In Chapters 2 through 4 I laid a foundation for learning the Java language. In Chapter 5 I build onto this foundation by introducing you to some of Java’s more advanced language features, specifically, those features related to nested types, packages, static imports, and exceptions. Additional advanced language features are covered in Chapter 6.

**Mastering Nested Types**

Classes that are declared outside of any class are known as *top-level classes*. Java also supports *nested classes*, which are classes that are declared as members of other classes or scopes. Nested classes help you implement top-level class architecture.

There are four kinds of nested classes: static member classes, nonstatic member classes, anonymous classes, and local classes. The latter three categories are known as *inner classes*.

In this section I introduce you to static member classes and inner classes. For each kind of nested class, I provide you with a brief introduction, an abstract example, and a more practical example. I then briefly examine the topic of nesting interfaces within classes.

**Static Member Classes**

A *static member class* is a static member of an enclosing class. Although enclosed, it doesn’t have an enclosing instance of that class and cannot access the enclosing class’s instance fields and invoke its instance methods. However, it can access the enclosing class’s static fields and invoke its static methods, even those members that are declared private. Listing 5-1 presents a static member class declaration.
Listing 5-1. Declaring a Static Member Class

class EnclosingClass
{
    private static int i;

    private static void m1()
    {
        System.out.println(i);
    }

    static void m2()
    {
        EnclosedClass.accessEnclosingClass();
    }

    static class EnclosedClass
    {
        static void accessEnclosingClass()
        {
            i = 1;
            m1();
        }

        void accessEnclosingClass2()
        {
            m2();
        }
    }
}

Listing 5-1 declares a top-level class named EnclosingClass with class field i, class methods m1() and m2(), and static member class EnclosedClass. Also, EnclosedClass declares class method accessEnclosingClass() and instance method accessEnclosingClass2().

Because accessEnclosingClass() is declared static, m2() must be prefixed with EnclosedClass and the member access operator to call this method.

Listing 5-2 presents the source code to an application class that demonstrates how to invoke EnclosedClass's accessEnclosingClass() class method and instantiate EnclosedClass and invoke its accessEnclosingClass2() instance method.

Listing 5-2. Invoking a Static Member Class's Class and Instance Methods

public class SMCDemo
{
    public static void main(String[] args)
    {
        EnclosingClass.EnclosedClass.accessEnclosingClass(); // Output: 1
        EnclosingClass.EnclosedClass ec = new EnclosingClass.EnclosedClass();
        ec.accessEnclosingClass2(); // Output: 1
    }
}