**Nested and inner classes**

The following program illustrates how to define and use an inner class. The class named

**Outer** has one instance variable named **outer\_x**, one instance method named **test( )**, and

defines one inner class called **Inner**.

// Demonstrate an inner class.

**class** Outer {

**int** outer\_x = 100;

**void** test() {

Inner inner = **new** Inner();

inner.display();

}

// this is an inner class

**class** Inner {

**void** display() {

System.*out*.println("display: outer\_x = " + outer\_x);

}

}

}

**class** InnerClassDemo {

**public** **static** **void** main(String args[]) {

Outer outer = **new** Outer();

outer.test();

}

}

**Result:**

display: outer\_x = 100

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Although we have been focusing on inner classes declared as members within an outer

class scope, it is possible to define inner classes within any block scope. For example, you

can define a nested class within the block defined by a method or even within the body of

a **for** loop, as this next program shows:

// Define an inner class within a for loop.

**class** Outer {

**int** outer\_x = 100;

**void** test() {

**for**(**int** i=0; i<10; i++) {

**class** Inner {

**void** display() {

System.*out*.println("display: outer\_x = " + outer\_x);

}

}

Inner inner = **new** Inner();

inner.display();

}

}

}

**class** InnerClassDemo {

**public** **static** **void** main(String args[]) {

Outer outer = **new** Outer();

outer.test();

}

}

**Result:**

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

display: outer\_x = 100

While nested classes are not applicable to all situations, they are particularly helpful

when handling events.