Note: *C How to Program*, Chapter 17 is a copy of *C++ How to Program* Chapter 9. We have not renumbered the PowerPoint Slides.

Chapter 9
Classes: A Deeper Look, Part 1

C++ How to Program, 8/e

9.2 Time Class Case Study

▸ Our first example (Figs. 9.1–9.3) creates class Time and a driver program that tests the class.

```
 1   // Fig. 9.1: Time.h
 2   // Time class definition.
 3   // Member functions are defined in Time.cpp
 4
 5   // prevent multiple inclusions of header
 6   #ifndef TIME_H
 7   #define TIME_H
 8
 9   // Time class definition
10   class Time
11   {
12   public:
13      Time(); // constructor
14      void setTime( int, int, int ); // set hour, minute and second
15      void printUniversal(); // print time in universal-time format
16      void printStandard(); // print time in standard-time format
17   private:
18      int hour; // 0 - 23 (24-hour clock format)
19      int minute; // 0 - 59
20      int second; // 0 - 59
21   }; // end class Time
22
23   #endif
```

Fig. 9.1 | Time class definition.

```
 1   // Fig. 9.2: Time.cpp
 2   // Member-function definitions for class Time.
 3   #include <iostream>
 4   #include <iomanip>
 5   #include <stdexcept> // for invalid_argument exception class
 6   #include "Time.h" // include definition of class Time from Time.h
 7
 8   using namespace std;
 9
10   // Time constructor initializes each data member to zero.
11   Time::Time()
12   {
13      hour = minute = second = 0;
14   } // end Time constructor
15
```

Fig. 9.2 | Time class member-function definitions. (Part 1 of 3.)

```
16    // set new Time value using universal time
17    void Time::setTime( int h, int m, int s )
18    {
19       // validate hour, minute and second
20       if ( ( h >= 0 && h < 24 ) && ( m >= 0 && m < 60 ) &&
21          ( s >= 0 && s < 60 ) )
22       {
23          hour = h;
24          minute = m;
25          second = s;
26       } // end if
27       else
28          throw invalid_argument(
29             "hour, minute and/or second was out of range" );
30    } // end function setTime
31
```

Fig. 9.2 | Time class member-function definitions. (Part 2 of 3.)

```
32    // print Time in universal-time format (HH:MM:SS)
33    void Time::printUniversal()
34    {
35       cout << setfill( '0' ) << setw( 2 ) << hour << ":"
36          << setw( 2 ) << minute << ":" << setw( 2 ) << second;
37    } // end function printUniversal
38
39    // print Time in standard-time format (HH:MM:SS AM or PM)
40    void Time::printStandard()
41    {
42       cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ) << ":"
43          << setfill( '0' ) << setw( 2 ) << minute << ":" << setw( 2 )
44          << second << ( hour < 12 ? " AM" : " PM" );
45    } // end function printStandard
```

Fig. 9.2 | Time class member-function definitions. (Part 3 of 3.)

9.2 Time Class Case Study (cont.)

▸ Once class Time has been defined, it can be used as a type in object, array, pointer and reference declarations as follows:

```
Time sunset; // object of type Time
Time arrayOfTimes[ 5 ]; // array of 5 Time objects
Time &dinnerTime = sunset; // reference to a Time object
Time *timePtr = &dinnerTime; // pointer to a Time object
```

```cpp
1   // Fig. 9.3: fig09_03.cpp
2   // Program to test class Time.
3   // NOTE: This file must be compiled with Time.cpp.
4   #include <iostream>
5   #include "Time.h" // include definition of class Time from Time.h
6   using namespace std;
7
8   int main()
9   {
10      Time t; // instantiate object t of class Time
11
12      // output Time object t's initial values
13      cout << "The initial universal time is ";
14      t.printUniversal(); // 00:00:00
15      cout << "\nThe initial standard time is ";
16      t.printStandard(); // 12:00:00 AM
17
18      t.setTime( 13, 27, 6 ); // change time
19
```

Fig. 9.3 | Program to test class Time. (Part 1 of 3.)

```
20        // output Time object t's new values
21        cout << "\n\nUniversal time after setTime is ";
22        t.printUniversal(); // 13:27:06
23        cout << "\nStandard time after setTime is ";
24        t.printStandard(); // 1:27:06 PM
25
26        // attempt to set the time with invalid values
27        try
28        {
29           t.setTime( 99, 99, 99 ); // all values out of range
30        } // end try
31        catch ( invalid_argument &e )
32        {
33           cout << "Exception: " << e.what() << endl << endl;
34        } // end catch
35
36        // output t's values after specifying invalid values
37        cout << "\n\nAfter attempting invalid settings:"
38           << "\nUniversal time: ";
39        t.printUniversal(); // 00:00:00
40        cout << "\nStandard time: ";
41        t.printStandard(); // 12:00:00 AM
42        cout << endl;
43     } // end main
```

Fig. 9.3 | Program to test class Time. (Part 2 of 3.)

```
The initial universal time is 00:00:00
The initial standard time is 12:00:00 AM

Universal time after setTime is 13:27:06
Standard time after setTime is 1:27:06 PM

Exception: hour, minute and/or second was out of range

After attempting invalid settings:
Universal time: 13:27:06
Standard time: 1:27:06 PM
```

Fig. 9.3 | Program to test class Time. (Part 3 of 3.)

9.6 Time Class Case Study: Constructors with Default Arguments

▸ Like other functions, constructors can specify default arguments.

▸ The default arguments to the constructor ensure that, even if no values are provided in a constructor call, the constructor still initializes the data.

▸ A constructor that defaults all its arguments is also a default constructor—i.e., a constructor that can be invoked with no arguments.

▸ There can be at most one default constructor per class.

```
 1   // Fig. 9.8: Time.h
 2   // Time class containing a constructor with default arguments.
 3   // Member functions defined in Time.cpp.
 4
 5   // prevent multiple inclusions of header
 6   #ifndef TIME_H
 7   #define TIME_H
 8
 9   // Time abstract data type definition
10   class Time
11   {
12   public:
13      Time( int = 0, int = 0, int = 0 ); // default constructor
14
15      // set functions
16      void setTime( int, int, int ); // set hour, minute, second
17      void setHour( int ); // set hour (after validation)
18      void setMinute( int ); // set minute (after validation)
19      void setSecond( int ); // set second (after validation)
20
```

Fig. 9.8 | Time class containing a constructor with default arguments. (Part 1 of 2.)

```
 1   // Fig. 9.9: Time.cpp
 2   // Member-function definitions for class Time.
 3   #include <iostream>
 4   #include <iomanip>
 5   #include <stdexcept>
 6   #include "Time.h" // include definition of class Time from Time.h
 7   using namespace std;
 8
 9   // Time constructor initializes each data member to zero
10   Time::Time( int hour, int minute, int second )
11   {
12      setTime( hour, minute, second ); // validate and set time
13   } // end Time constructor
14
15   // set new Time value using universal time
16   void Time::setTime( int h, int m, int s )
17   {
18      setHour( h ); // set private field hour
19      setMinute( m ); // set private field minute
20      setSecond( s ); // set private field second
21   } // end function setTime
22
```

Fig. 9.9 | Time class member-function definitions including a constructor that takes arguments. (Part 1 of 4.)

```
 1   // Fig. 9.10: fig09_10.cpp
 2   // Demonstrating a default constructor for class Time.
 3   #include <iostream>
 4   #include <stdexcept>
 5   #include "Time.h" // include definition of class Time from Time.h
 6   using namespace std;
 7
 8   int main()
 9   {
10      Time t1; // all arguments defaulted
11      Time t2( 2 ); // hour specified; minute and second defaulted
12      Time t3( 21, 34 ); // hour and minute specified; second defaulted
13      Time t4( 12, 25, 42 ); // hour, minute and second specified
14
```

Fig. 9.10 | Constructor with default arguments. (Part 1 of 4.)

```
15      cout << "Constructed with:\n\nt1: all arguments defaulted\n   ";
16      t1.printUniversal(); // 00:00:00
17      cout << "\n   ";
18      t1.printStandard(); // 12:00:00 AM
19
20      cout << "\n\nt2: hour specified; minute and second defaulted\n   ";
21      t2.printUniversal(); // 02:00:00
22      cout << "\n   ";
23      t2.printStandard(); // 2:00:00 AM
24
25      cout << "\n\nt3: hour and minute specified; second defaulted\n   ";
26      t3.printUniversal(); // 21:34:00
27      cout << "\n   ";
28      t3.printStandard(); // 9:34:00 PM
29
30      cout << "\n\nt4: hour, minute and second specified\n   ";
31      t4.printUniversal(); // 12:25:42
32      cout << "\n   ";
33      t4.printStandard(); // 12:25:42 PM
34
```

Fig. 9.10 | Constructor with default arguments. (Part 2 of 4.)
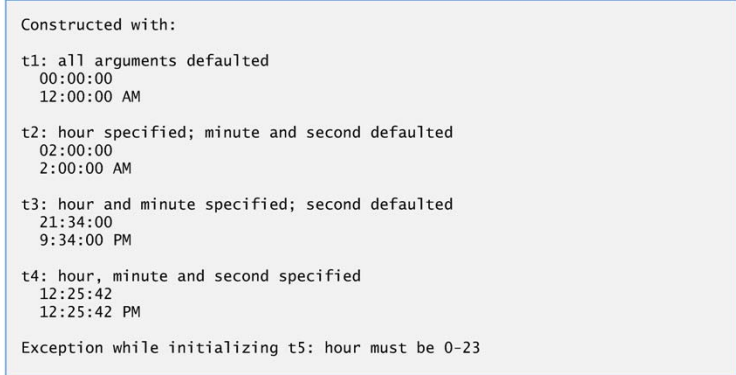
```
35      // attempt to initialize t6 with invalid values
36      try
37      {
38         Time t5( 27, 74, 99 ); // all bad values specified
39      } // end try
40      catch ( invalid_argument &e )
41      {
42         cout << "\n\nException while initializing t5: " << e.what() << endl;
43      } // end catch
44   } // end main
```

Fig. 9.10 | Constructor with default arguments. (Part 3 of 4.)

```
Constructed with:

t1: all arguments defaulted
   00:00:00
   12:00:00 AM

t2: hour specified; minute and second defaulted
   02:00:00
   2:00:00 AM

t3: hour and minute specified; second defaulted
   21:34:00
   9:34:00 PM

t4: hour, minute and second specified
   12:25:42
   12:25:42 PM

Exception while initializing t5: hour must be 0-23
```

Fig. 9.10 | Constructor with default arguments. (Part 4 of 4.)

9.7 Destructors

▸ The name of the destructor for a class is the tilde character (~) followed by the class name.
▸ Often referred to with the abbreviation "dtor" in the literature.
▸ Called implicitly when an object is destroyed.
▸ *The destructor itself does not actually release the object's memory—it performs termination housekeeping before the object's memory is reclaimed, so the memory may be reused to hold new objects.*
▸ Receives no parameters and returns no value.
▸ May not specify a return type—not even void.
▸ A class may have only one destructor.
▸ A destructor must be public.
▸ If you do not explicitly provide a destructor, the compiler creates an "empty" destructor.