Nested Class

The Java programming language allows you to define a class within another class. Such a class is called a *nested class* and is illustrated here:

class OuterClass {

...

class NestedClass {

...

}

}

Nested classes are divided into two categories: non-static and static.

Inner Class

Non-static nested classes are called *inner classes*.

Static Nested Class

Nested classes that are declared static are called *static nested classes*.

class OuterClass {

...

class InnerClass {

...

}

static class StaticNestedClass {

...

}

}

**Why Use Nested Classes?**

* **It is a way of logically grouping classes that are only used in one place**: If a class is useful to only one other class, then it is logical to embed it in that class and keep the two together. Nesting such "helper classes" makes their package more streamlined.
* **It increases encapsulation**: Consider two top-level classes, A and B, where B needs access to members of A that would otherwise be declared private. By hiding class B within class A, A's members can be declared private and B can access them. In addition, B itself can be hidden from the outside world.
* **It can lead to more readable and maintainable code**: Nesting small classes within top-level classes places the code closer to where it is used.

**Non-static nested classes are divided into two parts:**

**Local inner class and anonymous inner class.**

Local Inner Class

We can declare a class inside any block such as instance block or constructor or method or if block, such type of [inner classes](https://javabydeveloper.com/non-static-inner-class/) are called local inner classes

You can define a java local class inside any block. For example, you can define a local class in a method body, a for loop, or an if clause.

<https://javabydeveloper.com/java-local-class/>

Anonymous Inner Class

We can declare inner class without name such type of inner class are called anonymous class. Java anonymous class is an [inner class](https://javabydeveloper.com/non-static-inner-class/) with no name.

The main objective of anonymous inner classes is “just for instant use”.  
We may use java anonymous classes 3 ways

1. Anonymous inner class that extends a class.
2. Anonymous inner class that implements an interface.
3. Anonymous inner class that defined as arguments of method / constructor.

https://javabydeveloper.com/java-anonymous-class-with-examples/

Reflection Mechanism

In Java, reflection allows us to inspect and manipulate classes, interfaces, constructors, methods, and fields at run time.