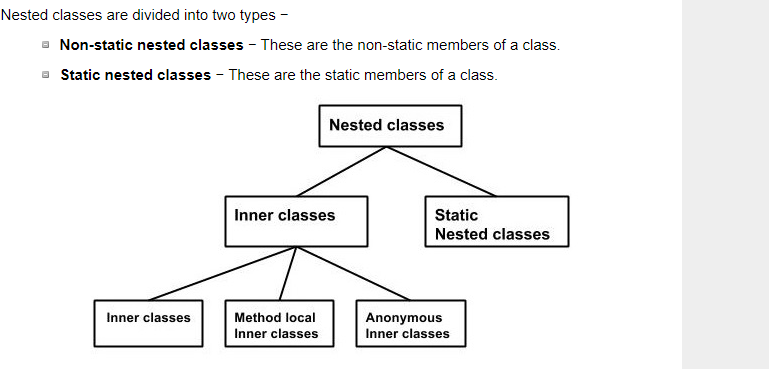
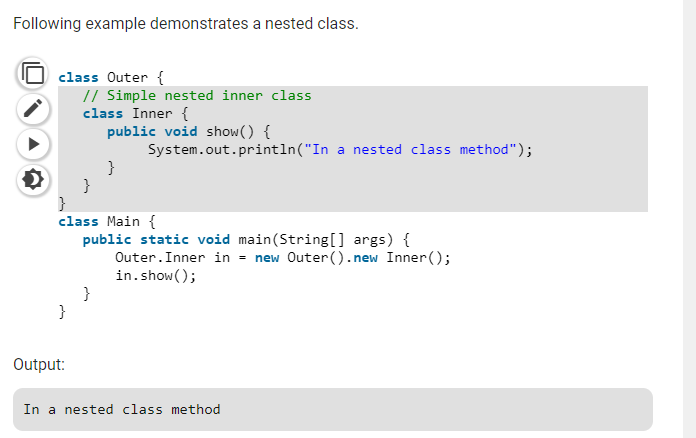
Inner class in java

Inner class means one class which is a member of another class. There are basically four types of inner classes in java.

1. Nested Inner class  
   2) Method Local inner classes  
   3) Anonymous inner classes  
   4) Static nested classes



**Nested Inner class** can access any private instance variable of outer class. Like any other instance variable, we can have access modifier private, protected, public and default modifier.  
Like class, interface can also be nested and can have access specifiers.



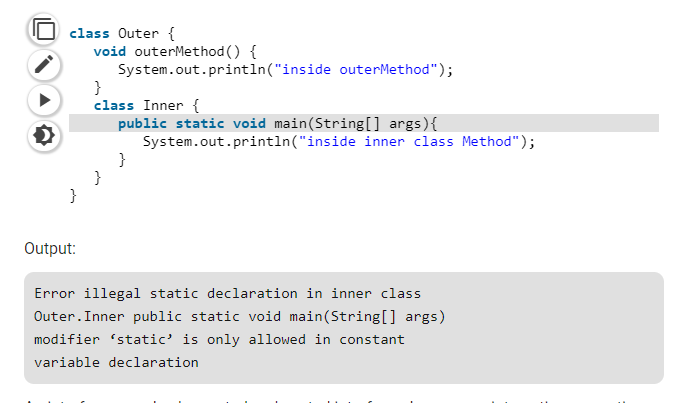
As a side note, we can’t have static method in a nested inner class because an inner class is implicitly associated with an object of its outer class so it cannot define any static method for itself. For example the following program doesn’t compile.

## **Difference between static and non static nested class in Java**

1) Nested static class doesn't need reference of Outer class but non static nested class or Inner class requires Outer class reference. You can not create instance of Inner class without creating instance of Outer class. This is by far most important thing to consider while making a nested class static or non static.

2) static class is actually [static member](http://javarevisited.blogspot.sg/2011/11/static-keyword-method-variable-java.html) of class and can be used in [static context](http://javarevisited.blogspot.sg/2012/02/why-non-static-variable-cannot-be.html) e.g. static method or static block of Outer class.

3) Another difference between static and non static nested class is that you can not access non static members e.g. method and field into nested static class directly. If you do you will get error like ["non static member can not be used in static context"](http://javarevisited.blogspot.sg/2012/02/why-non-static-variable-cannot-be.html). While Inner class can access both static and non static member of Outer class.



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

Additionally, it can access all the members of outer class including private data members and methods.

1. **class** Java\_Outer\_class{
2. //code
3. **class** Java\_Inner\_class{
4. //code
5. }
6. }

### **Advantage of java inner classes**

There are basically three advantages of inner classes in java. They are as follows:

1) Nested classes represent a special type of relationship that is **it can access all the members (data members and methods) of outer class** including private.

2) Nested classes are used **to develop more readable and maintainable code** because it logically group classes and interfaces in one place only.

3) **Code Optimization**: It requires less code to write.

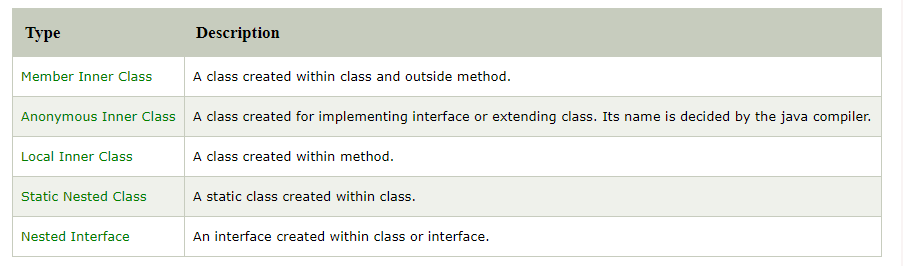
### **Difference between nested class and inner class in Java**

Inner class is a part of nested class. Non-static nested classes are known as inner classes.

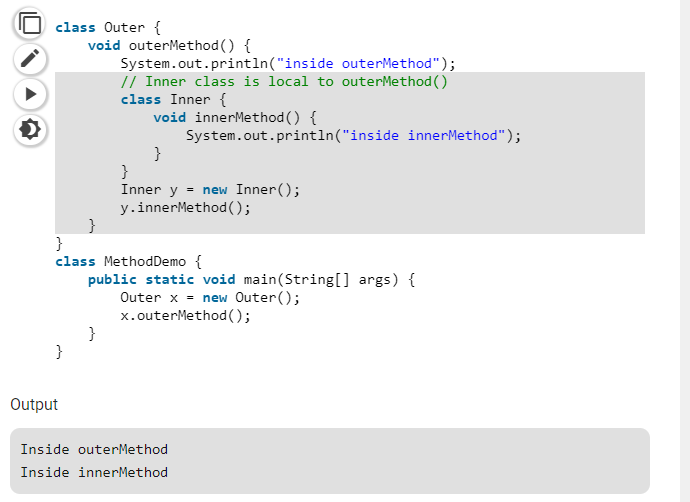
### **Types of Nested classes**

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

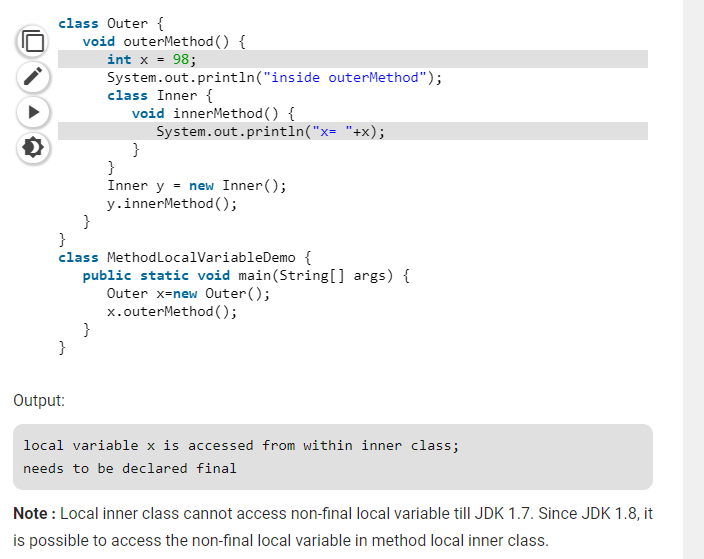
* Non-static nested class (inner class)
  1. Member inner class
  2. Anonymous inner class
  3. Local inner class
* Static nested class

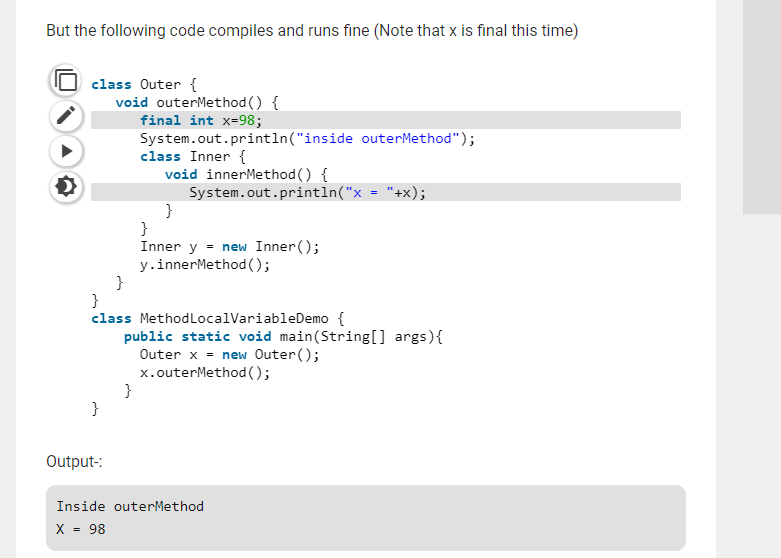


**Method Local inner classes**  
Inner class can be declared within a method of an outer class. In the following example, Inner is an inner class in outerMethod().



Method Local inner classes can’t use local variable of outer method until that local variable is not declared as final. For example, the following code generates compiler error (Note that x is not final in outerMethod() and innerMethod() tries to access it).





The main reason we need to declare a local variable as a final is that local variable lives on stack till method is on the stack but there might be a case the object of inner class still lives on the heap.  
Method local inner class can’t be marked as private, protected, static and transient but can be marked as abstract and final, but not both at the same time.

# Anonymous Inner Class in Java

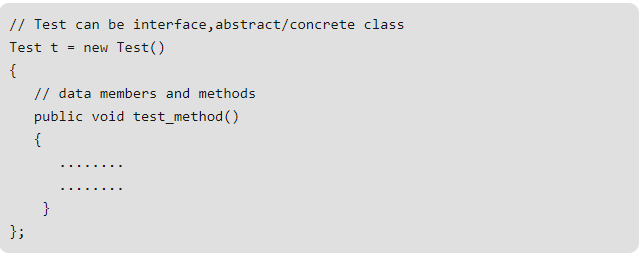
It is an inner class without a name and for which only a single object is created. An anonymous inner class can be useful when making an instance of an object with certain “extras” such as overloading methods of a class or interface, without having to actually subclass a class.

Anonymous inner classes are useful in writing implementation classes for listener interfaces in graphics programming.

Anonymous inner class are mainly created in two ways:

* Class (may be abstract or concrete)
* Interface

**Syntax:** The syntax of an anonymous class expression is like the invocation of a constructor, except that there is a class definition contained in a block of code.



/Java program to demonstrate need for Anonymous Inner class

interface Age

{

    int x = 21;

    void getAge();

}

class AnonymousDemo

{

    public static void main(String[] args)

    {

        // Myclass is implementation class of Age interface

        MyClass obj=new MyClass();

        // calling getage() method implemented at Myclass

        obj.getAge();

    }

}

// Myclass implement the methods of Age Interface

class MyClass implements Age

{

    @Override

    public void getAge()

    {

        // printing the age

        System.out.print("Age is "+x);

    }

}

In the program, interface Age is created with getAge() method and x=21.  Myclass is written as implementation class of Age interface. As done in Program, there is no need to write a  separate class Myclass. Instead,   directly copy the code of Myclass into this parameter, as shown here:

Age oj1 = new Age() {

@Override

public void getAge() {

System.out.print("Age is "+x);

}

};

Here, an object to Age is not created but an object of Myclass is created and copied in the entire class code as shown above. This is possible only with anonymous inner class. Such a class is called ‘anonymous inner class’, so here we call ‘Myclass’ as anonymous inner class.