Static Modifier:

- static is non-access modifiers and it is a keyword
- with the help of static modifiers, we declare variable, we can define methods , we can also define static block {upto Java 6}

Static Variables:

- A variable which is declare in a class with static modifier. Another name for static variables or class variables
- Non static variable also called Object Variable
- For Static variables only one copy of memory created and it can be shared by N.no.of Objects of Same type
- For static variable memory is not allocated with in the Object , rather memory is allocated with in "static constant pool"
- For static variables memory is allocated whenever the class is loaded in the memory
- Every static variable must be referred by either classname or Object reference whenever you want access it from outside of the class

Example:

```
class Test {
        static int x=10;
}
class Testing
{
        public static void main(String args[ ])
        {
                System.out.println("x val is : "+Test.x);
                Test t=new Test( );
                System.out.println("x val is : "+t.x);

        }
}
```

Example2 :

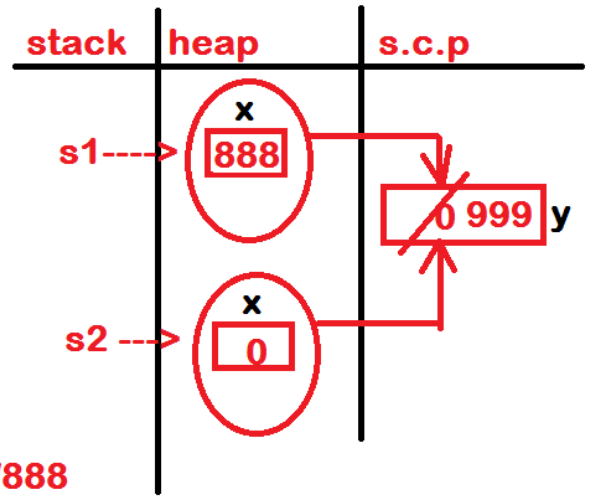
```
class Test
{
        static int x=111;
        public static void main(String args[ ])
        {
                System.out.println("x val is : "+x);
                Test t=new Test( );
                System.out.println("x val is : "+t.x);
                System.out.println("x val is : "+Test.x);
        }
}
```

Example 3:

```
class Sample {
        int x; //non static
  static int y; //static

public static void main(String args[ ]){
    Sample s1=new Sample( );
    Sample s2=new Sample( );
S.o.pln("y is : "+ s1.y); //0
S.o.pln("y is : "+ s2.y); //0
s1.y=999;
 S.o.pln("y is : "+s1.y); //999
 S.o.pln("y is : "+s2.y); //999

  }
 }
```



For an Idea:

```
s1.x=888;
S.o.p("x is : "+s1.x); //888
S.o.p("x is : "+s2.x); //0
```

Note : If you want store the individual data for Object then we need to use instance field, but if you want store the common data every object of same type then we have to use static variable

```
class Student{
        int sno; String sname;  //instance fields
        static String course="Java";

        void setStudent(int no,String name) //no,name
        { sno=no; sname=name; }

        void getStudent()
        { System.out.println("Sno is : "+sno);
          System.out.println("Sname is : "+sname);
          System.out.println("Course is : "+course); }

        public static void main(String args[ ])
        { Student s1=new Student( );
                  s1.setStudent(101,"Ramesh");

         Student s2=new Student( );
                  s2.setStudent(102,"Sudha");

        // Student.course="Android";

         System.out.println("Data From s1");
         s1.getStudent( );

         System.out.println("Data From s2");
         s2.getStudent( ); }
}
```

Static Methods:

- Methods which are declare with in the class with static modifier
- Static methods can perform the operations only on the static variables
- Non static methods | instance methods can perform the operations on both static and non-static variables
- Every Static methods must be referred by either class name or Object reference when ever you want access it from outside of the class

```
Example:
class Test
{
        static void method1()
        { System.out.println("Static Mtd-1 of Test"); }
}

class Testing{
        public static void main(String args[ ])
        {
                Test.method1();
        }
}

Example :2
class Maths
{
        static int sq(int x)  //static mtd
        {  return x*x; }

        public static void main(String args[ ])
        { int r=sq(9);
          System.out.println("Result is : "+r); }
}
```

```
Example :3
class Master
{
        int x=111;  //instance field

        static void method1()
        {   Master m=new Master( );
              System.out.println("x val is : "+m.x); }

        public static void main(String args[])
        {
       method1();
        }
}
```

Note: if you access the non static variable in the static context then we must required an object reference of that class