### 

A class declared inside a class is known as nested (Inner)class.

We use nested classes to logically group classes in one place so that it can be more readable and maintainable code.

It can access all the members of outer class including private members.  
  
**Syntax of Nested class**

1. **class** OuterClass {
2. ...
3. **class** NestedClass {
4. ...
5. }
6. }

***Why Use Nested Classes?***

1. It is a way of logically grouping classes that are only used in one place.
2. It increases encapsulation.
3. Nested classes can lead to more readable and maintainable code.

Types of Nested classes

There are two types of nested classes non-static and static nested classes.The non-static nested classes are also known as inner classes.

1. Non-static nested class(inner class)
   * a)Member inner class
   * b)Annomynous inner class
   * c)Local inner class
2. Static nested class

|  |  |
| --- | --- |
| **Type** | **Description** |
| [Member Inner Class](http://www.javatpoint.com/member-inner-class) | A class created within class and outside method. |
| [Anonymous Inner Class](http://www.javatpoint.com/anonymous-inner-class) | A class created for implementing interface or extending class. Its name is decided by the java compiler. |
| [Local Inner Class](http://www.javatpoint.com/local-inner-class) | A class created within method. |
| [Static Nested Class](http://www.javatpoint.com/static-nested-class) | A static class created within class. |
| [Nested Interface](http://www.javatpoint.com/nested-interface) | An interface created within class or interface. |

## **Java Member inner class example**

In this example, we are creating msg() method in member inner class that is accessing the private data member of outer class.

**class** MemberOuter {

**int** data = 30;

**class** Inner {

**void** msg() {

System.***out***.println("data is " + data);

}

}

}

**public** **class** TestMemberOuter {

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

MemberOuter obj = **new** MemberOuter(); // create outerclass object

MemberOuter.Inner in = obj.**new** Inner(); // create inner class object

in.msg();

}

}

* the inner class object can be created as follows:

OuterClassName> outerObj = **new** <OuterClassName>(arguments);

# outerclassName.<InnerClassName> innerObj = outerObj.new <InnerClassName>(arguments);

# Java Anonymous inner class

**Anonymous Inner Classes**

It is a type of inner class which

1. has no name
2. can be instantiated only once
3. is usually declared inside a method or a code block ,a curly braces ending with semicolon.
4. is accessible only at the point where it is defined.
5. does not have a constructor simply because it does not have a name
6. cannot be static

A class that have no name is known as anonymous inner class in java**.**

**It should be used if you have to override method of class or interface**

### Java anonymous inner class example using class

/\*abstract class Vehicle{

abstract void start();

}

\*/

**interface** Vehicle {

**void** start();

}

**public** **class** AnonymousClassDemo {

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

Vehicle v = **new** Vehicle() {

**public** **void** start() {

System.***out***.println("Car started");

}

};

v.start();

}

}

Well, the main thing is that it is quicker to just create an anonymous inner class rather than create a new separate class. Anonymous inner classes are especially useful when you only need to override a **small amount of functionality (like just one method) in a superclass**, and don’t want to deal with the overhead of creating an entire class for something so simple.

Anonymous inner classes are also helpful for adding behaviour to objects which already have names, such as AWT components (to which anonymous event handlers are added), and threads.

# Java Local inner class

A class i.e. created inside a method is called local inner class in java.

If you want to invoke the methods of local inner class, you must instantiate this class inside the method.

## **Java local inner class example**

**public** **class** localInner1{

**private** **int** data=30;//instance variable

**void** display(){

**class** Local{

**void** msg(){System.out.println(data);}

  }

  Local l=**new** Local();

  l.msg();

 }

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

  localInner1 obj=**new** localInner1();

  obj.display();

 }

}

#### Rule: Local variable can't be private, public or protected.

## **Rules for Java Local Inner class**

#### 1) Local inner class cannot be invoked from outside the method.

# 2) Local inner class cannot access non-final local variable till JDK 1.7. Since JDK 1.8, it is possible to access the non-final local variable in local inner class.

# Java static nested class

A static class i.e. created inside a class is called static nested class in java. It can be accessed by outer class name.

* It can access static data members of outer class including private.
* Static nested class cannot access non-static (instance) data member or method

## **Java static nested class example with instance method**

**class** Outer {

**static** **int** *data* = 30;

**static** **class** Inner {

**void** msg() {

System.***out***.println("data is " + *data*);

}

}

}

**public** **class** StaticInnerClassDemo {

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

Outer.Inner in = **new** Outer.Inner(); // create static inner class object

in.msg();

}

}

Accessing static inner class instance members

Outerclassname.InnerClassname innerobj=new OuterClassname.Innerclassname();

innerObj.membername;

## **Java static nested class example with static method**

If you have the static member inside static nested class, you don't need to create instance of static nested class.

**class** TestOuter2{

**static** **int** data=30;

**static** **class** Inner{

**static** **void** msg(){System.out.println("data is "+data);}

  }

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

  TestOuter2.Inner.msg();//no need to create the instance of static nested class

  }

}