Chapter 8

Additional Items

To date we have considered the following class items:

- controlling access - public and private data members;
- controlling access - public and private methods;
- constructors;
- overloaded constructors;

properties

A property is a special way to retrieve information, public or private, from an object in a class, or to assign a value to a data item, public or private, while controlling how the result is obtained.

An example from the text

```
// Fig. 8.6: Time3.cs
// Class Time2 provides overloaded constructors.

using System;

namespace TimeTest3
{
   // Time3 class definition
   public class Time3 : Object
   {
      private int hour;    // 0-23
      private int minute;  // 0-59
      private int second;  // 0-59
```

Constructors

```
// Time3 constructor initializes instance variables to
      // zero to set default time to midnight
      public Time3()
      {
         SetTime( 0, 0, 0 );
      }

      // Time3 constructor: hour supplied, minute and second
      // defaulted to 0
      public Time3( int hour )
      {
         SetTime( hour, 0, 0 );
      }
```

Constructors cont.

```
// Time3 constructor: hour and minute supplied, second
      // defaulted to 0
      public Time3( int hour, int minute )
      {
         SetTime( hour, minute, 0 );
      }

      // Time3 constructor: hour, minute and second supplied
      public Time3( int hour, int minute, int second )
      {
         SetTime( hour, minute, second );
      }

      // Time3 constructor: initialize using another Time3 object
      public Time3( Time3 time )
      {
         SetTime( time.Hour, time.Minute, time.Second );
      }
```

Method to set values for an existing object in the Time class

```
// Set new time value in 24-hour format. Perform validity
 // checks on the data. Set invalid values to zero.
      public void SetTime(
         int hourValue, int minuteValue, int secondValue )
      {
         Hour = hourValue;       // invoke Hour property set
         Minute = minuteValue;   // invoke Minute property set
         Second = secondValue;   // invoke Second property set
      }
```

Hour Property

```
// property Hour
      public int Hour
      {
         get
         {
            return hour;
         }

         set
         {
            hour = ( ( value >= 0 && value < 24 ) ? value : 0 );
         }

      } // end property Hour
```

Minute property

```
// property Minute
      public int Minute
      {
         get
         {
            return minute;
         }

         set
         {
            minute = ( ( value >= 0 && value < 60 ) ? value : 0 );
         }

      } // end property Minute
```

Second property

```
// property Second
      public int Second
      {
         get
         {
            return second;
         }

         set
         {
            second = ( ( value >= 0 && value < 60 ) ? value : 0 );
         }

      } // end property Second
```

Two Final Methods

```
// convert time to universal-time (24 hour) format string
      public string ToUniversalString()
      {
         return String.Format(
            "{0:D2}:{1:D2}:{2:D2}", Hour, Minute, Second );
      }

      // convert time to standard-time (12 hour) format string
      public string ToStandardString()
      {
         return String.Format( "{0}:{1:D2}:{2:D2} {3}",
            ( ( Hour == 12 || Hour == 0 ) ? 12 : Hour % 12 ),
            Minute, Second, ( Hour < 12 ? "AM" : "PM" ) );
      }

   } // end class Time3
}
```

this

"Every object can access a reference to itself, called the ***this*** reference." (Deitel)

To date, in writing methods we have used the ***this*** reference; e.g., this.numer and this.denom in our Rational class. Actually, in the cases where we used it, it was not necessary to include this. before the variable name, since it would be understood. We used it to emphasize that ***this*** references the object in question.

Why have it then?

- In a method, you might want to return a reference to the object.
- If a parameter in a method header has the exact name as one of the class data items, then this can be used to differentiate the parameter and the data item.

  Example: this.numer = numer;

  The first numer now means the object data, and the second the local parameter within the method.

static data members

Sometimes when developing classes for object oriented programming, it may be necessary to maintain one or more data items that are common to all objects that are instantiated at a given time.

Read the first paragraph on page 313 to see a good example of where this might occur.