Function Overriding

(unit-4)

- ~~You can use same fun name~~

O/L → overloading
O/R → overriding
param → parameter

- Same fun name can be used in 3 diffs ways in inh – 1) inherited fun
  2) O/Loaded fun
  3) O/Ridden fun.

1- Inherited Fun

$$\begin{tikzcd} \boxed{B} \arrow[d] & \text{inherited\_fun()} \arrow[d] \\ \boxed{D1} \arrow[d] & {''} \arrow[d] \\ \boxed{D2} & {''.} \end{tikzcd}$$

- Suppose a fun named inherited_fun() is created in base class.

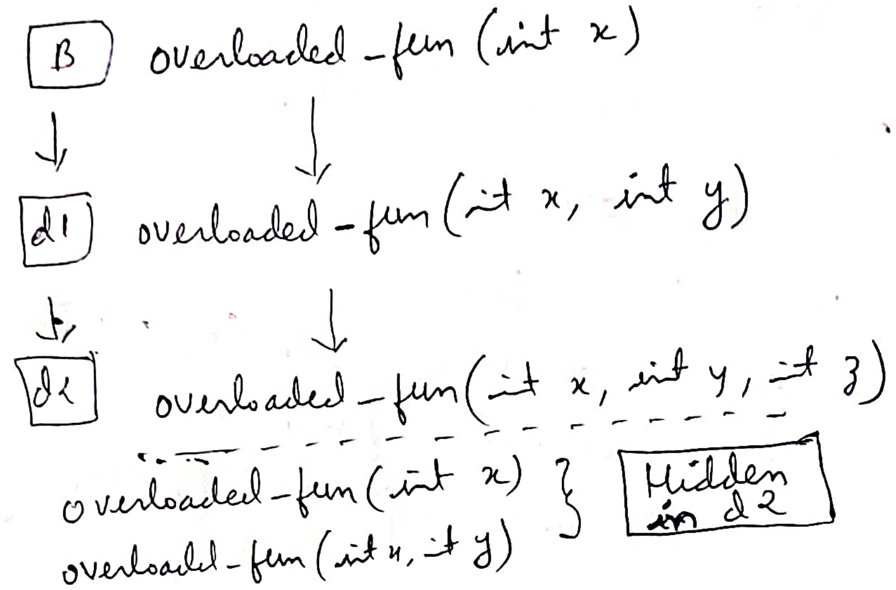
- This fun will not be created/~~redfun~~ redefined ~~in~~ again in derived class.

- This fun will be inherited by d2 class & obj of d2 class can call this fun.

- There is only one ~~copy~~ fun with name inherited_fun() in d2 class. So d2 obj can directly call this fun.

```
//Create obj of d2
      d2 obj_d2;
//Call the fun
      obj_d2.inherited_fun();
```

2- Fun overloading or overloaded fun


B
overloaded_fun (int x)
d1
overloaded_fun (int x, int y)
d2
overloaded_fun (int x, int y, int z)
overloaded_fun (int x)
overloaded_fun (int x, int y)
Hidden in d2

Note- ★★

- Any inherited fun with same name will be hidden in derived class.

- Eg:- • Suppose Base contains a fun named overloaded_fun ( ) with 1 param.

- If another overloaded_fun( ) is created / defined in d1 with 2 param, then
  overloaded_fun ( ) in base will be inherited by d1 but it will be hidden.

- Similarly if overloaded_fun( ) is created in d2 with 3 param, then
  d2 will contain 3 fun
  out of wh. 2 fun will be inherited
  & they will be hidden ★

• Hidden means that they r not visible to obj directly & class name should be speified to call those fun.

• Eg:- //Create obj of d2

```
d2 obj_d2;

//call these fun
// obj_d2.overloaded_fun(1);   } // Error these
// obj_d2.overloaded_fun(1,2); }    r hidden
obj_d2.overload_fun(1,2,3); // This will run.


obj_d2.base::overloaded_fun(1);
obj_d2.d1::overloaded_fun(1,2);
```

↓

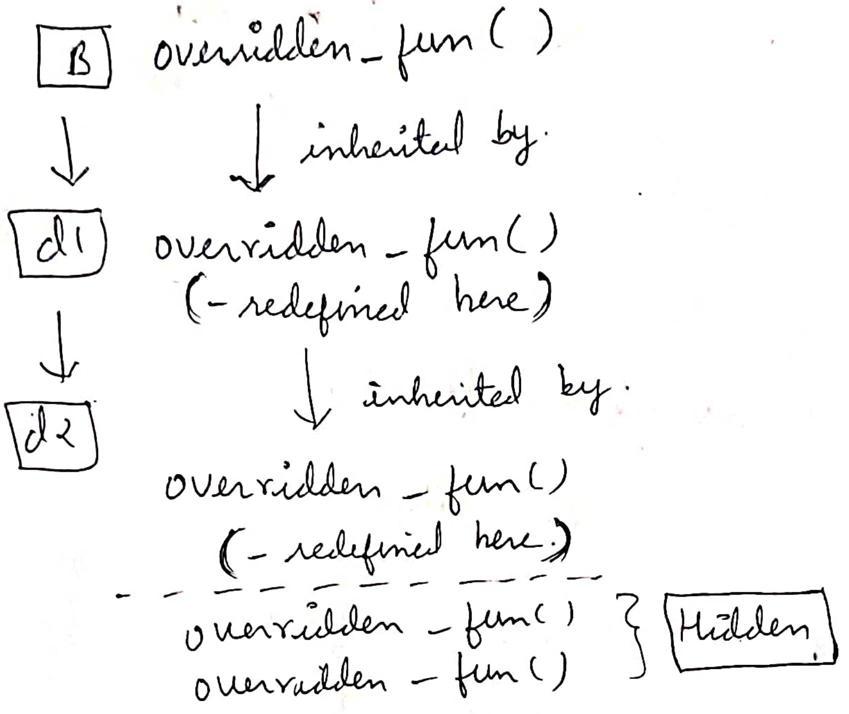
• hidden fun r accessed by using class name & scope resolution operator.

• Note ⋆⋆

• Even though hidden fun have diff No. of param, still class name needs to be speified to call these fun.

3 - Fun overriding or overridden fun

- Third way to use fun is through fun overriding.



- fun overriding means creating same fun in derived class with same no. of param
- Suppose there is a fun named overridden_fun() in base class.
- This fun is redefined in d1 & d2.
- overridden_fun is also inherited by d2 class
- So d2 will have 3 overridden_fun() with same no. of param.

- when obj of d2 calls overridden-fun then its own fun is called. overridden-fun() of B & d1 r hidden

- So d2's overridden-fun() hides d1 & base fun

- To access the hidden fun, class name & scope resolution opr needs to be used.

Eg d2 obj-d2;
obj-d2. overridden-fun();
↓
d2's fun called

obj-d2. base :: overridden-fun();
obj-d2. d1 :: overridden-fun();
↓
hidden fun r called.

- Fun O/R involves 2 conditions
  - Inh must be involved
    B, d1, d2 all must use same fun name.
  - fun param shud be same in all fun.

summary:-

- If you create a fun with same name in derived class, then base class fun will be hidden in derived class
  (~~whether~~ Applies to both, fun O/L and fun O/R ~~is done~~)

- To access hidden fun, use class name & ~~[illegible]~~ ::

Inherited fun	Overloaded fun	overridden fun.
• inh must be involved	• Can be done with, w/o inh	• inh must be involved
—	• No of param. shud be diff	• No of param shud be same.
• —	• O/L fun hidden in derived class	• O/R fun also hidden
• There is a single fun.	• There r mul fun with same name	• "
• "	• O/L fun r inherited by derived class	"

★ Whether fun r overloaded or overridden, they will be hidden in derived class.