static

- static member is defined for the class itself & exist independently of any object of that class
- We can declare both methods and variables to be static.
- To create such a member, precede its declaration with the keyword static.
- To call static member use the following syntax;

```
clsName.varName;
clsName.methodName(args);
```

Static variable :

Only one copy of variable is created.

For static variable,

int is initialized to 0,

float to 0.0,

boolean initialized to false,

String to null

```
class   FirstProgram
{
         static int i;
         static float f;
        static char c;
        static String s;
        static boolean b;
       public static void  main(String args[])
      {      System.out.println("int i="+i);
             System.out.println("float f="+f);
             System.out.println("char c="+c);
             System.out.println("string s="+s);
             System.out.println("boolean b="+b);
     }
}
```

OUTPUT :

```
 int i= 0
float f= 0.0
char c=
string s= null
boolean b= false
```

static method :

The Math class provides several static methods some are

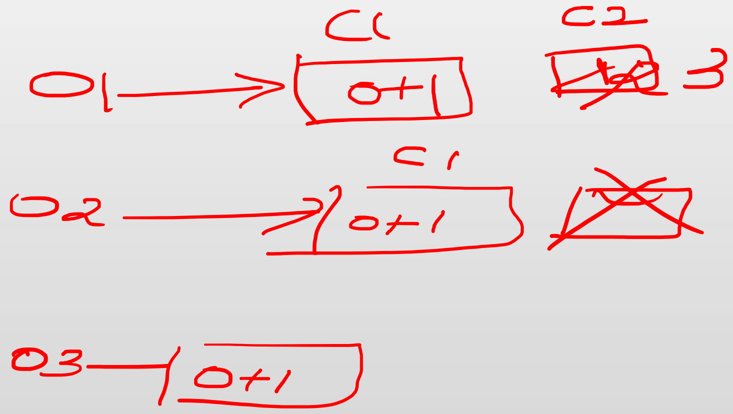
```
static int max(int I, int j);
static int min(int I, int j);
static double pow(double x, double y);
static double sqrt(double d);
static int abs(int i);
```

```
Class  prg
{
        public static void main(String args[])
        {
                System.out.println(" largest : "+Math.max(5,10));
                System.out.println(" smallest : "+Math.min(5,10));
                System.out.println(" power : "+Math.pow(2,3));
        }
}
```

```
largest : 10
 smallest : 5
 power : 8.0
```

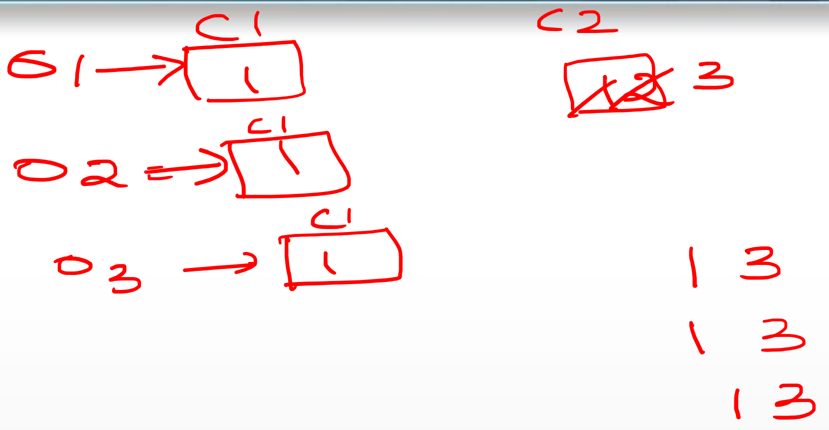
```
class StaticDemo
 {    static int  a = 42;
              static int  b = 99;
              static void callme()
              {          System.out.println("a = " + a);
              }
}
class StaticByName
{            public static void main(String args[])
              {
                            StaticDemo.callme();
                            System.out.println("b = " + StaticDemo.b);
              }
}
```

```
class Counter
{           int c1=0;
            static int c2;
           Counter()
           {            c1++;
                        c2++;     }
           void disp()
           {             System.out.print("First Counter : "+c1);
                         System.out.println("   Second Counter : "+c2);   }
}
class prg3
{          public static void main(String args[])
           {            Counter  o1 = new Counter();
                        o1.disp();
                        Counter  o2 = new Counter();
                        o2.disp()
                        Counter  o3 = new Counter();
                         o3.disp();
           }}
```



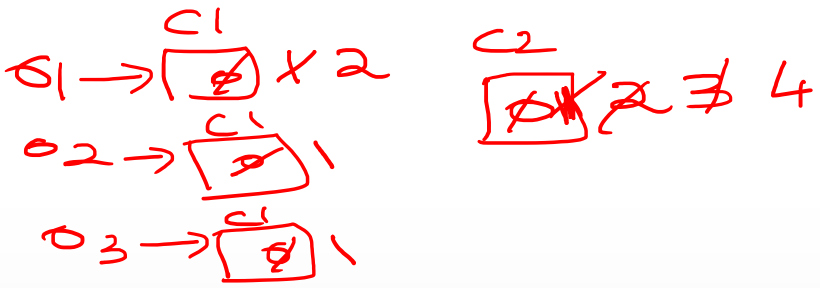
```
First Counter : 1   Second Counter : 1
First Counter : 1   Second Counter : 2
First Counter : 1   Second Counter : 3
```

```
class Counter
{           int c1=0;
            static int c2;
            Counter()
           {            c1++;
                        c2++;    }
           void disp()
           {            System.out.print("First Counter : "+c1);
                        System.out.println("   Second Counter : "+c2);    }
           }
class prg3
{          public static void main(String args[])
           {             Counter  o1 = new Counter();
                          Counter  o2 = new Counter();
                          Counter  o3 = new Counter();
                          o1.disp();
                           o2.disp();
                           03.disp();
            }}
```



```
First Counter : 1   Second Counter : 3
First Counter : 1   Second Counter : 3
First Counter : 1   Second Counter : 3
```

```
class Counter
{           int c1=0;
            static int c2;
           void increment()
           {              c1++;
                          c2++;  }
           void disp()
           {              System.out.println("First Counter : "+c1+"   Second Counter : "+c2);    }
}
class prg4
{          public static void main(String args[])
           {              Counter  o1 = new Counter();
                           Counter  o2 = new Counter();
                          Counter  o3 = new Counter();
                           o1.increment();
                           o1.increment();
                           o2.increment();
                           o3.increment();
                          o1.disp();   o2.disp();   o3.disp();        }   }
```



```
First Counter : 2   Second Counter : 4
First Counter : 1   Second Counter : 4
First Counter : 1   Second Counter : 4
```

Methods declared as static have following restrictions:

- They can only directly call other static methods.
- They can only directly access static data.
- They cannot refer to this or super.

```
class A {
static void v1( )
                { }
static void v2(){}
public static void main(
  {
   v1( );
   v2( );      }
```

~~A a = new A~~

~~a.nonstatic~~

CLASSES AND OBJECTS

When we require to do **computation** in order to **initialize static variables,** we can declare a **static block** that gets executed **exactly once,** when the **class is first loaded.**

```
class UseStatic
 {          static int a = 3;
            static int b;
            static void meth(int  x)
            {           System.out.println("x  = " + x);
                        System.out.println("a  = " + a);
                        System.out.println("b  = " + b);
            }
            static {
                        System.out.println("Static  block initialized.");
                        b = a * 4;
                  }
            public  static void main(String  args[])
            {           meth(42);
            }
  }
```

```
class UseStatic
 {          static int a = 3;
            static int b;
            static void meth(int  x)
            {           System.out.println("x  = " + x);
                        System.out.println("a  = " + a);
                        System.out.println("b  = " + b);
            }
            static {
                        System.out.println("Static  block initialized.");
                        b = a * 4;
                  }
            public  static void main(String  args[])
            {           meth(42);
            }
  }
```

```
Static block initialized.
x = 42
a = 3
b = 12
```

```
Class X
{           static int array[];
            static {
                  array = new int[6];
                  for(int i=0;i<6;i++)
                  array[i] = i;
            }
}
class prg
{              public  static void main(String  args[])
               {
                         for(int i=0;i<6;i++)
                         System.out.println(X.array[i]);
            }
}
```

```
Class X
{           static int array[];
            static {
                  array = new int[6];
                  for(int i=0;i<6;i++)
                  array[i] = i;
            }
}
class prg
{              public  static void main(String  args[])
               {
                          for(int i=0;i<6;i++)
                          System.out.println(X.array[i]);
            }
}
```

OUTPUT : 0 1 2 3 4 5

Nested Class

Nested class: class within another class.

The scope of a nested class is bounded by scope of enclosing class.

If class B is defined within class A, then B does not exist independently of A.

Nested class has access to the members, including the private members, of the class in which it is nested.

Enclosing class does not have access to the members of the nested class.

Nested class types:

- **static :** It must access the members of its enclosing class through an object. ie, it cannot refer to members (non static) of its enclosing class directly. It can access static data members of outer class including private.

- **non - static: Inner class**

  It has access to all of the variables and methods of its outer class and may refer to them in the same way that other non-static members of the outer class do

Nested classes are divided into two categories: static and non-static. Nested classes that are declared static are simply called *static nested classes*. Non-static nested classes are called *inner classes*.

```
class OuterClass
{
        static class StaticNestedClass
        {
                ...
        }
        class InnerClass
        {
                ...
        }
}
```

```
class Outer
 {              int  outer_x = 100;
               void test()
              {        Inner   inner = new Inner();
                       inner.display();
              }
               class Inner
             {        void display()
                     {         System.out.println("display:  outer_x = " + outer_x);    }
              }
   }
class InnerClassDemo
{        public  static void main(String  args[])
        {       Outer   outer = new Outer();
                outer.test();
         }  }
```

```
class Outer
{     int outer_x = 100;
       void test()
      {        Inner   inner = new Inner();
               inner.display();      }
       void showY()
      {          System.out.println(y);      // error, y not known  here!       }
      class Inner
      {        int y = 10;  //y is local to Inner
               void display()
               {               System.out.println("display:  outer_x = " + outer_x);       }
      }
}
class InnerClassDemo
{       public  static void main(String  args[])
       {       Outer   outer = new Outer();
               outer.test();
       }  }
```

The Object Class

- There is one special class, **Object**, defined by Java.
- All other classes are subclasses of **Object**.
- That is, **Object** is a superclass of all other classes.
- This means that a reference variable of type **Object** can refer to an object of any other class.
- Arrays are implemented as classes, a variable of type **Object** can also refer to any array.

object can be a member of another class.

```
class Contact          is also object                      Object r = new Employee
{     int mobileNo, landline;                                      (— — — —);
}
class Employee              Object
{
            int empId;
            String empName;
            Contact c = new Contact();
            Employee(int r, String nm, int n1, int n2)
           {
                        empId = r;
                        empName = nm;
                        c.mobileNo  = n1;
                        c.landline = n2;
           }
}
```

Create a class Marks with members Internal, EndSem and grade.

grade is either ‘P’ or ’F’.

Define void compute_grade() which computes the grade as 'P‘ if EndSem is above 18 and Sum of Internal and EndSem is above 50. Otherwise ‘F’

Create a class Student having regno,name and an object of Marks class as its member. Use appropriate constructors to inilialize student object.

Write a java program which instantiate array of 3 Student Objects and display student details whose grade is ‘P’.