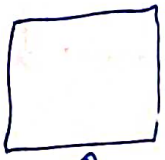
# Function Overriding and function overloading ***

Unit-2

## Obj & reference variable

- When we create obj, they act as reference variable

eg box obj1 = new box();
box obj2; // obj2 is a ref var. It doesn't point to any obj.
box ref; // ref is also a ref var.



obj1

(obj1 point to an obj)

obj2 → null;
ref → null;

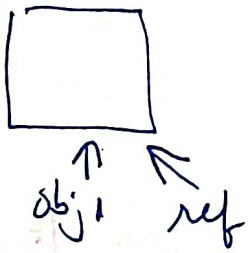
(obj2 & ref point to null they do not pt. to any obj)

## Ref var can point to other obj

- A ref can point to any other obj by using assignment operator.

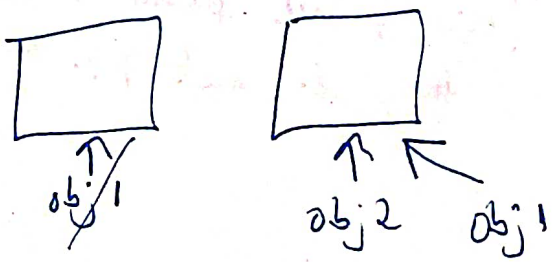
eg1

```
box obj1 = new box(); //box obj is created
box ref; // ref is created
ref = obj1; //ref now pt. to obj1
```




eg 2

```
box obj1 = new box(); // 2 obj r created
box obj2 = new box();
obj1 = obj2; // obj1 ref var now pt.
                to obj2
// obj2 is not copied, instead both
   now pt. to same obj
```

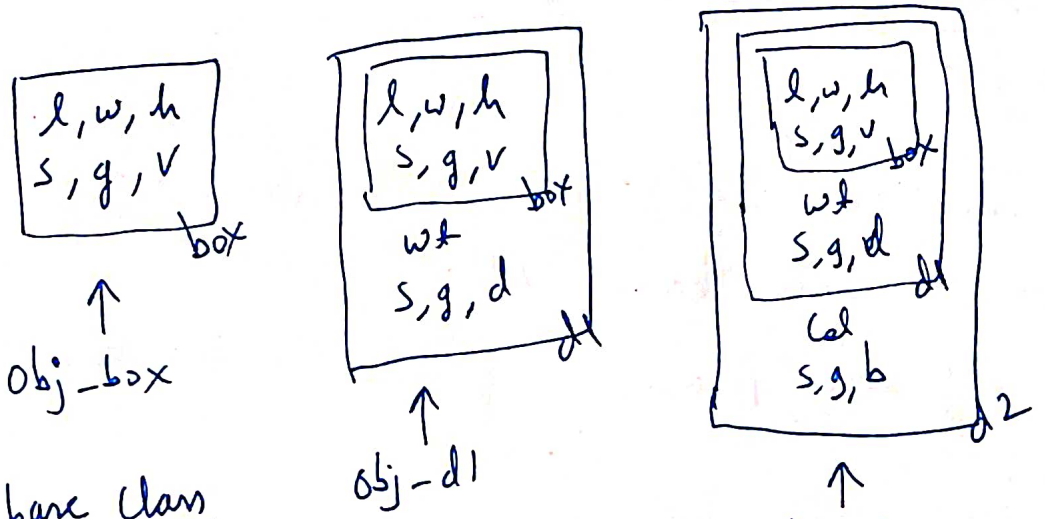


Derived class obj has a base part, but
base " " do not have a derived part.

box class :
var → l, w, h
fun → set, get, vol.

d1 class :
var → wt
fun → s, get, density
{ l, w, h ; set, get, vol } // there r inherited

d2 class :
var → col
fun → set, get, blackbox
{ l, w, h ; s, g, v ; wt ; s, g, den } // there r inherited

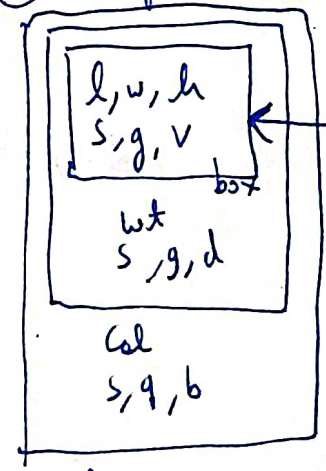


## base ref can pt. to a derived class obj

- A base ref can pt. to a derc obj, bcoz derc obj has a base part.
- But a derc ref can't pt. to a basec obj bcoz basec do not have derived part.
- when a basec ref pt. to a derc obj, it pts. to only the base part in that obj.
- A basec ref can't acces var or fun ovrride the base part in a derived obj.

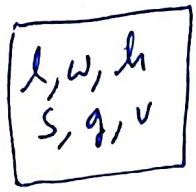
### How to point a base ref to derived class obj :-

(1) d2 obj_d2 = new d2(); // create a derc obj
(2) box ref; // create basec ref.
(3) ref = obj_box; // basec ref pts. to derclass obj.



eg-2:

```
box  obj_box = new box();  //create a base obj
d2 ref ;//create a ref of der class.
d2 = obj_box ;// ERROR!!  der ref can't pt.
                          to a base class obj.
```



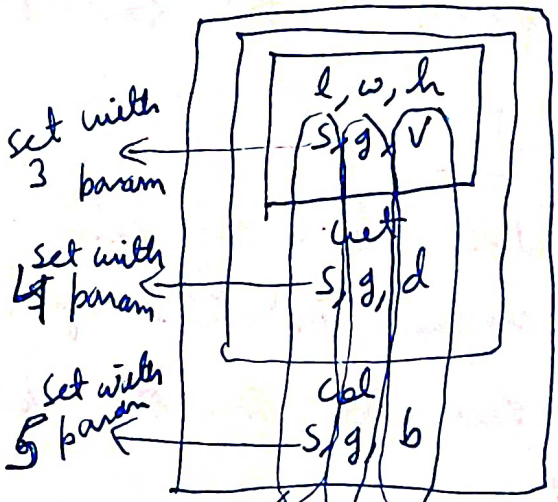

obj_box
(base obj)

ref
//This is a der class ref.
//It can't pt. to base obj, bcoz base doesn't have any derived part.

---

## Using Base Reference to call functions

- In the der obj, there r 3 types of fun
  - set is overloaded
  - get is overridden
  - val, den, blackbox r normal inherited fun.

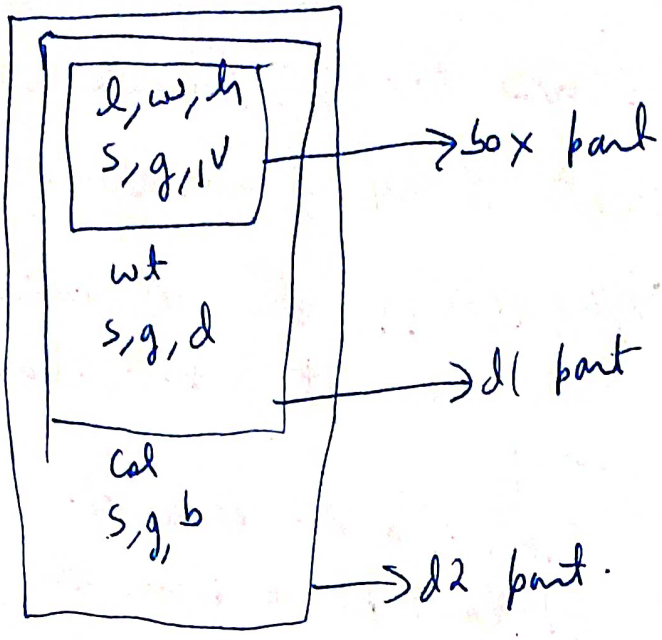



o/loaded

o/ridden

normal inherited fun.

d2_obj
(obj of der class)




A derived obj has ~~dyy~~ 3 base parts.

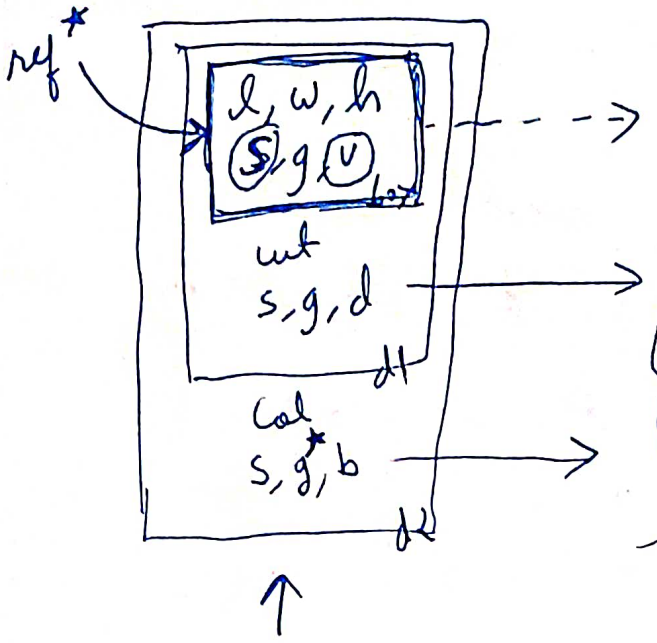
- A base ref can point to a derived obj
- when a base ref points to a der obj, it points to the base part in the obj.

Rule :-

- When base ref calls o/loaded fun (i.e set) or normal inherited fun (i.e vol, den, blackbox) then fun in the base part will be called

**

- But when base ref calls o/ridden fun (i.e get) then ~~fun~~ objects own o/Ridden fun will be called. Base's fun will not be called.




*get() will be called with bare ref, even though it is outside bare part.

---

Fun O/L$^g$ & fun O/R$^g$ in inh

overloading — overriding

Q) what is fun O/R$^g$?

A. fun O/R$^g$ is done in the context of inh
- A der class can define a fun w/ same name & param as in bare class.
- eg box class has a get fun & dl class defines another get fun w/ same name & param.

Q) What is fun o/L$^g$?

A) • o/L$^g$ means derC creates another fun w/ same name but diff param

eg box class has set with 3 param
d1 " " " " 4 "
d2 " " " " 5 "

• set is o/Loaded fun.

| o/L$^g$ | vs | o/R$^g$ |
|---|---|---|
| • derC defines a fun w/ same name but diff param | | • derC defines a fun w/ same name & same param. |
| • Can be done w/ or w/o inh | | • Can only be done w/ inh. |
| • implements compile time polymorphism | | • implements run time poly. |

***

## Dy. method dispatch, Dy. method resolution, Runtime poly, Late Binding, Dy. polymorphism

- Combination of base ref & O/R fun is used to implement run-time poly.
- When O/R fun in a derived obj is called using base ref, then RT poly occurs.
- Object's own O/R fun will be called, base's fun will not be called.

### What is runtime poly?

(1) When a base ref calls an O/R fun in derC, then that ~~obj's~~ objects own fun will be called. Base's fun will not be called.

(2) In this case, the fun call is resolved at runtime & not at compile time

### How is RT poly achieved

- By combination of base ref & O/R fun.
- When base ref calls O/Loaded fun, then compile time poly occurs.
- When base ref calls O/R fun, then runtime poly occurs.

eg1 ref.set(1,2,3); // set is overloaded fun.

bare ref + o/L fun // compile time poly occurs.

eg2 ref.get(); // get is o/R fun.

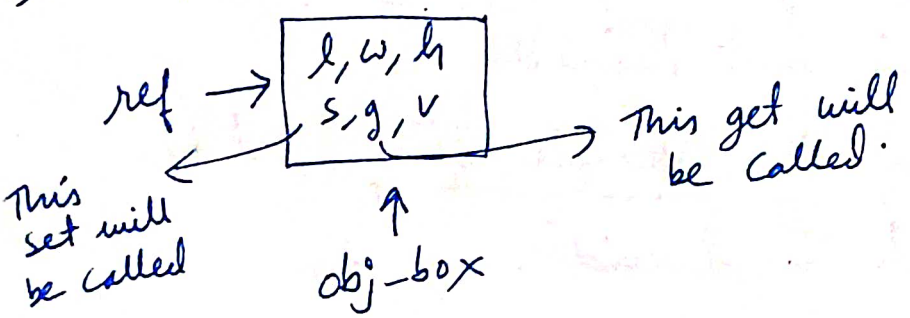
bare ref + o/R fun // runtime poly occurs

---

## What is the benefit of RT poly

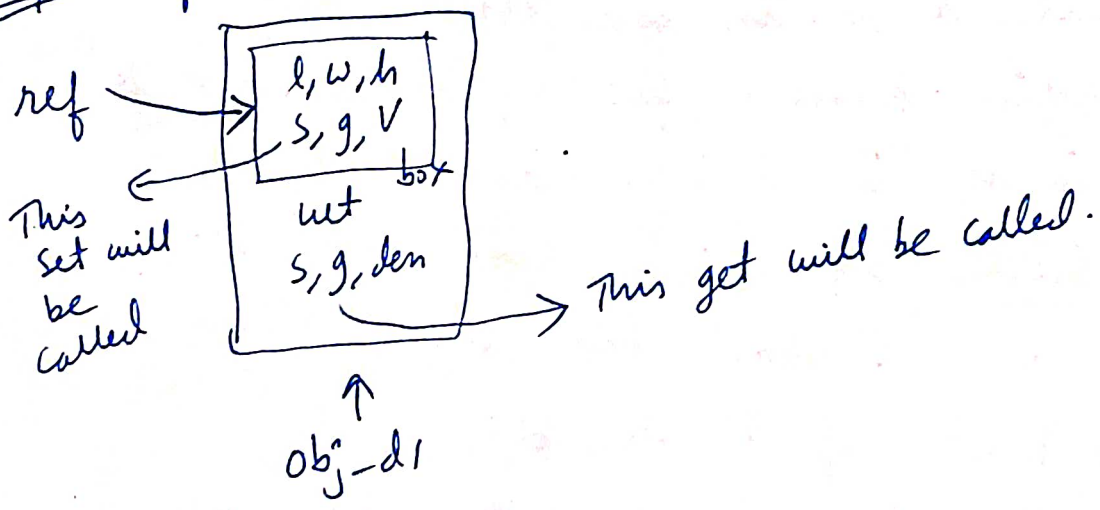
- A single ref can be used to point to any obj.
- For o/L fun, base's fun will be called. Object's own o/L fun can't be called using base ref.
- But for o/R fun, base's fun will not be called object's own fun will be called.
- Thus RT poly, allows to call correct fun of the obj thru base ref.

eg1 Ref pt. to box obj.



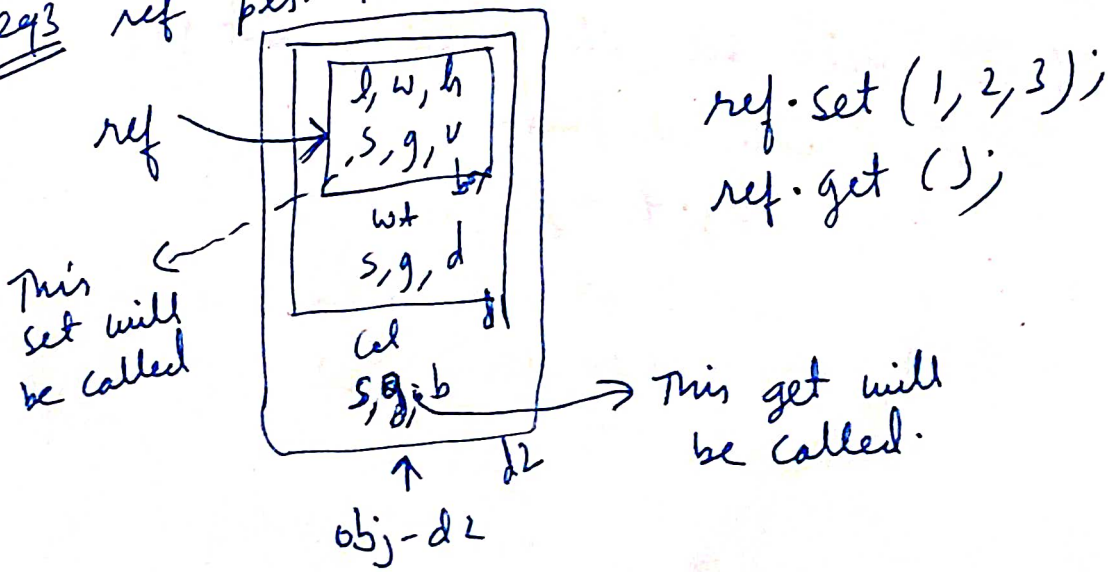

```
ref.set (1,2,3);
ref.get ();
```

eg2 ref points to d1 obj




```
ref.set (1,2,3);
ref.get ();
```

eg3 ref pts. to d2 obj




```
ref.set (1,2,3);
ref.get ();
```

- In all 3 cases, set of base fun will be called wh. is not desirable
but get() of the specific obj will be called wh. is desirable

---

WAP to show working of RT poly thru O/R fun

steps:-
1) Create a fun in base class
2) define another fun w/ same name & param in derived class.
3) Create obj of der class.
4) " a base class ref.
5) Point ref to der obj
6) Call the O/R fun thru base ref.

```
class box {  // base class
    protected int l, w, h;
    public void set(int x, int y, int z) {
        // set with 3 param
        l=x; w=y; h=z;
    }
```

```
public void get() {
    //get prints 3 var. This will be o/ridden by d1
    Sopln(l);
    Sopln(w);
    Sopln(h);
}
public void volume() {
    //Normal inherited fun.
    Sopln(l*w*h);
}
} //box ends
---------------------------------
class d1 extends box {
    protected ~~void~~
    protected int wt;
    public void set(int x, int y, int z, int u) {
    //~~overridden~~ ~~fun~~ overloaded fun.
    //This fun initializes 4 var.
        l=x; w=y; h=z; wt=u;
    }
    public void get() {
    //Overridden fun.
    //This fun prints 4 var.
        Sopln(l);
        Sopln(w);
        Sopln(h);
        Sopln(wt);
    }
    public void density() {
        Sopln( ~~l*w*h*1.0~~ wt*1.0/l*w*h);
    }
} //d1 ends.
```

```
class d2 extends d1 {
    //
    protected ~~int~~ ~~col~~ String col;
    public void set(int x, int y, int z,
                    int u, String p) {
        //o/loaded fun
        //This fun initializes 5 var.
        l=x; w=y; h=z; wt=u; col=p;
    }
    public void get(){
        //overridden fun
        //This fun prints 5 var
        Sopln(l);
          " (w);
            (h);
            (wt);
            (~~col~~);
    }
    public void blackbox(){
        col="black";
    }
} //d2 ends.
```

C ∅ { psvm }.

//Create obj of derC
d2 obj-d2 = new d2();

//Create a base ref.
box ref;

//Pt. ref to der obj
ref = obj-d2;

//Call O/L fun thru ref.
~~ref.get();~~
ref.get(1, 2, 3); //ref can call set with 3 param only.

//ref can't call set w/ 4 or 5 param.

//Call O/R fun thru ref.
ref.get(); // 5 var will be printed bcoz get() in der C prints 5 var.

//Runtime poly occurs.