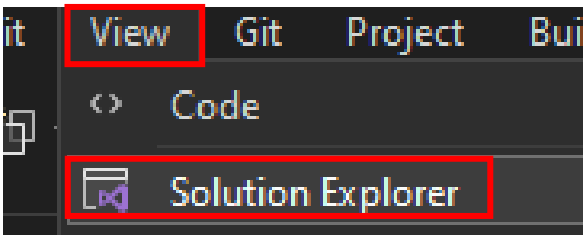
```
class clsAddress
{

private:
    string _AddressLine1;
    string _AddressLine2;
    string _City;
    string _Country;
public:


    clsAddress(string AddressLine1, string AddressLine2, string City, string Country)
    {
        _AddressLine1 = AddressLine1;
        _AddressLine2 = AddressLine2;
        _City = City;
        _Country = Country;

    }

    string setAddressLine1(string AddressLine1)
    {
        _AddressLine1 = AddressLine1;
    }

    string AddressLine1()
    {
        return _AddressLine1;
    }

    string setAddressLine2(string AddressLine2)
    {
        _AddressLine2 = AddressLine2;
    }

    string AddressLine2()
    {
        return _AddressLine2;
    }


    string setCity(string City)
    {
        _City = City;
    }

    string City()
    {
        return _City;
    }


    string setCountry(string Country)
    {
        _Country = Country;
    }

    string Country()
    {
        return _Country;
    }

    void Print()
    {
        cout << "\nAddress:\n";
        cout << _AddressLine1 << endl;
        cout << _AddressLine2 << endl;
        cout << _City << endl;
        cout << _Country << endl;
```

```
        }
    };


public:

    string setFullName(string FullName)
    {
        _FullName = FullName;
    }

    string FullName()
    {
        return _FullName;
    }

    clsAddress Address = clsAddress("", "", "", "");

    clsPerson(string FullName, string AddressLine1, string AddressLine2, string City,
string Country)
    {
        _FullName = FullName;

        //initiate address class by it's constructor
        Address = clsAddress(AddressLine1, AddressLine2, City, Country);

    }
};

int main()

{

    clsPerson Person1("Mohammed Abu-Hadhoud", "Building 10", "Queen Rania Street", "Amman",
"Jordan");

    Person1.Address.Print();

    system("pause>0");
    return 0;
}
```

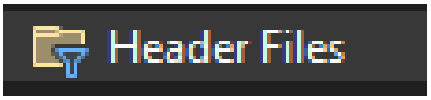
Separate Classes In Libraries

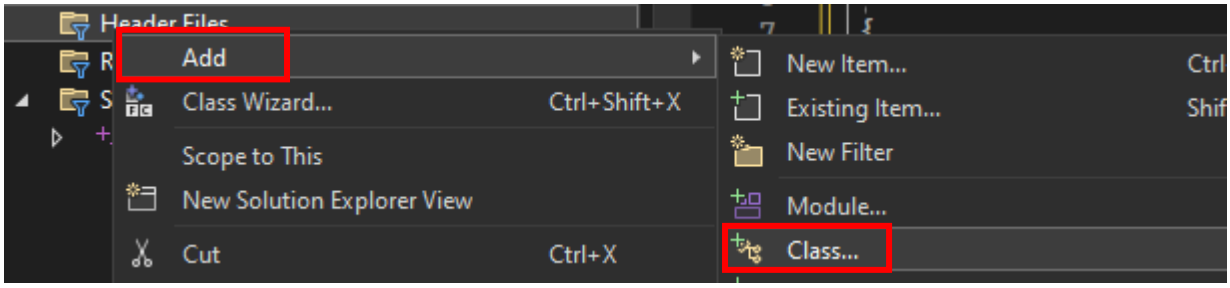
دلوقتي احنا كنا أي كلاس عاوزين نعمله كنا بنعمله في نفس الملف اللي فيه ال main
عاوزين نحط كل كلاس في مكان خاص بيه زي ماكنا بنعمل المكتبات كده
هنعملها ازاي

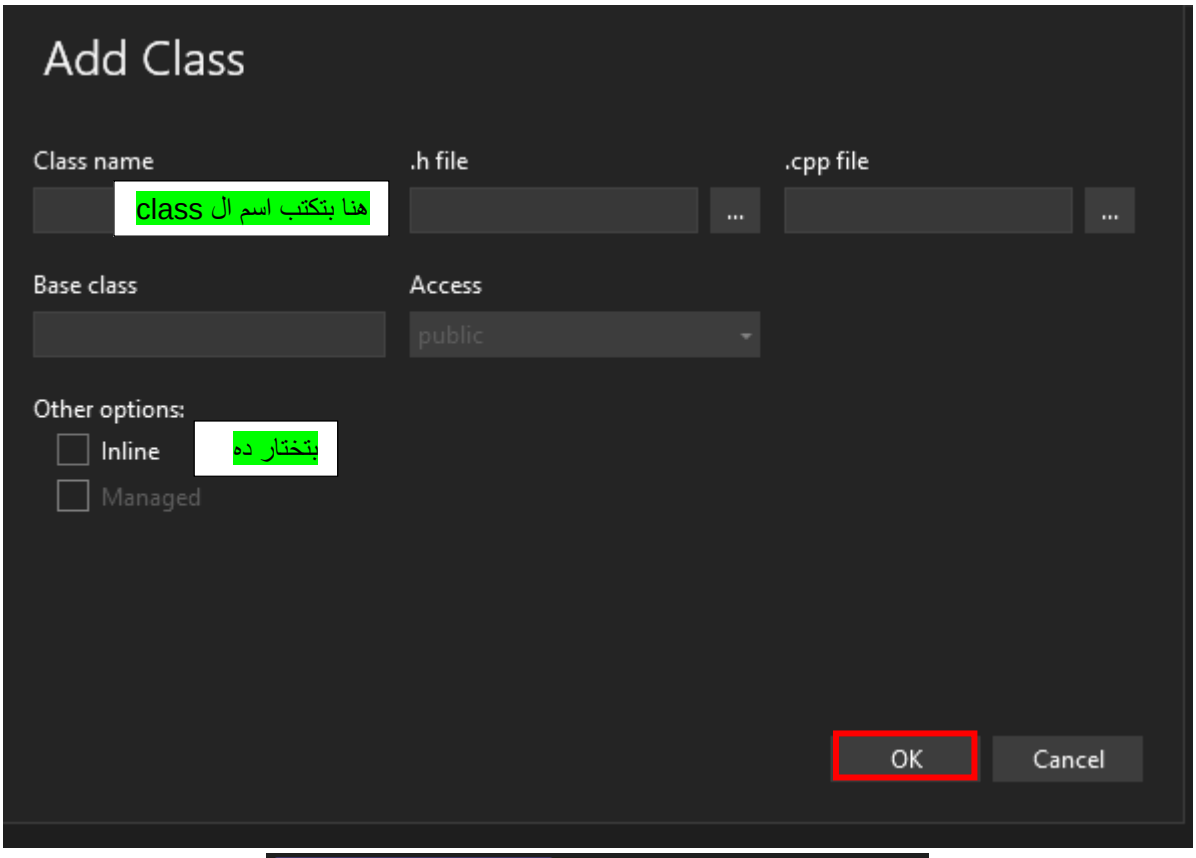
1- لو القايمة اللي في الجنب مش مفتوحه هنفتحها عن طريق
View>>solution explorer

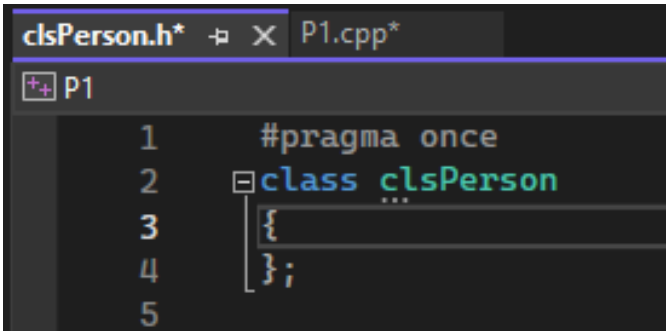


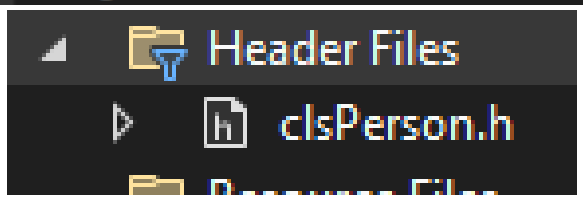
2- بتروح عالقايمة اللي في الجنب وكليك يمين علي header files
Header files>> add>> class



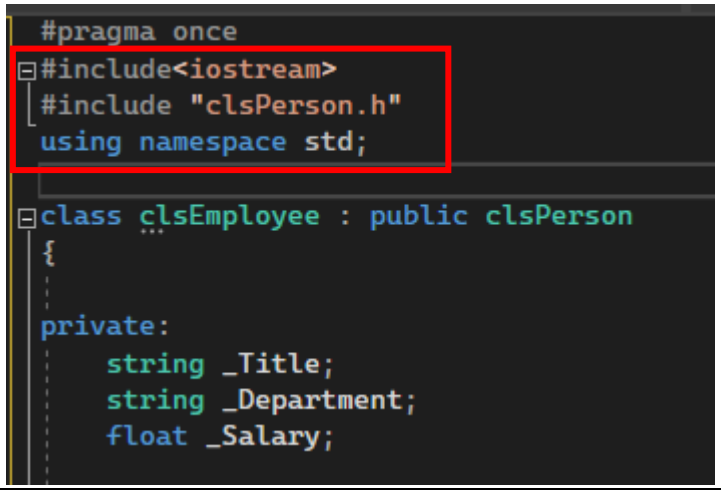








تقدر هنا بقي تكتب الكود بتاعك عادي
عشان تقدر تستخدم الكلاس ده لازم تعمله include في المكان اللي هتستخدمه فيه
يعني انا هنا هعمل كلاس تاني اسمه employee بيورث من الكلاس person فلازم اعمل include للكلاس person
نفس شغل المكتبات بالظبط



```
#pragma once
#include<iostream>
#include "clsPerson.h"
using namespace std;

class clsEmployee : public clsPerson
{

private:
    string _Title;
    string _Department;
    float _Salary;
```

هنا ده الكود بتاع كلاس ال person

```
#pragma once
#include<iostream>
using namespace std;

class clsPerson
{

private:
    int _ID;
    string _FirstName;
    string _LastName;
    string _Email;
    string _Phone;

public:

    clsPerson(int ID, string FirstName, string LastName, string Email,
string Phone)
    {
        _ID = ID;
        _FirstName = FirstName;
        _LastName = LastName;
        _Email = Email;
        _Phone = Phone;
    }

    //Read Only Property
    int ID()
    {
        return _ID;
    }

    //Property Set
    void setFirstName(string FirstName)
    {
        _FirstName = FirstName;
    }

    //Property Get
    string FirstName()
    {
        return _FirstName;
    }

    //Property Set
    void setLastName(string LastName)
    {
        _LastName = LastName;
    }

    //Property Get
    string LastName()
    {
```

```
        return _LastName;
    }

    //Property Set
    void setEmail(string Email)
    {
        _Email = Email;
    }

    //Property Get
    string Email()
    {
        return _Email;
    }

    //Property Set
    void setPhone(string Phone)
    {
        _Phone = Phone;
    }

    //Property Get
    string Phone()
    {
        return _Phone;
    }

    string FullName()
    {
        return _FirstName + " " + _LastName;
    }

    void Print()
    {
        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID       : " << _ID;
        cout << "\nFirstName: " << _FirstName;
        cout << "\nLastName : " << _LastName;
        cout << "\nFull Name: " << FullName();
        cout << "\nEmail    : " << _Email;
        cout << "\nPhone    : " << _Phone;
        cout << "\n____________________\n";

    }

    void SendEmail(string Subject, string Body)
    {

        cout << "\nThe following message sent successfully to email: " <<
_Email;
        cout << "\nSubject: " << Subject;
        cout << "\nBody: " << Body << endl;

    }

    void SendSMS(string TextMessage)
    {
        cout << "\nThe following SMS sent successfully to phone: " <<
_Phone;
        cout << "\n" << TextMessage << endl;
    }


};
```

هنا ده الكود بتاع كلاس

```
#pragma once
#include<iostream>
#include "clsPerson.h"
using namespace std;
```

ال
employee

```
class clsEmployee : public clsPerson
{

private:
    string _Title;
    string _Department;
    float _Salary;

public:

    clsEmployee(int ID, string FirstName, string LastName, string Email,
string Phone, string Title, string Department, float Salary)
        : clsPerson(ID, FirstName, LastName, Email, Phone)

    {

        _Title = Title;
        _Department = Department;
        _Salary = Salary;

    }

    //Property Set
    void setTitle(string Title)
    {
        _Title = Title;
    }

    //Property Get
    string Title()
    {
        return _Title;
    }

    //Property Set
    void setDepartment(string Department)
    {
        _Department = Department;
    }

    //Property Get
    string Department()
    {
        return _Department;
    }

    //Property Set
    void setSalary(float Salary)
    {
        _Salary = Salary;
    }

    //Property Get
    float Salary()
    {
        return _Salary;
    }

    void Print()
    {

        cout << "\nInfo:";
        cout << "\n____________________";
        cout << "\nID        : " << ID();
        cout << "\nFirstName : " << FirstName();
        cout << "\nLastName  : " << LastName();
        cout << "\nFull Name : " << FullName();
        cout << "\nEmail     : " << Email();
```

```
        cout << "\nPhone     : " << Phone();

        cout << "\nTitle     : " << _Title;
        cout << "\nDepartment: " << _Department;
        cout << "\nSalary    : " << _Salary;

        cout << "\n____________________\n";

    }

};
```

```
#include<iostream>
#include "clsEmployee.h"
using namespace std;


int main()

{
    clsEmployee Employee1(10, "Mohammed", "Abu-Hadhoud", "A@a.com",
"8298982", "CEO", "ProgrammingAdvices", 5000);

    Employee1.Print();

    system("pause>0");
    return 0;
}
```

هنا ده ال function بتاع ال main

الواجب

Separating Code and Classes in Libraries will make our life easier and we can control our code and organize it better.

True

False

We must user "#pragma once" in each header file to prevent the complier from loading the library more than one time and have repeated code included.

True

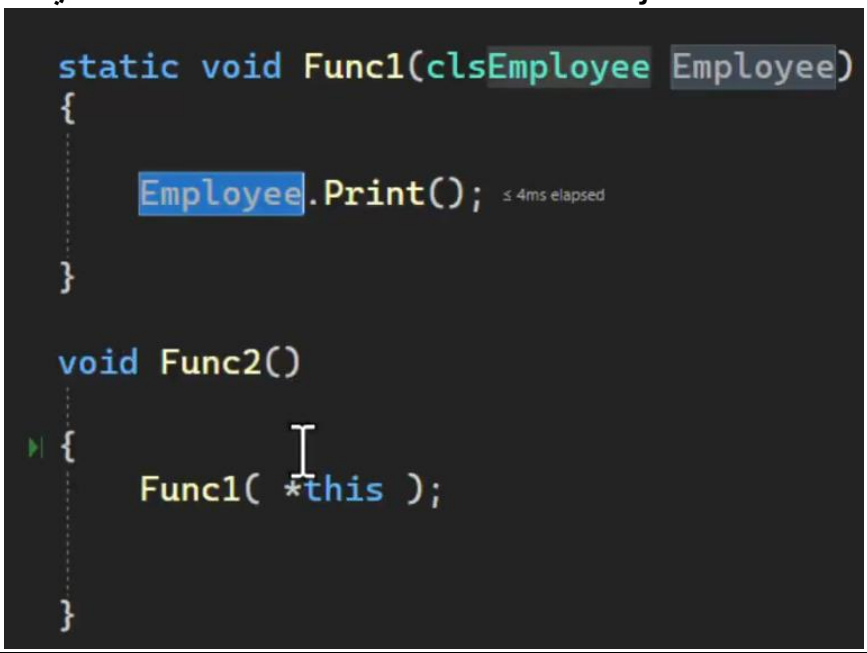
False

What is ‘this’ pointer ?

هنا بيقولك انه فيه حاجه اسمها this pointer وده وظيفته انك لو بتكتب كود جوه كلاس معين وعامل في الكلاس ده متغيرات او functions بتكتب this قبل اسم ال member عشان تعرف ال compiler انك تقصد ال member اللي موجود في الكلاس
لو ماكتبتهوش فال compiler بيعمله مكانك
ماتقدرش تعمل this بره الكلاس وماتقدرش تستخدمه مع ال friend member
فيه لغات تانيه زي ال visual basic بيكون اسمها me

فيه حالات بينفع فيها زي مثلا لو انت عامل parameterized constructor وكاتب ال parameters بنفس اسم المتغيرات
هنا لازم تقول This.a=a عشان تعرفه ان اللي بعد this ده المتغير اللي في بداية الكلاس

اقدر استخدمها عشان ابعت ال object من جوه الكلاس ل function تانيه زي كده




```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
using namespace std;

class clsEmployee

{

public:
    int ID;
    string Name;
    float Salary;

    clsEmployee(int ID, string Name, float Salary)
    {
        this->ID = ID;
        this->Name = Name;
        this->Salary = Salary;

    }

    static void Func1(clsEmployee Employee)
    {
```

```
        Employee.Print();

    }

    void Func2()

    {
        Func1(*this);


    }

    void Print()
    {
        cout << ID << "  " << Name << "  " << Salary << endl;
        // cout << this->ID << "  " << this->Name << "  " << this->Salary << endl;

    }

};

int main(void)

{
    clsEmployee Employee1(101, "Ali", 5000);
    Employee1.Print();

    Employee1.Func2();

    return 0;
}
```

الواجب

Every object in C++ has access to its own address through an important pointer called this pointer.

True

False

The this pointer is an implicit parameter to all member functions.

True

False

Therefore, inside a member function, this may be used to refer to the invoking object.

True

False

Friend functions do not have a this pointer, because friends are not members of a class. Only member functions/member data have a this pointer.

True

False

'this' pointer can be used to pass current object as a parameter to another method.

True

False

'this' pointer can be used to refer current class instance variable.

True

False

Passing Objects to Functions (ByRef/ByVal)

Objects can be treated like any data type such as int, bool, string ...etc,
They can be passed to functions as parameters either by reference or by value.

```
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#include<iostream>
```

```
using namespace std;

class clsA
{

public:
	int x;

	void Print()
	{
		cout << "The value of x=" << x << endl;
	}

};


//object sent by value, any updated will not b reflected
// on the original object
void Fun1(clsA A1)
{

	A1.x = 100;
}


//object sent by reference, any updated will be reflected
// on the original object
void Fun2(clsA& A1)
{

	A1.x = 200;
}


int main()

{
	clsA A1;

	A1.x = 50;
	cout << "\nA.x before calling function1: \n";
	A1.Print();


	//Pass by value, object will not be afected.
	Fun1(A1);
	cout << "\nA.x after calling function1 byval: \n";
	A1.Print();

	//Pass by value, object will be afected.
	Fun2(A1);
	cout << "\nA.x after calling function2 byref: \n";
	A1.Print();


	system("pause>0");

}
```

Objects and Vectors

Adding Objects to Vector

```
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#include<iostream>
#include<vector>

using namespace std;
```

```
class clsA
{
public:
	//Parametarized Constructor
	clsA(int value)
	{
		x = value;
	}

	int x;

	void Print()
	{
		cout << "The value of x=" << x << endl;
	}

};


int main()

{

	vector <clsA> v1;
	short NumberOfObjects = 5;

	// inserting object at the end of vector
	for (int i = 0; i < NumberOfObjects; i++)
	{
		v1.push_back(clsA(i));
	}

	// printing object content
	for (int i = 0; i < NumberOfObjects; i++)
	{
		v1[i].Print();

	}

	system("pause>0");

}
```

Objects and Dynamic Array

Objects and Dynamic Array

```
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#include<iostream>
#include<vector>

using namespace std;

class clsA
{
public:

	// dummy constructor
	clsA() {}

	//Parametarized Constructor
	clsA(int value)
	{
		x = value;
	}
```

```
	int x;

	void Print()
	{
		cout << "The value of x=" << x << endl;
	}

};


int main()

{
	short NumberOfObjects = 5;

	// allocating dynamic array
   // of Size NumberOfObjects using new keyword

	clsA* arrA = new clsA[NumberOfObjects];

	// calling constructor
	// for each index of array
	for (int i = 0; i < NumberOfObjects; i++) {
		arrA[i] = clsA(i);
	}

	// printing contents of array
	for (int i = 0; i < NumberOfObjects; i++) {
		arrA[i].Print();
	}

	return 0;


	system("pause>0");

}
```

Objects with Parameterized Constructor and Array

Objects with Parameterized Constructor and Array

```
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#include<iostream>
#include<vector>

using namespace std;

class clsA
{
public:

	//Parametarized Constructor
	clsA(int value)
	{
		x = value;
	}

	int x;

	void Print()
	{
		cout << "The value of x=" << x << endl;
	}

};
```

```cpp
int main()

{

	// Initializing 3 array Objects with function calls of
	// parameterized constructor as elements of that array
	clsA obj[] = { clsA(10), clsA(20), clsA(30) };

	// using print method for each of three elements.
	for (int i = 0; i < 3; i++) {
		obj[i].Print();
	}

	return 0;


	system("pause>0");

}
```

String Library Project (Requirements)

هنعمل كلاس اسمه clsString ونحط فيه كل ال functions اللي عملناها في ال string

```cpp
#include <iostream>
#include "clsString.h"

using namespace std;


int main()

{
    clsString String1;
    clsString String2 ("Mohammed");

    String1.Value = "Ali";

    cout << "String1 = " << String1.Value << endl;
    cout << "String2 = " << String2.Value << endl;

    system("pause>0");
    return 0;
};
```

```cpp
#pragma once

#include <iostream>
using namespace std;

class clsString
{

private:
    string _Value;

public:

    clsString()
    {

        _Value = "";
    }

    clsString(string Value)
    {

        _Value = Value;
    }
```

```cpp
    void SetValue(string Value) {
        _Value = Value;
    }

    string GetValue() {
        return _Value;
    }

    __declspec(property(get = GetValue, put = SetValue)) string Value
};
```

بعدين اعمل function بتعد الكلمات بتاخد string
وبتعمل واحدة تانيه فاضيه بتستدعي ال function اللي فاتت عشان تعد الكلمات في المتغير value اللي أصلا عملناه عشان نخزن فيه ال string

```cpp
cout << "Number of words: " << String1.CountWords();

cout << clsString::CountWords("Mohammed Abu-Hadhoud");
```

عشان اتغلب عال error اللي في السطر التاني بعمل ال function من النوع static

```cpp
static short CountWords(string S1) { ... }

    short CountWords()
    {
        return CountWords(_Value);
    };
```

ده ال main

```cpp
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsString.h"

using namespace std;


int main()

{
    clsString String1;
    clsString String2("Mohammed");

    String1.Value = "Ali Ahmed";

    cout << "String1 = " << String1.Value << endl;
    cout << "String2 = " << String2.Value << endl;


    cout << "Number of words: " << String1.CountWords() << endl;

    cout << "Number of words: " << String1.CountWords("Fadi ahmed rateb omer")
<< endl;

    cout << "Number of words: " <<
        clsString::CountWords("Mohammed Saqer Abu-Hadhoud");



    system("pause>0");
    return 0;
};
```

ده كلاس ال string

```cpp
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#pragma once

#include <iostream>
using namespace std;

class clsString
{

private:
    string _Value;

public:

    clsString()
    {

        _Value = "";
    }

    clsString(string Value)
    {

        _Value = Value;
    }


    void SetValue(string Value) {
        _Value = Value;
    }
```

```
    string GetValue() {
        return _Value;
    }

    __declspec(property(get = GetValue, put = SetValue)) string Value;

    static short CountWords(string S1)
    {

        string delim = " "; // delimiter
        short Counter = 0;
        short pos = 0;
        string sWord; // define a string variable

        // use find() function to get the position of the delimiters
        while ((pos = S1.find(delim)) != std::string::npos)
        {
            sWord = S1.substr(0, pos); // store the word
            if (sWord != "")
            {
                Counter++;
            }

            //erase() until positon and move to next word.
            S1.erase(0, pos + delim.length());
        }

        if (S1 != "")
        {
            Counter++; // it counts the last word of the string.
        }

        return Counter;

    }

    short CountWords()
    {
        return CountWords(_Value);
    };

};
```

String Library Project (Solution)

main ده ال

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud
#include <iostream>
#include "clsString.h"

using namespace std;

int main()

{
    clsString String1;


    clsString String2("Mohammed");

    String1.Value = "Ali Ahmed";

    cout << "String1 = " << String1.Value << endl;
    cout << "String2 = " << String2.Value << endl;

    cout << "Number of words: " << String1.CountWords() << endl;

    cout << "Number of words: " << String1.CountWords("Fadi ahmed rateb
omer") << endl;
```

```
    cout << "Number of words: " <<
        clsString::CountWords("Mohammed Saqer Abu-Hadhoud") << endl;

    //---------------
    clsString String3("hi how are you?");

    cout << "String 3 = " << String3.Value << endl;

    cout << "String Length = " << String3.Length() << endl;

    String3.UpperFirstLetterOfEachWord();
    cout << String3.Value << endl;

    //---------------

    String3.LowerFirstLetterOfEachWord();
    cout << String3.Value << endl;

    //---------------

    String3.UpperAllString();
    cout << String3.Value << endl;

    //---------------

    String3.LowerAllString();
    cout << String3.Value << endl;

    //---------------

    cout << "After inverting a : "
        << clsString::InvertLetterCase('a') << endl;

    //---------------

    String3.Value = "AbCdEfg";

    String3.InvertAllLettersCase();
    cout << String3.Value << endl;

    String3.InvertAllLettersCase();
    cout << String3.Value << endl;

    //---------------

    cout << "Capital Letters count : "
        << clsString::CountLetters("Mohammed Abu-Hadhoud",
clsString::CapitalLetters)
        << endl << endl;

    //---------------

    String3.Value = "Welcome to Jordan";
    cout << String3.Value << endl;

    cout << "Capital Letters count :" << String3.CountCapitalLetters() <<
endl;

    //---------------

    cout << "Small Letters count :" << String3.CountSmallLetters() << endl;

    //---------------

    cout << "vowels count :" << String3.CountVowels() << endl;

    //---------------
```

```
    cout << "letter E count :" << String3.CountSpecificLetter('E', false)
<< endl;

    //----------------

    cout << "is letter u vowel? " << clsString::IsVowel('a')
        << endl;

    //----------------

    cout << "Words Count" << String3.CountWords()
        << endl;

    //----------------


    vector<string> vString;

    vString = String3.Split(" ");

    cout << "\nTokens = " << vString.size() << endl;

    for (string& s : vString)
    {
        cout << s << endl;
    }

    //----------------

    //Tirms
    String3.Value = "    Mohammed Abu-Hahdoud     ";
    cout << "\nString     = " << String3.Value;

    String3.Value = "    Mohammed Abu-Hahdoud     ";
    String3.TrimLeft();
    cout << "\n\nTrim Left  = " << String3.Value;

    //----------------

    String3.Value = "    Mohammed Abu-Hahdoud     ";
    String3.TrimRight();
    cout << "\nTrim Right = " << String3.Value;

    //----------------

    String3.Value = "    Mohammed Abu-Hahdoud     ";
    String3.Trim();
    cout << "\nTrim       = " << String3.Value;

    //----------------

    //Joins
    vector<string> vString1 = { "Mohammed","Faid","Ali","Maher" };

    cout << "\n\nJoin String From Vector: \n";
    cout << clsString::JoinString(vString1, " ");


    string arrString[] = { "Mohammed","Faid","Ali","Maher" };

    cout << "\n\nJoin String From array: \n";
    cout << clsString::JoinString(arrString, 4, " ");

    //----------------

    String3.Value = "Mohammed Saqer Abu-Hahdoud";
    cout << "\n\nString     = " << String3.Value;

    String3.ReverseWordsInString();
```

```
    cout << "\nReverse Words : " << String3.Value
        << endl;

    //---------------

    String3.Value = "Mohammed Saqer Abu-Hahdoud";
    cout << "\nReplace : " << String3.ReplaceWord("Mohammed", "Sari")
        << endl;

    //---------------

    String3.Value = "This is: a sample text, with punctuations.";
    cout << "\n\nString     = " << String3.Value;

    String3.RemovePunctuations();
    cout << "\nRemove Punctuations : " << String3.Value
        << endl;

    //---------------
    system("pause>0");
    return 0;
};
```

ده الكلاس

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#pragma once

#include <iostream>
#include <vector>

using namespace std;

class clsString
{

private:
    string _Value;

public:

    clsString()
    {
        _Value = "";
    }

    clsString(string Value)
    {
        _Value = Value;
    }

    void SetValue(string Value) {
        _Value = Value;
    }

    string GetValue() {
        return _Value;
    }

    __declspec(property(get = GetValue, put = SetValue)) string Value;


    static short Length(string S1)
    {
        return S1.length();
    };

    short Length()
    {
```

```
        return _Value.length();
    };

    static short CountWords(string S1)
    {

        string delim = " "; // delimiter
        short Counter = 0;
        short pos = 0;
        string sWord; // define a string variable

        // use find() function to get the position of the delimiters
        while ((pos = S1.find(delim)) != std::string::npos)
        {
            sWord = S1.substr(0, pos); // store the word
            if (sWord != "")
            {
                Counter++;
            }

            //erase() until positon and move to next word.
            S1.erase(0, pos + delim.length());
        }

        if (S1 != "")
        {
            Counter++; // it counts the last word of the string.
        }

        return Counter;

    }

    short CountWords()
    {
        return CountWords(_Value);
    };

    static string  UpperFirstLetterOfEachWord(string S1)
    {

        bool isFirstLetter = true;

        for (short i = 0; i < S1.length(); i++)
        {

            if (S1[i] != ' ' && isFirstLetter)
            {
                S1[i] = toupper(S1[i]);

            }

            isFirstLetter = (S1[i] == ' ' ? true : false);

        }

        return S1;
    }

    void  UpperFirstLetterOfEachWord()
    {
        // no need to return value , this function will directly update the
object value
        _Value = UpperFirstLetterOfEachWord(_Value);
    }

    static string  LowerFirstLetterOfEachWord(string S1)
    {
```

```
        bool isFirstLetter = true;

        for (short i = 0; i < S1.length(); i++)
        {

            if (S1[i] != ' ' && isFirstLetter)
            {
                S1[i] = tolower(S1[i]);

            }

            isFirstLetter = (S1[i] == ' ' ? true : false);

        }

        return S1;
    }

    void  LowerFirstLetterOfEachWord()
    {


        // no need to return value , this function will directly update the
object value
        _Value = LowerFirstLetterOfEachWord(_Value);
    }

    static string  UpperAllString(string S1)
    {
        for (short i = 0; i < S1.length(); i++)
        {
            S1[i] = toupper(S1[i]);
        }
        return S1;
    }

    void  UpperAllString()
    {
        _Value = UpperAllString(_Value);
    }

    static string  LowerAllString(string S1)
    {
        for (short i = 0; i < S1.length(); i++)
        {
            S1[i] = tolower(S1[i]);
        }
        return S1;
    }

    void  LowerAllString()
    {
        _Value = LowerAllString(_Value);
    }

    static char  InvertLetterCase(char char1)
    {
        return isupper(char1) ? tolower(char1) : toupper(char1);
    }

    static string  InvertAllLettersCase(string S1)
    {
        for (short i = 0; i < S1.length(); i++)
        {
            S1[i] = InvertLetterCase(S1[i]);
        }
        return S1;
    }
```

```
    void  InvertAllLettersCase()
    {
        _Value = InvertAllLettersCase(_Value);
    }

    enum enWhatToCount { SmallLetters = 0, CapitalLetters = 1, All = 3 };

    static short CountLetters(string S1, enWhatToCount WhatToCount =
enWhatToCount::All)
    {


        if (WhatToCount == enWhatToCount::All)
        {
            return S1.length();
        }

        short Counter = 0;

        for (short i = 0; i < S1.length(); i++)
        {

            if (WhatToCount == enWhatToCount::CapitalLetters &&
isupper(S1[i]))
                Counter++;


            if (WhatToCount == enWhatToCount::SmallLetters &&
islower(S1[i]))
                Counter++;


        }

        return Counter;

    }

    static short  CountCapitalLetters(string S1)
    {

        short Counter = 0;

        for (short i = 0; i < S1.length(); i++)
        {

            if (isupper(S1[i]))
                Counter++;

        }

        return Counter;
    }

    short  CountCapitalLetters()
    {
        return CountCapitalLetters(_Value);
    }

    static short  CountSmallLetters(string S1)
    {

        short Counter = 0;

        for (short i = 0; i < S1.length(); i++)
        {

            if (islower(S1[i]))
                Counter++;
```

```
        }

        return Counter;
    }

    short  CountSmallLetters()
    {
        return CountSmallLetters(_Value);
    }

    static short  CountSpecificLetter(string S1, char Letter, bool
MatchCase = true)
    {

        short Counter = 0;

        for (short i = 0; i < S1.length(); i++)
        {

            if (MatchCase)
            {
                if (S1[i] == Letter)
                    Counter++;
            }
            else
            {
                if (tolower(S1[i]) == tolower(Letter))
                    Counter++;
            }

        }

        return Counter;
    }

    short  CountSpecificLetter(char Letter, bool MatchCase = true)
    {
        return CountSpecificLetter(_Value, Letter, MatchCase);
    }

    static bool IsVowel(char Ch1)
    {
        Ch1 = tolower(Ch1);

        return ((Ch1 == 'a') || (Ch1 == 'e') || (Ch1 == 'i') || (Ch1 ==
'o') || (Ch1 == 'u'));

    }

    static short  CountVowels(string S1)
    {

        short Counter = 0;

        for (short i = 0; i < S1.length(); i++)
        {

            if (IsVowel(S1[i]))
                Counter++;

        }

        return Counter;
    }

    short  CountVowels()
    {
        return CountVowels(_Value);
```

```
    }

    static vector<string> Split(string S1, string Delim)
    {

        vector<string> vString;

        short pos = 0;
        string sWord; // define a string variable

        // use find() function to get the position of the delimiters
        while ((pos = S1.find(Delim)) != std::string::npos)
        {
            sWord = S1.substr(0, pos); // store the word
            if (sWord != "")
            {
                vString.push_back(sWord);
            }

            S1.erase(0, pos + Delim.length());  /* erase() until positon
and move to next word. */
        }

        if (S1 != "")
        {
            vString.push_back(S1); // it adds last word of the string.
        }

        return vString;

    }

    vector<string> Split(string Delim)
    {
        return Split(_Value, Delim);
    }

    static string TrimLeft(string S1)
    {


        for (short i = 0; i < S1.length(); i++)
        {
            if (S1[i] != ' ')
            {
                return S1.substr(i, S1.length() - i);
            }
        }
        return "";
    }

    void TrimLeft()
    {
        _Value = TrimLeft(_Value);
    }

    static string TrimRight(string S1)
    {


        for (short i = S1.length() - 1; i >= 0; i--)
        {
            if (S1[i] != ' ')
            {
                return S1.substr(0, i + 1);
            }
        }
        return "";
    }
```

```
    void TrimRight()
    {
        _Value = TrimRight(_Value);
    }

    static string Trim(string S1)
    {
        return (TrimLeft(TrimRight(S1)));

    }

    void Trim()
    {
        _Value = Trim(_Value);
    }

    static string JoinString(vector<string> vString, string Delim)
    {

        string S1 = "";

        for (string& s : vString)
        {
            S1 = S1 + s + Delim;
        }

        return S1.substr(0, S1.length() - Delim.length());


    }

    static string JoinString(string arrString[], short Length, string
Delim)
    {

        string S1 = "";

        for (short i = 0; i < Length; i++)
        {
            S1 = S1 + arrString[i] + Delim;
        }

        return S1.substr(0, S1.length() - Delim.length());

    }

    static string ReverseWordsInString(string S1)
    {

        vector<string> vString;
        string S2 = "";

        vString = Split(S1, " ");

        // declare iterator
        vector<string>::iterator iter = vString.end();

        while (iter != vString.begin())
        {

            --iter;

            S2 += *iter + " ";

        }

        S2 = S2.substr(0, S2.length() - 1); //remove last space.
```

```
        return S2;
    }

    void ReverseWordsInString()
    {
        _Value = ReverseWordsInString(_Value);
    }

    static string ReplaceWord(string S1, string StringToReplace, string
sRepalceTo, bool MatchCase = true)
    {

        vector<string> vString = Split(S1, " ");

        for (string& s : vString)
        {

            if (MatchCase)
            {
                if (s == StringToReplace)
                {
                    s = sRepalceTo;
                }

            }
            else
            {
                if (LowerAllString(s) == LowerAllString(StringToReplace))
                {
                    s = sRepalceTo;
                }

            }

        }

        return JoinString(vString, " ");
    }

    string ReplaceWord(string StringToReplace, string sRepalceTo)
    {
        return ReplaceWord(_Value, StringToReplace, sRepalceTo);
    }

    static string RemovePunctuations(string S1)
    {

        string S2 = "";

        for (short i = 0; i < S1.length(); i++)
        {
            if (!ispunct(S1[i]))
            {
                S2 += S1[i];
            }
        }

        return S2;

    }

    void RemovePunctuations()
    {
        _Value = RemovePunctuations(_Value);
    }


};
```

Date Library Project (Requirements)

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud
#include <iostream>
#include "clsDate.h"

using namespace std;

int main()

{
    clsDate Date1;
    Date1.Print();

    clsDate Date2("31/1/2022");
    Date2.Print();

    clsDate Date3(20, 12, 2022);
    Date3.Print();

    clsDate Date4(250,2022);
    Date4.Print();
```

```
Date3.PrintMonthCalendar();
Date4.PrintMonthCalendar();
```

```
Date3.PrintYearCalendar();
```

اعمل كلاس حط فيه كل ال functions بتاعت التاريخ وبدل ما كنت بتعمل struct date اعمله class date

Date Library Project (Solution)

ده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud
#include <iostream>
#include "clsDate.h"

using namespace std;

int main()

{
    clsDate Date1;
    Date1.Print();

    clsDate Date2("31/1/2022");
    Date2.Print();

    clsDate Date3(20, 12, 2022);
    Date3.Print();

    clsDate Date4(250, 2022);
    Date4.Print();
```

```
    Date1.IncreaseDateByOneMonth();
    Date1.Print();

    //Date3.PrintYearCalendar();
    //cout << Date2.IsValid() << endl;

    /*cout << "My Age InDays:" <<
        clsDate::CalculateMyAgeInDays(  clsDate(6, 11, 1977) );*/
        //You can try any method at your own..


    system("pause>0");
    return 0;
};
```

ده
الكلاس

```
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#pragma warning(disable : 4996)
#pragma once

#include<iostream>
#include<string>
#include "clsString.h"

using namespace std;

class clsDate
{

private:

        short _Day = 1;
        short _Month = 1;
        short _Year = 1900;

public:

        clsDate()
        {
                time_t t = time(0);
                tm* now = localtime(&t);
                _Day = now->tm_mday;
                _Month = now->tm_mon + 1;
                _Year = now->tm_year + 1900;
        }

        clsDate(string sDate)
        {

                vector <string> vDate;
                vDate = clsString::Split(sDate, "/");

                _Day = stoi(vDate[0]);
                _Month = stoi(vDate[1]);
                _Year = stoi(vDate[2]);

        }

        clsDate(short Day, short Month, short Year)
        {

                _Day = Day;
                _Month = Month;
                _Year = Year;

        }

        clsDate(short DateOrderInYear, short Year)
```

```
{
	//This will construct a date by date order in year
	clsDate Date1 = GetDateFromDayOrderInYear(DateOrderInYear, Year);
	_Day = Date1.Day;
	_Month = Date1.Month;
	_Year = Date1.Year;
}

void SetDay(short Day) {
	_Day = Day;
}

short GetDay() {
	return _Day;
}
__declspec(property(get = GetDay, put = SetDay)) short Day;

void SetMonth(short Month) {
	_Month = Month;
}

short GetMonth() {
	return _Month;
}
__declspec(property(get = GetMonth, put = SetMonth)) short Month;


void SetYear(short Year) {
	_Year = Year;
}

short GetYear() {
	return _Year;
}
__declspec(property(get = GetYear, put = SetYear)) short Year;

void Print()
{
	cout << DateToString() << endl;
}

static clsDate GetSystemDate()
{
	//system date
	time_t t = time(0);
	tm* now = localtime(&t);

	short Day, Month, Year;

	Year = now->tm_year + 1900;
	Month = now->tm_mon + 1;
	Day = now->tm_mday;

	return clsDate(Day, Month, Year);
}

static	bool IsValidDate(clsDate Date)
{

	if (Date.Day < 1 || Date.Day>31)
		return false;

	if (Date.Month < 1 || Date.Month>12)
		return false;

	if (Date.Month == 2)
```

```
                {
                        if (isLeapYear(Date.Year))
                        {
                                if (Date.Day > 29)
                                        return false;
                        }
                        else
                        {
                                if (Date.Day > 28)
                                        return false;
                        }
                }

                short DaysInMonth = NumberOfDaysInAMonth(Date.Month, Date.Year);

                if (Date.Day > DaysInMonth)
                        return false;

                return true;

        }

        bool IsValid()
        {
                return IsValidDate(*this);
        }

        static string DateToString(clsDate Date)
        {
                return  to_string(Date.Day) + "/" + to_string(Date.Month) + "/" +
to_string(Date.Year);
        }

        string DateToString()
        {
                return  DateToString(*this);
        }

        static bool isLeapYear(short Year)
        {

                // if year is divisible by 4 AND not divisible by 100
          // OR if year is divisible by 400
          // then it is a leap year
                return (Year % 4 == 0 && Year % 100 != 0) || (Year % 400 == 0);
        }

        bool isLeapYear()
        {
                return isLeapYear(_Year);
        }

        static short NumberOfDaysInAYear(short Year)
        {
                return  isLeapYear(Year) ? 365 : 364;
        }

        short NumberOfDaysInAYear()
        {
                return  NumberOfDaysInAYear(_Year);
        }

        static short NumberOfHoursInAYear(short Year)
        {
                return  NumberOfDaysInAYear(Year) * 24;
        }
```

```
short NumberOfHoursInAYear()
{
	return  NumberOfHoursInAYear(_Year);
}

static int NumberOfMinutesInAYear(short Year)
{
	return  NumberOfHoursInAYear(Year) * 60;
}

int NumberOfMinutesInAYear()
{
	return  NumberOfMinutesInAYear(_Year);
}

static int NumberOfSecondsInAYear(short Year)
{
	return  NumberOfMinutesInAYear(Year) * 60;
}

int NumberOfSecondsInAYear()
{
	return  NumberOfSecondsInAYear();
}

static short NumberOfDaysInAMonth(short Month, short Year)
{

	if (Month < 1 || Month>12)
		return  0;

	int days[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
	return (Month == 2) ? (isLeapYear(Year) ? 29 : 28) : days[Month - 1];

}

short NumberOfDaysInAMonth()
{
	return NumberOfDaysInAMonth(_Month, _Year);
}

static short NumberOfHoursInAMonth(short Month, short Year)
{
	return  NumberOfDaysInAMonth(Month, Year) * 24;
}

short NumberOfHoursInAMonth()
{
	return  NumberOfDaysInAMonth(_Month, _Year) * 24;
}

static int NumberOfMinutesInAMonth(short Month, short Year)
{
	return  NumberOfHoursInAMonth(Month, Year) * 60;
}

int NumberOfMinutesInAMonth()
{
	return  NumberOfHoursInAMonth(_Month, _Year) * 60;
}

static int NumberOfSecondsInAMonth(short Month, short Year)
{
	return  NumberOfMinutesInAMonth(Month, Year) * 60;
}
```

```
int NumberOfSecondsInAMonth()
{
	return  NumberOfMinutesInAMonth(_Month, _Year) * 60;
}

static short DayOfWeekOrder(short Day, short Month, short Year)
{
	short a, y, m;
	a = (14 - Month) / 12;
	y = Year - a;
	m = Month + (12 * a) - 2;
	// Gregorian:
	//0:sun, 1:Mon, 2:Tue...etc
	return (Day + y + (y / 4) - (y / 100) + (y / 400) + ((31 * m) / 12)) % 7;
}

short DayOfWeekOrder()
{
	return DayOfWeekOrder(_Day, _Month, _Year);
}

static string DayShortName(short DayOfWeekOrder)
{
	string arrDayNames[] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder];

}

static string DayShortName(short Day, short Month, short Year)
{

	string arrDayNames[] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder(Day, Month, Year)];

}

string DayShortName()
{

	string arrDayNames[] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder(_Day, _Month, _Year)];

}

static string MonthShortName(short MonthNumber)
{
	string Months[12] = { "Jan", "Feb", "Mar",
	                      "Apr", "May", "Jun",
	                      "Jul", "Aug", "Sep",
	                      "Oct", "Nov", "Dec"
	};

	return (Months[MonthNumber - 1]);
}

string MonthShortName()
{

	return MonthShortName(_Month);
}

static void PrintMonthCalendar(short Month, short Year)
```

```
{
        int NumberOfDays;

        // Index of the day from 0 to 6
        int current = DayOfWeekOrder(1, Month, Year);

        NumberOfDays = NumberOfDaysInAMonth(Month, Year);

        // Print the current month name
        printf("\n  _________________%s_________________\n\n",
                MonthShortName(Month).c_str());

        // Print the columns
        printf("  Sun  Mon  Tue  Wed  Thu  Fri  Sat\n");

        // Print appropriate spaces
        int i;
        for (i = 0; i < current; i++)
                printf("     ");

        for (int j = 1; j <= NumberOfDays; j++)
        {
                printf("%5d", j);


                if (++i == 7)
                {
                        i = 0;
                        printf("\n");
                }
        }

        printf("\n  ________________________________________\n");

}

void PrintMonthCalendar()
{
        PrintMonthCalendar(_Month, _Year);
}

static void PrintYearCalendar(int Year)
{
        printf("\n  ________________________________________\n\n");
        printf("          Calendar - %d\n\n", Year);
        printf("  ________________________________________\n");


        for (int i = 1; i <= 12; i++)
        {
                PrintMonthCalendar(i, Year);
        }

        return;
}

void PrintYearCalendar()
{
        printf("\n  ________________________________________\n\n");
        printf("          Calendar - %d\n\n", _Year);
        printf("  ________________________________________\n");


        for (int i = 1; i <= 12; i++)
        {
                PrintMonthCalendar(i, _Year);
```

```
        }

        return;
    }

    static short DaysFromTheBeginingOfTheYear(short Day, short Month, short Year)
    {


        short TotalDays = 0;

        for (int i = 1; i <= Month - 1; i++)
        {
            TotalDays += NumberOfDaysInAMonth(i, Year);
        }

        TotalDays += Day;

        return TotalDays;
    }

    short DaysFromTheBeginingOfTheYear()
    {


        short TotalDays = 0;

        for (int i = 1; i <= _Month - 1; i++)
        {
            TotalDays += NumberOfDaysInAMonth(i, _Year);
        }

        TotalDays += _Day;

        return TotalDays;
    }

    static clsDate GetDateFromDayOrderInYear(short DateOrderInYear, short Year)
    {

        clsDate Date;
        short RemainingDays = DateOrderInYear;
        short MonthDays = 0;

        Date.Year = Year;
        Date.Month = 1;

        while (true)
        {
            MonthDays = NumberOfDaysInAMonth(Date.Month, Year);

            if (RemainingDays > MonthDays)
            {
                RemainingDays -= MonthDays;
                Date.Month++;
            }
            else
            {
                Date.Day = RemainingDays;
                break;
            }

        }

        return Date;
    }
```

```
        void AddDays(short Days)
        {


                short RemainingDays = Days + DaysFromTheBeginingOfTheYear(_Day, _Month,
_Year);
                short MonthDays = 0;

                _Month = 1;

                while (true)
                {
                        MonthDays = NumberOfDaysInAMonth(_Month, _Year);

                        if (RemainingDays > MonthDays)
                        {
                                RemainingDays -= MonthDays;
                                _Month++;

                                if (_Month > 12)
                                {
                                        _Month = 1;
                                        _Year++;

                                }
                        }
                        else
                        {
                                _Day = RemainingDays;
                                break;
                        }

                }


        }

        static bool IsDate1BeforeDate2(clsDate Date1, clsDate Date2)
        {
                return  (Date1.Year < Date2.Year) ? true : ((Date1.Year == Date2.Year) ?
(Date1.Month < Date2.Month ? true : (Date1.Month == Date2.Month ? Date1.Day < Date2.Day :
false)) : false);
        }

        bool IsDateBeforeDate2(clsDate Date2)
        {
                //note: *this sends the current object :-)
                return  IsDate1BeforeDate2(*this, Date2);

        }

        static bool IsDate1EqualDate2(clsDate Date1, clsDate Date2)
        {
                return  (Date1.Year == Date2.Year) ? ((Date1.Month == Date2.Month) ?
((Date1.Day == Date2.Day) ? true : false) : false) : false;
        }

        bool IsDateEqualDate2(clsDate Date2)
        {
                return  IsDate1EqualDate2(*this, Date2);
        }

        static bool IsLastDayInMonth(clsDate Date)
        {
```

```
        return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));

}

bool IsLastDayInMonth()
{

        return IsLastDayInMonth(*this);

}

static bool IsLastMonthInYear(short Month)
{
        return (Month == 12);
}

static clsDate AddOneDay(clsDate Date)
{
        if (IsLastDayInMonth(Date))
        {
                if (IsLastMonthInYear(Date.Month))
                {
                        Date.Month = 1;
                        Date.Day = 1;
                        Date.Year++;
                }
                else
                {
                        Date.Day = 1;
                        Date.Month++;
                }
        }
        else
        {
                Date.Day++;
        }

        return Date;
}

void AddOneDay()

{
        *this = AddOneDay(*this);
}

static void  SwapDates(clsDate& Date1, clsDate& Date2)
{

        clsDate TempDate;
        TempDate = Date1;
        Date1 = Date2;
        Date2 = TempDate;

}

static int GetDifferenceInDays(clsDate Date1, clsDate Date2, bool IncludeEndDay = false)
{
        //this will take care of negative diff
        int Days = 0;
        short SawpFlagValue = 1;

        if (!IsDate1BeforeDate2(Date1, Date2))
        {
                //Swap Dates
                SwapDates(Date1, Date2);
```

```
                SawpFlagValue = -1;

            }

            while (IsDate1BeforeDate2(Date1, Date2))
            {
                Days++;
                Date1 = AddOneDay(Date1);
            }

            return IncludeEndDay ? ++Days * SawpFlagValue : Days * SawpFlagValue;
        }

        int GetDifferenceInDays(clsDate Date2, bool IncludeEndDay = false)
        {
            return GetDifferenceInDays(*this, Date2, IncludeEndDay);
        }

        static short CalculateMyAgeInDays(clsDate DateOfBirth)
        {
            return GetDifferenceInDays(DateOfBirth, clsDate::GetSystemDate(), true);
        }
        //above no need to have nonstatic function for the object because it does not depend on
any data from it.

        static clsDate IncreaseDateByOneWeek(clsDate& Date)
        {

            for (int i = 1; i <= 7; i++)
            {
                Date = AddOneDay(Date);
            }

            return Date;
        }

        void IncreaseDateByOneWeek()
        {
            IncreaseDateByOneWeek(*this);
        }

        clsDate IncreaseDateByXWeeks(short Weeks, clsDate& Date)
        {

            for (short i = 1; i <= Weeks; i++)
            {
                Date = IncreaseDateByOneWeek(Date);
            }
            return Date;
        }

        void IncreaseDateByXWeeks(short Weeks)
        {
            IncreaseDateByXWeeks(Weeks, *this);
        }

        clsDate IncreaseDateByOneMonth(clsDate& Date)
        {

            if (Date.Month == 12)
            {
                Date.Month = 1;
                Date.Year++;
            }
            else
            {
```

```
                Date.Month++;
        }

        //last check day in date should not exceed max days in the current month
        // example if date is 31/1/2022 increasing one month should not be 31/2/2022, it
should
        // be 28/2/2022
        short NumberOfDaysInCurrentMonth = NumberOfDaysInAMonth(Date.Month,
Date.Year);
        if (Date.Day > NumberOfDaysInCurrentMonth)
        {
                Date.Day = NumberOfDaysInCurrentMonth;
        }

        return Date;
    }

    void IncreaseDateByOneMonth()
    {

        IncreaseDateByOneMonth(*this);

    }

    clsDate IncreaseDateByXDays(short Days, clsDate& Date)
    {

        for (short i = 1; i <= Days; i++)
        {
                Date = AddOneDay(Date);
        }
        return Date;
    }

    void IncreaseDateByXDays(short Days)
    {

        IncreaseDateByXDays(Days, *this);
    }

    clsDate IncreaseDateByXMonths(short Months, clsDate& Date)
    {

        for (short i = 1; i <= Months; i++)
        {
                Date = IncreaseDateByOneMonth(Date);
        }
        return Date;
    }

    void IncreaseDateByXMonths(short Months)
    {
        IncreaseDateByXMonths(Months, *this);
    }

    static clsDate IncreaseDateByOneYear(clsDate& Date)
    {
        Date.Year++;
        return Date;
    }

    void IncreaseDateByOneYear()
    {
        IncreaseDateByOneYear(*this);
    }
```

```
clsDate IncreaseDateByXYears(short Years, clsDate& Date)
{
	Date.Year += Years;
	return Date;

}

void IncreaseDateByXYears(short Years)
{
	IncreaseDateByXYears(Years);
}

clsDate IncreaseDateByOneDecade(clsDate& Date)
{
	//Period of 10 years
	Date.Year += 10;
	return Date;
}

void IncreaseDateByOneDecade()
{
	IncreaseDateByOneDecade(*this);
}

clsDate IncreaseDateByXDecades(short Decade, clsDate& Date)
{
	Date.Year += Decade * 10;
	return Date;
}

void IncreaseDateByXDecades(short Decade)
{
	IncreaseDateByXDecades(Decade, *this);
}

clsDate IncreaseDateByOneCentury(clsDate& Date)
{
	//Period of 100 years
	Date.Year += 100;
	return Date;
}

void IncreaseDateByOneCentury()
{
	IncreaseDateByOneCentury(*this);
}

clsDate IncreaseDateByOneMillennium(clsDate& Date)
{
	//Period of 1000 years
	Date.Year += 1000;
	return Date;
}

clsDate IncreaseDateByOneMillennium()
{
	IncreaseDateByOneMillennium(*this);
}

static clsDate DecreaseDateByOneDay(clsDate Date)
{
	if (Date.Day == 1)
	{
		if (Date.Month == 1)
		{
			Date.Month = 12;
```

```
                        Date.Day = 31;
                        Date.Year--;
                }
                else
                {

                        Date.Month--;
                        Date.Day = NumberOfDaysInAMonth(Date.Month, Date.Year);
                }
        }
        else
        {
                Date.Day--;
        }

        return Date;
}

void DecreaseDateByOneDay()
{
        DecreaseDateByOneDay(*this);
}

static clsDate DecreaseDateByOneWeek(clsDate& Date)
{

        for (int i = 1; i <= 7; i++)
        {
                Date = DecreaseDateByOneDay(Date);
        }

        return Date;
}

void DecreaseDateByOneWeek()
{
        DecreaseDateByOneWeek(*this);
}

static clsDate DecreaseDateByXWeeks(short Weeks, clsDate& Date)
{

        for (short i = 1; i <= Weeks; i++)
        {
                Date = DecreaseDateByOneWeek(Date);
        }
        return Date;
}

void DecreaseDateByXWeeks(short Weeks)
{
        DecreaseDateByXWeeks(Weeks, *this);
}

static clsDate DecreaseDateByOneMonth(clsDate& Date)
{

        if (Date.Month == 1)
        {
                Date.Month = 12;
                Date.Year--;
        }
        else
                Date.Month--;
```

```
		//last check day in date should not exceed max days in the current month
	// example if date is 31/3/2022 decreasing one month should not be 31/2/2022, it should
	// be 28/2/2022
		short NumberOfDaysInCurrentMonth = NumberOfDaysInAMonth(Date.Month,
Date.Year);
		if (Date.Day > NumberOfDaysInCurrentMonth)
		{
			Date.Day = NumberOfDaysInCurrentMonth;
		}


		return Date;
	}

	void DecreaseDateByOneMonth()
	{
		DecreaseDateByOneMonth(*this);
	}

	static clsDate DecreaseDateByXDays(short Days, clsDate& Date)
	{

		for (short i = 1; i <= Days; i++)
		{
			Date = DecreaseDateByOneDay(Date);
		}
		return Date;
	}

	void DecreaseDateByXDays(short Days)
	{
		DecreaseDateByXDays(Days, *this);
	}

	static clsDate DecreaseDateByXMonths(short Months, clsDate& Date)
	{

		for (short i = 1; i <= Months; i++)
		{
			Date = DecreaseDateByOneMonth(Date);
		}
		return Date;
	}

	void DecreaseDateByXMonths(short Months)
	{
		DecreaseDateByXMonths(Months, *this);
	}

	static clsDate DecreaseDateByOneYear(clsDate& Date)
	{

		Date.Year--;
		return Date;
	}

	void DecreaseDateByOneYear()
	{
		DecreaseDateByOneYear(*this);
	}

	static clsDate DecreaseDateByXYears(short Years, clsDate& Date)
	{

		Date.Year -= Years;
		return Date;
```

```
}

void DecreaseDateByXYears(short Years)
{
	DecreaseDateByXYears(Years, *this);
}

static clsDate DecreaseDateByOneDecade(clsDate& Date)
{
	//Period of 10 years
	Date.Year -= 10;
	return Date;
}

void DecreaseDateByOneDecade()
{
	DecreaseDateByOneDecade(*this);
}

static clsDate DecreaseDateByXDecades(short Decades, clsDate& Date)
{

	Date.Year -= Decades * 10;
	return Date;
}

void DecreaseDateByXDecades(short Decades)
{
	DecreaseDateByXDecades(Decades, *this);
}

static clsDate DecreaseDateByOneCentury(clsDate& Date)
{
	//Period of 100 years
	Date.Year -= 100;
	return Date;
}

void DecreaseDateByOneCentury()
{
	DecreaseDateByOneCentury(*this);
}

static clsDate DecreaseDateByOneMillennium(clsDate& Date)
{
	//Period of 1000 years
	Date.Year -= 1000;
	return Date;
}

void DecreaseDateByOneMillennium()
{
	DecreaseDateByOneMillennium(*this);
}


static short IsEndOfWeek(clsDate Date)
{
	return  DayOfWeekOrder(Date.Day, Date.Month, Date.Year) == 6;
}

short IsEndOfWeek()
{
	return IsEndOfWeek(*this);
}
```

```
static bool IsWeekEnd(clsDate Date)
{
        //Weekends are Fri and Sat
        short DayIndex = DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
        return  (DayIndex == 5 || DayIndex == 6);
}

bool IsWeekEnd()
{
        return  IsWeekEnd(*this);
}

static bool IsBusinessDay(clsDate Date)
{
        //Weekends are Sun,Mon,Tue,Wed and Thur

  /*
        short DayIndex = DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
        return  (DayIndex >= 5 && DayIndex <= 4);
  */

  //shorter method is to invert the IsWeekEnd: this will save updating code.
        return !IsWeekEnd(Date);

}

bool IsBusinessDay()
{
        return  IsBusinessDay(*this);
}

static short DaysUntilTheEndOfWeek(clsDate Date)
{
        return 6 - DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
}

short DaysUntilTheEndOfWeek()
{
        return  DaysUntilTheEndOfWeek(*this);
}

static short DaysUntilTheEndOfMonth(clsDate Date1)
{

        clsDate EndOfMontDate;
        EndOfMontDate.Day = NumberOfDaysInAMonth(Date1.Month, Date1.Year);
        EndOfMontDate.Month = Date1.Month;
        EndOfMontDate.Year = Date1.Year;

        return GetDifferenceInDays(Date1, EndOfMontDate, true);

}

short DaysUntilTheEndOfMonth()
{
        return DaysUntilTheEndOfMonth(*this);
}

static short DaysUntilTheEndOfYear(clsDate Date1)
{

        clsDate EndOfYearDate;
        EndOfYearDate.Day = 31;
        EndOfYearDate.Month = 12;
        EndOfYearDate.Year = Date1.Year;
```

```
		return GetDifferenceInDays(Date1, EndOfYearDate, true);

	}

	short DaysUntilTheEndOfYear()
	{
		return  DaysUntilTheEndOfYear(*this);
	}

	//i added this method to calculate business days between 2 days
	static short CalculateBusinessDays(clsDate DateFrom, clsDate DateTo)
	{

		short Days = 0;
		while (IsDate1BeforeDate2(DateFrom, DateTo))
		{
			if (IsBusinessDay(DateFrom))
				Days++;

			DateFrom = AddOneDay(DateFrom);
		}

		return Days;

	}

	static short CalculateVacationDays(clsDate DateFrom, clsDate DateTo)
	{
		/*short Days = 0;
		while (IsDate1BeforeDate2(DateFrom, DateTo))
		{
			if (IsBusinessDay(DateFrom))
				Days++;

			DateFrom = AddOneDay(DateFrom);
		}*/

		return CalculateBusinessDays(DateFrom, DateTo);

	}
	//above method is eough , no need to have method for the object

	static clsDate CalculateVacationReturnDate(clsDate DateFrom, short VacationDays)
	{

		short WeekEndCounter = 0;

		for (short i = 1; i <= VacationDays; i++)
		{

			if (IsWeekEnd(DateFrom))
				WeekEndCounter++;

			DateFrom = AddOneDay(DateFrom);
		}
		//to add weekends
		for (short i = 1; i <= WeekEndCounter; i++)
			DateFrom = AddOneDay(DateFrom);

		return DateFrom;
	}

	static bool IsDate1AfterDate2(clsDate Date1, clsDate Date2)
	{
		return (!IsDate1BeforeDate2(Date1, Date2) && !IsDate1EqualDate2(Date1,
Date2));
```

```
        }

        bool IsDateAfterDate2(clsDate Date2)
        {
                return IsDate1AfterDate2(*this, Date2);
        }

        enum enDateCompare { Before = -1, Equal = 0, After = 1 };

        static enDateCompare CompareDates(clsDate Date1, clsDate Date2)
        {
                if (IsDate1BeforeDate2(Date1, Date2))
                        return enDateCompare::Before;

                if (IsDate1EqualDate2(Date1, Date2))
                        return enDateCompare::Equal;

                /* if (IsDate1AfterDate2(Date1,Date2))
                         return enDateCompare::After;*/

                         //this is faster
                return enDateCompare::After;

        }

        enDateCompare CompareDates(clsDate Date2)
        {
                return CompareDates(*this, Date2);
        }



};
```

Period Class

The following is source code for Period class and they way to use it

ده ال main

```
#include <iostream>
#include "clsPeriod.h"

int main()

{

    clsPeriod Period1(clsDate(1, 1, 2022), clsDate(10, 1, 2022));
    Period1.Print();

    cout << "\n";

    clsPeriod Period2(clsDate(3, 1, 2022), clsDate(5, 1, 2022));
    Period2.Print();


    //You can check like this
    cout << Period1.IsOverLapWith(Period2) << endl;


    //Also you can call the static method and send period 1 and period 2
    cout << clsPeriod::IsOverlapPeriods(Period1, Period2) << endl;


    return 0;
}
```

ده ال
period

```
#pragma once
#include "clsDate.h"

class clsPeriod
{
public:

    clsDate StartDate;
    clsDate EndDate;

    clsPeriod(clsDate StartDate, clsDate DateTo)
    {
        this->StartDate = StartDate;
        this->EndDate = EndDate;

    }

    static bool IsOverlapPeriods(clsPeriod Period1, clsPeriod Period2)
    {

        if (
            clsDate::CompareDates(Period2.EndDate, Period1.StartDate) ==
clsDate::enDateCompare::Before
            ||
            clsDate::CompareDates(Period2.StartDate, Period1.EndDate) ==
clsDate::enDateCompare::After
            )
            return false;
        else
            return true;

    }


    bool IsOverLapWith(clsPeriod Period2)
    {
        return IsOverlapPeriods(*this, Period2);
    }

    void Print()
    {
        cout << "Period Start: ";
        StartDate.Print();


        cout << "Period End: ";
        EndDate.Print();


    }

};
```

ده ال
date

```
//ProgrammingAdivces.com
//Mohammed Abu-Hadhoud
#pragma warning(disable : 4996)
#pragma once

#include<iostream>
#include<string>
#include "clsString.h"

using namespace std;

class clsDate
{

private:
```

```
	short _Day = 1;
	short _Month = 1;
	short _Year = 1900;

public:

	clsDate()
	{
		time_t t = time(0);
		tm* now = localtime(&t);
		_Day = now->tm_mday;
		_Month = now->tm_mon + 1;
		_Year = now->tm_year + 1900;
	}

	clsDate(string sDate)
	{

		vector <string> vDate;
		vDate = clsString::Split(sDate, "/");

		_Day = stoi(vDate[0]);
		_Month = stoi(vDate[1]);
		_Year = stoi(vDate[2]);

	}

	clsDate(short Day, short Month, short Year)
	{

		_Day = Day;
		_Month = Month;
		_Year = Year;

	}

	clsDate(short DateOrderInYear, short Year)
	{
		//This will construct a date by date order in year
		clsDate Date1 = GetDateFromDayOrderInYear(DateOrderInYear, Year);
		_Day = Date1.Day;
		_Month = Date1.Month;
		_Year = Date1.Year;
	}

	void SetDay(short Day) {
		_Day = Day;
	}

	short GetDay() {
		return _Day;
	}
	__declspec(property(get = GetDay, put = SetDay)) short Day;

	void SetMonth(short Month) {
		_Month = Month;
	}

	short GetMonth() {
		return _Month;
	}
	__declspec(property(get = GetMonth, put = SetMonth)) short Month;


	void SetYear(short Year) {
```

```
		_Year = Year;
	}

	short GetYear() {
		return _Year;
	}
	__declspec(property(get = GetYear, put = SetYear)) short Year;

	void Print()
	{
		cout << DateToString() << endl;
	}

	static clsDate GetSystemDate()
	{
		//system date
		time_t t = time(0);
		tm* now = localtime(&t);

		short Day, Month, Year;

		Year = now->tm_year + 1900;
		Month = now->tm_mon + 1;
		Day = now->tm_mday;

		return clsDate(Day, Month, Year);
	}

	static	bool IsValidDate(clsDate Date)
	{

		if (Date.Day < 1 || Date.Day>31)
			return false;

		if (Date.Month < 1 || Date.Month>12)
			return false;

		if (Date.Month == 2)
		{
			if (isLeapYear(Date.Year))
			{
				if (Date.Day > 29)
					return false;
			}
			else
			{
				if (Date.Day > 28)
					return false;
			}
		}

		short DaysInMonth = NumberOfDaysInAMonth(Date.Month, Date.Year);

		if (Date.Day > DaysInMonth)
			return false;

		return true;

	}

	bool IsValid()
	{
		return IsValidDate(*this);
	}

	static string DateToString(clsDate Date)
```

```
	{
		return  to_string(Date.Day) + "/" + to_string(Date.Month) + "/" +
to_string(Date.Year);
	}

	string DateToString()
	{
		return  DateToString(*this);
	}

	static bool isLeapYear(short Year)
	{

		// if year is divisible by 4 AND not divisible by 100
	  // OR if year is divisible by 400
	  // then it is a leap year
		return (Year % 4 == 0 && Year % 100 != 0) || (Year % 400 == 0);
	}

	bool isLeapYear()
	{
		return isLeapYear(_Year);
	}

	static short NumberOfDaysInAYear(short Year)
	{
		return  isLeapYear(Year) ? 365 : 364;
	}

	short NumberOfDaysInAYear()
	{
		return  NumberOfDaysInAYear(_Year);
	}

	static short NumberOfHoursInAYear(short Year)
	{
		return  NumberOfDaysInAYear(Year) * 24;
	}

	short NumberOfHoursInAYear()
	{
		return  NumberOfHoursInAYear(_Year);
	}

	static int NumberOfMinutesInAYear(short Year)
	{
		return  NumberOfHoursInAYear(Year) * 60;
	}

	int NumberOfMinutesInAYear()
	{
		return  NumberOfMinutesInAYear(_Year);
	}

	static int NumberOfSecondsInAYear(short Year)
	{
		return  NumberOfMinutesInAYear(Year) * 60;
	}

	int NumberOfSecondsInAYear()
	{
		return  NumberOfSecondsInAYear();
	}

	static short NumberOfDaysInAMonth(short Month, short Year)
	{
```

```
        if (Month < 1 || Month>12)
                return  0;

        int days[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
        return (Month == 2) ? (isLeapYear(Year) ? 29 : 28) : days[Month - 1];

}

short NumberOfDaysInAMonth()
{
        return NumberOfDaysInAMonth(_Month, _Year);
}

static short NumberOfHoursInAMonth(short Month, short Year)
{
        return  NumberOfDaysInAMonth(Month, Year) * 24;
}

short NumberOfHoursInAMonth()
{
        return  NumberOfDaysInAMonth(_Month, _Year) * 24;
}

static int NumberOfMinutesInAMonth(short Month, short Year)
{
        return  NumberOfHoursInAMonth(Month, Year) * 60;
}

int NumberOfMinutesInAMonth()
{
        return  NumberOfHoursInAMonth(_Month, _Year) * 60;
}

static int NumberOfSecondsInAMonth(short Month, short Year)
{
        return  NumberOfMinutesInAMonth(Month, Year) * 60;
}

int NumberOfSecondsInAMonth()
{
        return  NumberOfMinutesInAMonth(_Month, _Year) * 60;
}

static short DayOfWeekOrder(short Day, short Month, short Year)
{
        short a, y, m;
        a = (14 - Month) / 12;
        y = Year - a;
        m = Month + (12 * a) - 2;
        // Gregorian:
        //0:sun, 1:Mon, 2:Tue...etc
        return (Day + y + (y / 4) - (y / 100) + (y / 400) + ((31 * m) / 12)) % 7;
}

short DayOfWeekOrder()
{
        return DayOfWeekOrder(_Day, _Month, _Year);
}

static string DayShortName(short DayOfWeekOrder)
{
        string arrDayNames[] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

        return arrDayNames[DayOfWeekOrder];
```

```
}

static string DayShortName(short Day, short Month, short Year)
{

        string arrDayNames[] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

        return arrDayNames[DayOfWeekOrder(Day, Month, Year)];

}

string DayShortName()
{

        string arrDayNames[] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

        return arrDayNames[DayOfWeekOrder(_Day, _Month, _Year)];

}

static string MonthShortName(short MonthNumber)
{
        string Months[12] = { "Jan", "Feb", "Mar",
                                        "Apr", "May", "Jun",
                                        "Jul", "Aug", "Sep",
                                        "Oct", "Nov", "Dec"
        };

        return (Months[MonthNumber - 1]);
}

string MonthShortName()
{

        return MonthShortName(_Month);
}

static void PrintMonthCalendar(short Month, short Year)
{
        int NumberOfDays;

        // Index of the day from 0 to 6
        int current = DayOfWeekOrder(1, Month, Year);

        NumberOfDays = NumberOfDaysInAMonth(Month, Year);

        // Print the current month name
        printf("\n  _________________%s_________________\n\n",
                MonthShortName(Month).c_str());

        // Print the columns
        printf("  Sun  Mon  Tue  Wed  Thu  Fri  Sat\n");

        // Print appropriate spaces
        int i;
        for (i = 0; i < current; i++)
                printf("     ");

        for (int j = 1; j <= NumberOfDays; j++)
        {
                printf("%5d", j);


                if (++i == 7)
                {
                        i = 0;
```

```
                        printf("\n");
                }
        }

        printf("\n  _____\n");

}

void PrintMonthCalendar()
{
        PrintMonthCalendar(_Month, _Year);
}

static void PrintYearCalendar(int Year)
{
        printf("\n  _____\n\n");
        printf("          Calendar - %d\n", Year);
        printf("  _____\n");


        for (int i = 1; i <= 12; i++)
        {
                PrintMonthCalendar(i, Year);
        }

        return;
}

void PrintYearCalendar()
{
        printf("\n  _____\n\n");
        printf("          Calendar - %d\n", _Year);
        printf("  _____\n");


        for (int i = 1; i <= 12; i++)
        {
                PrintMonthCalendar(i, _Year);
        }

        return;
}

static short DaysFromTheBeginingOfTheYear(short Day, short Month, short Year)
{


        short TotalDays = 0;

        for (int i = 1; i <= Month - 1; i++)
        {
                TotalDays += NumberOfDaysInAMonth(i, Year);
        }

        TotalDays += Day;

        return TotalDays;
}

short DaysFromTheBeginingOfTheYear()
{


        short TotalDays = 0;

        for (int i = 1; i <= _Month - 1; i++)
```

```
                {
                        TotalDays += NumberOfDaysInAMonth(i, _Year);
                }

                TotalDays += _Day;

                return TotalDays;
        }

        static clsDate GetDateFromDayOrderInYear(short DateOrderInYear, short Year)
        {

                clsDate Date;
                short RemainingDays = DateOrderInYear;
                short MonthDays = 0;

                Date.Year = Year;
                Date.Month = 1;

                while (true)
                {
                        MonthDays = NumberOfDaysInAMonth(Date.Month, Year);

                        if (RemainingDays > MonthDays)
                        {
                                RemainingDays -= MonthDays;
                                Date.Month++;
                        }
                        else
                        {
                                Date.Day = RemainingDays;
                                break;
                        }

                }

                return Date;
        }

        void AddDays(short Days)
        {


                short RemainingDays = Days + DaysFromTheBeginingOfTheYear(_Day,
_Month, _Year);
                short MonthDays = 0;

                _Month = 1;

                while (true)
                {
                        MonthDays = NumberOfDaysInAMonth(_Month, _Year);

                        if (RemainingDays > MonthDays)
                        {
                                RemainingDays -= MonthDays;
                                _Month++;

                                if (_Month > 12)
                                {
                                        _Month = 1;
                                        _Year++;

                                }
                        }
                        else
```

```
                        {
                                _Day = RemainingDays;
                                break;
                        }

                }


        }

        static bool IsDate1BeforeDate2(clsDate Date1, clsDate Date2)
        {
                return  (Date1.Year < Date2.Year) ? true : ((Date1.Year == Date2.Year) ?
(Date1.Month < Date2.Month ? true : (Date1.Month == Date2.Month ? Date1.Day < Date2.Day :
false)) : false);
        }

        bool IsDateBeforeDate2(clsDate Date2)
        {
                //note: *this sends the current object :-)
                return  IsDate1BeforeDate2(*this, Date2);

        }

        static bool IsDate1EqualDate2(clsDate Date1, clsDate Date2)
        {
                return  (Date1.Year == Date2.Year) ? ((Date1.Month == Date2.Month) ?
((Date1.Day == Date2.Day) ? true : false) : false) : false;
        }

        bool IsDateEqualDate2(clsDate Date2)
        {
                return  IsDate1EqualDate2(*this, Date2);
        }

        static bool IsLastDayInMonth(clsDate Date)
        {

                return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));

        }

        bool IsLastDayInMonth()
        {

                return IsLastDayInMonth(*this);

        }

        static bool IsLastMonthInYear(short Month)
        {
                return (Month == 12);
        }

        static clsDate AddOneDay(clsDate Date)
        {
                if (IsLastDayInMonth(Date))
                {
                        if (IsLastMonthInYear(Date.Month))
                        {
                                Date.Month = 1;
                                Date.Day = 1;
                                Date.Year++;
                        }
                        else
                        {
```

```
                                Date.Day = 1;
                                Date.Month++;
                        }
                }
                else
                {
                        Date.Day++;
                }

                return Date;
        }

        void AddOneDay()

        {
                *this = AddOneDay(*this);
        }

        static void  SwapDates(clsDate& Date1, clsDate& Date2)
        {

                clsDate TempDate;
                TempDate = Date1;
                Date1 = Date2;
                Date2 = TempDate;

        }

        static int GetDifferenceInDays(clsDate Date1, clsDate Date2, bool IncludeEndDay =
false)
        {
                //this will take care of negative diff
                int Days = 0;
                short SawpFlagValue = 1;

                if (!IsDate1BeforeDate2(Date1, Date2))
                {
                        //Swap Dates
                        SwapDates(Date1, Date2);
                        SawpFlagValue = -1;

                }

                while (IsDate1BeforeDate2(Date1, Date2))
                {
                        Days++;
                        Date1 = AddOneDay(Date1);
                }

                return IncludeEndDay ? ++Days * SawpFlagValue : Days * SawpFlagValue;
        }

        int GetDifferenceInDays(clsDate Date2, bool IncludeEndDay = false)
        {
                return GetDifferenceInDays(*this, Date2, IncludeEndDay);
        }

        static short CalculateMyAgeInDays(clsDate DateOfBirth)
        {
                return GetDifferenceInDays(DateOfBirth, clsDate::GetSystemDate(), true);
        }
        //above no need to have nonstatic function for the object because it does not depend on
any data from it.

        static clsDate IncreaseDateByOneWeek(clsDate& Date)
        {
```

```
		for (int i = 1; i <= 7; i++)
		{
			Date = AddOneDay(Date);
		}

		return Date;
	}

	void IncreaseDateByOneWeek()
	{
		IncreaseDateByOneWeek(*this);
	}

	clsDate IncreaseDateByXWeeks(short Weeks, clsDate& Date)
	{

		for (short i = 1; i <= Weeks; i++)
		{
			Date = IncreaseDateByOneWeek(Date);
		}
		return Date;
	}

	void IncreaseDateByXWeeks(short Weeks)
	{
		IncreaseDateByXWeeks(Weeks, *this);
	}

	clsDate IncreaseDateByOneMonth(clsDate& Date)
	{

		if (Date.Month == 12)
		{
			Date.Month = 1;
			Date.Year++;
		}
		else
		{
			Date.Month++;
		}

		//last check day in date should not exceed max days in the current month
		// example if date is 31/1/2022 increasing one month should not be 31/2/2022, it
should
		// be 28/2/2022
		short NumberOfDaysInCurrentMonth = NumberOfDaysInAMonth(Date.Month,
Date.Year);
		if (Date.Day > NumberOfDaysInCurrentMonth)
		{
			Date.Day = NumberOfDaysInCurrentMonth;
		}

		return Date;
	}

	void IncreaseDateByOneMonth()
	{

		IncreaseDateByOneMonth(*this);

	}

	clsDate IncreaseDateByXDays(short Days, clsDate& Date)
	{
```

```
        for (short i = 1; i <= Days; i++)
        {
            Date = AddOneDay(Date);
        }
        return Date;
}

void IncreaseDateByXDays(short Days)
{

        IncreaseDateByXDays(Days, *this);
}

clsDate IncreaseDateByXMonths(short Months, clsDate& Date)
{

        for (short i = 1; i <= Months; i++)
        {
            Date = IncreaseDateByOneMonth(Date);
        }
        return Date;
}

void IncreaseDateByXMonths(short Months)
{
        IncreaseDateByXMonths(Months, *this);
}

static clsDate IncreaseDateByOneYear(clsDate& Date)
{
        Date.Year++;
        return Date;
}

void IncreaseDateByOneYear()
{
        IncreaseDateByOneYear(*this);
}

clsDate IncreaseDateByXYears(short Years, clsDate& Date)
{
        Date.Year += Years;
        return Date;

}

void IncreaseDateByXYears(short Years)
{
        IncreaseDateByXYears(Years);
}

clsDate IncreaseDateByOneDecade(clsDate& Date)
{
        //Period of 10 years
        Date.Year += 10;
        return Date;
}

void IncreaseDateByOneDecade()
{
        IncreaseDateByOneDecade(*this);
}

clsDate IncreaseDateByXDecades(short Decade, clsDate& Date)
{
        Date.Year += Decade * 10;
```

```
        return Date;
}

void IncreaseDateByXDecades(short Decade)
{
        IncreaseDateByXDecades(Decade, *this);
}

clsDate IncreaseDateByOneCentury(clsDate& Date)
{
        //Period of 100 years
        Date.Year += 100;
        return Date;
}

void IncreaseDateByOneCentury()
{
        IncreaseDateByOneCentury(*this);
}

clsDate IncreaseDateByOneMillennium(clsDate& Date)
{
        //Period of 1000 years
        Date.Year += 1000;
        return Date;
}

clsDate IncreaseDateByOneMillennium()
{
        IncreaseDateByOneMillennium(*this);
}

static clsDate DecreaseDateByOneDay(clsDate Date)
{
        if (Date.Day == 1)
        {
                if (Date.Month == 1)
                {
                        Date.Month = 12;
                        Date.Day = 31;
                        Date.Year--;
                }
                else
                {

                        Date.Month--;
                        Date.Day = NumberOfDaysInAMonth(Date.Month, Date.Year);
                }
        }
        else
        {
                Date.Day--;
        }

        return Date;
}

void DecreaseDateByOneDay()
{
        DecreaseDateByOneDay(*this);
}

static clsDate DecreaseDateByOneWeek(clsDate& Date)
{

        for (int i = 1; i <= 7; i++)
```

```
		{
			Date = DecreaseDateByOneDay(Date);
		}

		return Date;
	}

	void DecreaseDateByOneWeek()
	{
		DecreaseDateByOneWeek(*this);
	}

	static clsDate DecreaseDateByXWeeks(short Weeks, clsDate& Date)
	{

		for (short i = 1; i <= Weeks; i++)
		{
			Date = DecreaseDateByOneWeek(Date);
		}
		return Date;
	}

	void DecreaseDateByXWeeks(short Weeks)
	{
		DecreaseDateByXWeeks(Weeks, *this);
	}

	static clsDate DecreaseDateByOneMonth(clsDate& Date)
	{

		if (Date.Month == 1)
		{
			Date.Month = 12;
			Date.Year--;
		}
		else
			Date.Month--;


		//last check day in date should not exceed max days in the current month
	  // example if date is 31/3/2022 decreasing one month should not be 31/2/2022, it
should
	  // be 28/2/2022
		short NumberOfDaysInCurrentMonth = NumberOfDaysInAMonth(Date.Month,
Date.Year);
		if (Date.Day > NumberOfDaysInCurrentMonth)
		{
			Date.Day = NumberOfDaysInCurrentMonth;
		}


		return Date;
	}

	void DecreaseDateByOneMonth()
	{
		DecreaseDateByOneMonth(*this);
	}

	static clsDate DecreaseDateByXDays(short Days, clsDate& Date)
	{

		for (short i = 1; i <= Days; i++)
		{
			Date = DecreaseDateByOneDay(Date);
		}
```

```
        return Date;
    }

    void DecreaseDateByXDays(short Days)
    {
        DecreaseDateByXDays(Days, *this);
    }

    static clsDate DecreaseDateByXMonths(short Months, clsDate& Date)
    {

        for (short i = 1; i <= Months; i++)
        {
            Date = DecreaseDateByOneMonth(Date);
        }
        return Date;
    }

    void DecreaseDateByXMonths(short Months)
    {
        DecreaseDateByXMonths(Months, *this);
    }

    static clsDate DecreaseDateByOneYear(clsDate& Date)
    {

        Date.Year--;
        return Date;
    }

    void DecreaseDateByOneYear()
    {
        DecreaseDateByOneYear(*this);
    }

    static clsDate DecreaseDateByXYears(short Years, clsDate& Date)
    {

        Date.Year -= Years;
        return Date;
    }

    void DecreaseDateByXYears(short Years)
    {
        DecreaseDateByXYears(Years, *this);
    }

    static clsDate DecreaseDateByOneDecade(clsDate& Date)
    {
        //Period of 10 years
        Date.Year -= 10;
        return Date;
    }

    void DecreaseDateByOneDecade()
    {
        DecreaseDateByOneDecade(*this);
    }

    static clsDate DecreaseDateByXDecades(short Decades, clsDate& Date)
    {

        Date.Year -= Decades * 10;
        return Date;
    }
```

```
void DecreaseDateByXDecades(short Decades)
{
	DecreaseDateByXDecades(Decades, *this);
}

static clsDate DecreaseDateByOneCentury(clsDate& Date)
{
	//Period of 100 years
	Date.Year -= 100;
	return Date;
}

void DecreaseDateByOneCentury()
{
	DecreaseDateByOneCentury(*this);
}

static clsDate DecreaseDateByOneMillennium(clsDate& Date)
{
	//Period of 1000 years
	Date.Year -= 1000;
	return Date;
}

void DecreaseDateByOneMillennium()
{
	DecreaseDateByOneMillennium(*this);
}


static short IsEndOfWeek(clsDate Date)
{
	return  DayOfWeekOrder(Date.Day, Date.Month, Date.Year) == 6;
}

short IsEndOfWeek()
{
	return IsEndOfWeek(*this);
}

static bool IsWeekEnd(clsDate Date)
{
	//Weekends are Fri and Sat
	short DayIndex = DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
	return  (DayIndex == 5 || DayIndex == 6);
}

bool IsWeekEnd()
{
	return  IsWeekEnd(*this);
}

static bool IsBusinessDay(clsDate Date)
{
	//Weekends are Sun,Mon,Tue,Wed and Thur

  /*
	short DayIndex = DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
	return  (DayIndex >= 5 && DayIndex <= 4);
  */

  //shorter method is to invert the IsWeekEnd: this will save updating code.
	return !IsWeekEnd(Date);

}
```

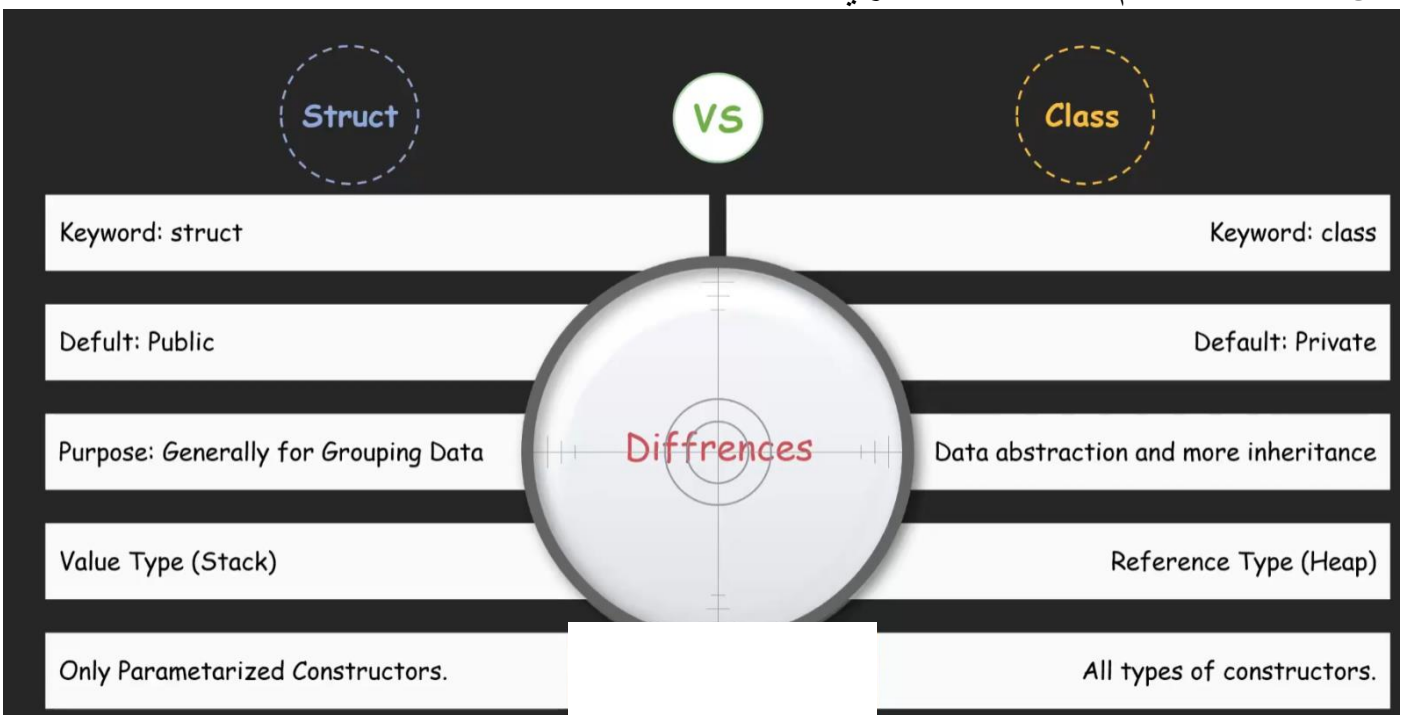
```
bool IsBusinessDay()
{
	return  IsBusinessDay(*this);
}

static short DaysUntilTheEndOfWeek(clsDate Date)
{
	return 6 - DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
}

short DaysUntilTheEndOfWeek()
{
	return  DaysUntilTheEndOfWeek(*this);
}

static short DaysUntilTheEndOfMonth(clsDate Date1)
{

	clsDate EndOfMontDate;
	EndOfMontDate.Day = NumberOfDaysInAMonth(Date1.Month, Date1.Year);
	EndOfMontDate.Month = Date1.Month;
	EndOfMontDate.Year = Date1.Year;

	return GetDifferenceInDays(Date1, EndOfMontDate, true);

}

short DaysUntilTheEndOfMonth()
{
	return DaysUntilTheEndOfMonth(*this);
}

static short DaysUntilTheEndOfYear(clsDate Date1)
{

	clsDate EndOfYearDate;
	EndOfYearDate.Day = 31;
	EndOfYearDate.Month = 12;
	EndOfYearDate.Year = Date1.Year;

	return GetDifferenceInDays(Date1, EndOfYearDate, true);

}

short DaysUntilTheEndOfYear()
{
	return  DaysUntilTheEndOfYear(*this);
}

//i added this method to calculate business days between 2 days
static short CalculateBusinessDays(clsDate DateFrom, clsDate DateTo)
{

	short Days = 0;
	while (IsDate1BeforeDate2(DateFrom, DateTo))
	{
		if (IsBusinessDay(DateFrom))
			Days++;

		DateFrom = AddOneDay(DateFrom);
	}

	return Days;

}
```

```
	static short CalculateVacationDays(clsDate DateFrom, clsDate DateTo)
	{
		/*short Days = 0;
		while (IsDate1BeforeDate2(DateFrom, DateTo))
		{
			if (IsBusinessDay(DateFrom))
				Days++;

			DateFrom = AddOneDay(DateFrom);
		}*/

		return CalculateBusinessDays(DateFrom, DateTo);

	}
	//above method is eough , no need to have method for the object

	static clsDate CalculateVacationReturnDate(clsDate DateFrom, short VacationDays)
	{

		short WeekEndCounter = 0;

		for (short i = 1; i <= VacationDays; i++)
		{

			if (IsWeekEnd(DateFrom))
				WeekEndCounter++;

			DateFrom = AddOneDay(DateFrom);
		}
		//to add weekends
		for (short i = 1; i <= WeekEndCounter; i++)
			DateFrom = AddOneDay(DateFrom);

		return DateFrom;
	}

	static bool IsDate1AfterDate2(clsDate Date1, clsDate Date2)
	{
		return (!IsDate1BeforeDate2(Date1, Date2) && !IsDate1EqualDate2(Date1,
Date2));

	}

	bool IsDateAfterDate2(clsDate Date2)
	{
		return IsDate1AfterDate2(*this, Date2);
	}

	enum enDateCompare { Before = -1, Equal = 0, After = 1 };

	static enDateCompare CompareDates(clsDate Date1, clsDate Date2)
	{
		if (IsDate1BeforeDate2(Date1, Date2))
			return enDateCompare::Before;

		if (IsDate1EqualDate2(Date1, Date2))
			return enDateCompare::Equal;

		/* if (IsDate1AfterDate2(Date1,Date2))
			return enDateCompare::After;*/

			//this is faster
		return enDateCompare::After;

	}
```

```
        enDateCompare CompareDates(clsDate Date2)
        {
                return CompareDates(*this, Date2);
        }


};
```

What is the difference between Class and Structure

لغة ال c++ هي امتداد للغة ال c
ال c++ بتدعم ال oop وال c لا عشان كده كان ال structure في ال c محدود يدوب بيجمع الداتا مع بعضها ومكانش ينفع تحط functions جوه ال structure
فعملوا تحسينات علي الكلاس من حيث الوراثه واضافه ال functions وخلافه
طيب امتي استخدم structure وامتي استخدم ال class ؟
هنا بيقولك استخدم ال structure في تجميع الداتا الصغيره
لكن لو هتعمل ولو function واحده يبقي استخدم الكلاس
مش كل اللغات بتدعم ال structure زي الجافا



Course Completed, Thank You.

هنا المفروض تعمل مشاريع عشان تطبق عاللي اخدته عشان كده الكورس القادم هيكون في الكورس القادم