Constructors

→ class Test
{
1. Variables — Local / Instance / Static
2. Methods — Instance / Static
3. Constructors
4. Instance blocks
5. Static Blocks
}

→ Using 'new' Keyword to create object

```
Test    t   = new   Test();
 ↑      ↑      ↑     ↑
Classname  Reference Variable (object Name)   Keyword   Constructor
```

→ Rules to declare constructor in JAVA :
(1) Constructor Name & class name must be same.
(2) Constructor able to take parameters
(3) Constructor not allowed return types.

Eg.
```
class Test
{
 void m1()
 {
  sop("m1");
 }
 Public static void main(string arg[])
 {
  Test t= new Test();
  t.m1();
 }
}
```

~~This keyword~~

- After compilation of above program, JAVA compiler generate one default constructor with empty implementn.

→
```
class Test
{
  void m1()
  {
    sop("m1");
  }
 /* Test class default Constructor

   Test()
    {
      // empty implementation
    }
 */
  public static void main (String args[])
   {
      Test t = new Test();
      t.m1();
   }
}
```

→ Types of Constructors:
- Default Constructor (zero Argument constructor)
- User defined Constructor
  (zero argument, Parameterized Constructor)

Note:
Default constructor always generated by compiler at Compile Time & Executed by JVM at RunTime

Eg. User defined Constructor

```
class Test
{
 void m1()
 {
  sop("m1");
 }
 Test()
 {
   sop("zero arg constructor");
 }
 Test(int a)
 {
   sop(" One arg Constructor");
 }
 public static void main (string args[])
 {
    Test t= new Test();
    Test t1= new Test(10);
    t.m1();
    t1.m1();
 }
}
```

<u>Output</u>:

```
zero arg Constructor
One arg Constructor
m1
m1
```

②
```
class Test
{
    Test (int a)
    {
        sop (" 1-arg constructor");
    }
    public static void main(String args[])
    {
        Test t = new Test(); ⟶ compiler error
        Test t1 = new Test(10);
    }
}
```

Output:
Compilation Error

Note:
- Inside the class if we are not declaring atleast one constructor then default constructor is generated by compiler.

- If we declare atleast one constructor then default constructor is not generated.

→ Advantage of constructors :
(1) Use to initialize instance variables.

Case I:

Problem: default values are printed even object is created.

```
class Emp
{
        int eid;    // instance variables
        string ename;

        Void disp()
        {
            sop ("eid =" + eid);
            sop ("ename = " + ename);
        }

        public static Void main (string args[])
        {
            Emp e = new Emp();
            e. disp();
        }
}
```

Output:
```
eid = 0
ename = null
```

Case II:

- To overcome above problem, during object creation we are 'initializing values.

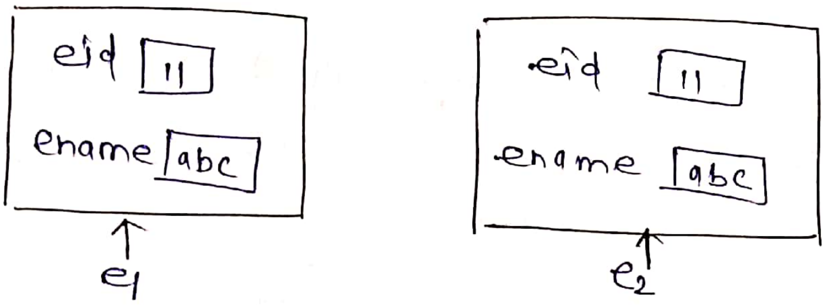
```
class Emp
{
    int eid;
    String ename;
    Emp ()
    {
        eid = 11;
        ename = "abc";
    }
    Void disp ()
    {
        sop ("eid =" + eid);
        Sop ("ename=" + ename);
    }
public static void main (String args[])
{
    Emp e=new Emp ();
    e.disp ();
  }
}
```

Output:
```
eid=11
ename=abc
```

Problem : For multiple objects same value initialized.

Eg.

```
public static void main (String args [])
{
    Emp e1 =new Emp ();
    e1.disp();
    Emp e2=new Emp ();
    e2.disp();
}
```



case III : To overcome above ~~problem~~ i.e Every object should have different value. We go for parameterized constructor.

```
class Emp
{
    int eid;        // instance variable
    string ename;

    Emp (int eid, string ename) // Local Variable
    {
    // conversion of local variables to instance Variable
            this.eid = eid;
            this.ename = ename;
    }
    void disp()
    {
            Sop ("eid ="+eid);
            Sop ("ename="+ename);
    }
    public static void main (string args [])
    {
        Emp e1= new Emp (1,"abc");
        Emp e2 = new Emp (2, "xy2");
        e1. disp();
        e2. disp();
    }
}
```

Output:

```
eid=1
ename=abc
eid=2
ename=xy2
```

Constructor Calling

Case I :

```
class Test
{
  Test()
   {
     sop(" 0-arg constuctor");
   }

   Test(int a)
   {
     sop(" 1-arg constructor");
   }
    Test (int a, float c)
    {
      sop("2-arg constructor");
    }
    public static void main(String args[])
    {
      Test t1 = new Test();
      Test t2 = new Test(10);
      Test t3 = new Test(10, 4.5);
    }
}
```

Output :

```
0-arg Constructor
1-arg constructor
2-arg constructor
```

Case II : Use "this" keyword

```
class Test
{
   Test()
   {
      this (10);
      sop(" 0-arg constructor");
   }
```

Scanned by CamScanner

```
Test(int a)
 {
  this (10,20);
  sop("1-arg constructor");
 }
 Test (int a, int b)
 {
   sop ("2-arg Constructor");
 }
 public static void main (string args [])
 {
  Test t= new Test();
 }
}
```

Output:

```
2-arg constructor
1-arg constructor
0-arg Constructor
```

Note :
this must be first statement in constructor. Inside the method this can place anywhere.

Eg.

```
class Test
 {
   Test()
   { this (10);
     this (10,20); → C.T error
   }
   ==
 }
```

One constructor is able to call one constructor at a time.