<http://docs.oracle.com/javase/tutorial/java/javaOO/nested.html>

**Nested Classes**

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 {

...

}

}

**Terminology:** Nested classes are divided into two categories: static and non-static. Nested classes that are declared static are simply called *static nested classes*. Non-static nested classes are called *inner classes*.

class OuterClass {

...

static class StaticNestedClass {

...

}

class InnerClass {

...

}

}

A nested class is a member of its enclosing class. Non-static nested classes (inner classes) have access to other members of the enclosing class, even if they are declared private. Static nested classes do not have access to other members of the enclosing class. As a member of the OuterClass, a nested class can be declared private, public, protected, or *package private*. (Recall that outer classes can only be declared public or *package private*.)

**Why Use Nested Classes?**

There are several compelling reasons for using nested classes, among them:

* It is a way of logically grouping classes that are only used in one place.
* It increases encapsulation.
* Nested classes can lead to more readable and maintainable code.

**Logical grouping of classes**—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.

**Increased 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.

**More readable, maintainable code**—Nesting small classes within top-level classes places the code closer to where it is used.

**Static Nested Classes**

As with class methods and variables, a static nested class is associated with its outer class. And like static class methods, a static nested class cannot refer directly to instance variables or methods defined in its enclosing class — it can use them only through an object reference.

**Note:** A static nested class interacts with the instance members of its outer class (and other classes) just like any other top-level class. In effect, a static nested class is behaviorally a top-level class that has been nested in another top-level class for packaging convenience.

Static nested classes are accessed using the enclosing class name:

OuterClass.StaticNestedClass

For example, to create an object for the static nested class, use this syntax:

OuterClass.StaticNestedClass nestedObject =

new OuterClass.StaticNestedClass();

**Inner Classes**

As with instance methods and variables, an inner class is associated with an instance of its enclosing class and has direct access to that object's methods and fields. Also, because an inner class is associated with an instance, it cannot define any static members itself.

Objects that are instances of an inner class exist *within* an instance of the outer class. Consider the following classes:

class OuterClass {

...

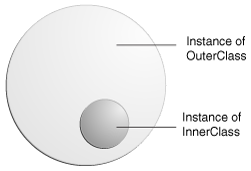
class InnerClass {

...

}

}

An instance of InnerClass can exist only within an instance of OuterClass and has direct access to the methods and fields of its enclosing instance. The next figure illustrates this idea.



An Instance of InnerClass Exists Within an Instance of OuterClass

To instantiate an inner class, you must first instantiate the outer class. Then, create the inner object within the outer object with this syntax:

OuterClass.InnerClass innerObject = outerObject.new InnerClass();

Additionally, there are two special kinds of inner classes: local classes and anonymous classes (also called anonymous inner classes). Both of these will be discussed briefly in the next section.

**Note:** If you want more information on the taxonomy of the different kinds of classes in the Java programming language (which can be tricky to describe concisely, clearly, and correctly), you might want to read Joseph Darcy's blog: [Nested, Inner, Member and Top-Level Classes](https://blogs.oracle.com/darcy/entry/nested_inner_member_and_top).