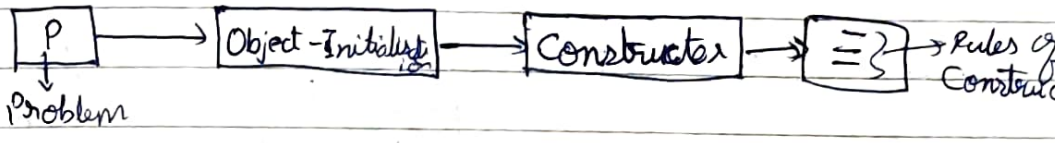
27/06/2017

CONSTRUCTOR

Constructer are spl. member f^n's of class which are used to initialise the object.



Object Initialisation (O-I)

Intial Value of Object → The value to properties given to object during its constr^n is known as Initial Value i.e. before
This process is known as Initialisation.

Def^n : General

Putting the values in properties of object while creating it. This is known as O-I.

Technical

Putting the values in to the Data members of a object while creating it.

```
Class Emp
{
int salary;

void get()
{
salary = 1500;
}
```

→ Normal f^n

Problems

1. It is created after creating object, any runs
2. It is called 4 times hence can be initialised any no. of times but it has to be initialised only once.
3. No guarantee it would run

Now, we have to alter the Normal f^n.

Also we can't write outside the f^n. a single line of code.

Alterations Made

1. We have to initialize at the given place, its not programmer dependent/wish.
   eg. So, defined place is:- new get()
   f^n
2. In above we see after 'new' we used 'get()' so we program not doesn't Know class name.
   So, Contructer name is changed to same as that of class

Example

```
Class Emp
{
int salary;
Emp(Emp this)
{
this.salary = 15000; (return this;)
```

→ implicitly passed.

RID

```
Emp
e = new Emp()
```

Java allows blank return;

3. There is no 'return type'.

Prog that constructor returns 'this'

(1)

```
class Emp.
{
 int salary;
 Emp()
 {
  salary = 15000;
 }
 Void show()
 {
  Sop(salary);
 }
```

```
psvm()
{
Sop(new Emp().salary);
```

Anonymous object - as it can't be reused. becomes unreachable

```
Sop(new Emp().show());
}
```

(2)

```
Class Emp
{
int salary;

Emp()
{
Sop (this);
}
```

```
psvm()
{
Emp e1 = new Emp();
  Sop (e1);
}
}
```

★ Constructor is helping out 'new' operator to construct a new object, hence the name 'constructor'.

Date / /
Page

Cor

Constructor Def Acc to ~~Byg~~ Program :-

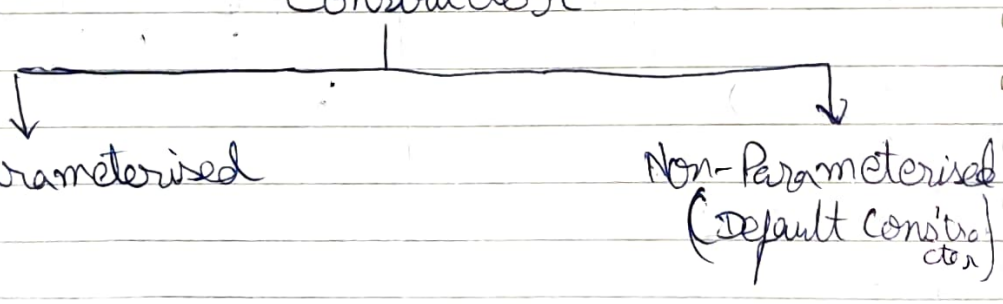
(i) If any task you want to perform only once in the life cycle of an object, then put the code of that task into the constructor.

(ii) If you want to provides the certain resources to each object of class while creating it then put the code of these resources, into the constructor.

28/6/2017

☆(i) Constructors are by nature static in Java.

(ii) It's a spl static fn. As this is only static fn that uses 'this'.



Date / /
Page

Non-parameterised:

```
class Temp
{
void show()
{
sop ("show");
}

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

Que → Since there is no Temp() constructor in prog. but it runs? Why?
Ans → Compiler inserts a Non-parameterised constructor on its own when class has no constructor.

- No Class can exist without any Constructor
- **Rule**
  If there is no constructor in a class then one non-parameterised Constructor (Default) is inserted into the class by the Compiler.

  Proof (PTO)

Date / /

Page

Compilation error as there is two cond' one para

```
Class Temp
{
Void show()
{
Sop ("show");
}
Temp(int x)
{
Sop(x);
}
```

Parameterised Constructor

```
Psvm
{
Temp t = New Temp();
t.show();
}
}
```

It can be corrected by

```
Temp()
{
Sop("default");
}
```

```
class Temp
{
Temp int x;
     int y;

Temp()
{
x=50;
y=100;
}

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

```
psvm
{
Temp t1 = new Temp()
t1.show();
Temp t2 = new Temp()
t2.show();
}
}
```

x = 50
y = 100

50
100

- If you want to initialise the datam member of each object with same values, then always use default constructer, and this concept is known as static initialisation of non-static data members.

- Non-static data meter members can also be initialised at class level.

```
class Temp()
{
int x=50;
int y=100;

void show()
{
Sop(x);
Sop(y);
}
```

psvm

After compilation above changes to

```
class Temp()
{
int x;
int y;

Temp()
{
x=50;
y=100;
}

psvm
```

» If more constructors are present compiler inserts initial values in all constructors.

Date / /
Page

- In Java methods can be called at class level.

eg

```
Class Temp
{
  int x = get();   ---> this.get() implicitly.

  int get()
  {
    Sop ("get");
    return 10;
  }
  Temp()
  {
    Sop("Constru");
    Sop(x);
  }
```

```
psvm()
{
  new Temp();
```

- In Java we can also call the method at class level to initialise the data methods of class.
  If the Data Member is a static then methods must be ' '.
  If Data Mem are non-stat then methods must be ' ' '

⇒ Compiler does this → it

```
→ class temp
  { int x;
    :
    Temp()
      { x = get(); }
```

```
Sop(x);
}
```

Date / /
Page

Practice

```
class B
{
    int x = this.getX();

int getX()
 {
 Sop(x);
 Sop("via get function");
 return 10;
 }

  B()
{
Sop(this.x + " via constructor");
}

PSVM()
{
new B();
}
}
```

Date / /
Page

Example

```
Class Temp{
{
int x;
int y;

Temp (int x, int y)
  {
  this.x = x;
  this.y = y;
  }

void show()
  {
  Sop(x);
  Sop(y);
  }

psvm()
{
```

```
psvm()
        RID
Temp t1 = new Temp(10,20)
t1.show();   x=10 y=20
        RID      x=100 y=200
Temp t2 = new Temp(100,200)
t2.show()

}
}
```

- If you want to initialise data members of each object with a diff. diff. values then always use Parameterised Constructer, and this concept is known as Dynamic initialisation of non-static data members

29/06/2017

```
class Temp
{
Temp( )
{
Sop("default");
}
void Temp()
 {
 Sop ("Hello");
}
psvm( )
{
new Temp( ); new Temp().Temp();
}
}
```

This should be compilation error :-

(i) There is a return type of ~~Temp~~ constructor.

(ii) There are two '~~Temps~~' fn's of same name - 'Temp'

But there is no error.

★ Only Java allows fn. with same name of class.

Here One is constructor of

Rule

- In Java we can also make a method with the same name as class name.
- Java check with 'program' only that methods which have no return type if the method is with new & the method which is after '.' should have a return Type.

COPY CONSTRUCTOR [& PARAMETERISED]

Copy Constructor is used to create the duplicate copy of some existing object. Its like a photostate machine.

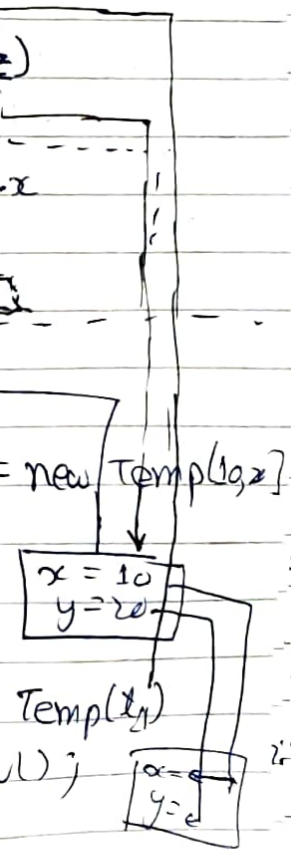
Eg.

```
Class Temp
{
 int x;
 int y;

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

```
Temp (Temp z)
{
 this.x = z.x
 this.y = z.y
 z.x = 1000
 psvm( )
Temp {
   Temp t1 = new Temp(10,20)
  t1.show();

Temp t2 = new Temp(t1)
     t2.show();
 }
}
```



Call by Value

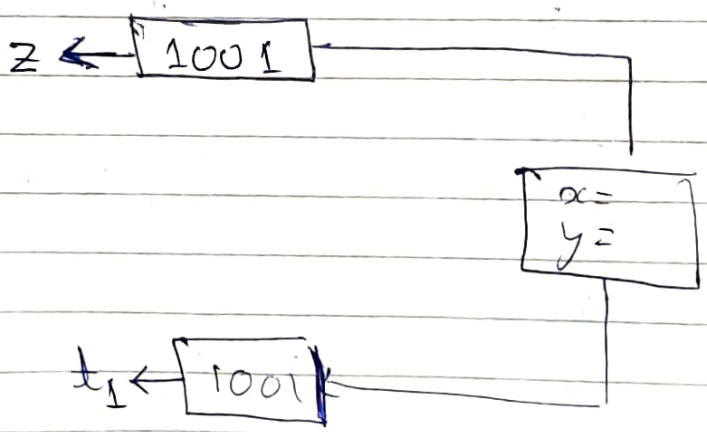
Whenever you are passing actual value of variable in any f^n when calling it, is known as call by value

Call by Ref.

Whenever you are passing the address of any variable as a argument in any f^n, while calling it then it is known as call by Ref.

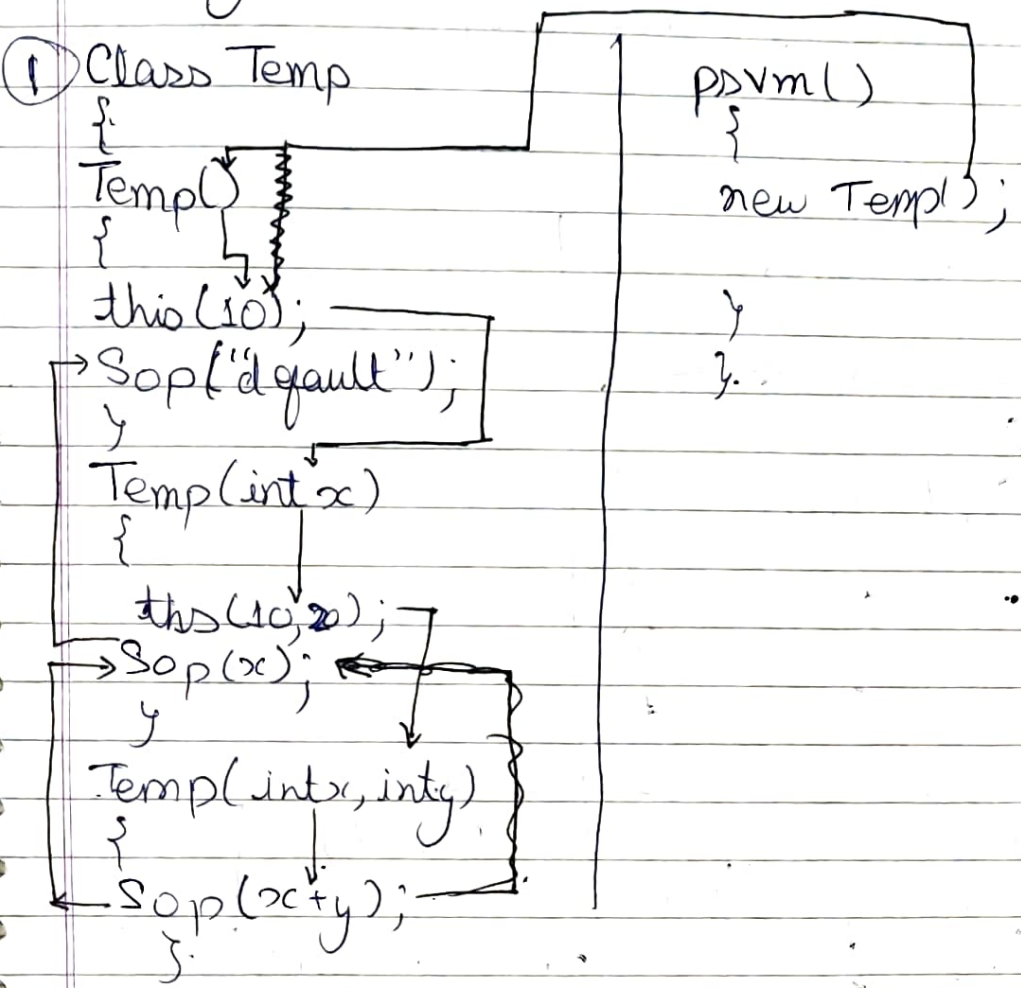
☆ Java doesn't support call by Reference.

Eg. → In above prgm.



CONSTRUCTOR CHAINING

Calling of one constructor from another constructor with respect to a current object is known a Constructor Chaining.



RULES

- Whenever you are doing a constructor chaining using 'this()' then it must be the 1st line in any constructor.
- Whever you are doing a constructer chaining using a 'this()' there must be.

at least 1 constructer which is not having a 'this()' as a 1st argume-nt.

- constructer ~~ch~~ chaining can be achived in ~~any~~ any order.

Eg:

(2)
```
class Temp
{
Temp()
{
Sop("default");
}
Temp(int x)
{
this();
Sop(x);
}
Temp(int, int y)
{
this(10);
Sop(x+y)
}
psvm()
{
new Temp(10,20);
}
}
```

→ Here we could have written 'Temp(10)' instead of 'this(10)'. But why not?

Ans → Because 'Temp(10)' could have pointed to any Temp named fn

Date / /
Page

- Here we have used 'this()' because we have used only 1 object ~~on~~ & using other keyword could have confused Progmer & here only 1 obj is there & this points to that 'current'-'One' object.

- Whenever you want to create so many resources in a single constructor, rather than putting code of each resource in a single contructor, create a separate constructor for each resource & then make their chain.

- We can also create this chaining when ~~con~~ a result of one constructor depends on result of another ".