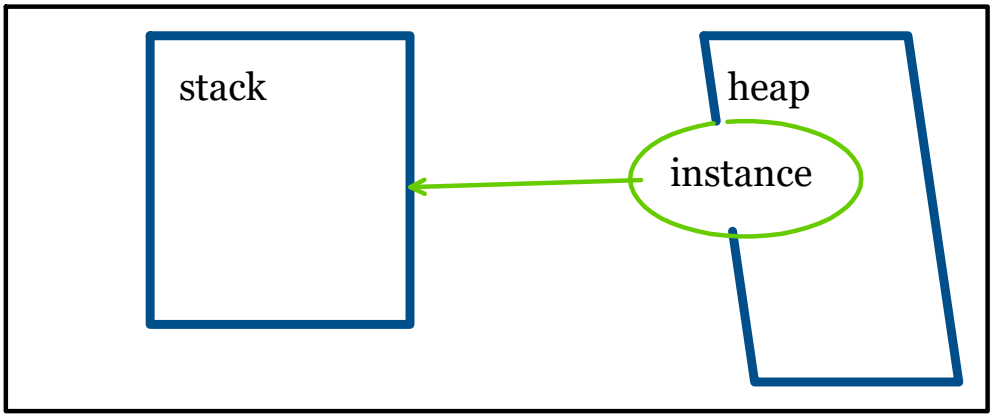
Non Static members of class

As we know the non static members are saved in the heap, but the heap is not connected to the class/stack, so we need to create an instance(a reference) between the heap and the stack.



If and only if the instance is created then the non static members get their memory and are accessible.

To Create an Instance

new **ClassName();**

This keyword creates an empty memory block for the instance

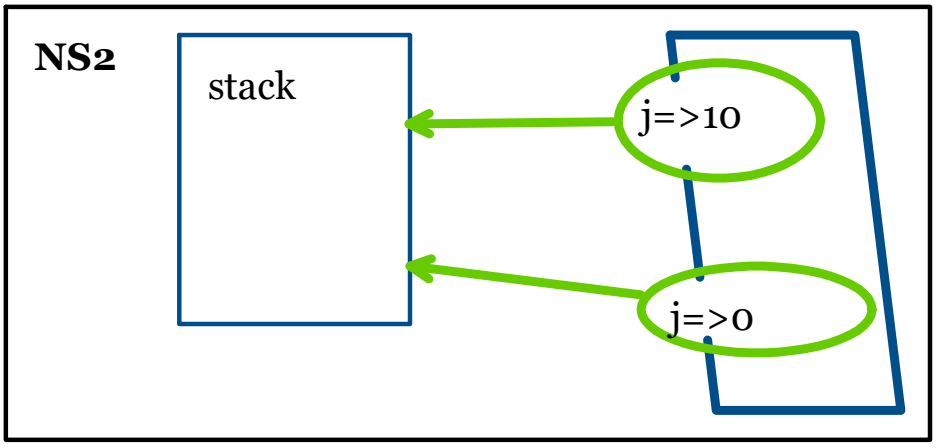
This is called as a *constructor*, and it is used to assign the reference in the memory.
Then the memory block which contains reference of class is called as Instance

```
public class NS1
{
   public static void main(String[] args)
   {
      System.out.println( new NS1() );
      /*  O/p:
```

```
        NS1@15bfd87

        Here, we are saying that, the reference of the
        class NS1 is saved in the address 15bfd87
       */
    }
}
```

```
public class NS2
{
    static int i=10;
    int j;
    public static void main(String[] args)
    {
        System.out.println( new NS2() );
        System.out.println( new NS2() );
    }
}
```

In the above program for a single class we are using multiple instances, that means multiple memory block are going to point to the class stack.



As the instance is volatile and non reusable, we go ahead and try to save the instance in a class variable called as object.

Class Variable	=	*Instance*
ClassName Var_Name	**=**	***new ClassName()***

An instance which is saved in a variable is called as Object Reference variable or just ***Object.***

Note: For the purpose of reusing the instance, we are going to save it in a variable.

```
public class NS3
{
   int i;
   public static void main(String[] args)
   {
      //class Variable =  instance
      NS3      obj    =   new   NS3();

      System.out.println(obj+"\n");
      System.out.println(obj.i);//0
      obj.i=10;
      System.out.println(obj.i);//10
   }
}
```

```
PS C:\java> java NS3.java
NS3@71a794e5

0
10
```

To use the Non static members,

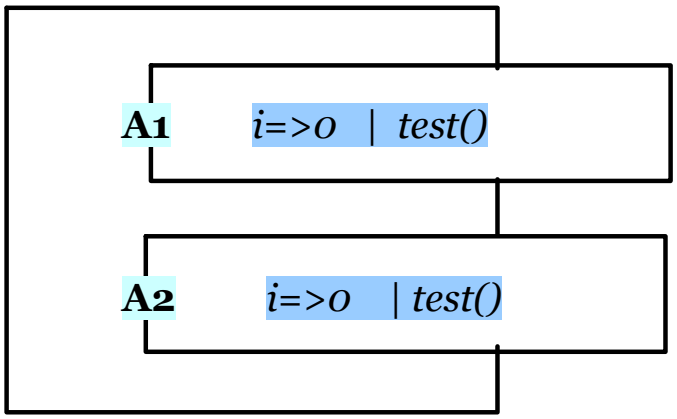
1. we need to create an instance so that the non static members get their memory
2. The instance should be reusable so we need to save it in a variable called as *Object*.

3. Finally we can use the object reference variable to access the non static members.

Note: If we want multiple access to the data (which means if *person a* uses the data it'll be something else and if *person b* uses the data, it'll be something else) is possible with the help multiple instances.

If the number of instances created is M and the number of elements in a class is N, then the number of elements created in the memory is ***M * N.***

```
class A
{
    int i;

    void test();

    main()
    {
        A A1=new A();
        A A2=new A();
    }
}
```



Object Oriented Programming

We develop program based the real time scenarios. Here every entity is considered as an *Object*.

Object: A real time entity (something which exists)

An object always has two things:

i. Something to identified
ii. Something to do

Ex:

Object : Apple

Identify -> color, taste, price, size.....

Task -> toeat

Every Object will have properties and behaviors. Where property means having a value for identification and behavior means the task done by the object.

Object	Properties	Behaviors
School	Name, Id, address..	Educate, conduct exams...
Teacher	Name, E_id, Subject, Look	Take classes, conduct exams, provide notes, grade the student.....
Student	Name, roll_no, class, section,.....	Attend classes, study, give exams.....

To use the Object Oriented Programming in java we need to have a class for object.

Class: A class is a blueprint for creating an object

```
public class Student//object of Student
{

   //non static variable are called as Properties
   int roll_no;
   String name;

   //non static method are called as behaviors
   void study()
   {
      System.out.println("Roll no: "+roll_no);
      System.out.println(name+" is Studying");
   }

   public static void main(String[] args)
   {
      Student s1=new Student();
      s1.name="Abcd";
      s1.roll_no=123;
      s1.study();

      Student s2=new Student();
```

```
        s2.name="Def";
        s2.roll_no=124;
        s2.study();
    }
}
```

Non static elements of a class

Any member of a class written without the static keyword are non static members and these get their memories in the instance of heap

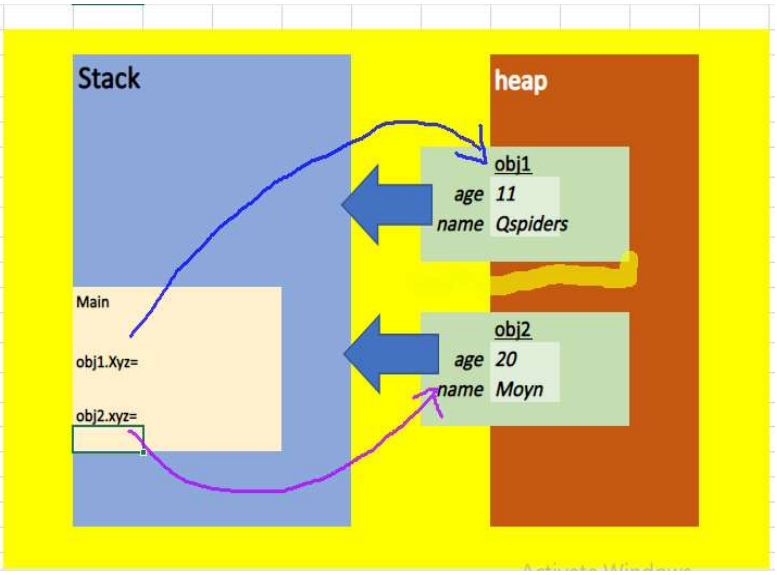
-> non static variables/instance variables
-> non static anonymous block/ Instance initializer block
-> non static methods / instance methods
-> Constructor.

1. Non static variables:

- These variables are created whenever the instance in created.
- The instance variables are different for every object/instance.
- A non static variable should be called by object or an instance.1

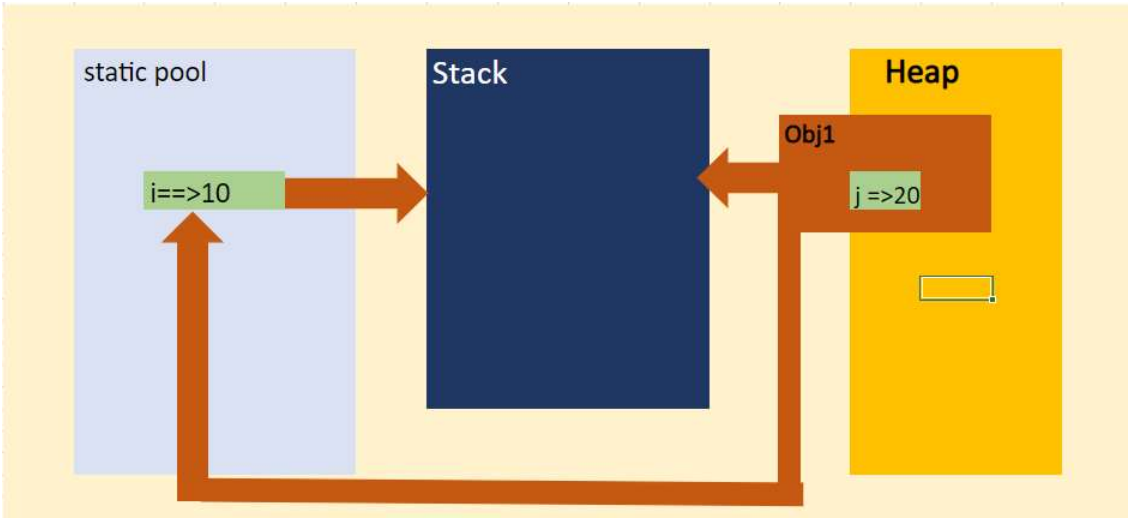
```
class A
{

   //declare the variable once
   //These are non static variables
   int age;
   String name;

   public static void main(String[] args)
   {
      //use it N number of times according to instances
      A obj1 = new A();
      obj1.age=30;
      obj1.name="moyn";
      System.out.println(obj1.age+" "+obj1.name);

      A obj2=new A();
      obj2.age=11;
      obj2.name="Qspiders";
      System.out.println(obj2.age+" "+obj2.name);
   }
}
```

```java
class A
{
    //declare the variable once
    //These are non static variables
    int age;
    String name;

    public static void main(String[] args)
    {
        //use it N number of times according to instances
        A obj1 = new A();
        obj1.age=30;
        obj1.name="moyn";
        System.out.println(obj1.age+" "+obj1.name);

        A obj2=new A();
        obj2.age=11;
        obj2.name="Qspiders";
        System.out.println(obj2.age+" "+obj2.name);
    }
```



To use the class members

	Main()	Static block	Non static block	Constructor
Local variables	**Directly**	**Directly**	**Directly**	**directly**
Global Static members	**Directly, With class name**	**Directly, With Class Name**	**Directly, With Class name**	**Directly, With Class name**
Global Non members	**With instance, with object**	**With instance, with object**	**Directly, *this keyword***	**Directly, *this keyword***



As specified in the above diagram, we can access any static member with the help of instance/Object also.

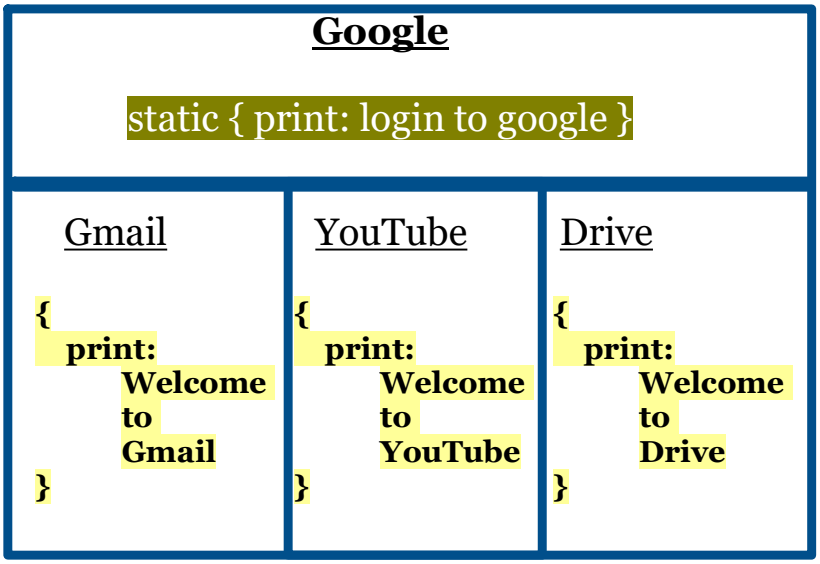
2. Non static Anonymous Block

This is a block without any name and we cannot call it, it gets executed every time an instance is created.

All pre-preparations of an object can be done in this Block.1

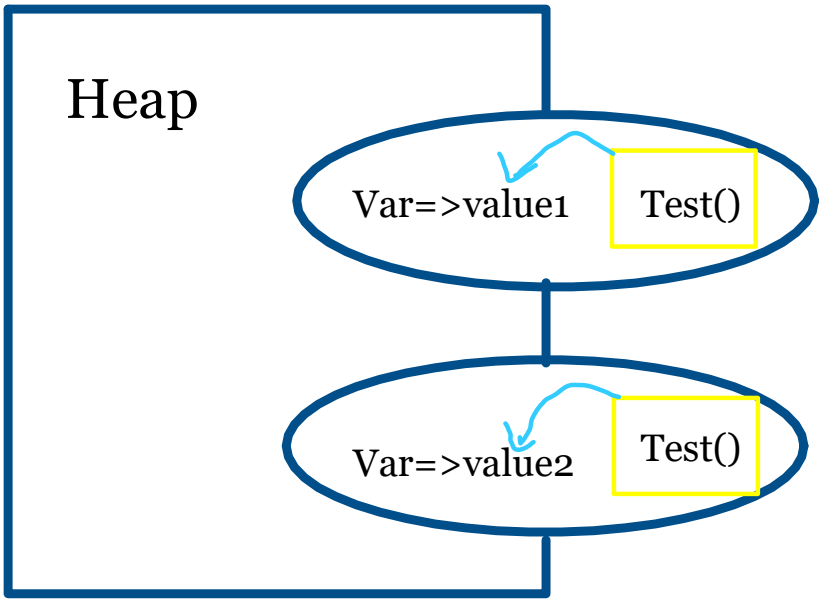
```
class Student
{
   int roll_no;
   String name;
   //instance initializer block
   {
      System.out.println("Student has joined the class");
   }
   public static void main(String[] args)
   {
      Student s1=new Student();
      s1.name="Abcd";
      s1.roll_no=101;

      Student s2=new Student();
      s2.name="Def";
      s2.roll_no=102;
      System.out.println();
      System.out.println(s1.name+"\t"+s1.roll_no);
      System.out.println(s2.name+"\t"+s2.roll_no);
   }
}
```

Note: A non static block gets executed and the JVM will enter the instance, the same instance is going to have values and those values can be accessed.



3. Instance methods

The methods written without the static keyword are the non static methods, these get their memories in the instance and as they are present in the instance they can also access their corresponding variables directly




```
public class Car //Factory
{
   static int i=0;//car count

   static//greeting
   {
      System.out.println("Welcome to the factory");
   }
   public static void repair(Car c)//one mechanic for all car
   {
      System.out.println(c.name+ "is being repaired" );
   }
```

```
    String name, type;
    int mileage, toSpeed;

    {
        System.out.println("Car no:"+ ++i+" is manfactured\n");
    }
    public void carDetails()
    {
        System.out.println("Car Name  :\t"+name);
        System.out.println("Type      :\t"+type);
        System.out.println("Top Speed :\t"+toSpeed+"KMPH");
        System.out.println("Mileage   :\t"+mileage+"KMPL\n");
    }

    public static void main(String[] args)
    {
        Car c1=new Car();
        c1.name="Nano";
        c1.type="Compact";
        c1.toSpeed=50;
        c1.mileage=25;

        Car c2=new Car();
        c2.name="Lamborghini";
        c2.type="Elite";
        c2.toSpeed=500;
        c2.mileage=5;
        c1.carDetails();
        c2.carDetails();

        repair(c1);
        repair(c2);
    }
}
```