

MEMBER FUNCTION - this

Passing Parameters

```
public class Emp
{
    int eid;
    float esal;
    Emp(int x, float y,
    {
        eid = x;
        esal = y;
    }
    void disp()
    {
        System.out.println("Emp id : =" +eid)
        System.out.println("Emp salary : =" +esal;
    }
    public static void main(String args[])
    {
        Emp e1 = new Emp(11,10678.77f);
        e1.disp();
        Emp e2 = new Emp(12,20678.77f);
        e2.disp();
    }
}
```

The diagram illustrates the flow of data in the provided Java code. It features two orange-bordered boxes at the top right: one labeled 'eid' containing the value '12', and another labeled 'esal' containing the values '20678.7' and '7' on separate lines. Two large, curved blue arrows with green outlines originate from these boxes. One arrow starts from the 'eid' box and points to the 'int x' parameter in the 'Emp' constructor call 'new Emp(11, 10678.77f)' within the 'main' method. The other arrow starts from the 'esal' box and points to the 'float y' parameter in the same constructor call. A third, similar curved blue arrow with a green outline starts from the 'esal' box and points to the 'float esal' parameter in the 'disp()' method call 'e2.disp()'. These arrows visually represent how the values are passed from the main method to the object's constructor and then to its methods.

Output:

```
Emp id : =11
Emp salary : =10678.77
Emp id : =12
Emp salary=20678.77
```

Passing Parameter

```
public class Emp
{
    int eid;
    float esal;
    Emp(int eid, float esal)
    {
        this->eid = eid;
        this->esal = esal;
    }
    void disp()
    {
        System.out.println("Emp id : =" + eid);
        System.out.println("Emp salary : =" + esal);
    }
    public static void main(String args[])
    {
        Emp e1 = new Emp(11, 10678.77f);
        e1.disp();
        Emp e2 = new Emp(12, 20678.77f);
        e2.disp();
    }
}
```

11

10.678.77

Output:

Emp id : =11

Emp salary : =10678.77

Emp id : =12

Emp salary=20678.77

Constructor Calling

```
public class Test
{
    Test()
    {
        this(10);
        System.out.println("0-arg");
    }
    Test(int a)
    {
        System.out.println("1-arg");
    }
    public static void main(String args[])
    {
        Test t = new Test();
    }
}
```

The diagram illustrates the execution flow of the code. Red arrows indicate the following sequence of calls: 1. From the `main` method to the no-argument constructor `Test()`. 2. From `Test()` to the one-argument constructor `Test(int a)`. 3. From `Test(int a)` back to the no-argument constructor `Test()`. 4. From the `new Test()` statement in `main` to the no-argument constructor `Test()`.

Output:
1-arg
0-arg

Constructor Calling

```
public class Test  
{
```

```
    Test()  
    {
```

```
        System.out.println("0-arg");  
    }  
}
```

BUG!!

Output:

error: call to this must be first statement in constructor

this(10);

^1 error

```
public static void main(String args[])  
{
```

```
    Test t = new Test();  
}
```



Constructor calling

```
public class Test
{
    Test()
    {
        this(10);
        this(30);
        System.out.println("0-arg");
    }
    Test(int a)
    {
        System.out.println("1-arg");
    }
    public static void main(String args[])
    {
        Test t = new Test();
    }
}
```

Error!! Multiple this statement throws error

To invoke current class method:

```
class A
{
    void m()
    {
        System.out.println("hello m");
    }
    void n()
    {
        System.out.println("hello n");
        //m(); //same as this.m()
        this.m();
    }
}

public class TestThis4
{
    public static void main(String args[])
    {
        A a=new A();
        a.n();
    }
}
```

Output: hello n hello m



Thank you

