**Understanding structure and class differences**

A structure and a class are syntactically similar, but there are a few important differences. Let’s look at

some of these variances:

■ ■ You can’t declare a default constructor (a constructor with no parameters) for a structure. The

following example would compile if Time were a class, but because Time is a structure, it does

not:

*struct Time*

*{*

*public Time() { ... } // compile-time error*

*...*

*}*

The reason you can’t declare your own default constructor for a structure is that the compiler

always generates one. In a class, the compiler generates the default constructor only if you

don’t write a constructor yourself. The compiler-generated default constructor for a structure

always sets the fields to 0, false, or null—just as for a class. Therefore, you should ensure that

a structure value created by the default constructor behaves logically and makes sense with

these default values. This has some ramifications that you will explore in the next exercise.

You can initialize fields to different values by providing a nondefault constructor. However,

when you do this, your nondefault constructor must explicitly initialize all fields in your

structure; the default initialization no longer occurs. If you fail to do this, you’ll get a compile-

time error. For example, although the following example would compile and silently initialize

seconds to 0 if Time were a class, because Time is a structure, it fails to compile:

*struct Time*

*{*

*private int hours, minutes, seconds;*

*...*

*public Time(int hh, int mm)*

*{*

*this.hours = hh;*

*this.minutes = mm;*

*} // compile-time error: seconds not initialized*

*}*

■ ■ In a class, you can initialize instance fields at their point of declaration. In a structure, you can-

not. The following example would compile if Time were a class, but because Time is a struc-

ture, it causes a compile-time error:

*struct Time*

*{*

*private int hours = 0; // compile-time error*

*private int minutes;*

*private int seconds;*

*...*

*}*

***Important*** Inheritance applies only to classes, not structures. You cannot define your own

inheritance hierarchy with structures, and you cannot define a structure that derives from a

class or another structure.

All structures actually inherit from an abstract class called System.ValueType.