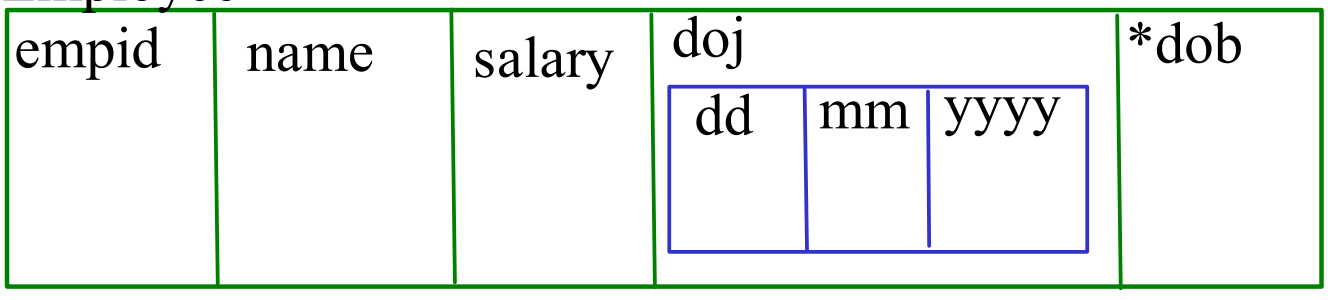
Hirerachy -> reusability
has-a -> Association
is-a -> Inheritance

Association
Employee has-a DOJ

//composition
(Dependent)Human has-a heart (Dependency)

/// Aggegration
Room has-a window

Employee



Day 6 -> Inheritance
Upcasting
Downcasting
Virtual
Virtual Dtor

Exception Handling
Operator loading
Template -> class -> Stack
STL
vector
stack
queue
map

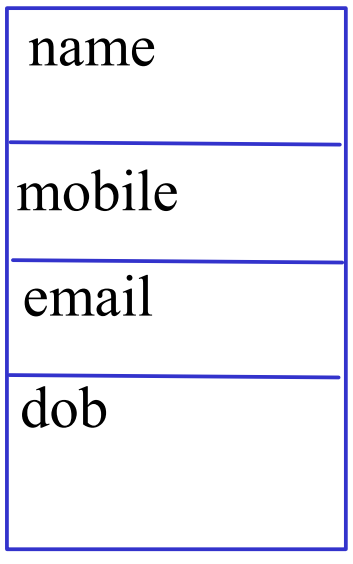
+	()
-	[]
>>	->
<<	
=	

RTTI
Advanced casting operator
Nested and local class
File IO

Inheritance

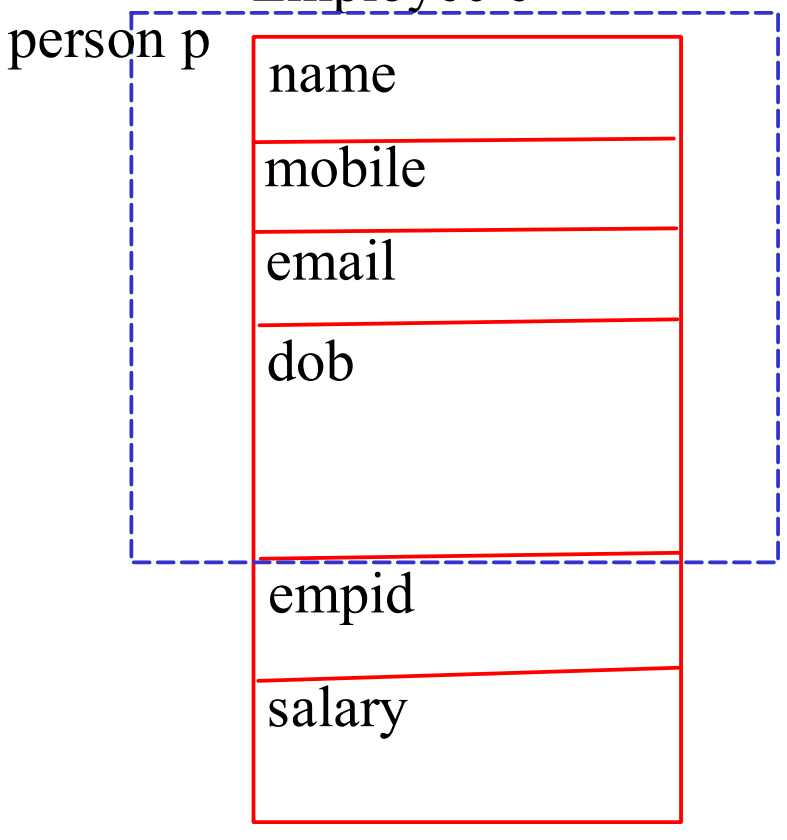
is-a relationship

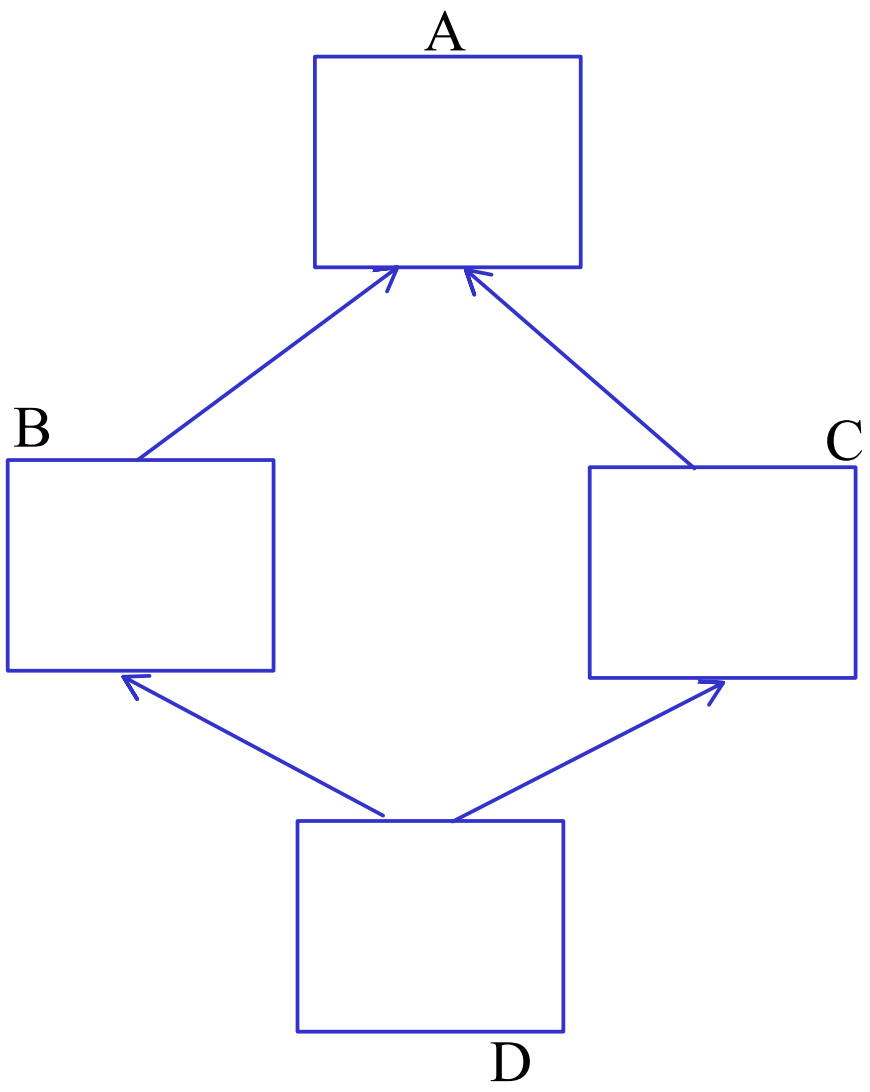
Person p




```
class Person
{
string name;
string mobile;
string email;
Date dob;
}
class Employee : Person{
int empid;
double salary;
}
```

Employee e

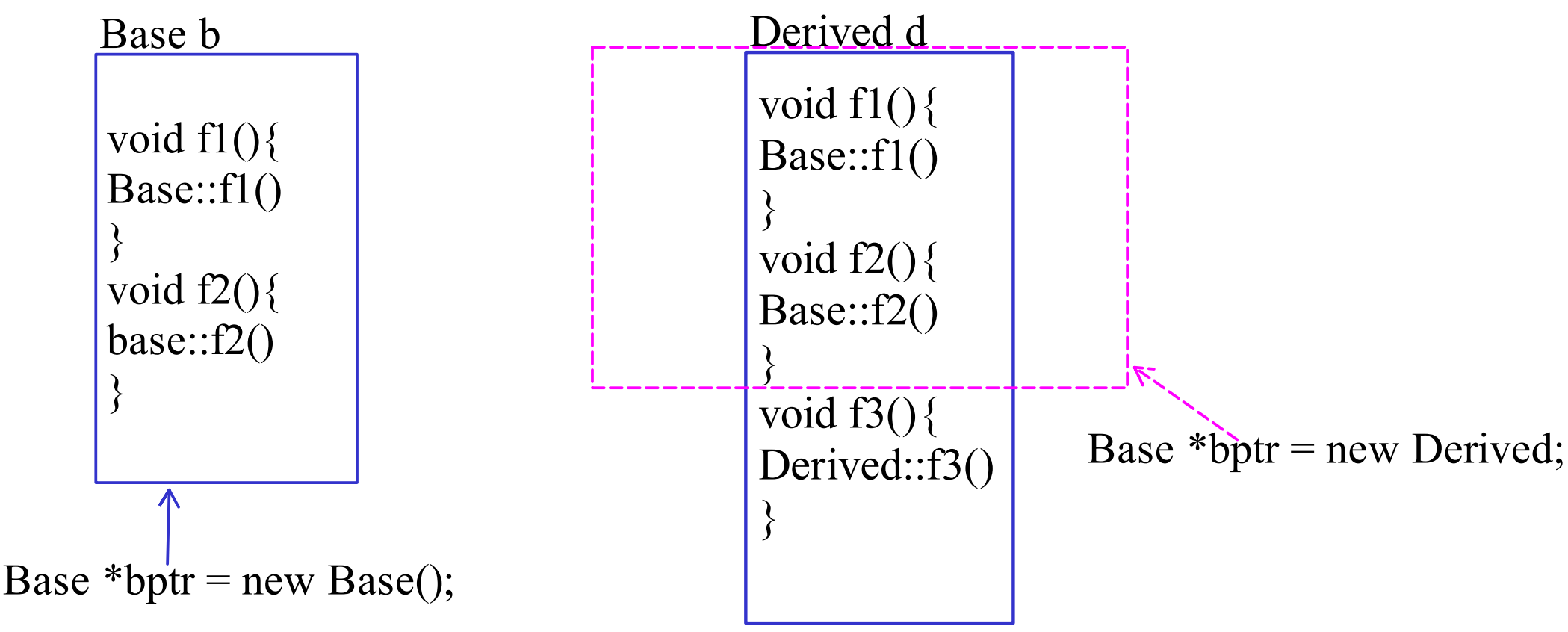




- When we have a Hybrid inheritance then the members of the indirect base class are inherited multiple times in the indirect derived class.

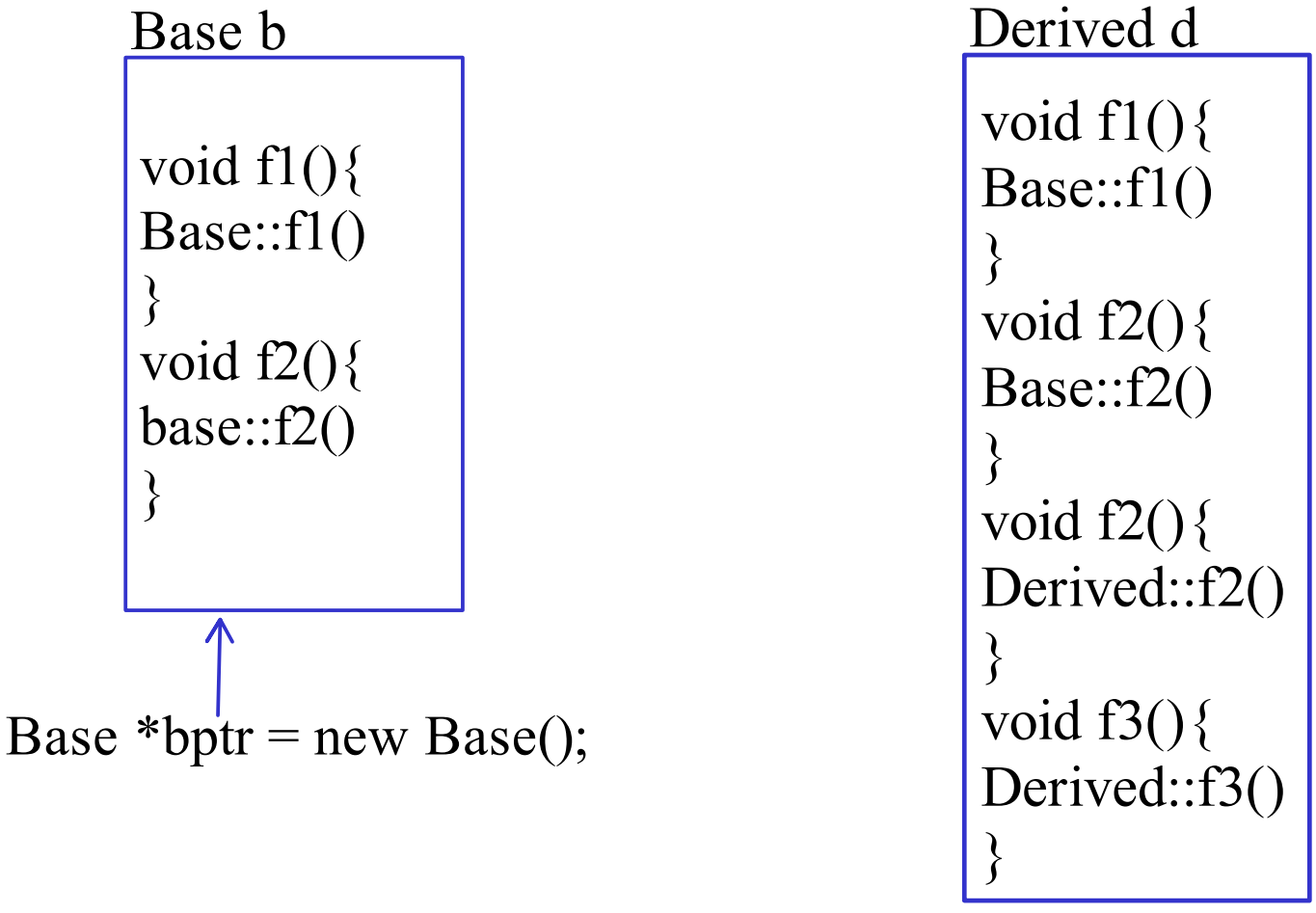
- This is what we refer to as diamond problem

- This can be resolved by making the base class as virtual



Upcasting-
It is a process of storing object of derived class into the base class pointer or reference
When upcasting is done then the base class pointer can only point to the members of base class inherited into the derived class.
It cannot point to the members of the derived class. this is called as object slicing

Downcasting-
Process of converting base class pointer into the derived class pointer is called as downcasting
At the time of downcasting explicit typecasting is mandatory

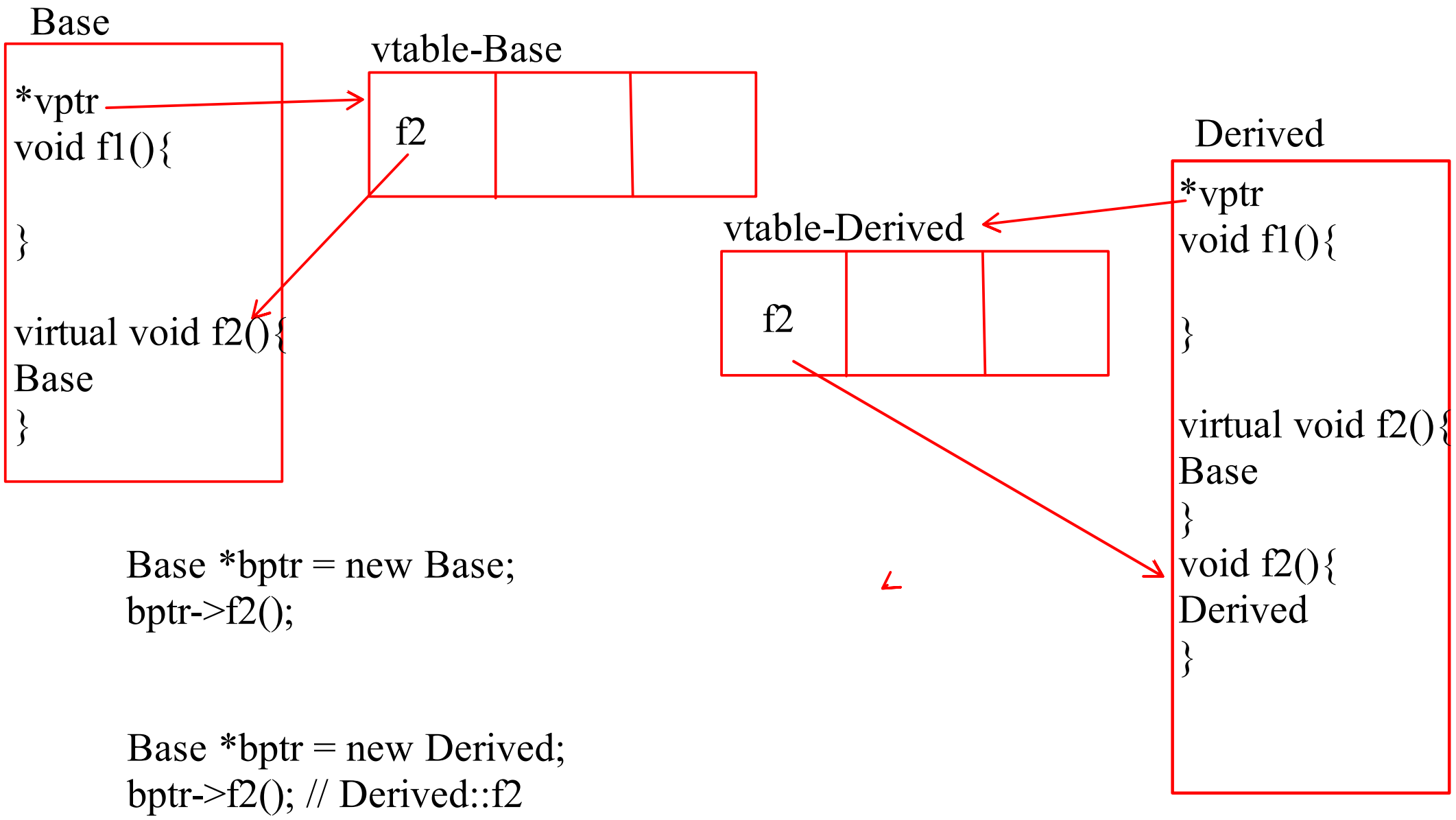


Function Overriding
- Redefining the function of base class once again into the derived class with same name and signature is called as fuction overriding
- Function overriding is an example of Run time polymorphism

Early Binding
- The call to the function is resolved based on the type of pointer that is created.
this is called as early binding

Late Binding
- If we want to resolve the call to the function depending on the object that is created rather than the pointer, we need to perform late binding.
To perform late binding declare the function of the base class as virtual

- Why to perform function overriding?
1. If the implemenetion of the base class function is partial complete
2. If the implementation of the base class function is 100 % incomplete
3. If we want the implementation of derived class function completly different from the base class function




```
Base *bptr = new Base;
bptr->f2();
```

```
Base *bptr = new Derived;
bptr->f2(); // Derived::f2
```

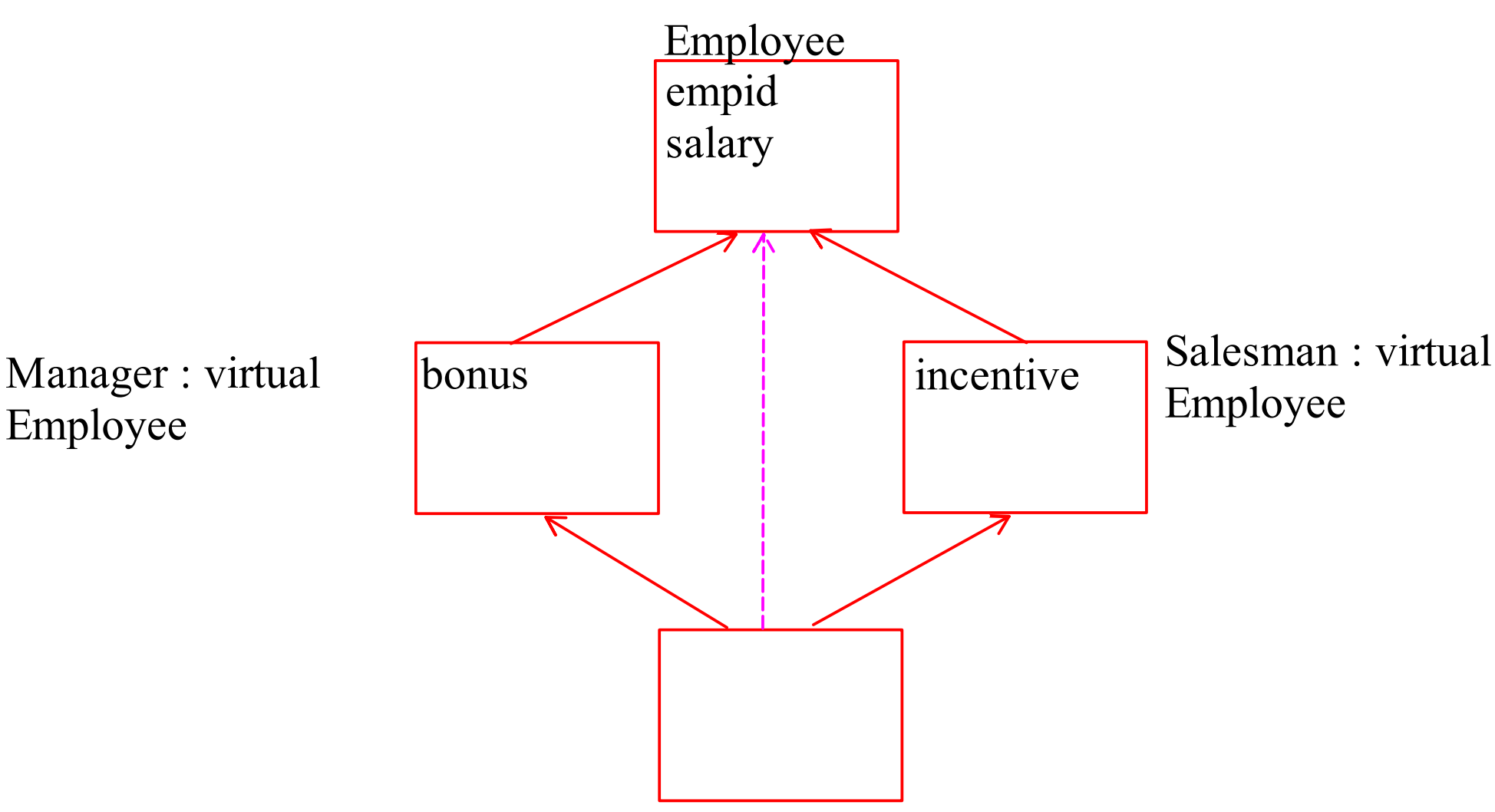
```
Circle:public Shape
{
accept()
calculate()
}
Rectangle: public Shape{
accept()
calculate()
}
```

```
Shape{
void accept();
void calculate();
}
```

Inheritance -> is-a
(Derived/Child)Employee is-a Person (Base/Parent)

```
class Person{
string name;
}

Employee : public Person{
int empid;
double salary;
}
```




```
Employee * eptr = new SalesManager;
eptr->accept();
```

Hybrid Assignement Question

RTTI
Person,Employee,Student

Advanced casting operator

Exception Handling

```
int search(Strudent **arr){
int rollno;
cout<<Enter rollno to search
cin>>

for(int i =0; i<5 ;i++){
if(rollno == arr[i]->getRollno())
        return i;
}
return -1;

}
```

```
Student *arr[5];
int index = search(arr);
if(index!=-1)
//display
else
//not found
```