```
class Point2D
{
String getDetails(){
return "x="+x+"y="+y;
}

boolean isEquals(Point2D p){
        return this.x == p.x && this.y == p.y
}

void calculateDistance(Point2D p){
        sysout(dist);
}
}
```

```
Point2D p1 =  new Point2D(2,3);
Point2D p2 = new Point2D(3,4);

if(p1.isEquals(p2))
        sysout("Equal");
else
        sysout("Not Equal");
```

```
int [][] arr = new int[2][];
arr[0] = new int[2];
arr[1]= new int[3];
```

```
void method(int ... arr){
}

method(10,20)
```

Initializers Final Singleton
Field
Object Static
Ctor -

14 days
+3 -> JDBC

5-> easy
5->Mid
4->Hard

OOP
Hirerachy

1. has-a (Association)
2. is-a (Inheritance)

Human has-a Heart
Employee has-a Doj

Employee is-a Person
Manager is-a Employee

```
class Date{

}
```

```
class Employee{
int empid;
String name;
double salary;
Date dob;
Date doj;
Date dol;
}
```

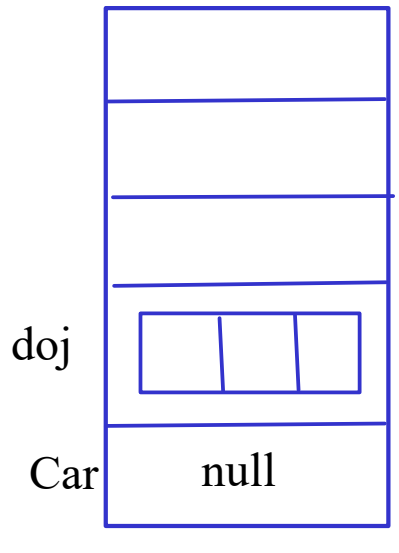
```
class Manager : Employee {

}
```

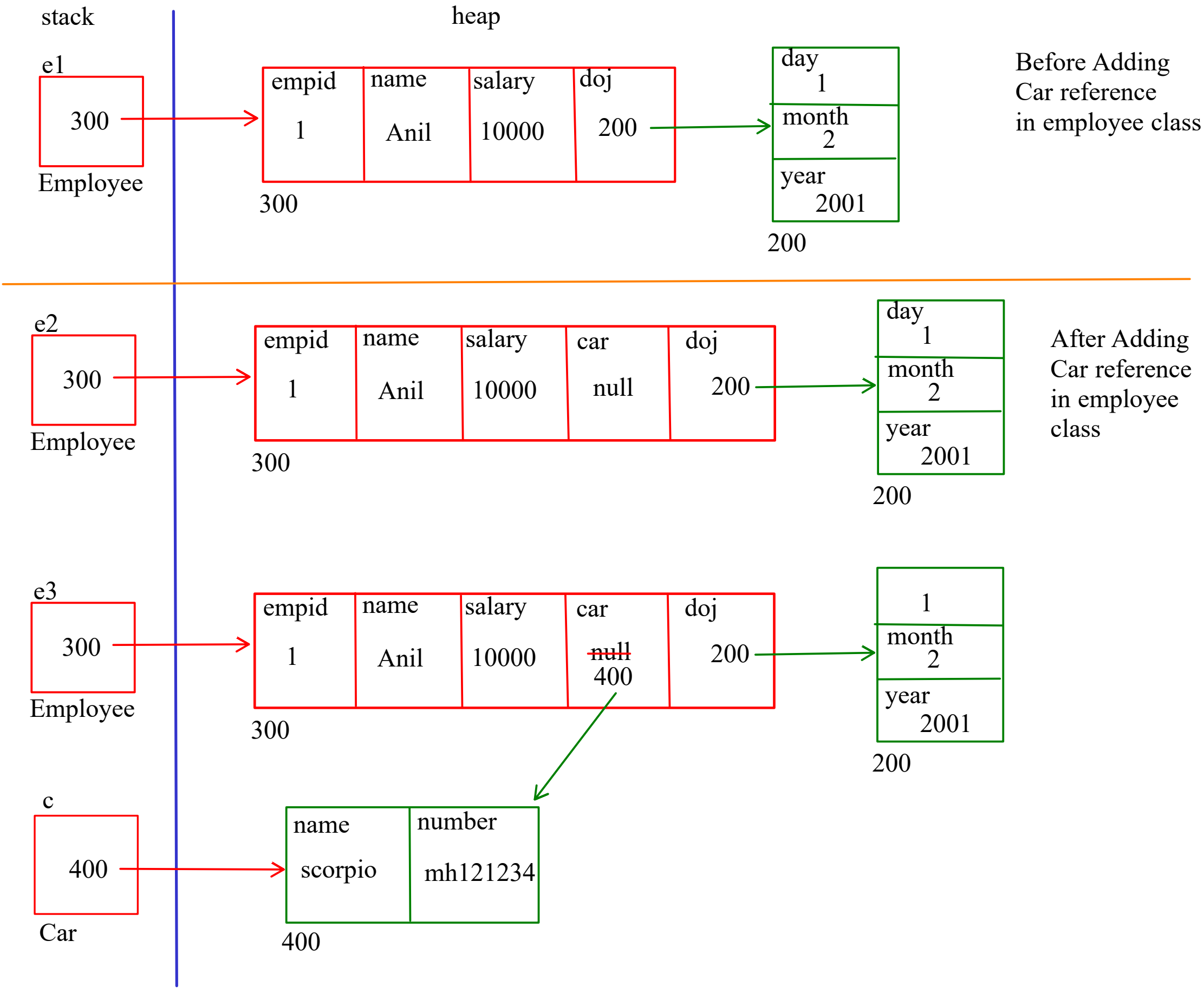
```
System{
PrintStream err;
PrintStream out;
}
```

has-a relationship -> Association
1. Composition
        - Entities are tightly coupled
2. Aggegration
        - Entities are loosely coupled

```
class Date{

}
```

```
class Employee{
Date doj;
Car c;
}
```



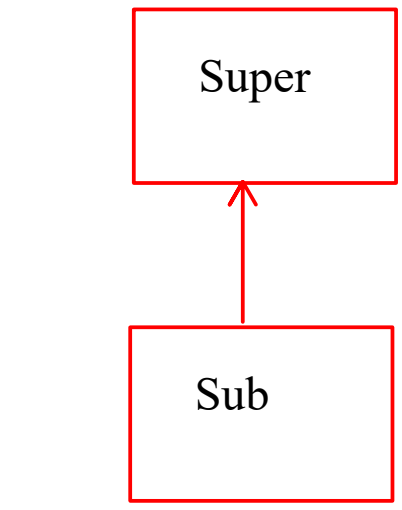


Inheritance (is-a)

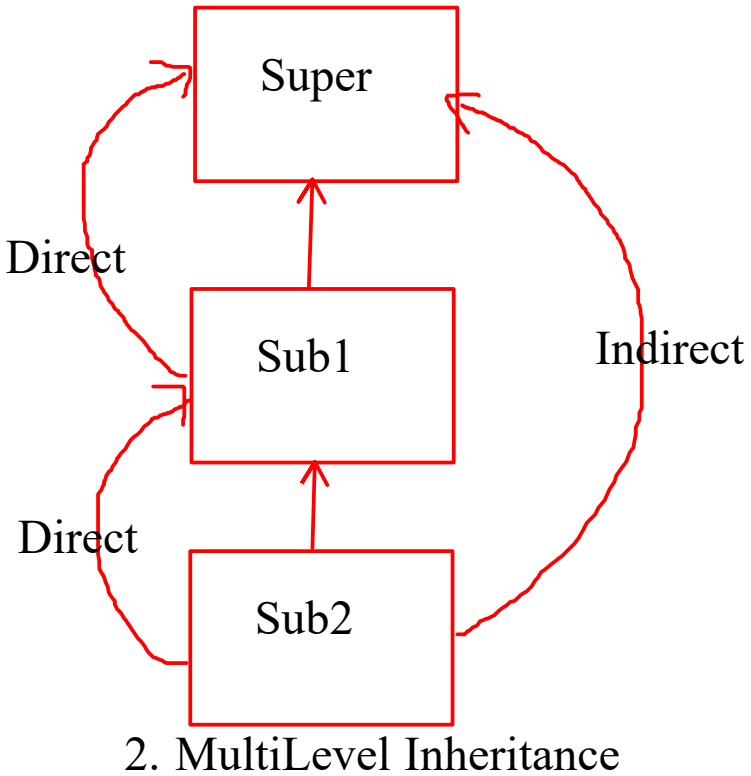
Parent -> Base (Super)
Child -> Derived (Sub)

```
class Parent
{
int n1;
}

class Child : public Parent
{
int n2;
}
```

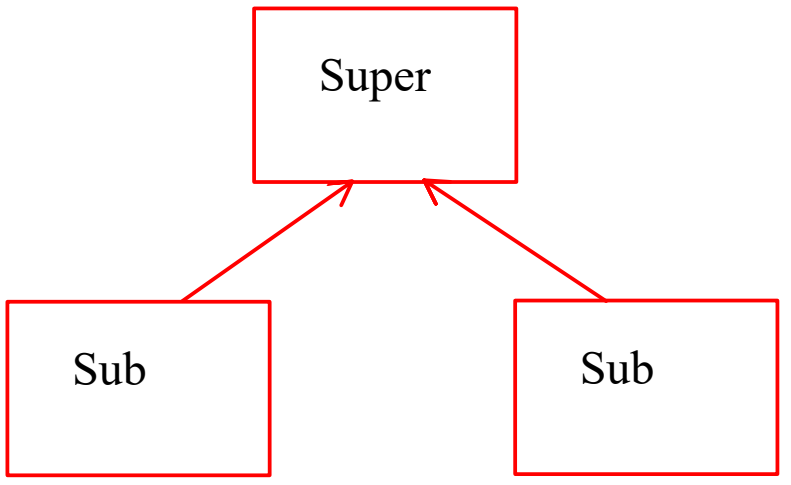
```
class Parent
{
int n1;
}

class Child extends Parent
{
int n2;
}
```



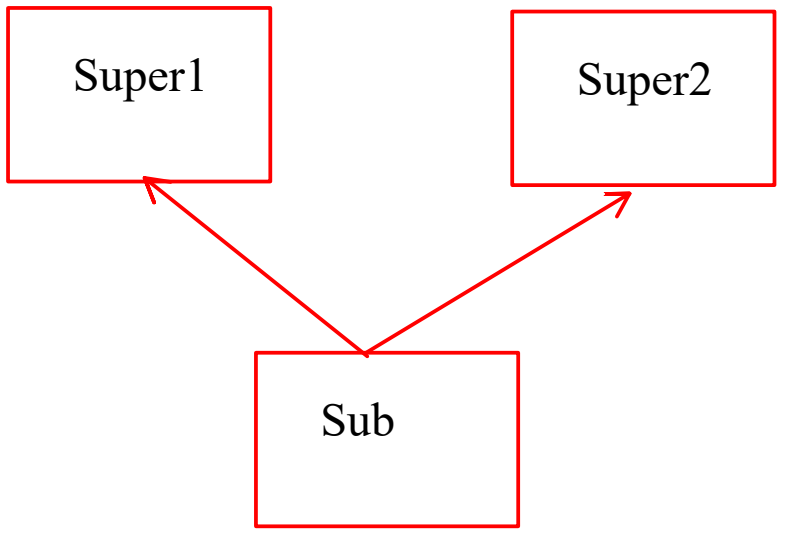
1. Single Inheritance



2. MultiLevel Inheritance



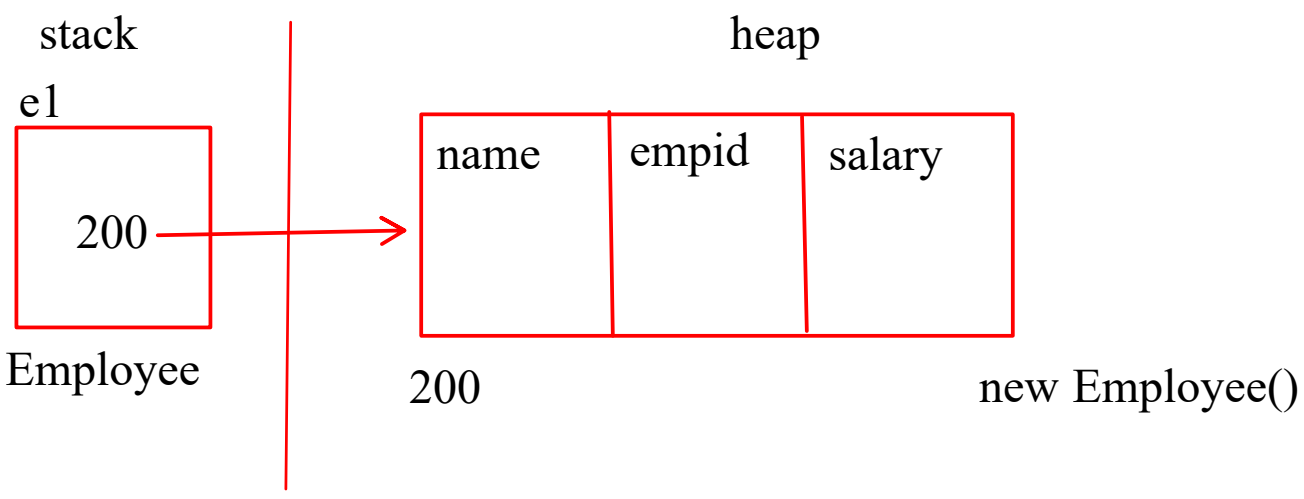
3. Hirerachical Inheritance



4. Multiple Inheritance

Java does not support Multiple class Inheritance
However It does support Multiple Interface Inheritance

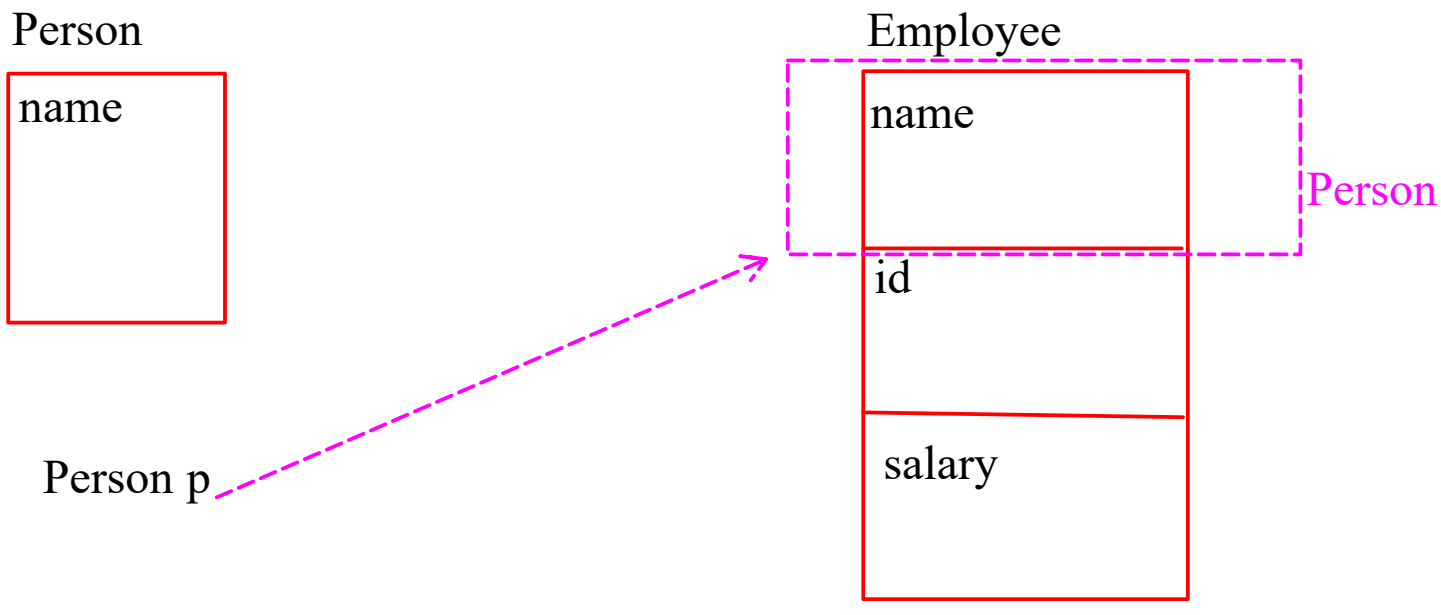
5. Hybrid Inheritance
   - Combination of any two different types of inheritance creates an hybrid inheritance



Method Overriding
- Redefining the methods of the super class once again into the sub class with same name and signature is called as method overriding
- It is an example of run time polymorphism
- Why to perform method overriding
    1. Implementation of super class method is 100% incomplete
    2. Implementation of the super class method is partial complete
    3. If we want implementation of the sub class method completly different than the super class method.

Rules of Method Overriding
1. Name and signature of the super class method should be kept same in sub class
2. Return type of overriden method should be same as that of super class method or it shoould be its sub type
3. Visibility modifer of the overriden subclass method should be same as that of super class method or it should be of wider type.
4. private methods and static methods are not designed for overriding

super keyword
- To call the parameterized ctor of the super class from the sub class ctor we use super();
- to unhide the implementation of the super class methods in the subclass we use super.



Early Binding
Late Binding

if(typeid(p)==typeid(Employee))

Product arr[] = new Product[3];

Lab work
complete the classwork
complete the pending assignemnt of point 2D
complete the assignemnt of Product, book and tape of cpp in java
complete the demo of shape, rectangle and circle