**Inner class** : sometime we declare a class inside another class called as inner classes. The purpose is to group classes that belong together, which makes your code more readable and maintainable. . >>Inner classes concept introduced in 1.1 version to fix GUI bugs.

**Where to use inner classes** : Without existing one type of object, if there is no chance of existing another type of object then we should go for inner classes. Ex. :

Class University

{

Class department

{

}

}

University consists of several departments. Without existing university, there is no existence for department. Hence, we have to declare department class inside university class. Same for car and engine. Car is outer class and engine is inner class.

**Note** : Without existing outer class object, there is no chance of existing inner class object.

Inner classes are of 4 categories :

1. Normal / regular
2. Method local inner classes (declared inside method)
3. Anonymous inner classes (without name)
4. Static nested classes
5. **Normal / regular inner classes** :

If we are declaring any named class directly inside a class without static modifier, such type of inner class is called regular / normal inner class.

Example 1 :

Class Outer

{

Class Inner

{

public void m1() {} // normal method

}

Public static void main(String args[])

{

Syso(“in main”);

}

}

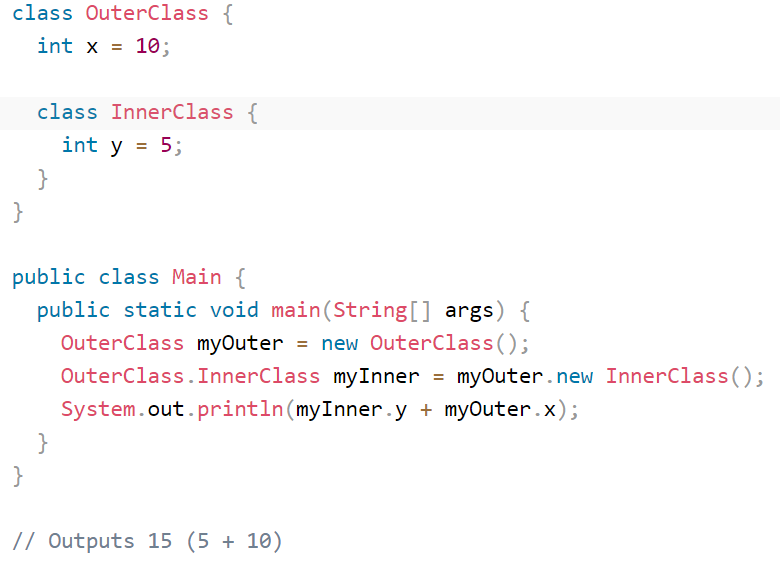
>>> Create object of inner and outer class :

Outer o = new Outer();

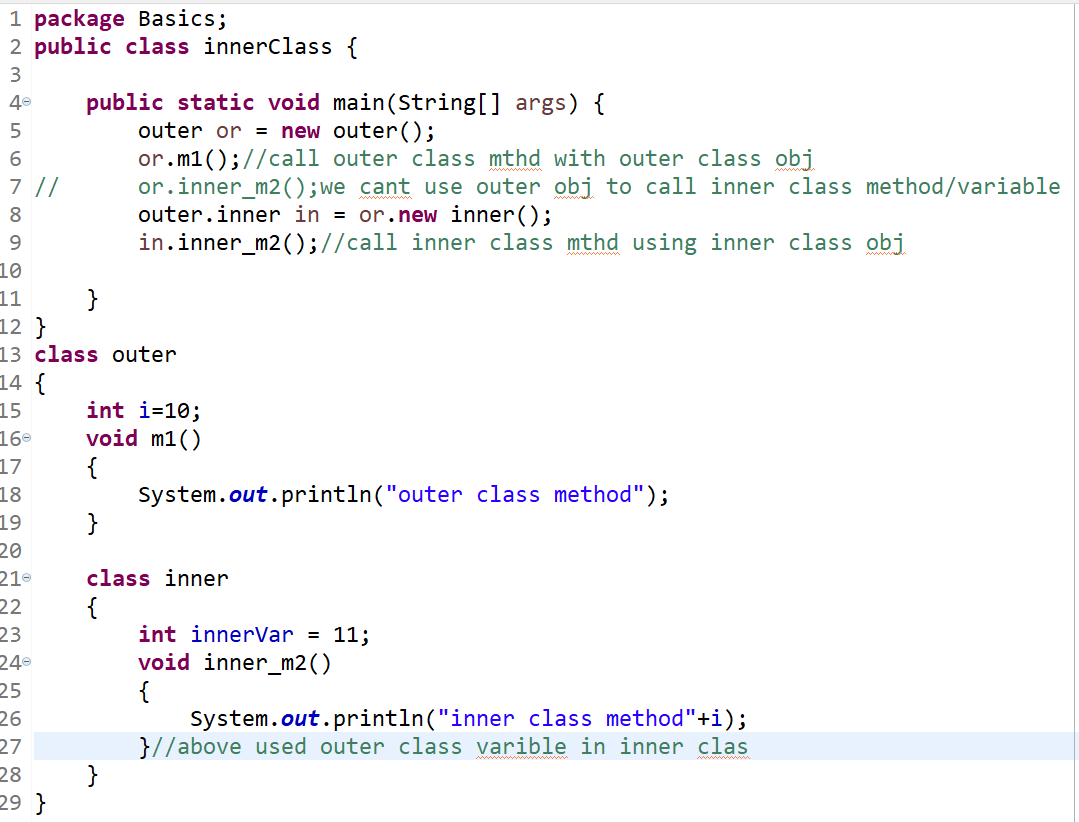
Outer.Inner i = o.new Inner();

i.m1();

Example 2 :



Example 2 :



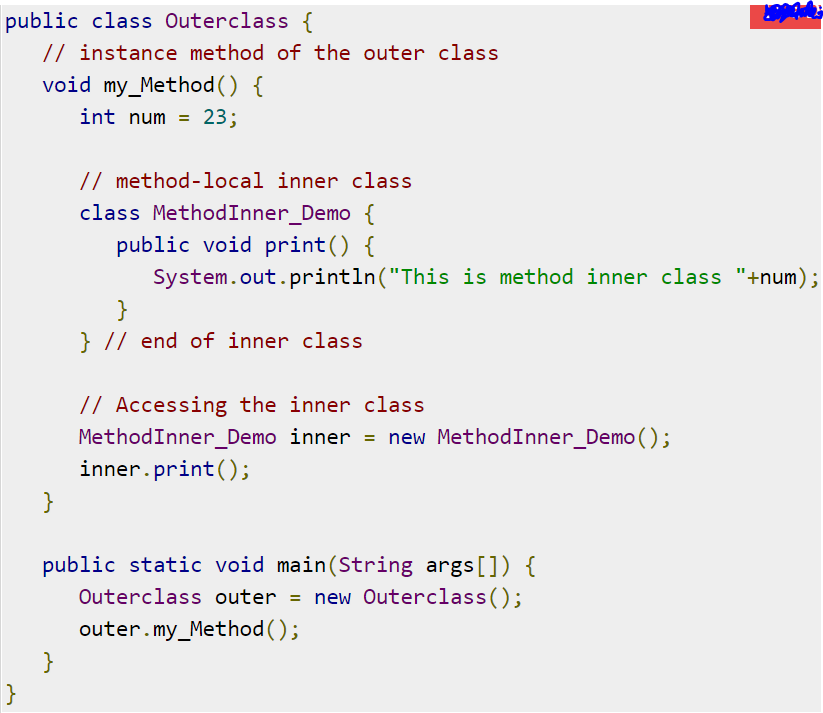
**Note** :

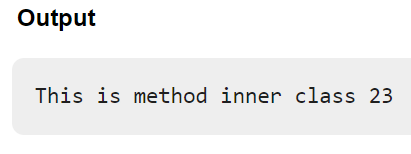
1. Inside inner class, we can’t declare any **static** members. Hence, we cannot declare main method and we can’t run inner class directly from command prompt.
2. **Static classes** : Java doesn't allow you to create top-level static classes; only nested (inner) static classes are allowed.

## Method-local Inner Class

In Java, we can write a class within a method and this will be a local type. Like local variables, the scope of the inner class is restricted within the method. A method-local inner class can be instantiated **only within the method** where the inner class is defined.

Example :





1. **Anonymous inner classes** (without name) : Sometimes we can declare inner class without name, such type of inner classes are called Anonymous inner classes for which only a single object is created. In case of anonymous inner classes, we declare and instantiate them at the same time. The main purpose of Anonymous inner classes is just for instant use i.e. one time usage.

**Constructor : (Do not tell to interviewer, if he asks say not able to recall) :** Since an anonymous inner class has no name, it is not possible to define a constructor for it within the class body.

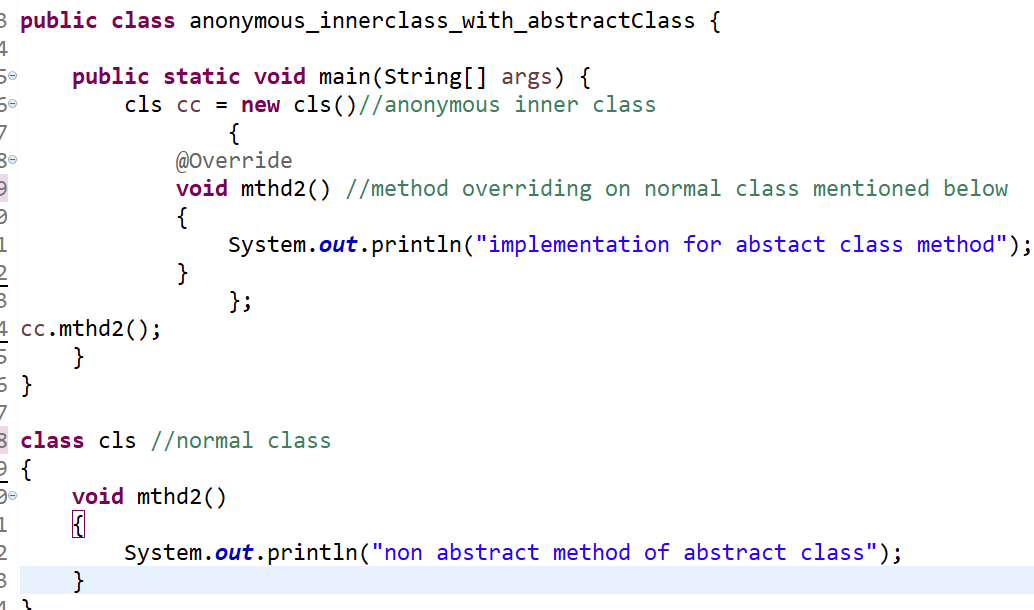
3 types of anonymous inner classes : 1. anonymous inner class that extends a class 2. anonymous inner class that implements an interface 3. anonymous inner class that defined inside arguments.

1. anonymous inner class that extends a (normal)class Example :

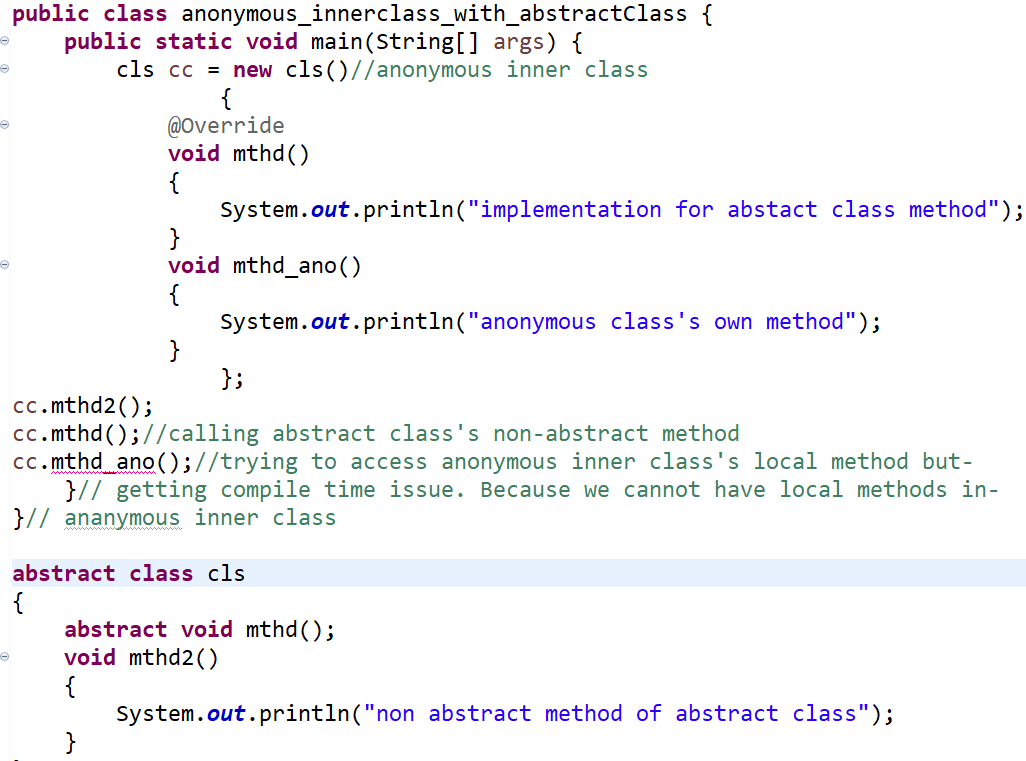
PopCorn p = new PopCorn() //anonymous inner class it is a class without name, it just extends a { //popcorn class. For that we are creating object with popcorn classe’s. . .//reference. So new object is object of anonymous inner class.

};

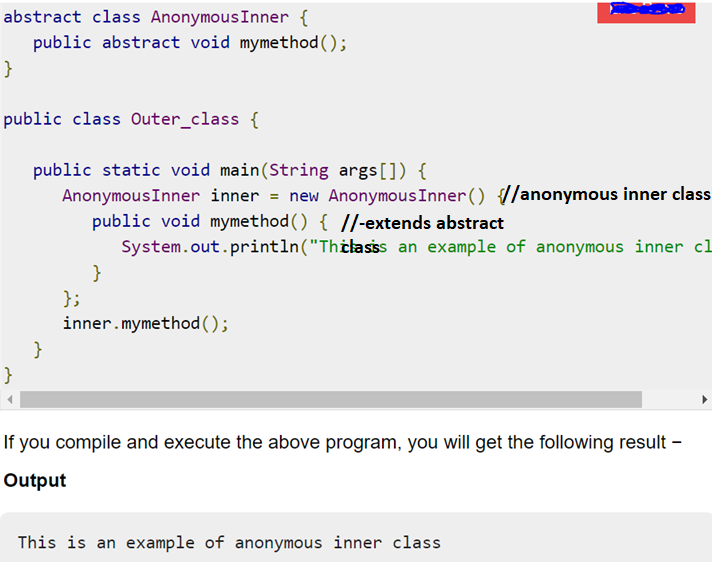
Example : anonymous inner class example using NORMAL class



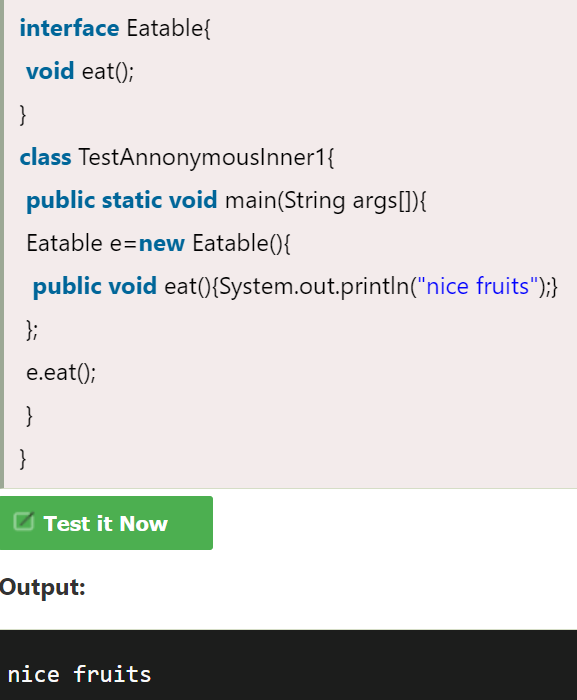
**VIMP : Local methods of anonymous inner class**. In anonymous inner class, we can override methods of class / interface / abstract class. We have to provide implementation to abstract methods of interface / abstract class. But we can not write local methods in anonymous inner class. If we write and try to access it with anonymous inner class’s object, it doesn’t allow us to do that. Example :



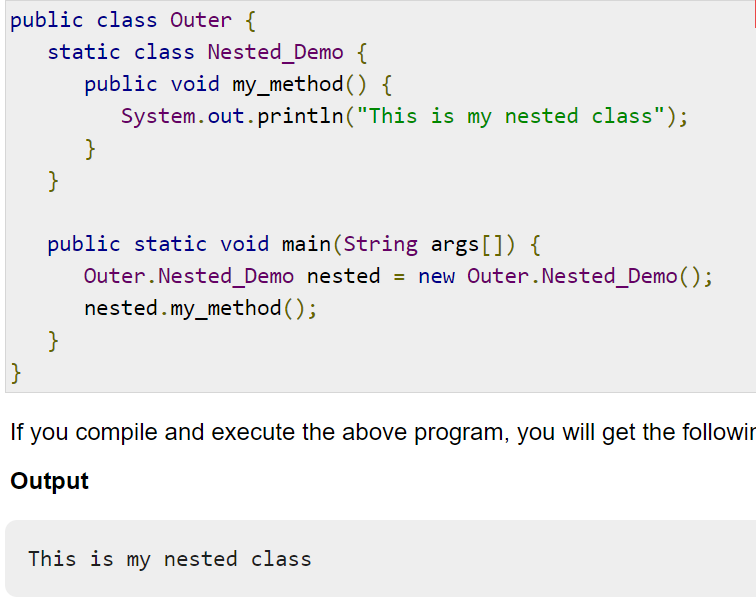
### Example1 :Java anonymous inner class example using Abstract class



## Example 2 : anonymous inner class example using interface



1. **Static nested class** : A static inner class is a nested class which is a static member of the outer class. It can be accessed without instantiating the outer class, using other static members. Just like static members, a static nested class does not have access to the instance variables and methods of the outer class.



NOTE : . there are 2 types of anonymous things in java : 1. **Anonymous inner classes** and 2. **Anonymous Array**