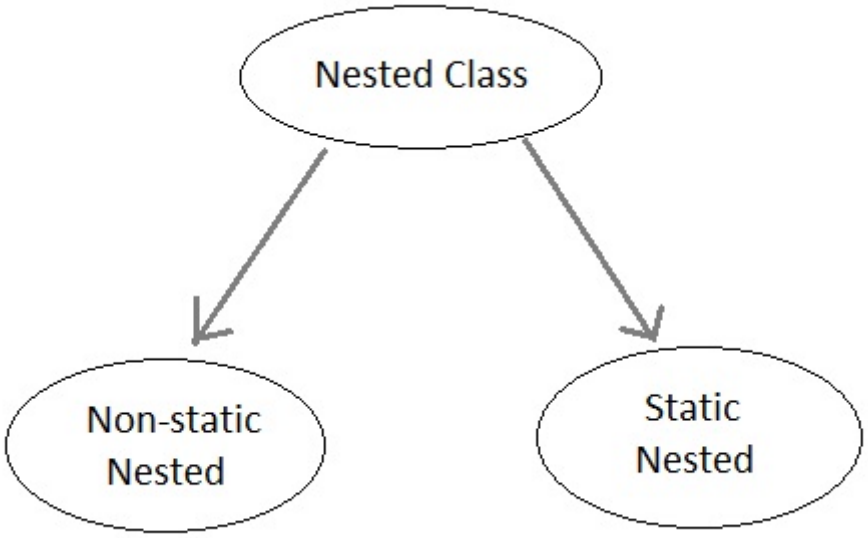
Nested classes in Java | Core Java Tutorial'

Nested Class

A class within another class is known as Nested class. The scope of the nested is bounded by the scope of its enclosing class.



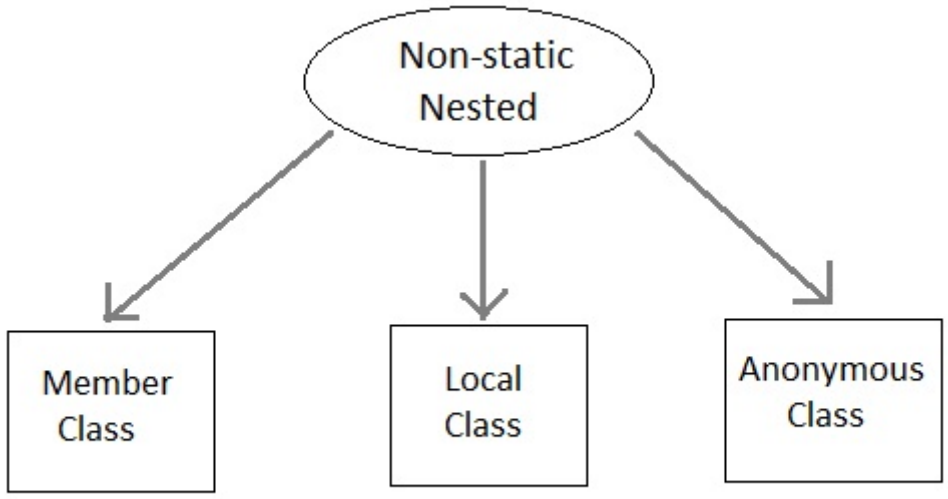
Static Nested Class

A satic nested class is the one that has **static** modifier applied. Because it is static it cannot refer to non-static members of its enclosing class directly. Because of this restriction static nested class is seldom used.

Non-static Nested class

Non-static Nested class is most important type of nested class. It is also known as **Inner** class. It has access to all variables and methods of **Outer** class and may refer to them directly. But the reverse is not true, that is, **Outer** class cannot directly access members of **Inner** class.

One more important thing to notice about an **Inner** class is that it can be created only within the scope of **Outer** class. Java compiler generates an error if any code outside **Outer** class attempts to instantiate **Inner** class.



Example of Inner class

```
class Outer
{
 public void display()
 {
  Inner in=new Inner();
  in.show();
 }

 class Inner
 {
  public void show()
  {
   System.out.println("Inside inner");
  }
 }
}

class Test
{
 public static void main(String[] args)
 {
  Outer ot=new Outer();
  ot.display();
 }
}
```

```
Output :
Inside inner
```

Example of Inner class inside a method

```
class Outer
{
 int count;
 public void display()
 {
  for(int i=0;i<5;i++)
  {
   class Inner       //Inner class defined inside for loop
   {
    public void show()
    {
     System.out.println("Inside inner "+(count++));
    }
   }
   Inner in=new Inner();
   in.show();
  }
 }
}

class Test
{
 public static void main(String[] args)
 {
  Outer ot=new Outer();
  ot.display();
 }
}
```

```
Output :
Inside inner 0
Inside inner 1
Inside inner 2
Inside inner 3
Inside inner 4
```

Example of Inner class instantiated outside Outer class

```
class Outer
{
 int count;
 public void display()
 {
  Inner in=new Inner();
  in.show();
 }
```

```
 class Inner
 {
  public void show()
  {
   System.out.println("Inside inner "+(++count));
  }
 }
}

class Test
{
 public static void main(String[] args)
 {
  Outer ot=new Outer();
  Outer.Inner in= ot.new Inner();
  in.show();
 }
}
```

```
Output :
Inside inner 1
```

Annonymous class

A class without any name is called Annonymous class.

```
interface Animal
{
 void type();
}
public class ATest {
 public static void main(String args[])
 {
  Animal an = new Animal(){         //Annonymous class created
  public void type()
  {
   System.out.println("Annonymous animal");
  }
 };
  an.type();
 }
}
```

```
Output :
Annonymous animal
```

Here a class is created which implements **Animal** interace and its name will be decided by the compiler. This annonymous class will provide implementation of **type()** method.