

Note: *C How to Program*, Chapter 18is a copy of *C++ How to Program* Chapter 10. We have not renumbered the PowerPoint Slides.

Chapter 10 Classes: A Deeper Look, Part 2

C++ How to Program, 8/e

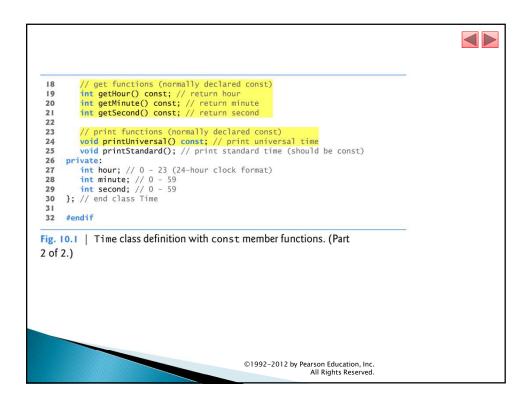
©1992-2012 by Pearson Education, Inc. All Rights Reserved



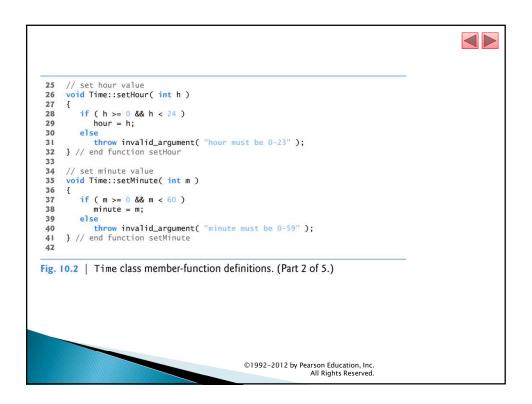
10.2 const (Constant) Objects and const Member Functions

- You may use keyword **CONSt** to specify that an object is not modifiable and that any attempt to mod-ify the object should result in a compilation error.
- ▶ C++ disallows member function calls for **CONSt** objects unless the member functions themselves are also declared **CONST**.
 - True even for *get member functions that do not modify the object.*
- A member function is specified as **const** both in its prototype and in its definition.

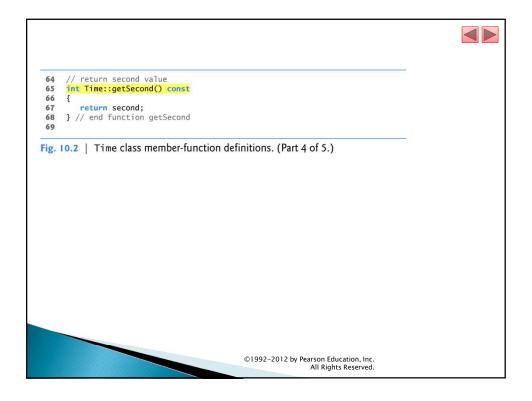
```
// Fig. 10.1: Time.h
// Time class definition with const member functions.
       // Member functions defined in Time.cpp.
#ifndef TIME_H
#define TIME_H
        class Time
  8
       public:
 10
            Time( int = 0, int = 0, int = 0 ); // default constructor
 11
 12
            // set functions
            // set functions
void setTime( int, int, int ); // set time
void setHour( int ); // set hour
void setMinute( int ); // set minute
void setSecond( int ); // set second
 13
 15
 16
Fig. 10.1 | Time class definition with const member functions. (Part
I of 2.)
                                                                   ©1992-2012 by Pearson Education, Inc.
                                                                                        All Rights Reserved.
```



```
// Fig. 10.2: Time.cpp
// Time class member-function definitions.
       // Time class memoer-function definitions.
#include <iostream>
#include <iomanip>
#include <stdexcept>
#include "Time.h" // include definition of class Time
       using namespace std;
       // constructor function to initialize private data;
       // colls member function setTime to set variables; // default values are 0 (see class definition)
 10
 11
 12
       Time::Time( int hour, int minute, int second )
 13
            setTime( hour, minute, second );
 15
      } // end Time constructor
 16
       // set hour, minute and second values
void Time::setTime( int hour, int minute, int second )
 17
 18
 19
       {
 20
            setHour( hour );
           setMinute( minute );
setSecond( second );
 21
 22
      } // end function setTime
 23
Fig. 10.2 | Time class member-function definitions. (Part 1 of 5.)
                                                              ©1992-2012 by Pearson Education, Inc.
                                                                                  All Rights Reserved.
```



```
// set second value
void Time::setSecond( int s )
 44
 45
      {
          if ( s >= 0 && s < 60 )
 46
 47
48
              second = s;
          else
 49
              throw invalid_argument( "second must be 0-59" );
 50 } // end function setSecond
 51
      // return hour value
int Time::getHour() const // get functions should be const
 52
 53
54 {
55
     return hour;
} // end function getHour
 56
 57
      // return minute value
int Time::getMinute() const
 58
 59
 60 {
 61
     return minute;
} // end function getMinute
 62
Fig. 10.2 | Time class member-function definitions. (Part 3 of 5.)
                                                       ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```



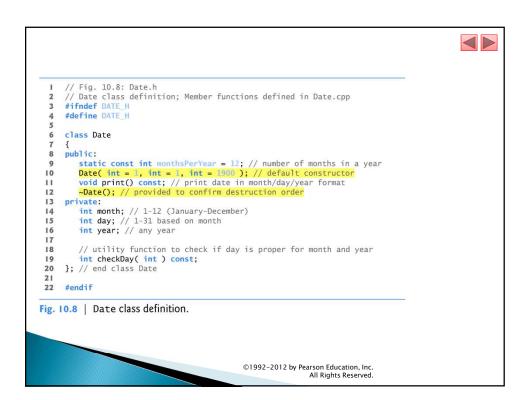
```
// print Time in universal-time format (HH:MM:SS)
void Time::printUniversal() const
 71
 72
       {
      73
 74
75
       // print Time in standard-time format (HH:MM:SS AM or PM) void Time::printStandard() // note lack of const declaration
 77
 78
 79
           cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
      << ":" << setfill( '0' ) << setw( 2 ) << minute
      << ":" << setw( 2 ) << second << ( hour < 12 ? " AM" : " PM" );</pre>
 80
 81
 82
 83 } // end function printStandard
Fig. 10.2 | Time class member-function definitions. (Part 5 of 5.)
                                                              ©1992-2012 by Pearson Education, Inc.
                                                                                  All Rights Reserved.
```

```
// Fig. 10.3: fig10_03.cpp
// Attempting to access a const object with non-const member functions.
#include "Time.h" // include Time class definition
  3
  4
5
      int main()
  6
         Time wakeUp( 6, 45, 0 ); // non-constant object
const Time noon( 12, 0, 0 ); // constant object
         MEMBER FUNCTION
 10
 11
 12
 13
         noon.setHour( 12 ); // const
 14
15
         wakeUp.getHour();
                                    // non-const const
         17
                                                     const
 18
                                                     const
 19
    noon.printStandard(); // const
} // end main
 20
                                                     non-const
 21
Fig. 10.3 | const objects and const member functions. (Part I of
2.)
                                                   ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```

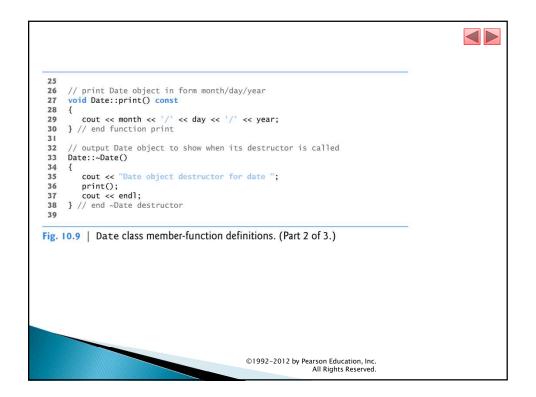


10.3 Composition: Objects as Members of Classes

- **▶** Composition
 - Sometimes referred to as a has-a relationship
 - A class can have objects of other classes as members
- An object's constructor can pass arguments to memberobject constructors via member initializers.



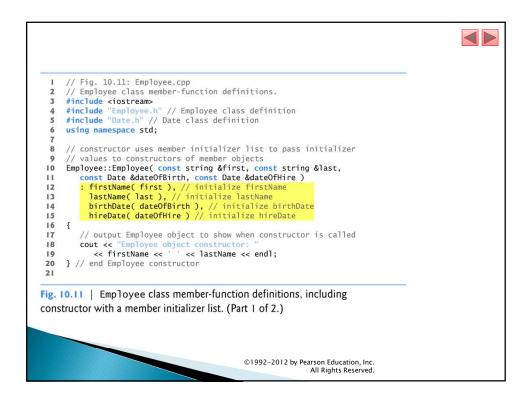
```
// Fig. 10.9: Date.cpp
// Date class member-function definitions.
      #include <iostream>
#include <stdexcept>
#include "Date.h" // include Date class definition
      using namespace std;
      // constructor confirms proper value for month; calls
// utility function checkDay to confirm proper value for day
Date::Date( int mn, int dy, int yr )
 10
 11
           if ( mn > 0 && mn <= months
PerYear ) // validate the month
 12
 13
              month = mn;
 15
              throw invalid_argument( "month must be 1-12" );
 16
 17
           year = yr; // could validate yr
 18
           day = checkDay( dy ); // validate the day
 19
           // output Date object to show when its constructor is called
 20
 21
           cout << "Date object constructor for date ";</pre>
 22
          print();
     cout << endl;
} // end Date constructor</pre>
 23
Fig. 10.9 | Date class member-function definitions. (Part 1 of 3.)
                                                         ©1992-2012 by Pearson Education, Inc.
                                                                            All Rights Reserved.
```

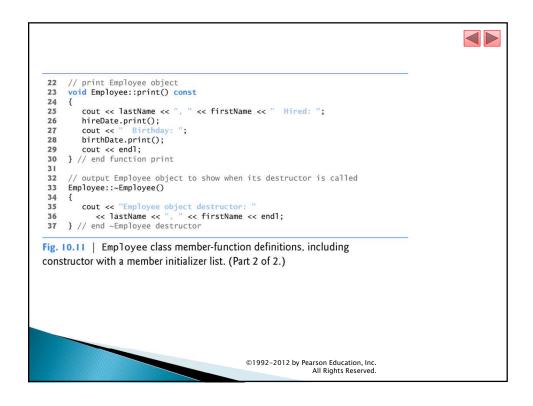


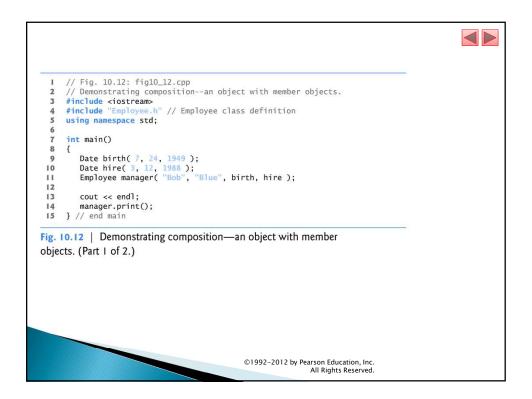
```
// utility function to confirm proper day value based on
// month and year; handles leap years, too
int Date::checkDay( int testDay ) const
 41
 42
 43
             static const int daysPerMonth[ monthsPerYear + 1 ] =
    { 0, 31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31, };
 44
45
  46
             // determine whether testDay is valid for specified month
if ( testDay > 0 && testDay <= daysPerMonth[ month ] )
    return testDay;</pre>
  47
 48
  49
  50
             // February 29 check for leap year
if ( month == 2 && testDay == 29 && ( year % 400 == 0 ||
    ( year % 4 == 0 && year % 100 != 0 ) ) )
 51
 52
  53
 54
55
                  return testDay;
             throw invalid_argument( "Invalid day for current month and year" );
 57 } // end function checkDay
Fig. 10.9 | Date class member-function definitions. (Part 3 of 3.)
                                                                        ©1992-2012 by Pearson Education, Inc.
                                                                                               All Rights Reserved.
```

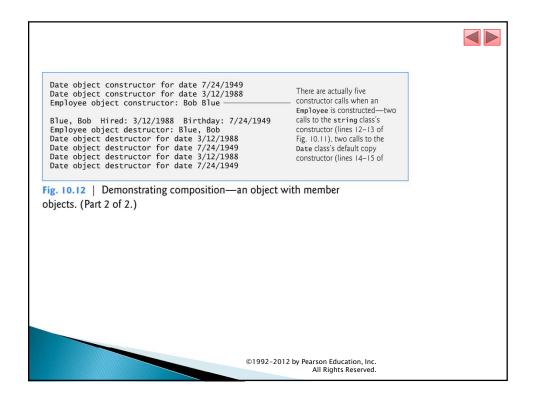


```
class Employee
 12
       public:
 13
           Employee( const string &, const string &,
           const Date &, const Date & );
void print() const;
~Employee(); // provided to confirm destruction order
 15
 16
  17
 18
       private:
           string firstName; // composition: member object
string lastName; // composition: member object
 19
 20
           const Date birthDate; // composition: member object
const Date hireDate; // composition: member object
 21
 22
 23
       }; // end class Employee
 24
 25
Fig. 10.10 | Employee class definition showing composition. (Part 2
of 2.)
                                                              ©1992-2012 by Pearson Education, Inc.
                                                                                  All Rights Reserved.
```







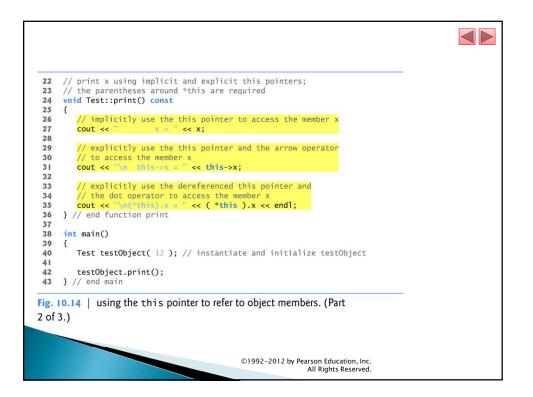


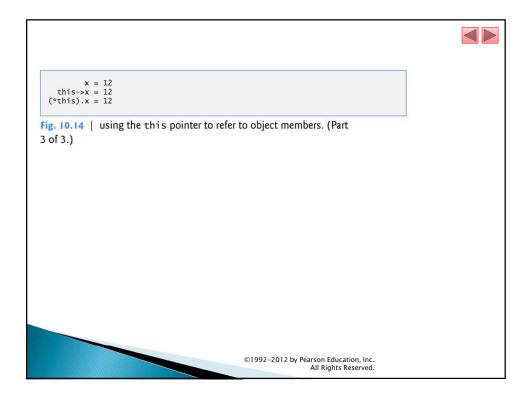


10.5 Using the this Pointer

- How do member functions know *which* object's data members to manipulate? Every object has access to its own address through a pointer called this (a C++ keyword).
- The this pointer is not part of the object itself.
 - The this pointer is passed (by the compiler) as an implicit argument to each of the object's non-static member functions.
- Description Objects use the this pointer implicitly or explicitly to reference their data members and member functions.
- The type of the this pointer depends on the type of the object and whether the member function in which this is used is declared const.

```
// Fig. 10.14: fig10_14.cpp
// Using the this pointer to refer to object members.
#include <iostream>
       using namespace std;
  6
       class Test
       public:
  8
           Test( int = 0 ); // default constructor
void print() const;
  11
 12
      int x;
}; // end class Test
 13
       // constructor
Test::Test( int value )
    : x( value ) // initialize x to value
 15
 16
 18
           // empty body
 19
      } // end constructor Test
 20
Fig. 10.14 | using the this pointer to refer to object members. (Part
I of 3.)
                                                           ©1992-2012 by Pearson Education, Inc.
                                                                              All Rights Reserved.
```



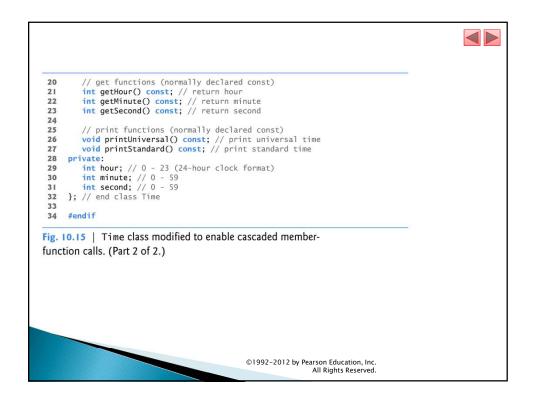




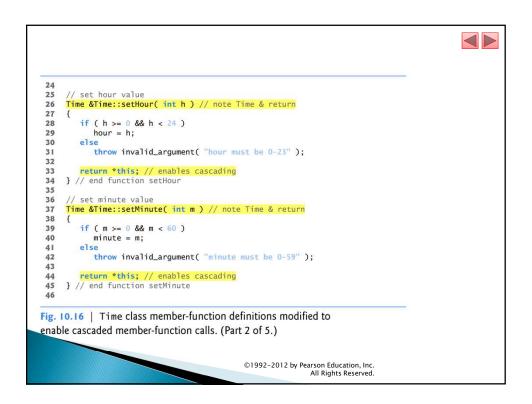
10.5 Using the this Pointer (cont.)

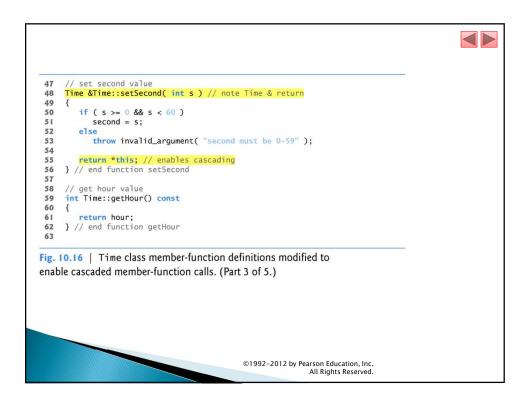
- Another use of the this pointer is to enable cascaded member-function calls
 - invoking multiple functions in the same statement
- ▶ The program of Figs. 10.15–10.17 modifies class Time's set functions setTime, setHour, set-Minute and setSecond such that each returns a reference to a Time object to enable cascaded member-function calls.

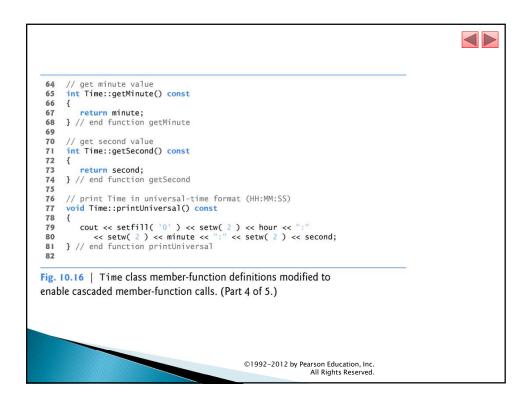
```
// Fig. 10.15: Time.h
// Cascading member function calls.
      // Time class definition.
      // Member functions defined in Time.cpp.
       #ifndef TIME_H
       #define TIME_H
      class Time
  9
 10
 11
      public:
          Time( int = 0, int = 0, int = 0 ); // default constructor
 12
 13
             set functions (the Time & return types enable cascading)
          Time &setMour( int ); // set hour
Time &setMour( int ); // set hour
Time &setMinute( int ); // set minute
Time &setSecond( int ); // set second
 15
 16
 18
 19
Fig. 10.15 | Time class modified to enable cascaded member-
function calls. (Part 1 of 2.)
                                                       ©1992-2012 by Pearson Education, Inc.
                                                                         All Rights Reserved.
```

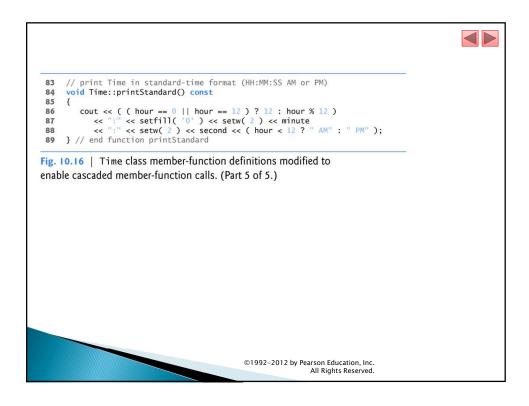


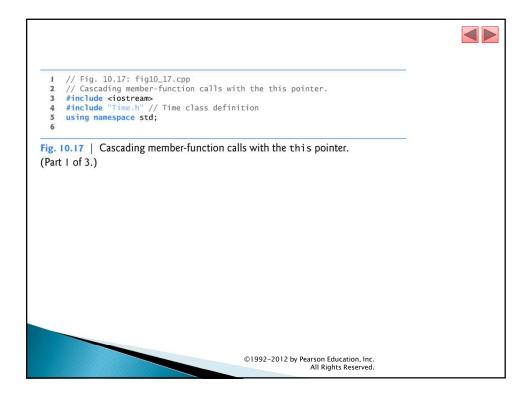
```
// Fig. 10.16: Time.cpp
// Time class member-function definitions.
       #include <iostream>
#include <iomanip>
#include "Time.h" //
                                   // Time class definition
        using namespace std;
       // constructor function to initialize private data;
// calls member function setTime to set variables;
// default values are 0 (see class definition)
  11
        Time::Time( int hr, int min, int sec )
 12
 13
            setTime( hr, min, sec );
 15
        // set values of hour, minute, and second
Time &Time::setTime( int h, int m, int s ) // note Time & return
 16
  17
 18
            setHour( h );
setMinute( m );
 19
 20
      setSecond( s );
return *this; // enables cascading
} // end function setTime
 21
 22
 23
Fig. 10.16 | Time class member-function definitions modified to
enable cascaded member-function calls. (Part 1 of 5.)
                                                                   ©1992-2012 by Pearson Education, Inc.
                                                                                        All Rights Reserved.
```











```
int main()
  7
            Time t; // create Time object
 10
11
12
13
            // cascaded function calls
t.setHour( 18 ).setMinute( 30 ).setSecond( 22 );
            // output time in universal and standard formats
cout << "Universal time: ";
t.printUniversal();</pre>
 14
15
16
17
            cout << "\nStandard time: ";
t.printStandard();</pre>
 18
 19
 21
22
            cout << "\n\nNew standard time: ";</pre>
            // cascaded function calls
      t.setTime( 20, 20, 20 ).printStandard();
cout << endl;
} // end main</pre>
 24
25
 26
Fig. 10.17 | Cascading member-function calls with the this pointer.
(Part 2 of 3.)
                                                                 ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```

