Note: Important to solve all lessons question

OOP Concept Brief

مبادئ ال OOP الأربعة الرئيسية هي:

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism
5. : Encapsulation

* Arrange all related (Variables-Methods) under the Class
* Access the class only through the objects

1. : Abstraction

* I only “expose” only the necessary class member (Variables - Methods) to the other users (developers), and “hides” the unnecessary details.
* Abstraction! = Abstract Class
* بخفي عنك كتير methods and props مابهموك مشان ماوجعلك راسك فيهن

1. Inheritance:

* Maximize reusability/maintainability
* Super/Base Class <<<-----Inherits----- Sub/Derived Class
* Inheritance Types:

1. Single Inheritance Person: Employee
2. Multi-Level Inheritance:

Person->Employee->Developer (**Not like Multi Inheritance**)

1. Hierarchal Inheritance:

Employee->Doctor Employee->Developer Employee->Tester (All Inherits from base class Employee)

1. Multiple Inheritance (danger only with C++)
2. Hypered Inheritance (more danger than 4)
3. Polymorphism:

* Consistent/Standard code.
* عندي print() فنكشن واحدة الها وظائف مختلفة وتنفذ حسب مايتم استدعائهامن أي object
* تعدد الأوجه للكود : يعني بنفس الاسم للميثود فيني اعمل اشياء مختلفة
* same method can do different thing based on situation
* can be done through 3 concepts:

1. Compile Time Polymorphism:

* Function Overloading
* Operator Overloading:

5+3 = 8

"Saleem" + "Kassab" = "saleemKassab";

1. Run Time Polymorphism:
   1. Function Overriding
   2. Virtual Functions
2. Inheritance (يعتبر شكل من أشكال ال Polymorphism)

Person is Employee and User

فإذا البيرسون صار اله وجهين مرة بيلعب دور الموظف ومرة بيلعب دور اليوزر

word Ploy means "Many" and word Morphism means "Form" so it means "Many Forms"

Important to Understand

1. Access Modifiers:

* set the accessibility of classes, methods, and other members.
* Types:

- private: class

- protected: private for out class and public for class and childs

- public: All

1. Constructors:

* special type of member function that is called automatically when an object is created
* same name as that of the class and it does not have a return type
* Constructor Types:
* Default Constructor: A constructor with no parameters
* Parameterized Constructor:

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

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

بيضملي انه مايكون عندي اوبجيكت فاضي

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

1. Destructors:

last function called before the destroy the object from memory

1. Static Member:
   * Static Member is a variable that is shared for all objects, any object modifies

it gets modified for all other objects.

(Static variable = Shared variable = global variable)

* + Static members **are on the class level not for each object**

ex: count all objects from class

1. Read only property doesn't have set method
2. Function Overriding:
   * Replace main function code with new implementation
   * Just create function in sub class with the same signature --> automatically override
3. Up Casting vs Down Casting:

* always convert from big to small (**child is the big** because it has member more than base class)
* base -> derived
* up casting Example:

Person p = new Employee();

p.print();

Person e = new Student();

e.print();

// Output

Hi, i am an Employee

Hi, i am a Student

1. Virtual Functions:

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

1. Interfaces/ Abstract Class:

* ابرام الصفقات والعقود
* ممكن يحتوي على Normal + Abstract Methods
* اجبر الديفلوبر لما يعمل كلاس يكون عنده Certain Methods
* Abstract Class is the same concept of Interface Class and they are both contracts --> done by Pure Virtual Functions
* A pure virtual function: function that has no definition.
* **If you have one pure virtual function in a class then it will be converted to abstract class.**
* **Abstract Class/Interface Class is a class with pure virtual functions.**
* I CANOOT inherited from Interface (Just Implement it)
* Must implements all functions, and class can have own methods
* Interfaces named by this name because **we only declare the methods interface** - Reference OOP Course11 - Project 3: Bank System - Extension 10 : Abstract Class/Interface Practical Example

1. Friend Class:

* A friend class can access both private and protected members.
* 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.
* We can use a friend Class in C++ using the "friend" keyword.

1. Differences between Class and Structure:

Class Structure

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

Member Access Private Public

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

Purpose Data Abstraction, Inheritance... Grouping Data

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

Stored Reference Type (Heap) Value type(Stack)

يعني بيعطي مساحات اكبرفي الرن تايم مشان أعمل اللي بدي ياه :Reference Type (Heap)

الأشياء الصغيرة بستخدملها ستركتشر والأشياء الكبيرة بستخدملها كلاس

**ملخص خبرة الأستاذ: استخدم الستركتشر فقط لل Data Grouping واذا عندي حتى لو ميثود واحد بحوله لكلاس**

1. Template Functions:

* موضوعها جدا بسيط ويوفر علينا كتابة كود
* if i want to create function to find max 2 numbers (int,int) OR (double, double)....

مافي داعي اعمل اربعة فانكشننز والحل اني بخلي الداتاتايب شو ماكان نوعه

int, double, float, char...

1. Composition:

فكرة مهمة من الكورس الثالث عشر

13 - Algorithms & Problem Solving Level 5

وهي في البروجيكت الثاني - درس انشاء الكيو

لما بدي اعمله بالأساس هو مبني على الدبل لينكد ليست وهنا لم اقم بالوراثة منه لكنني قمت بانشاء كلاس من اللينكد ليست داخل الكلاس كيو

طيب السؤال ليش ماعملت وراثة؟

ببساطة لأني مابدي استخدم كل الميثودات تبع اللينكليست داخل الكيو كلاس

* Conclusion: Composition is a design pattern in object-oriented programming where Expose Certain Methods to another Class