

## OOP Course

Created by : Israa Abdelghany

LinkedIn : [www.linkedin.com/in/israa-abdelghany-4872b0222](https://www.linkedin.com/in/israa-abdelghany-4872b0222)

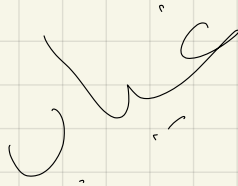
GitHub : <https://github.com/IsraaAbdelghany9>

# Lecture 1

```
void setId (int _id) {  
    id = _id;  
}  
  
int getAge ()  
{ return age; }
```

```
void setId (int id)  
{ this -> id = id; }
```

Picture in iPhone



## Structured programming has some problems:-

- \* structured programming languages consists of some functions.

So you will be distracted between a lot of functions and variables (local or global)

- \* related functions are not related so you may need some global data which can make code check and debug harder

↳ no restrictions on global data

- \* when you are trying to create global variable you may need it for specific number of functions not all  $\Rightarrow$  which can't be done in structure programming language



object oriented programming

- \* All previous issues can be solved here.

1. \* **Encapsulation**:- is the first fundamental in OOP meaning compress (encapsulate) some variables and functions under some name (class) and you can create objects from

- \* **class**:- new data type to create objects from (containing variables and functions that can define or modify it)

- \* object is an instance of the class

2. \* **Abstraction**:- is the process of hiding the internal implementation details of a class and exposing only the essential features to the user. it focuses on what an object does rather than how it does it.

- it focuses on hiding implementation details of both variables and functions

while exposing only what is necessary.

③ • Inheritance :- Improve the reusability

④ • polymorphism :- متعدد الأشكال

override

over load

override

VS

over load

→ Change the behaviour  
of subclass (child) that  
is already defined in the  
parent class

→ happens in inheritance

```
class Animal {  
public:  
    virtual void sound() {  
        cout << "Animal makes a sound" << endl;  
    }  
};  
  
class Dog : public Animal {  
public:  
    void sound() override {  
        cout << "Dog barks" << endl;  
    }  
};
```

\* defining multiple methods with  
the same name but different parameters  
(number or type)

\* happens in same class

```
class Calculator {  
public:  
    int add(int a, int b) { return a + b; }  
    double add(double a, double b) { return a + b; }  
};
```