Object Oriented Programming

C → procedural oriented

- Security
- Reusable

OOP

What is OOPs

OOPs is a programming technique which mainly revolve around real life objects

- Data Type

(i) primitive
(ii) Non - primitive
(iii) User defined

int x = 'y' (char

int arr()

Classes & Objects

Class

It is user defined data type, which contains data members (variables) & methods (functions).

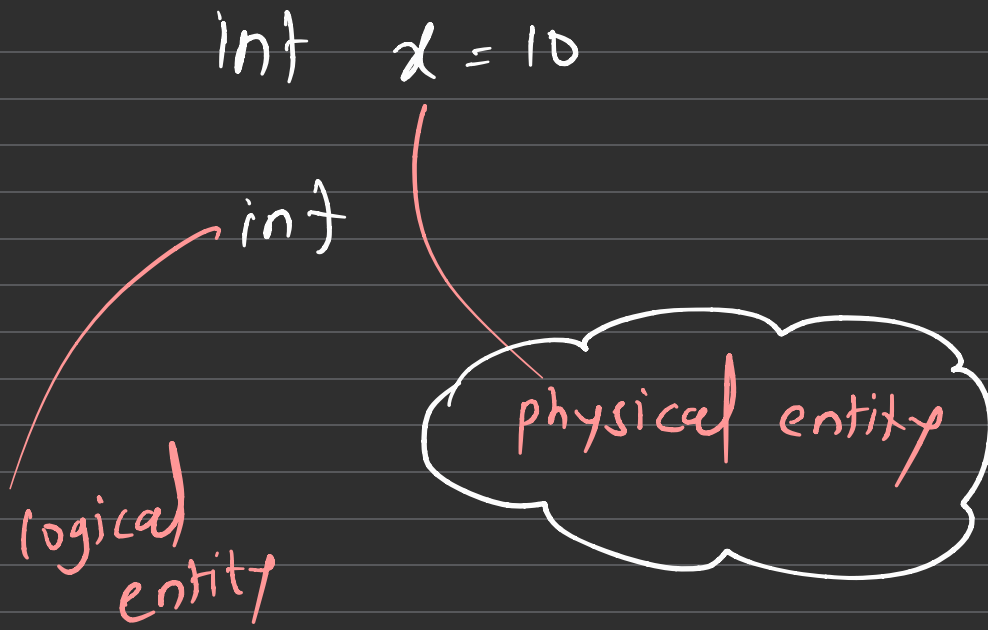
Syntax:

```
class ClassName
{
    // data member

    // methods
}
```

```
class Person
{
    int age
    sting nm

    void show()
    {
        print (nm, age)
    }
}
```



Object

It is a instance of the class - which tells what kind of operation we can perform on the class.

CPP

Syntax:

```
className  obj

Person  pramod

        int  x
```

How to access members of the class

```
class Person
{
    int age
    string nm

    void show()
    {
        print (nm, age)
    }
}
```

Person obj

(.) dot operator

obj.age

obj.nm

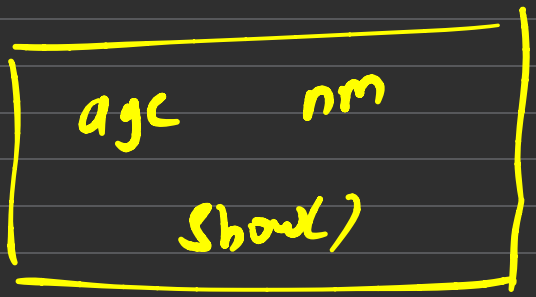
Obj.show()

int x = 10

x

cout << x ⟶ 10

Obj



cout << obj

Access Specifiers

Mainly decides the scope of accessibility of members of the class

~~default~~

private (by default)

public

Protected

inheritance

Private.

- By default all members are private
- can be accessed only in class.

Public

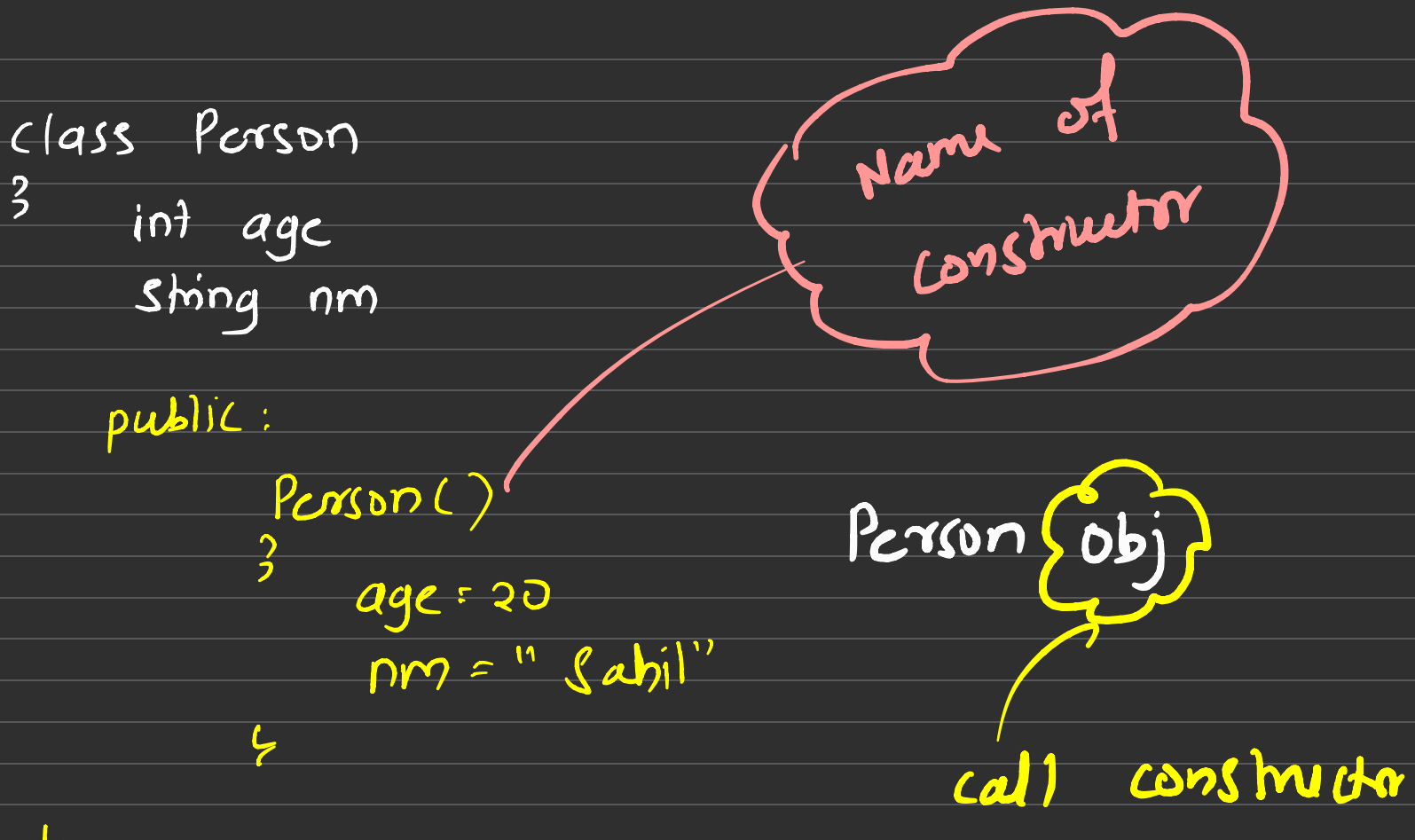
- Anyone can accessed (inside / outside) class

Implict → compiler Explict → program

Constructor & this pointer

mainly helps to initialize object of class

1. always public
2. Name of constructor = name of class
3. does not have return type → not even void
4. calls automatically → moment you create your object



- if no constructor is defined → then behind the Scene default Constructor Exist

- The moment you declar your own constructor → default no longer Exist

1. Default
2. Parameterized
3. Copy constructor

This pointer

```
class Person{

private:
    int age;
    string name;
    int marks;


public:

    // paratemized constructor
    Person(int age, string name, int marks){

        age = age;
        name = name;
        marks = marks;
    }
```

- mainly used to initialize current obj

- also used to separate local variable & class variable

Person (int age, String name, int marks)

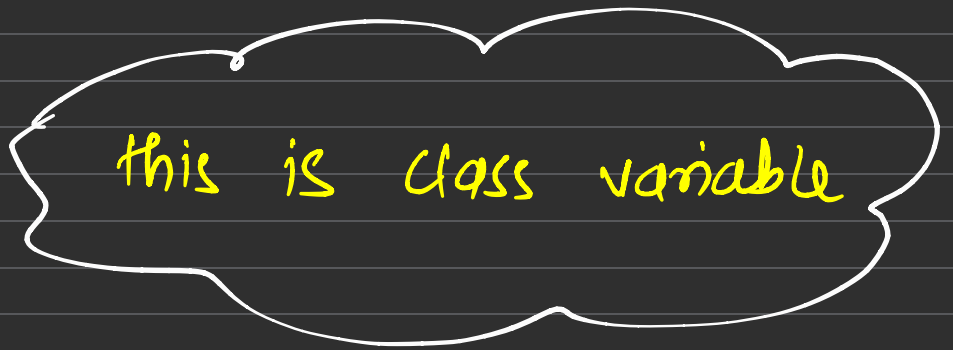
{

this → age = age

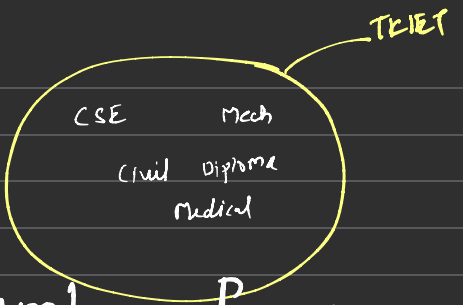
this → name = name

this → marks = marks

}



Encapsulation



Binding of data member and number functions which performs operation on data members together in same class

```
class Person
{
    int age, marks

    void show()
    {

    }
};
```

- Mainly helps us for data Hiding

```
class Person
{
  private
      int age
      string nm
public
      void show()
      {
          print()
      }
}
```

```
class Area
{
  private
      int radius
      int dia
public:
      void findArea(int height)
      {
          dia
          height = height x100
      }
};
```