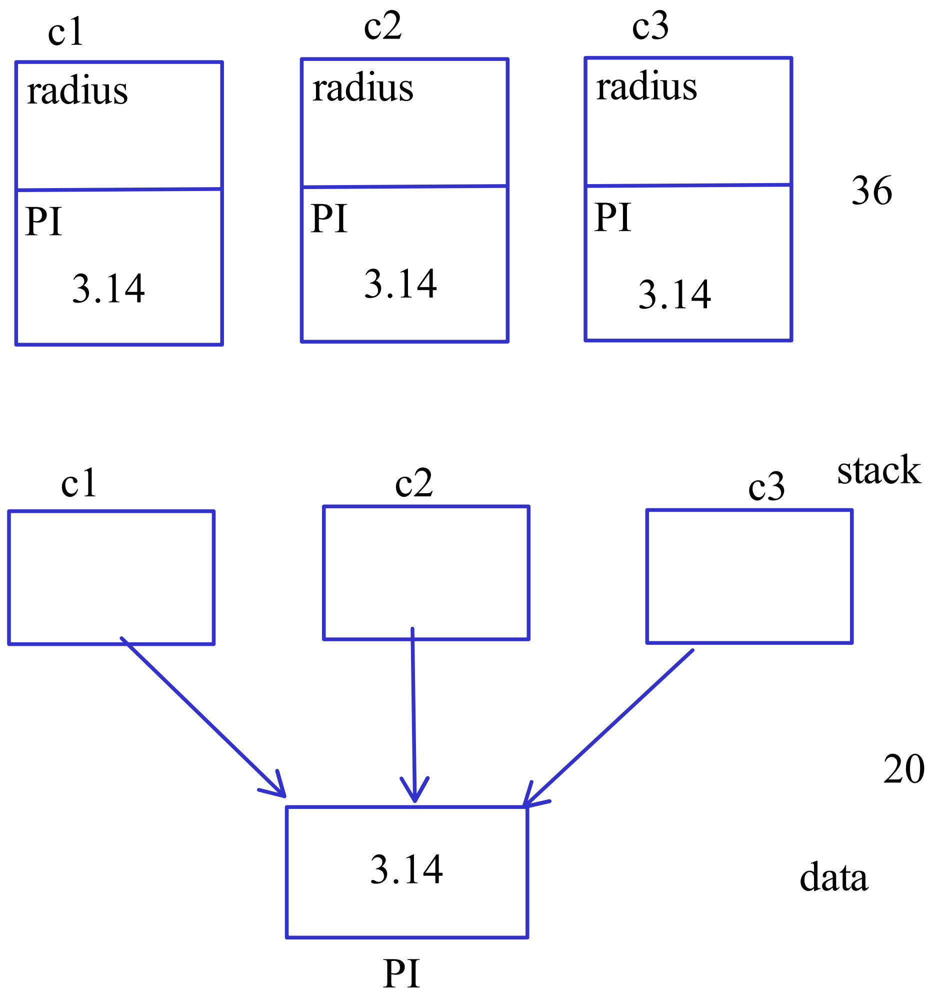
```
class Stack{
int top;
int size;
int *ptr;
Stack(){
top=-1;
this->size = 5;
ptr = new int[this->size];
}
Stack(int size){
top=-1;
this->size = size;
ptr = new int[this->size];
}

isFull();
isEmpty();
push();
pop();
peek();

}
```

Static
- Data Members
  - Memory is allocated on data section at the time of program loading only once
  - initialization should be done outside the class on global scope using classname ::
  - we can make the static data members as constant
- Member function static
  - static member functions are designed to call on class name using ::
  - these functions do not get this pointer
  - We cannot make the static member functions as constant

```
class Circle{
int radius;
static const  double PI ;

}
```

```
Circle c1;
Circle c2;
Circle c3;
```



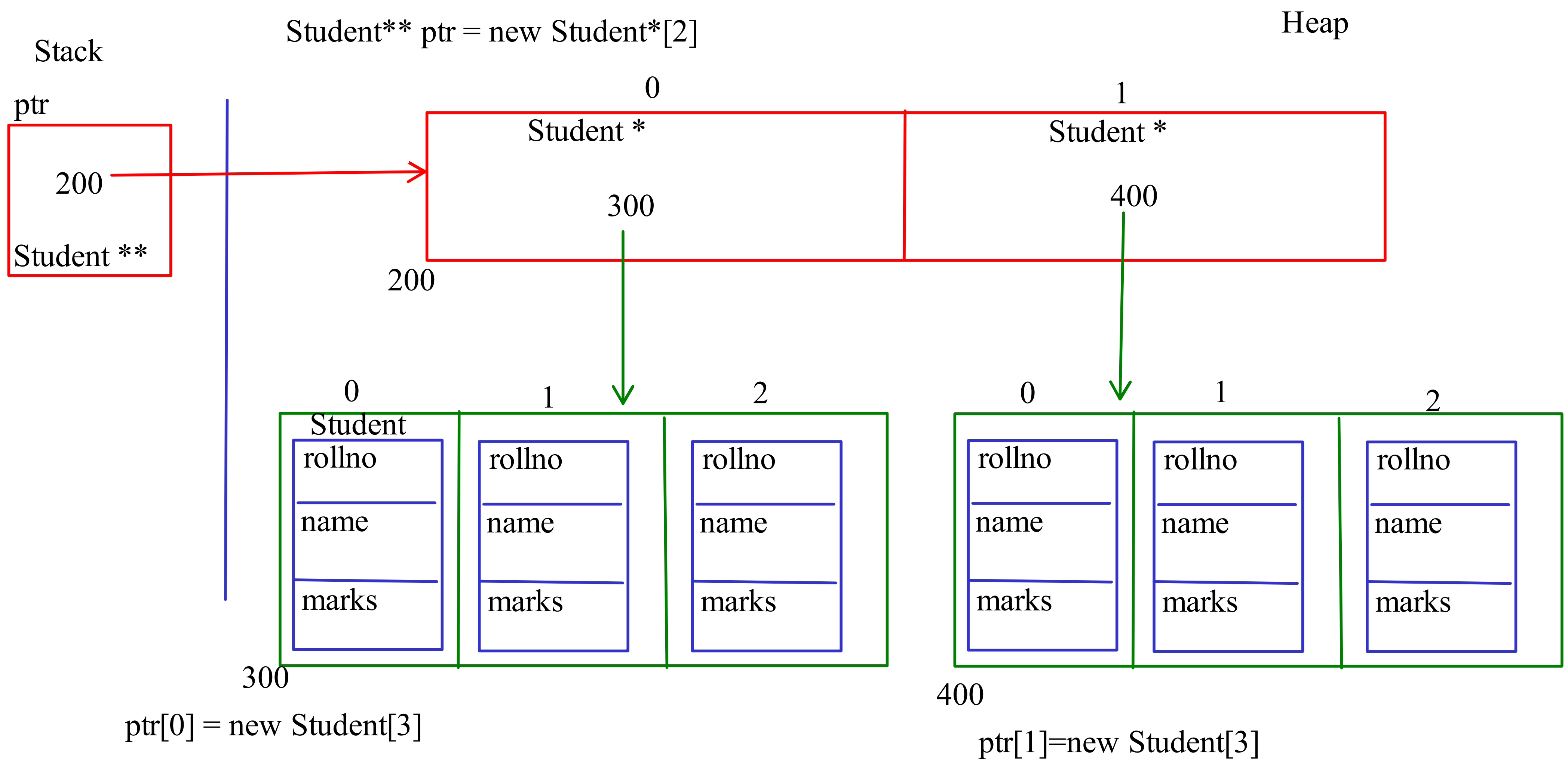
stack -> local variables
heap -> dynamic memory allocation
data -> global and static

Student

DITISS = 60
DBDA = 60

Institute -> Arrray of courses
-> Array of students in every course

Student** ptr = new Student*[2]



ptr[0] = new Student[3]

ptr[1]=new Student[3]

ptr[0][0].displayStudent()

# Hirerachy
- Has-a -> Association
- is-a -> Inheritance

```
Class Date{
day
month
year

}

class Person{
name
dob

}
```

Person has-a Date of Birth

```
class Employee{
id;
name
dob
doj
}
```

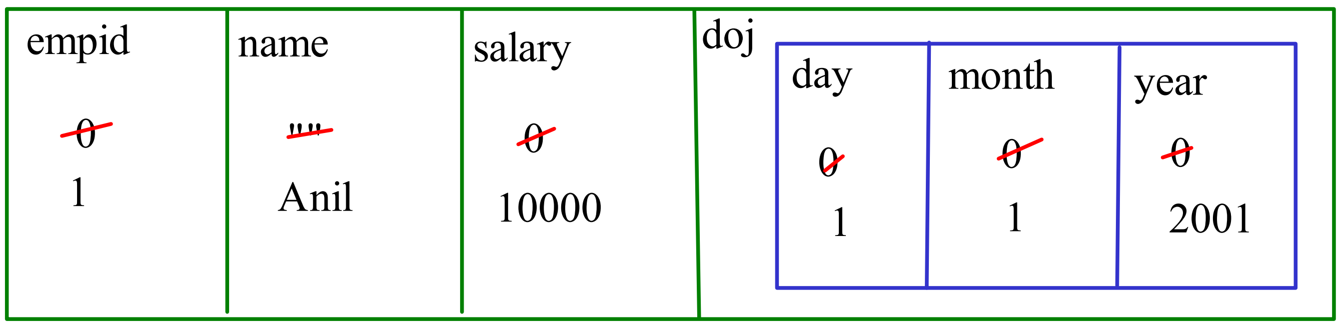
Employee is-a Person
Employee has-a Date of Joining

# Association (has-a)
- It is represented by has-a relation ship between two entities
- Composition
  - Tight coupling
- Aggegration
  - Loose Coupling

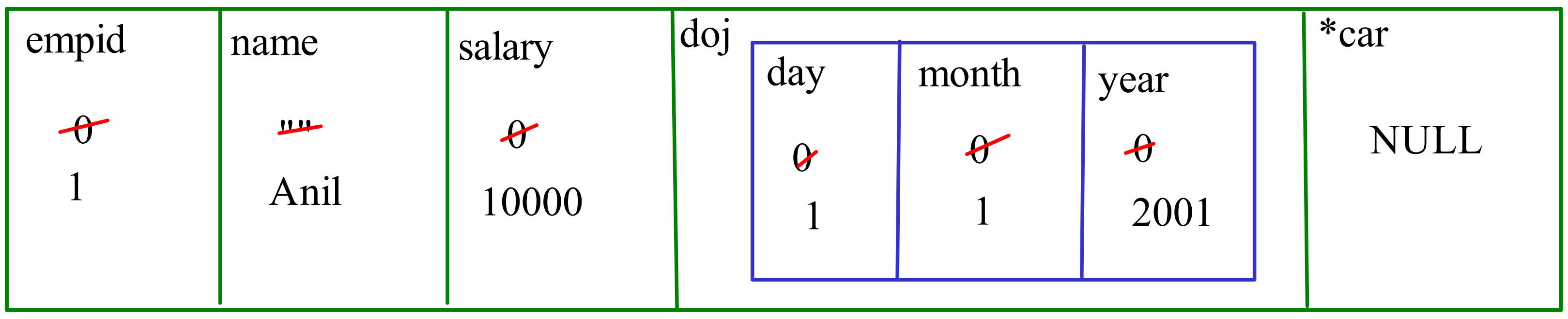
```
Employee{

Date doj // composition
car // aggegration
}
```

Employee has -a DateofJoining(Date)
Dependent --> Dependency

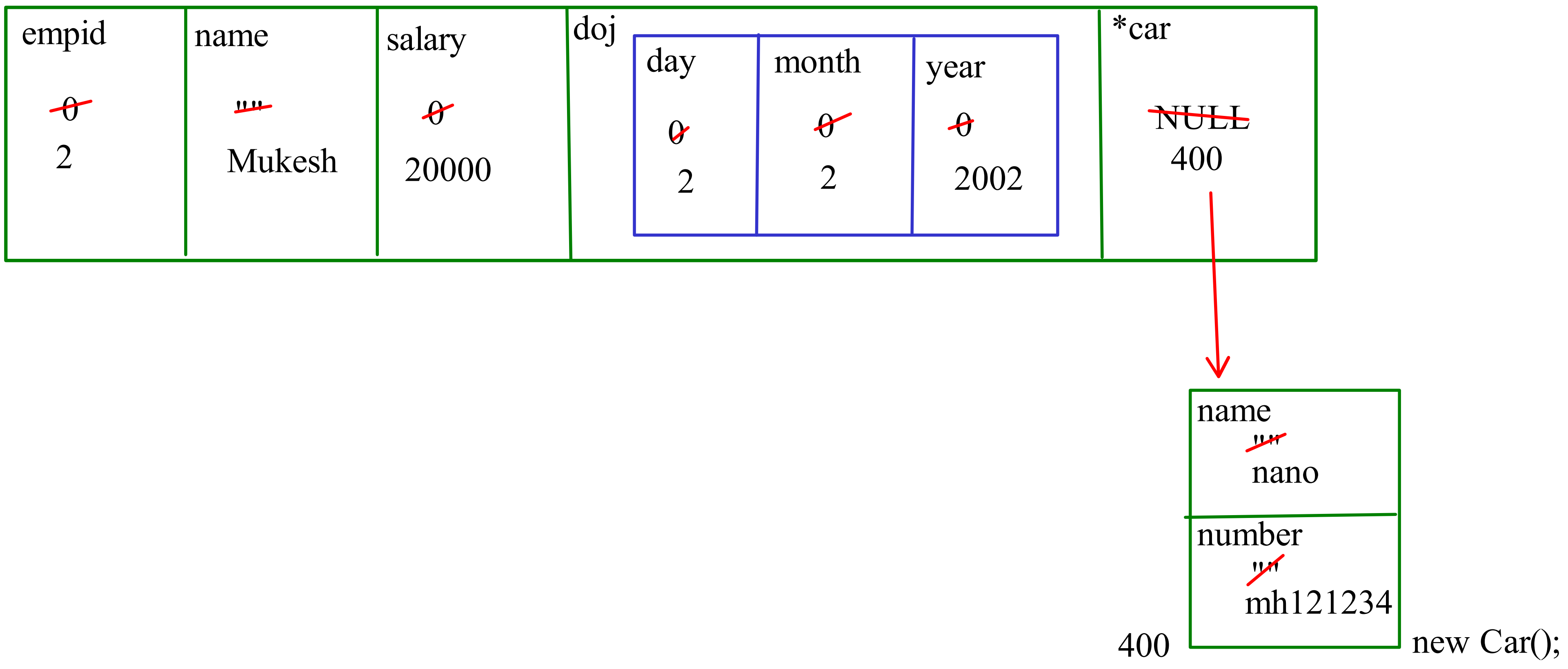
Employee e



Employee e1



Employee e2



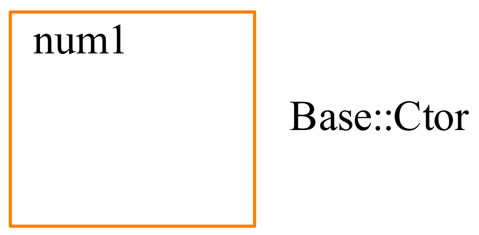
Parent -> Child

Employee is-a Person
Circle ia-a Shape

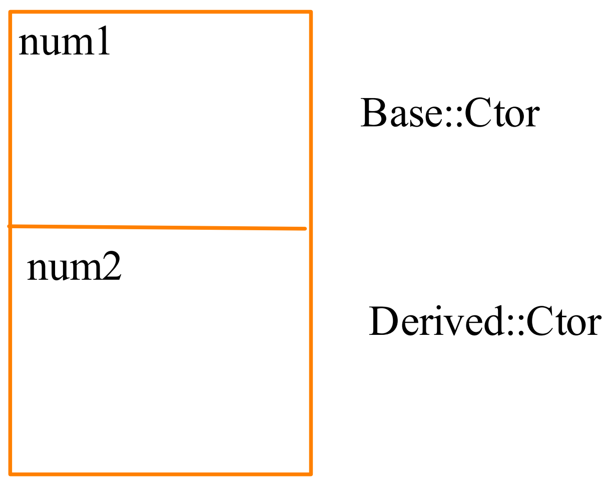
Person is inherited in to the Employee

// Recommended statement
Employee is derived from Person

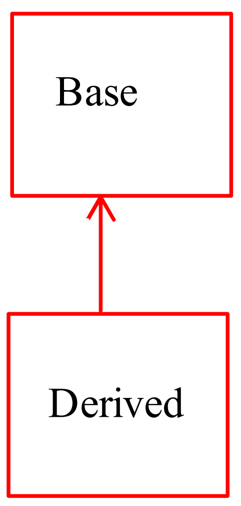
```
class Person // Parent-> Base
{

}

class  Employee : Person // Child->Derived
{

}
```



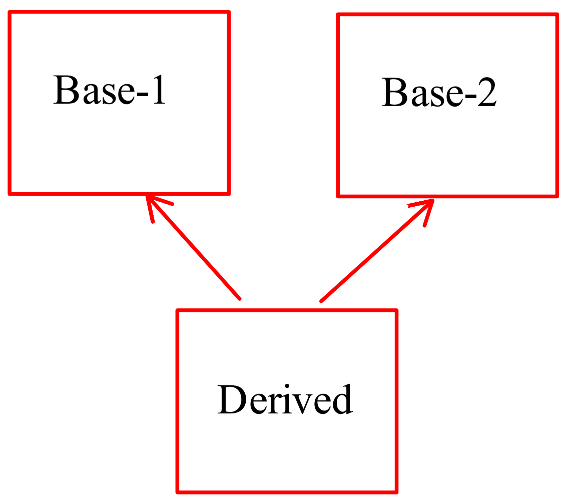
Base b1;



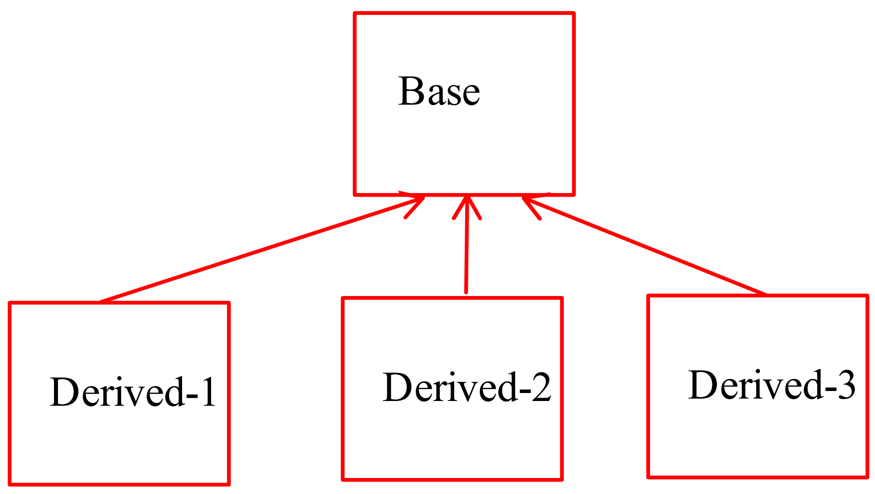
Derived d1;



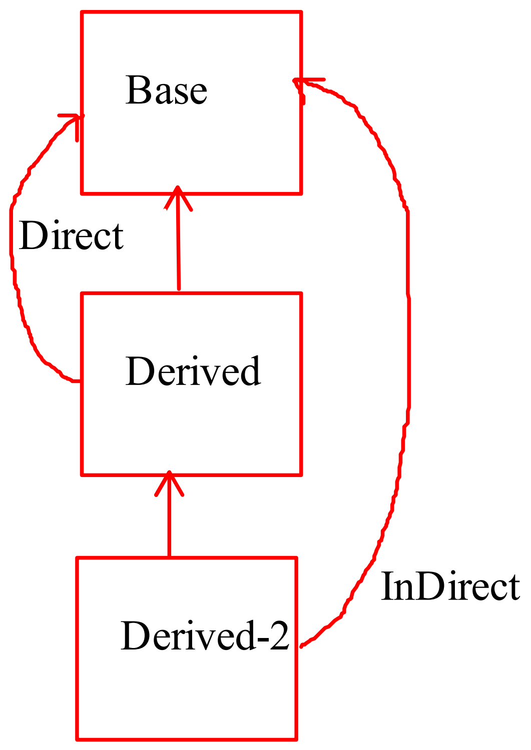
1. Single Inheritance



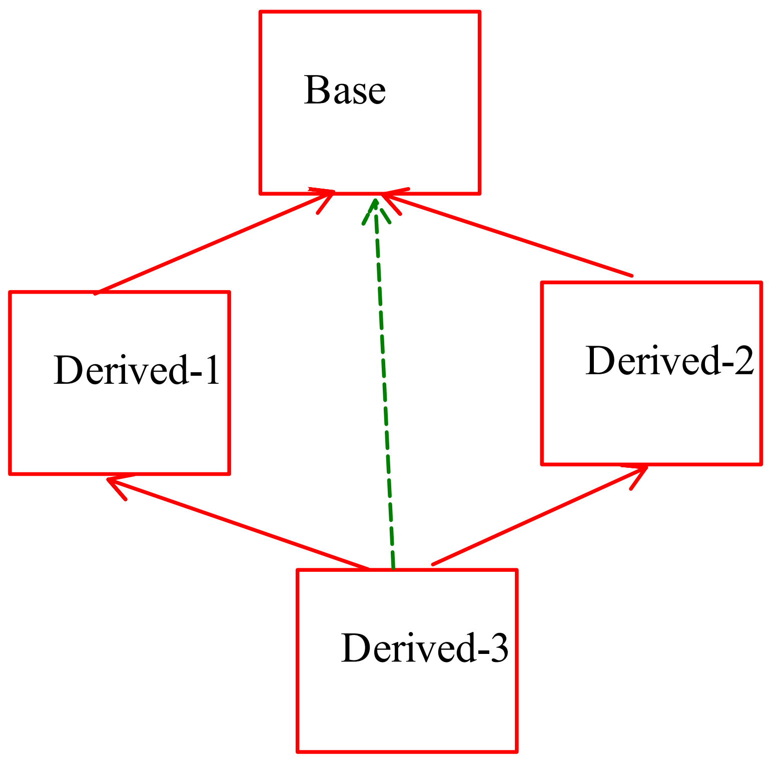
2. Multiple Inheritance



3. Hirerachical Inheritance



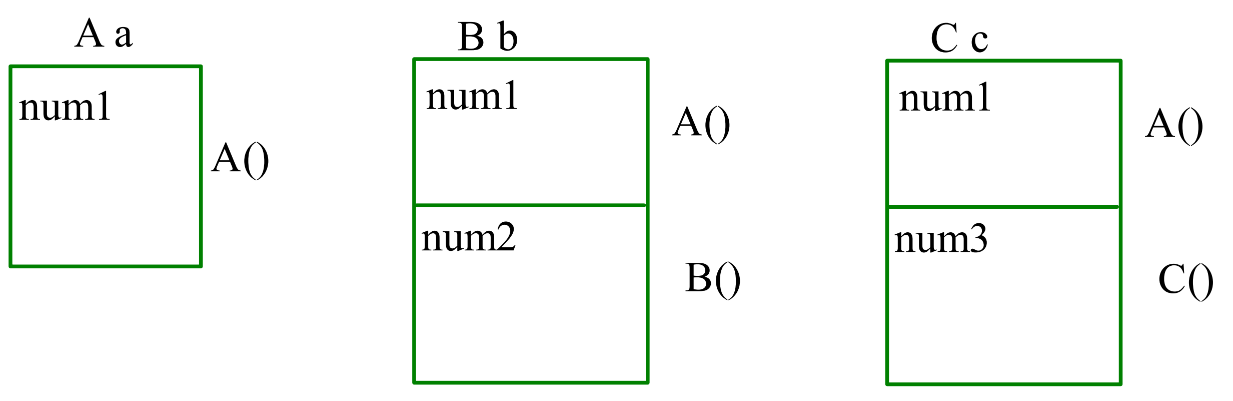
4. Multilevel Inheritance

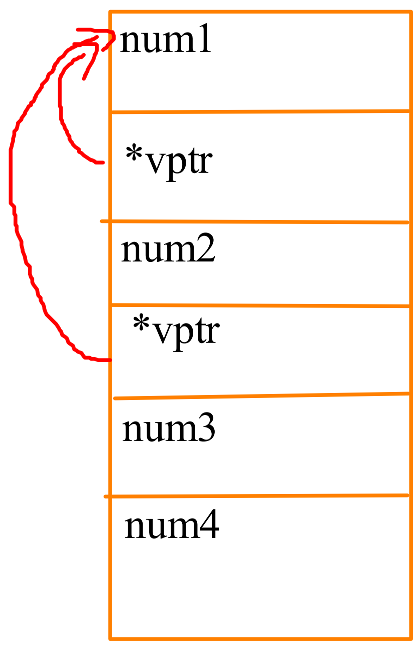


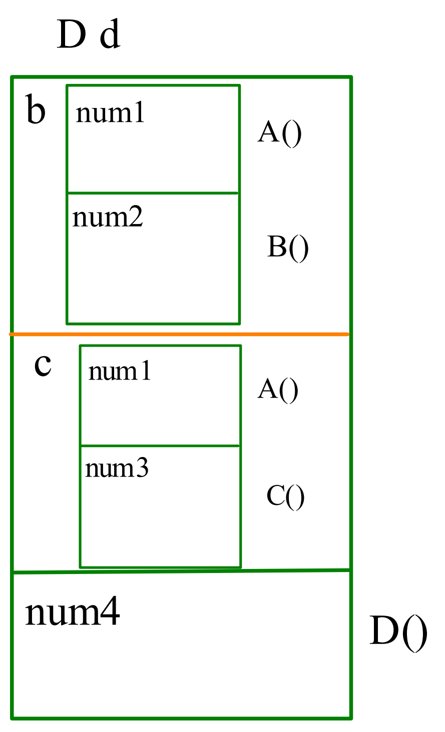
5. Hybrid Inheritance

Class, Ctor, Dtor
Object -> memory
Base *ptr

Access Specifiers
private
public
protected







Person

Manager

Salesman

Salesmanager