**Inner classes in java: Anonymous inner and static nested class**

[**OOPS CONCEPT**](http://beginnersbook.com/category/oops-concept/)

**What is an inner class?**

**Inner class** are defined inside the body of another class (known as **outer class**). These classes can have access modifier or even can be marked as abstract and final. Inner classes have special relationship with outer class instances. This relationship allows them to have access to outer class members including private members too.

**Inner classes can be defined in four different following ways as mentioned below:**

1) Inner class  
2) Method – local inner class  
3) Anonymous inner class  
4) Static nested class

**1) Inner class**

An inner class is declared inside the curly braces of another enclosing class. Inner class is coded inside a Top level class as shown below:-

//Top level class definition

class MyOuterClassDemo {

private int myVar= 1;

// inner class definition

class MyInnerClassDemo {

public void seeOuter () {

System.out.println("Value of myVar is :" + myVar);

}

} // close inner class definition

} // close Top level class definition

**Inner class** acts as a member of the enclosing class and can have any access modifiers: abstract, final, public, protected, private, static.  
Inner class can access all members of the outer class including those marked private as shown in the above example where inner class is accessing the private variable "myVar" of outer class.

**Instantiating an inner class**  
To instantiate an instance of inner class, there should be a live instance of outer class. An inner class instance can be created only from an outer class instance. An inner class shares a special relationship with an instance of the enclosing class.  
Instantiating an inner class from within code in outer class:

class MyOuterClassDemo {

private int x= 1;

public void innerInstance()

{

MyInnerClassDemo inner = new MyInnerClassDemo();

inner. seeOuter();

}

public static void main(String args[]){

MyOuterClassDemo obj = new MyOuterClassDemo();

obj.innerInstance();

}

// inner class definition

class MyInnerClassDemo {

public void seeOuter () {

System.out.println("Outer Value of x is :" + x);

}

} // close inner class definition

} // close Top level class definition

Output:

Outer Value of x is :1

**Instantiating an inner class from outside the outer class Instance Code:**  
The public static void main code in the above example can be replaced with this one. It will also give the same output.

public static void main(String args[]){

MyOuterClassDemo.MyInnerClassDemo inner = new MyOuterClassDemo().new MyInnerClassDemo();

inner. seeOuter();

}

**2) Method–Local inner classes**

A method local inner class is defined within a method of the enclosing class. If you want to use inner class , you must instantiate the inner class in the same method, but after the class definition code. Only two modifiers are allowed for method-local inner class which are abstract and final.The inner class can use the local variables of the method (in which it is present), only if they are marked final.

//Top level class definition

class MyOuterClassDemo {

private int x= 1;

public void doThings(){

String name ="local variable";

// inner class defined inside a method of outer class

class MyInnerClassDemo {

public void seeOuter () {

System.out.println("Outer Value of x is :" + x);

System.out.println("Value of name is :" + name);//compilation error!!

} //close inner class method

} // close inner class definition

} //close Top level class method

} // close Top level class

The above code will throw a compilation error as Inner class cannot use the non-final variables of the method, in which it is defined.  
This is how it can be fixed: If we mark the variable as final then inner class can use it.

final String name ="local variable";// inner object can use it

**3) 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

Example:

package beginnersbook.com;

class Pizza{

public void eat()

{

System.out.println("pizza");

}

}

class Food {

/\* There is no semicolon(;)

\* semicolon is present at the curly braces of the method end.

\*/

Pizza p = new Pizza(){

public void eat()

{

System.out.println("anonymous pizza");

}

};

}

**4) Static Nested Classes**

A static nested classes are the inner classes marked with static modifier.Because this is static in nature so this type of inner class doesn’t share any special kind of relationship with an instance of outer class.A static nested class cannot access non static members of outer class.

Example:

class Outer{

static class Nested{}

}

A **static nested class** can be instantiated like this:

class Outer{// outer class

static class Nested{}// static nested class

}

class Demo{

public static void main(string[] args){

// use both class names

Outer.Nested n= new Outer.Nested();

}

}