Date / /
Page

~~INHERETENCE~~

'this' → 'this' always points to current object.

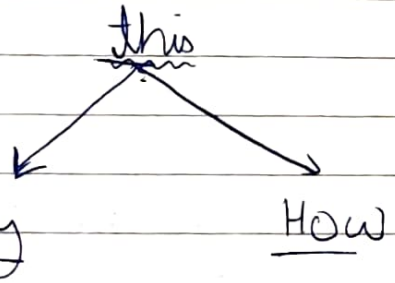
Def^n acc. to OOPs

Object being be used.

~~this else~~

Acc. to Java.

this always holds the Reference ID of current Object.



Eg

```
Class Temp
{
  int x=10;

  Void show(int y)
  {
   Sop(x);
   Sop(y);
  }

   psvm()
  { Temp t=new Temp();
      t.show(20); }}
```

★ Whenever a class level variable & local variable have same name, then this concept is known as 'Data Shadowing'.

Eg →

```
Class Temp
{
int x=10;

Void show(int x)
{
Sop(x);
Sop(x);
}
```

```
psvm()
{
Temp t=new Temp();
t.show(20);
}
}
```

Ans → 20,20

★ Priority always goes to Local Variable

Now

Using above prog. we want to print (10,20) using 'show' f^n only.

```
class Temp
{
int x=10;
Void show(int x, Temp z)
{
Sop(z.x);
Sop(x)
```

```
psvm()
{
Temp t = new Temp();
t.show(20,t);
Sop(t.x);
```

actual arg.

x=10

☆ GOLDEN RULE

1. The value of object (value at in its box) can only be accessed by using '.' of Ref. ID

2. Object is neither local, global, private, public. It always goes in 'Heap'.

Now,

```
Class Temp.
{
int x = 10;
void show(int y, Temp this)
{
Sop (this.x);
Sop (y);
}
```

```
psvm ()
{
Temp t1 = new Temp();
Sop (t1.x)
t1.show(20, t1);
```

* We can use 'this' or not in case of 'non-shadowing' as compiler

Date / /
Page

RULE

(1) 'this' can't be used in any static fn

eg-①

```
Class Temp
 {
  int x=10;

 Void show(int y)
  {
   Sop (this);
  }
```

```
psvm()
 {
Temp t1 = new Temp();
   Sop(t1);
t1.show(20);
```

Passed as argument to fn show

eg-②

```
class Temp
  {
    int x=10
 Void show (~~Ew~~ Temp this, int y)
 {
      Sop (x);
      Sop (this);
      Sop (y);
 }
```

2. If you pass 'this' as a argument in any non-static f^n then it must be the first argument.

Eg

```
class MyThis
{
int x = 10;

void get (MyThis this, int g)
  {
    Sop (this+"get");
    Sop (this.x);
    Sop (x);
  }

  psvm()
  {
  MyThis mt = new MyThis();
    Sop (mt+"main");
    mt.get (10);
  }
  }
```