OOPs

↓

Definition + Eg

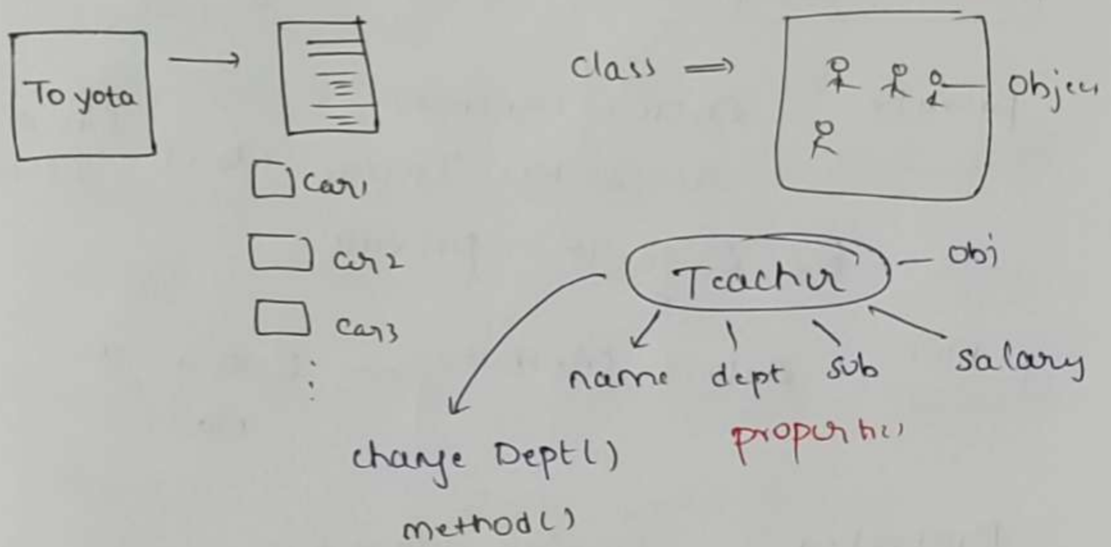
Real life Scenario ⇒ C++ code → Oops

Vector
String } stl
Stack

Classes & Objects:

Objects are Entities in real World ⇒ Obj

class → Blue print of these Entities



string t1Name   string t2Name
string t1dept   string t2dept

→ 50

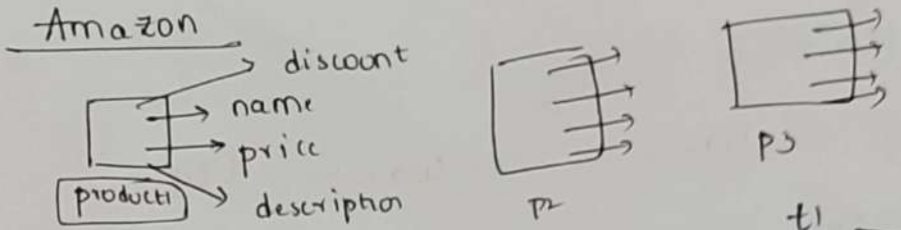
Class → Blueprint

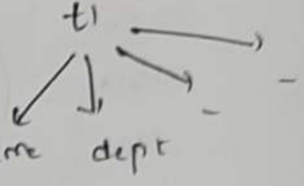
↓

obj 1   T1
obj 2   T2
obj 3   T3

↓ 50/100/12

Amazon





```
class Teacher {


};
```

- ⊛ Encapsulation
- ⊛ Abstraction
- ⊛ Inheritance
- ⊛ Polymorphism

Access Modifiers

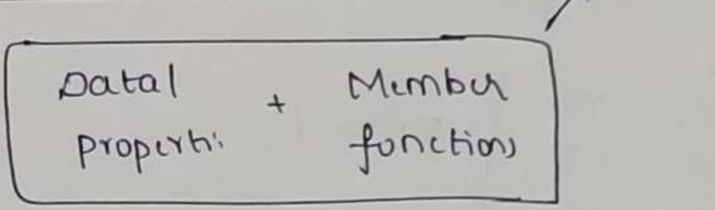
private: Data & Methods Accessible Inside class — Only in Class

By Default private

public: Data & Methods — Outside the class

Protected: Inside class + Derived class

Encapsulation:



Wrapping up of Data & Member functions in a single Unit

Data Hiding

Sensitive Info

Constructor

Special Method —— invoked automatically —— time of Object creation

→ Initialisation

Same name as class

No return type

Only called once — Obj creation

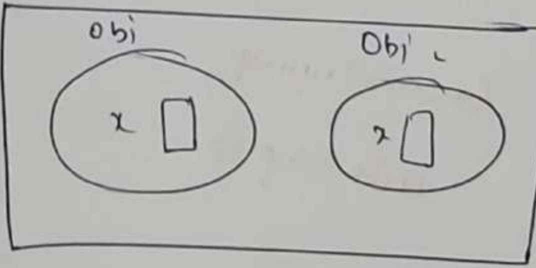
Constructor() → public

Memory Allocation —— when construc - called

```
class A {
    int x;

}
```

—— Memory - not allocated

A obj1 → Now alloc



Constructor → Non Parameterized
→ Parameterized
→ Copy

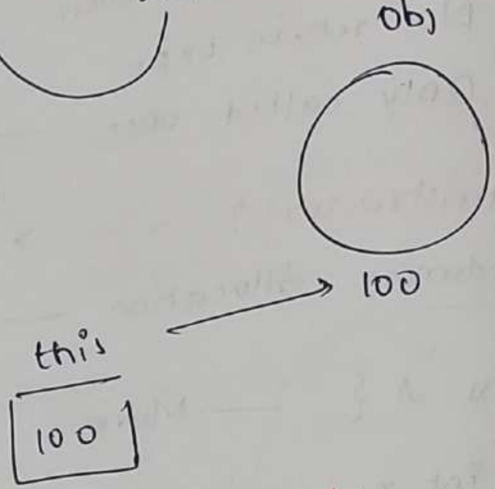
type : $\underbrace{\text{Constructor Overloading}}_{\text{Polymorphism}}$

<u>This</u> : current obj

prop

obj.func()

this



*(this).prop

* dereferencing operator

*ptr = &x

(*this).prop

this → prop

copy constructor ⟶ obj

```
class {
                one obj
    prop ⟶ other obj



}
```

Teacher t1 ("Shradha" , "Computer", "C++", 25000)

Teachu t2(t1)

orig obj

// pass by reference

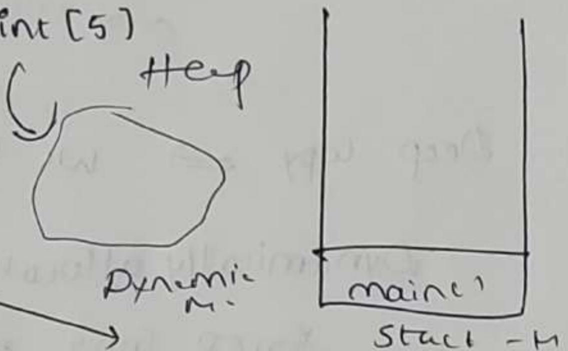
```
Teacher ( Teacher &obj ) {
```

Teacher t2 (t1)

Run time memo all

new int [5]

}

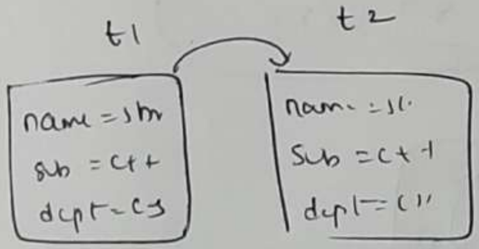


2 copies

Shallow copy

Deep copy

ऊपर

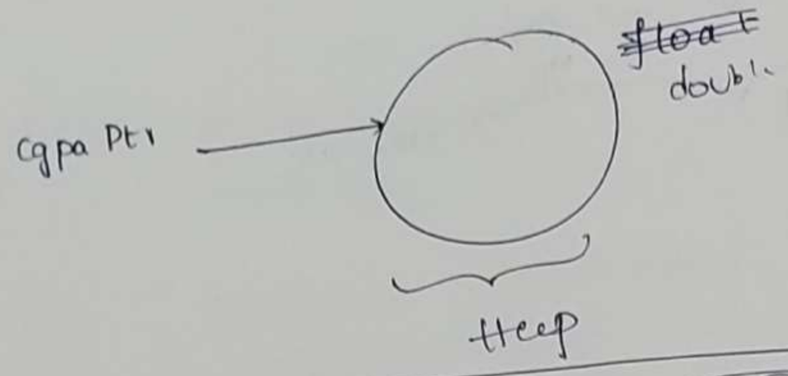


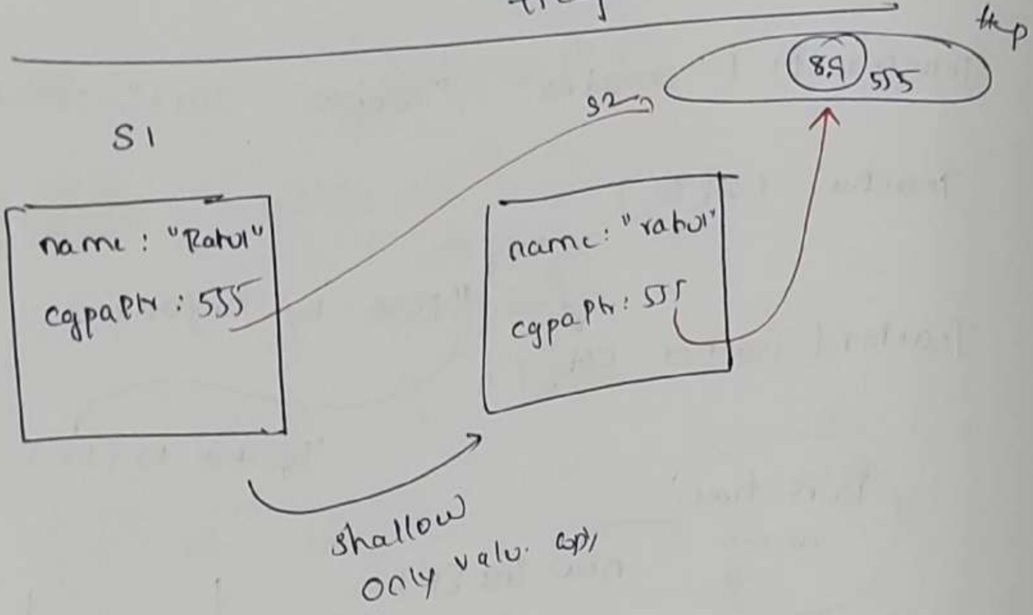
name = shr
sub = C++
dept = CS

name = sh.
sub = C++
dept = CS

Dynamic Memory Allocation

చేసేటప్పుడు లేదా వస్తు



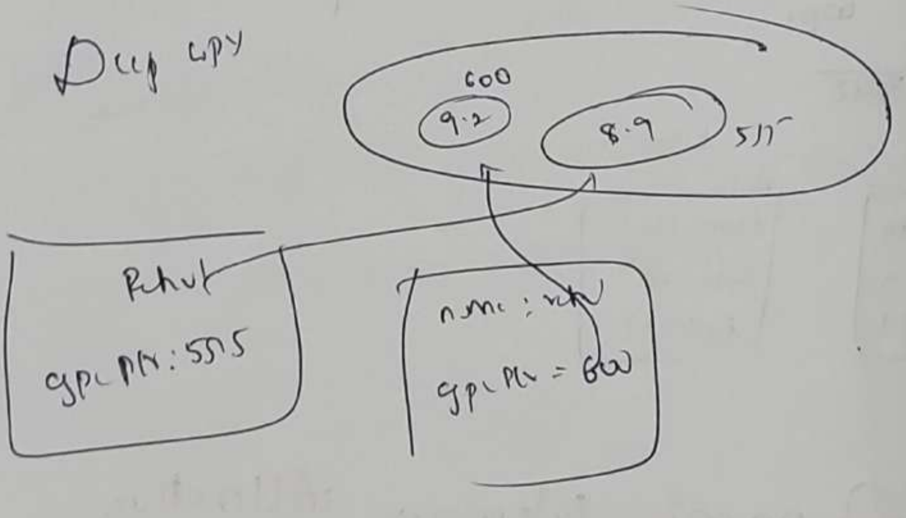


Deep copy ⟹ we have to make

Dynamically allocate చేసిన Memory కి

కూడా copy చేయాలి

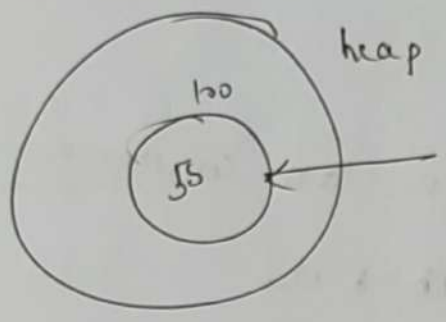
Deep copy



Malli Dynamic ga Allocate చేస్తాం

Destructor → Deallocate

- → Only deallocates Static Memory alloc
- → Dynamic — new delete



delete ptr;

ptr => delete చెప్పాలి
Memory d l..

~Student

delete cgpaPtr

Inheritence

Code Reusabili

Properties &
Memb fun } class → other cls

intx class A (Parent, Base)

↓ ↓

class B (Child, Derived)

B -obj() → Base class construct
→ Child cls

First

Constructor
first — parent
2nd → child

Destruc
1st — child
2nd → parent

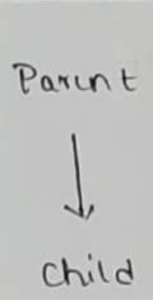
Derived → Mode

Parent	Private	Protected	Public
private	NOt Inherited		
Protected	Private	Protected	Protected
Public	Private	Protected	Public

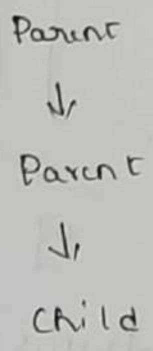
private => Inherit అవ్వదు

protected Make chy

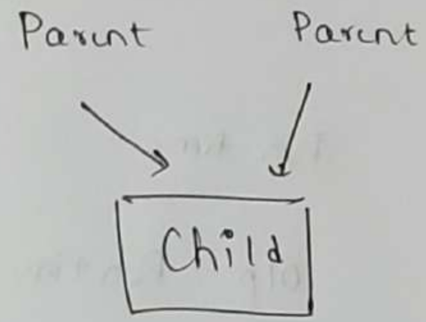
=> Single Inheritance

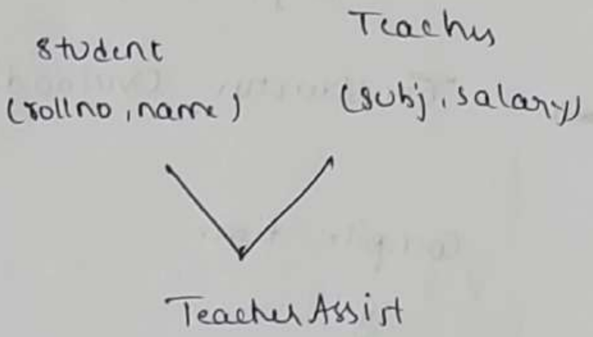


Multile

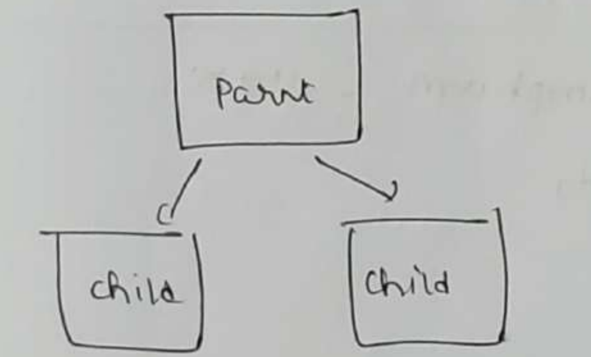


Multipl

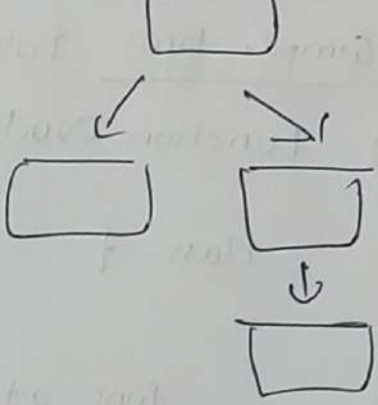




Hierarchical Inherit



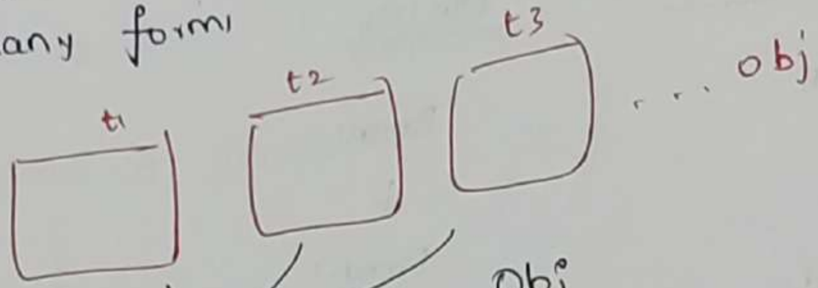
Hybrid Inheri



Polymorphism:

↓

Many form



Ability of Obj

Diff forms | Behave in dift ways depending on the Context

⇒ Constructor Overloading

Compile time

⇓

Output get to know in compile time Only

Run-time

O/p Run time లో తెలుస్తుంది

Compile time Polymorphism — static

(*) Function Overloading

```
class {

    func add ( )          number - dift
                          type - dift

    fun add ( , )
```

Operator Overlo

int y = 10
int x = y

string b = "abc"
string a = b
copy

Run Time Polymorphi

Function Overriding

Overloading   Overrid

class   Inheritnc

↙ ↘   P ✓ ←

↓

C ✓ ←

P

↓

C → parent class func — Overriden

Virtual functions

Dynamic in Nature

Virtual key word

Base om Declare

Child om Override చేస్తాం

Runtime

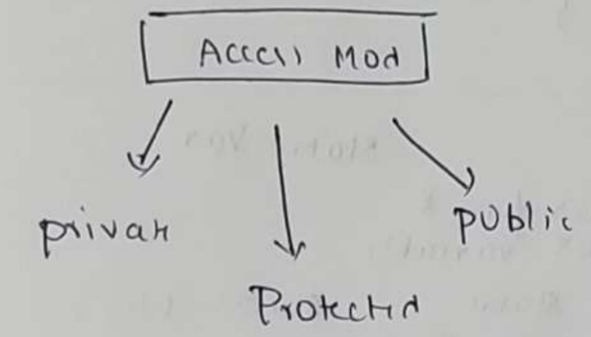
```
class parent {
    virtual void show() {

    }
};
class child {
    void show(){
    }
}
```

Abstraction

└→ Access Modifiers

Hiding all Unnecessary details &

showing only the Imp part



Abstrac

~~అనవసరమైనవి~~

Imp వి చూపిస్తాది

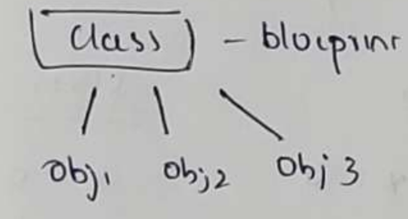
Data Hiding

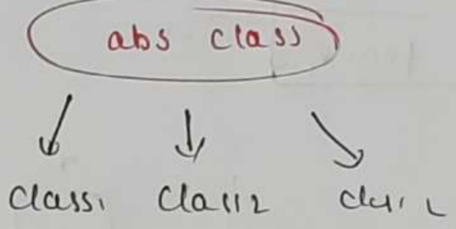
Only Hiding

Abstract classes

```
Abstract classA {

};
```

⇒ ఎప్పుడు Object ని Create చేయరు





Base class

Can't be Instantiated

అంటే Objects crete చేయలేం

→ from which వేరే class are Derived

Blueprint for derived Class

```
abstractShape {
```

```
    draw ()

};
```

```
class Circle : public shape {
        draw () {

        }

};
```

Static Var

Static Keyword

ఒక func &
ఒక variable

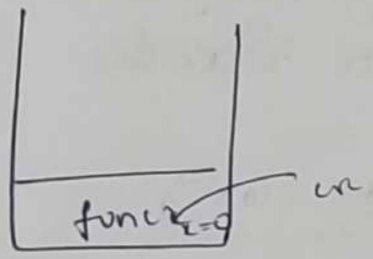
జీవిత కాలంలో

static
అవతే
అవి

static Class

Once created &
Once Initialised
for lifetim

```
void func () {
        int x = 0;
        cout << x << endl;
        x ++;
}
```



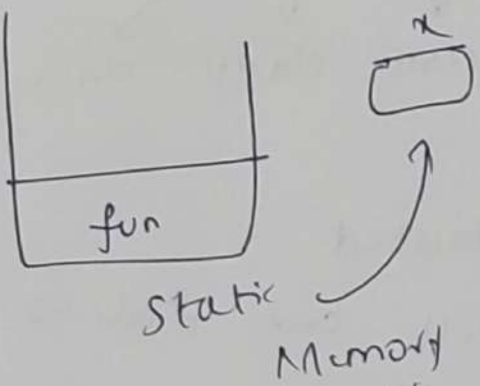
Norm

O/p

x : 0
x : 0
x : 0

Static

x = 0
x = 1
x = 2



Memory
Waste Decr

```
if (true) {
    static ABC obj;

}
cout << "Main end" << endl;
```

→ O/P

Construc

Main.

Destruc

⊗ Norm cu.

friend fnx & find class