Inner classes were introduced in version 1.1 of Java.

**Nested classes enable us to logically group classes that are only used in one place, write more readable and maintainable code and increase encapsulation.**

Class Outer{

Class Inner{

}}

Nested classes are of 4 types:

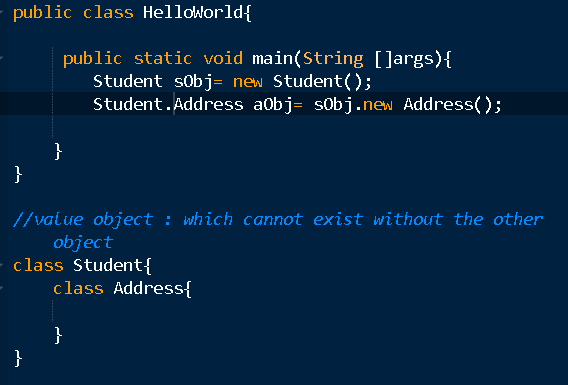
1) Normal inner class: any nested class which is not static, not inside a block and has a name

2) Anonymous inner class: a class without a name

3) Method local inner class: inner class defined inside a block

4) Static nested class: inner class, declared to be static

* Static nested class: if this inner class is static
* Inner class : if the inner class is not static



compiler : Student.class, Student$Address.class

**2. Static Nested Classes**

Nested class which is declared to be static

****

We can create object of static nested class without creating the object of the outer class, hence it is not termed as an inner class. When there is not such a strong association between your outer class and your inner class.

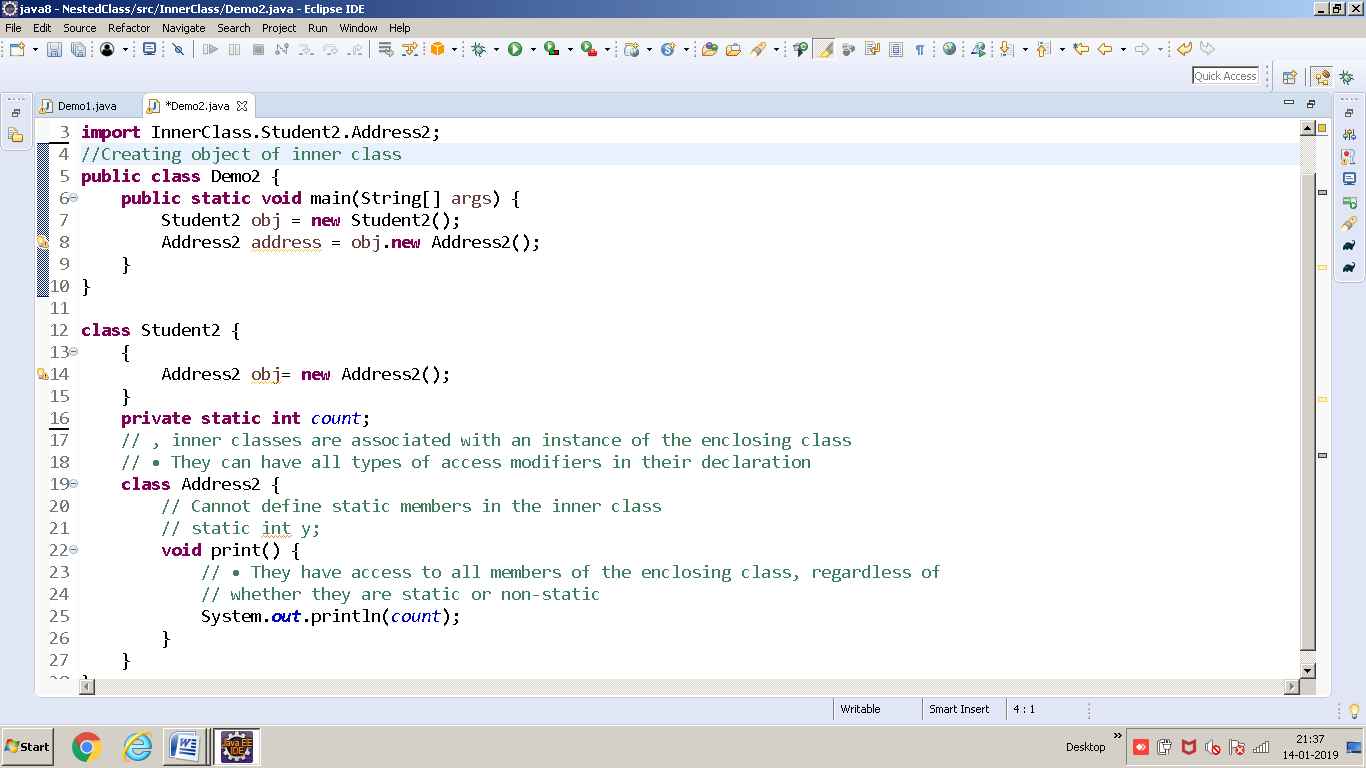
Here are a few points to remember about static nested classes:

* As with static members, these belong to their enclosing class, and not to an instance of the class
* They can have all types of access modifiers in their declaration
* They only have access to static members in the enclosing class
* They can define both static and non-static members

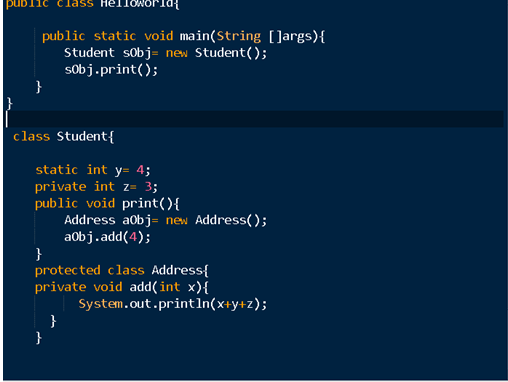
**3. Non-Static Nested Classes**

Next, here are a few quick points to remember about non-static nested classes:

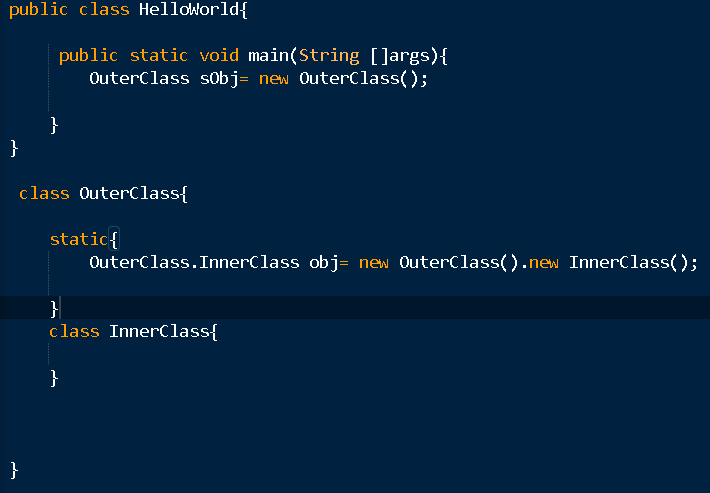
* They are also called inner classes
* They can have all types of access modifiers in their declaration
* Just like instance variables and methods, inner classes are associated with an instance of the enclosing class
* They have access to all members of the enclosing class, regardless of whether they are static or non-static
* They can only define non-static members
* Can also access private members of enclosing class
* You cannot use private or protected for the top level class, but can use any access modifier for the inner class.
* You cannot create the object of inner class without the object of outer class, hence it cannot be created in a static block/method in your outer class or from outside the class. But from the instance area of the outer class, I can create the object of the inner class directly.

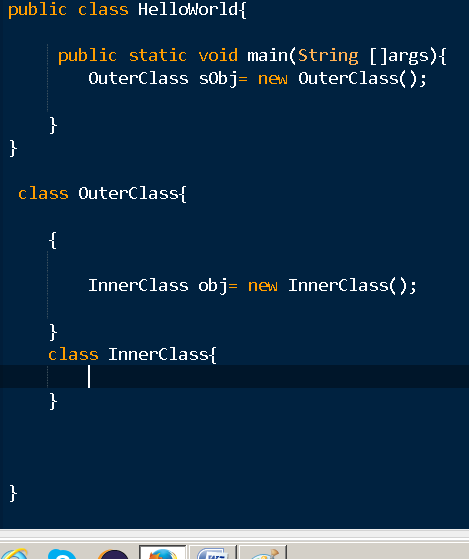


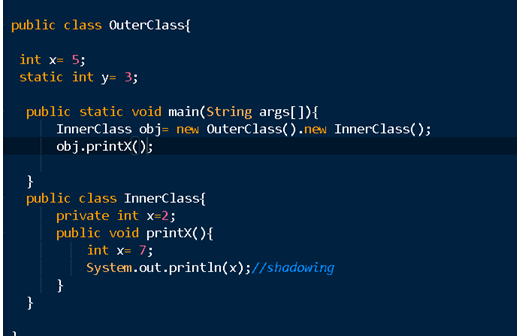
Q) What should be the output?

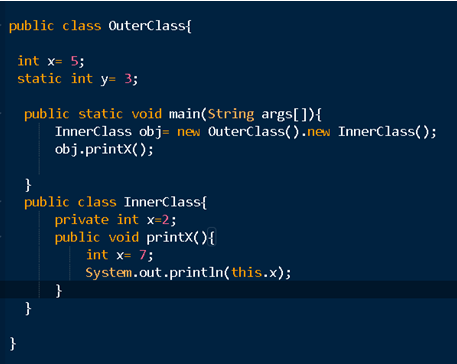


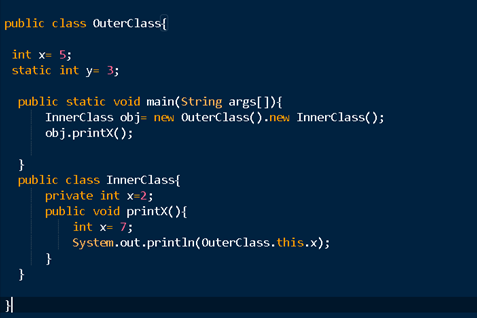
Q) What should be the output?

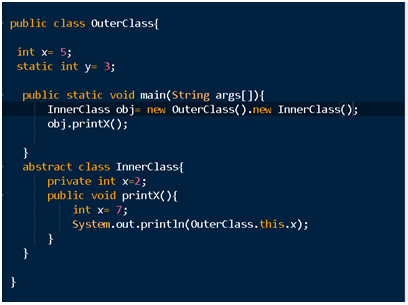


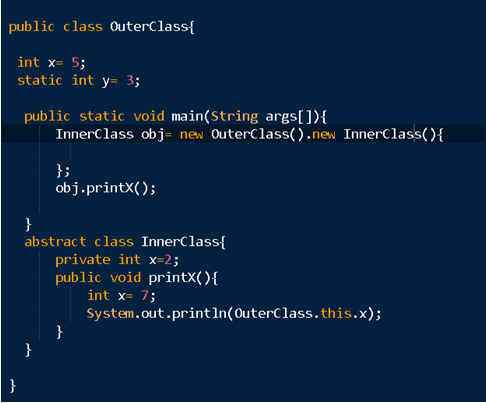












### ****3.1. Local Classes****

Local classes are a special type of inner classes – in which **the class is defined inside a method** or scope block.

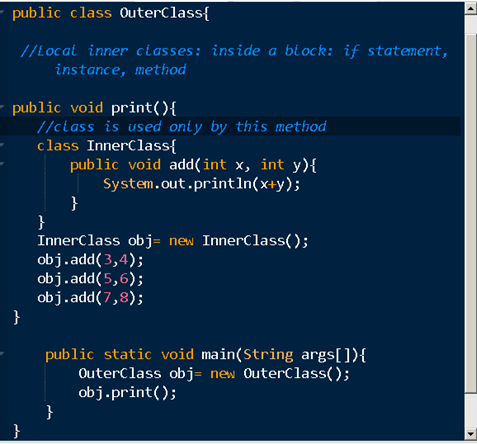
Let’s see a few points to remember about this type of class:

* They cannot have access modifiers in their declaration
* They have access to both static and non-static members in the enclosing context
* They can only define instance members
* Local class can only access local variables that are final or are effectively final.

Inner classes can access the all the members of your outer class but if your inner class is inside a static block, then it would only be able to access static members of the outer class

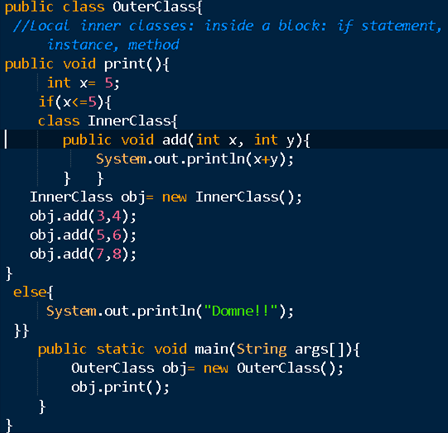
We cannot declare any static member in our inner classes.

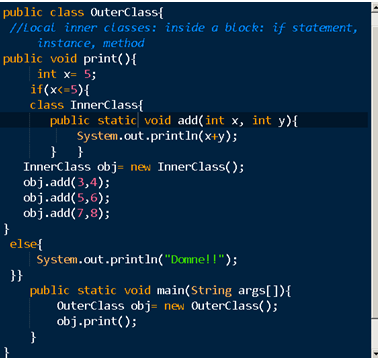
* You can use static inside a local inner class but should be used only to declare constants
* Interfaces by-default are static, so we cannot put them in a block.

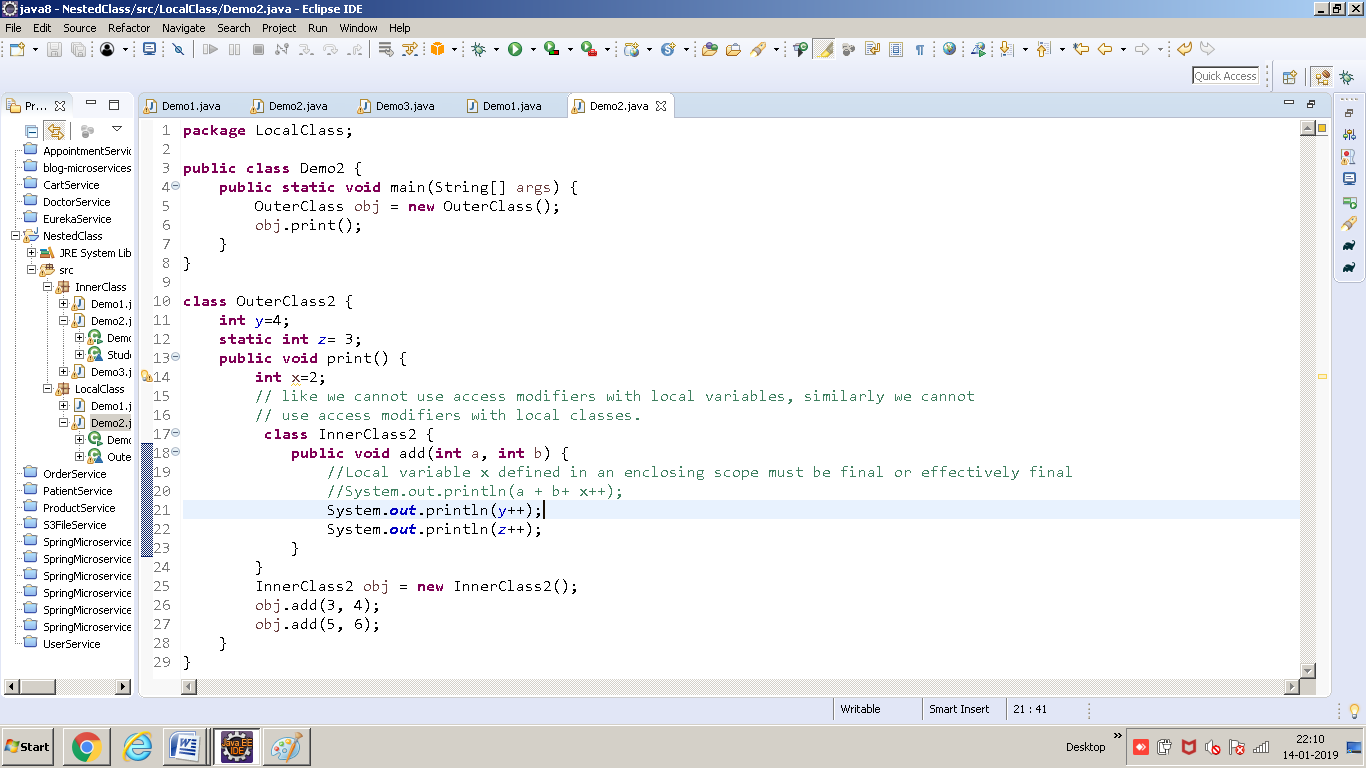


InnerClass can be used only inside the add method

We cannot use static keyword in inner local classes







### ****3.2****. Anonymous Classes

Anonymous classes can be used to define an implementation of an interface or an abstract class without having to create a reusable implementation. Anonymous inner classes should be used if we want to use them locally only once.

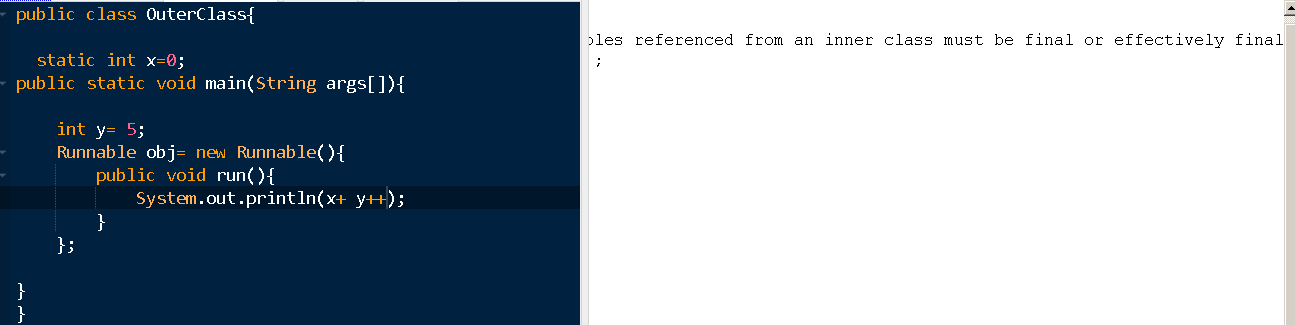
Let’s list a few points to remember about anonymous classes:

* They cannot have access modifiers in their declaration
* They have access to both static and non-static members in the enclosing context
* They can only define instance members
* They’re the only type of nested classes that cannot define constructors or extend/implement other classes or interfaces

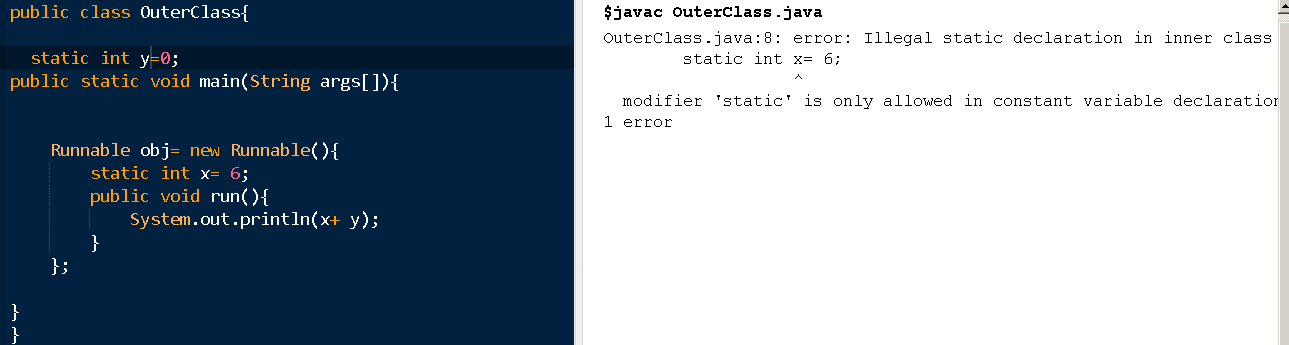
Inner class which does not have name

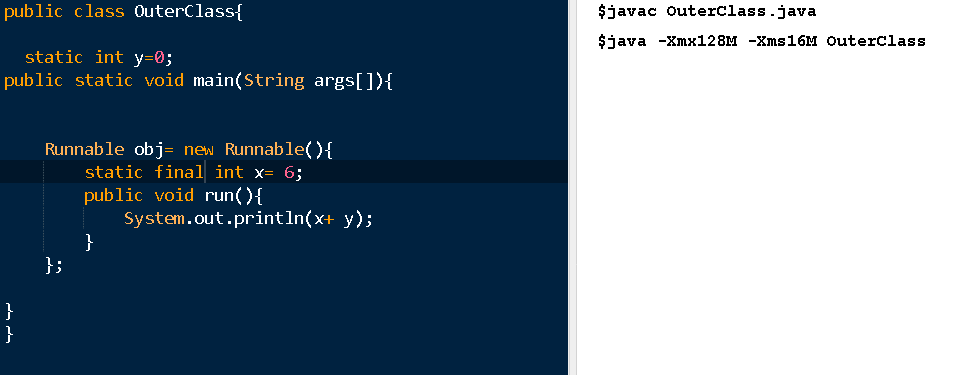
1. Which extends a class
2. Which implements an interface
3. That defines inside arguments

Q2) Do u think I can access local variables inside it?: final/ effectively final



Q3) Can I declare static members in the anonymous class? No if they are not constant. Means can only use it with final keyword





Q) Can I declare a constructor inside my anonymous block? No, bcoz we donno the name of the class

## ****4. Shadowing****

**The declaration of the members of an inner class shadow those of the enclosing class** if they have the same name.

In this case, the this keyword refers to the instances of the nested class and the members of the outer class can be referred to using the name of the outer class.

**5. Serialization**

To avoid a *java.io.NotSerializableException* while attempting to serialize a nested class, we should:

* Declare the nested class as *static*
* Make both the nested class and the enclosing class implement *Serializable*

**static nested class vs non-static nested class**

* 1) Nested static class doesn’t need reference of Outer class, but Non-static nested class or Inner class requires Outer class reference.
* 2) Inner class(or non-static nested class) can access both static and non-static members of Outer class. A static class cannot access non-static members of the Outer class. It can access only static members of Outer class.