

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 Reserve

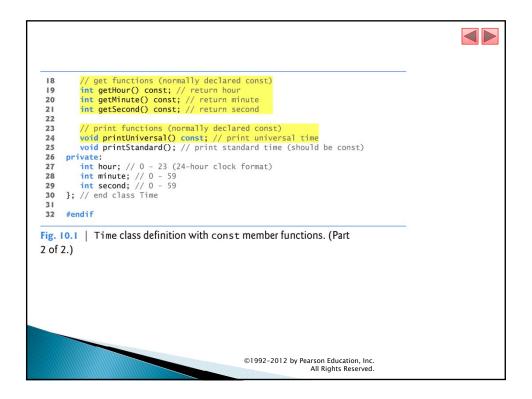


## 10.2 const (Constant) Objects and const Member Functions

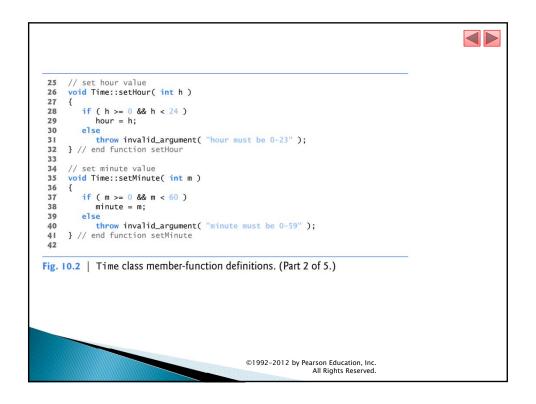
- You may use keyword const to specify that an object is not modifiable and that any attempt to mod-ify the object should result in a compilation error.
- C++ disallows member function calls for const objects unless the member functions themselves are also declared const.
  - True even for *get member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

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

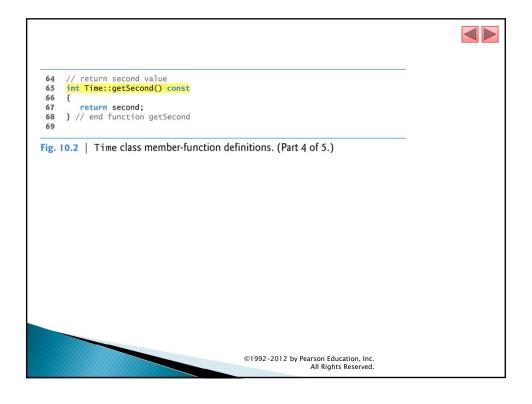
```
// Fig. 10.1: Time.h
// Time class definition with const member functions.
// Member functions defined in Time.cpp.
         #ifndef TIME_H
         #define TIME_H
         class Time
   8
        public:
   9
  10
             Time( int = 0, int = 0, int = 0 ); // default constructor
  11
             // set functions
void setTime( int, int, int ); // set time
void setHour( int ); // set hour
void setMinute( int ); // set minute
void setSecond( int ); // set second
  12
  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.
       #include <iostream>
#include <iomanip>
       #include <stdexcept>
#include "Time.h" // include definition of class Time
       using namespace std;
      // constructor function to initialize private data;
// calls member function setTime to set variables;
// default values are 0 (see class definition)
 10
 11
       Time::Time( int hour, int minute, int second )
 12
 13
           setTime( hour, minute, second );
      } // end Time constructor
 15
 16
       // set hour, minute and second values
void Time::setTime( int hour, int minute, int second )
 17
 18
 19
       {
 20
           setHour( hour );
 21
           setMinute( minute );
           setSecond( second );
 22
      } // end function setTime
 23
 24
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 )
second = s;
 46
 47
48
           else
              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
      {
 55    return hour;
56 } // end function getHour
 57
     // return minute value
int Time::getMinute() const
 58
 59
 60
 61    return minute;
62 } // end function getMinute
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
   77
78
79
  80
81
82
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
      int main()
  6
          Time wakeUp( 6, 45, 0 ); // non-constant object
const Time noon( 12, 0, 0 ); // constant object
  7
  8
          wakeUp.setHour( 18 ); // OBJECT MEMBER FUN
non-const
 10
                                                        MEMBER FUNCTION
 12
 13
          noon.setHour( 12 ); // const
 14
15
          wakeUp.getHour();
                                     // non-const const
          noon.getMinute();  // const
noon.printUniversal(); // const
 17
                                                        const
 18
                                                        const
 19
non-const
Fig. 10.3 | const objects and const member functions. (Part I of
2.)
                                                     ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```

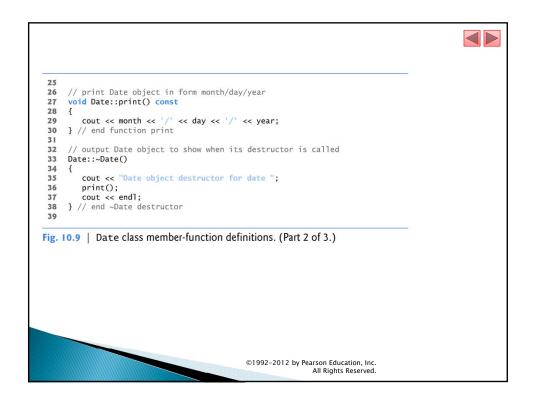
## 10.3 Composition: Objects as Members of Classes

- **▶** 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.

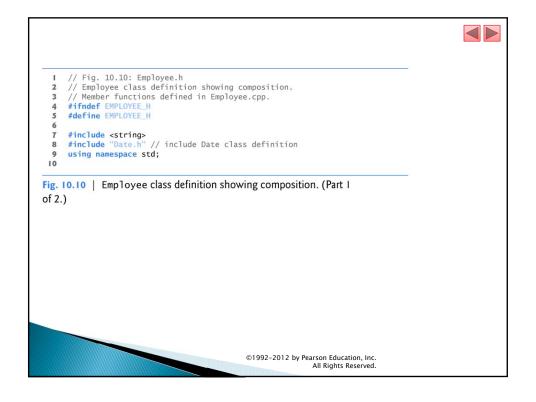
©1992-2012 by Pearson Education, Inc. All Rights Reserved.

```
// Fig. 10.8: Date.h
         // Date class definition; Member functions defined in Date.cpp
         #ifndef DATE_H
         #define DATE_H
         class Date
   7
        public:
              static const int monthsPerYear = 12; // number of months in a year
Date( int = 1, int = 1, int = 1900 ); // default constructor
void print() const; // print date in month/day/year format
~Date(); // provided to confirm destruction order
  11
  13
              int month; // 1-12 (January-December)
int day; // 1-31 based on month
int year; // any year
 14
15
  17
              // utility function to check if day is proper for month and year int\ checkDay(\ int\ )\ const;
  18
  19
 20
       }; // end class Date
 21
        #endif
Fig. 10.8 | Date class definition.
                                                                           ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```

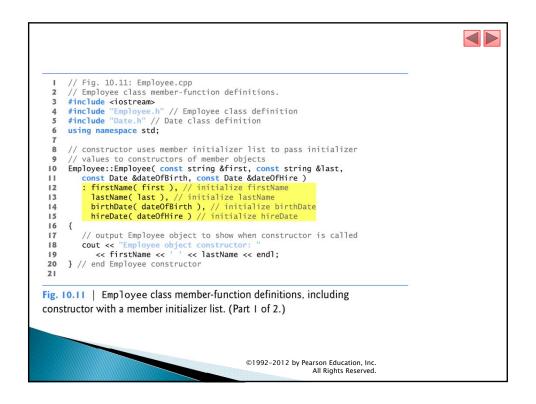
```
// Fig. 10.9: Date.cpp
// Date class member-function definitions.
       #include <iostream>
      #include <stdexcept>
#include "Date.h" // include Date class definition
using namespace std;
      // constructor confirms proper value for month; calls
// utility function checkDay to confirm proper value for day
Date::Date( int mn, int dy, int yr )
 10
 11
           if ( mn > 0 && mn \leftarrow monthsPerYear ) // validate the month
 12
              month = mn;
 13
  14
              throw invalid_argument( "month must be 1-12" );
 15
 16
 17
           year = yr; // could validate yr
 18
           day = checkDay( dy ); // validate the day
 19
 20
           // output Date object to show when its constructor is called
           cout << "Date object constructor for date ";</pre>
 22
           print();
           cout << endl;
 23
 24
      } // end Date constructor
Fig. 10.9 | Date class member-function definitions. (Part 1 of 3.)
                                                          ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```



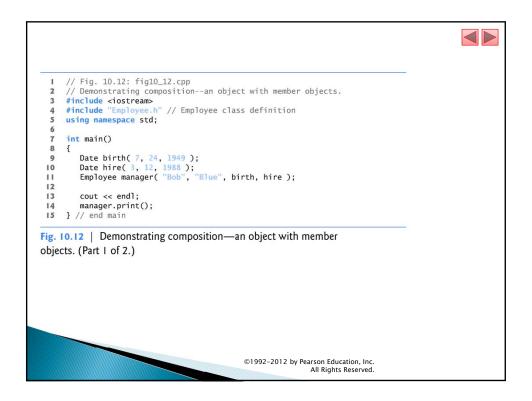
```
// utility function to confirm proper day value based on
// month and year; handles leap years, too
int Date::checkDay( int testDay ) const
 41
 42
  43
             static const int daysPerMonth[ monthsPerYear + 1 ] =
    { 0, 31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31, };
 44
45
             // determine whether testDay is valid for specified month if ( testDay > 0 && testDay <= daysPerMonth[ month ] )
  47
 48
  49
                  return testDay;
 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;
 56
             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.
```

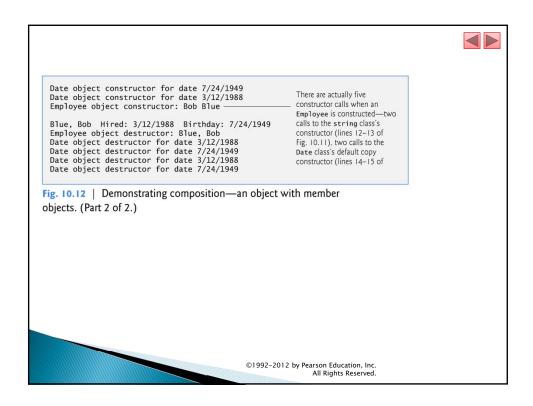


```
class Employee
 12
      public:
 13
 14
          Employee( const string &, const string &,
           const Date &, const Date & );
void print() const;
 15
 16
           ~Employee(); // provided to confirm destruction order
 18
     private:
          string firstName; // composition: member object 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
     #endif
 25
Fig. 10.10 | Employee class definition showing composition. (Part 2
of 2.)
                                                         ©1992-2012 by Pearson Education, Inc.
                                                                            All Rights Reserved.
```



```
// print Employee object
void Employee::print() const
 23
 24
 25
           cout << lastName << ", " << firstName << " Hired: ";</pre>
          hireDate.print();
 26
 27
           cout <<
           birthDate.print();
     cout << endl;
} // end function print</pre>
 29
 30
 32
       // output Employee object to show when its destructor is called
 33
       Employee::~Employee()
 34
     cout << "Employee object destructor: "
     << lastName << ", " << firstName << endl;
} // end ~Employee destructor</pre>
 35
 36
 37
Fig. 10.11 | Employee class member-function definitions, including
constructor with a member initializer list. (Part 2 of 2.)
                                                          ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```





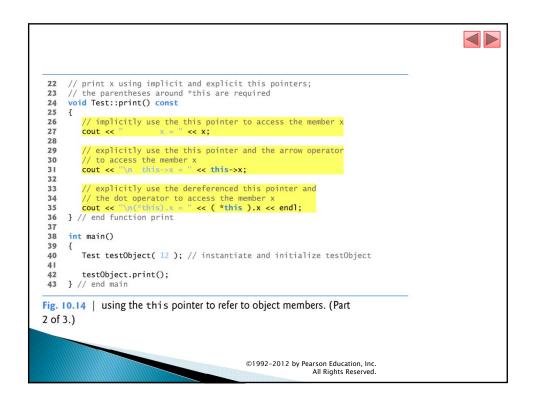


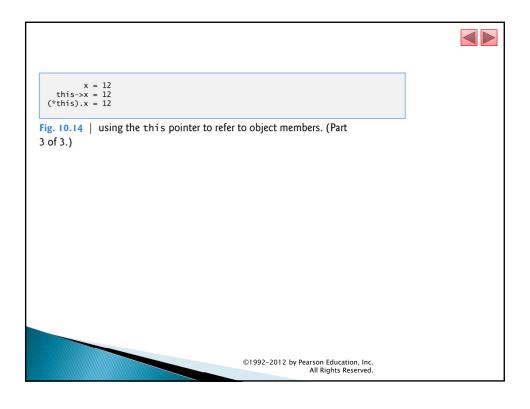
## 10.5 Using the this Pointer

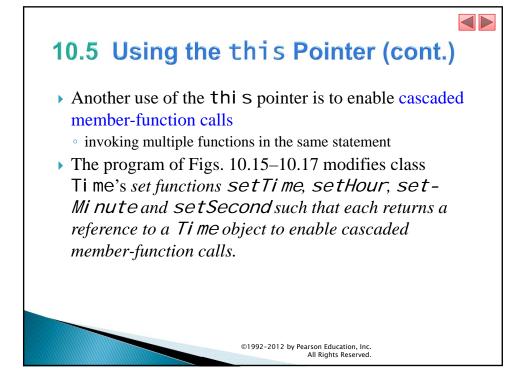
- How do member functions know *which* object's data members to manipulate? Every object has access to its own address through a pointer called this (a C++ keyword).
- The thi s pointer is not part of the object itself.
  - The thi s pointer is passed (by the compiler) as an implicit argument to each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

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

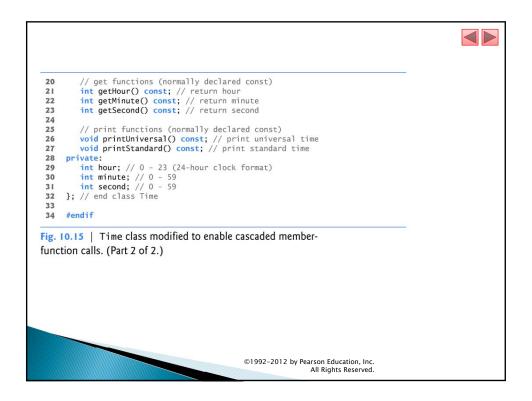
```
// Fig. 10.14: fig10_14.cpp
// Using the this pointer to refer to object members.
#include <iostream>
       using namespace std;
       class Test
       public:
  8
          Test( int = 0 ); // default constructor
void print() const;
 10
 11
       private:
      int x;
}; // end class Test
 12
 13
      // constructor
Test::Test( int value )
   : x( value ) // initialize x to value
 15
 16
 17
 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.
```



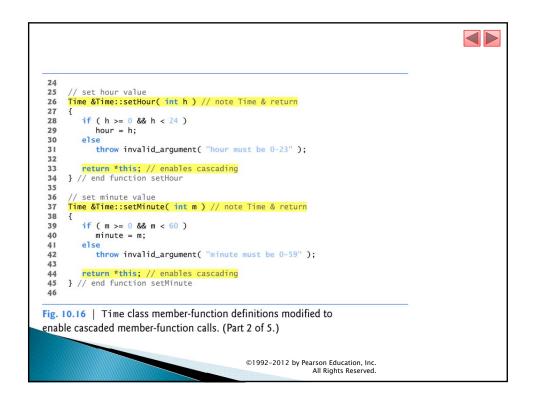




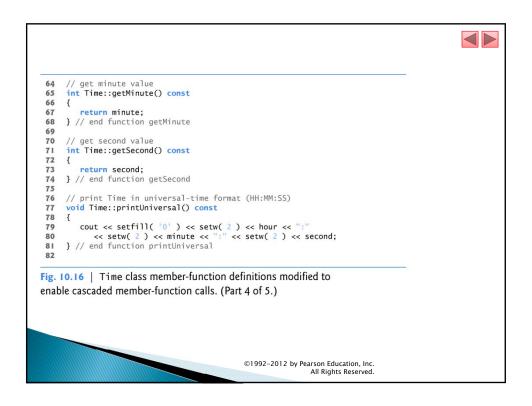
```
// Fig. 10.15: Time.h
// Cascading member function calls.
       // Time class definition.
       // Member functions defined in Time.cpp.
#ifndef TIME_H
#define TIME_H
       class Time
 10
       public:
 11
 12
           Time( int = 0, int = 0, int = 0 ); // default constructor
 13
             // set functions (the Time & return types enable cascading)
           Time &setTime(int); // set hour, minute, second
Time &setMinute(int); // set hour
Time &setMinute(int); // set minute
Time &setSecond(int); // set second
 15
 16
 17
 18
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.
```

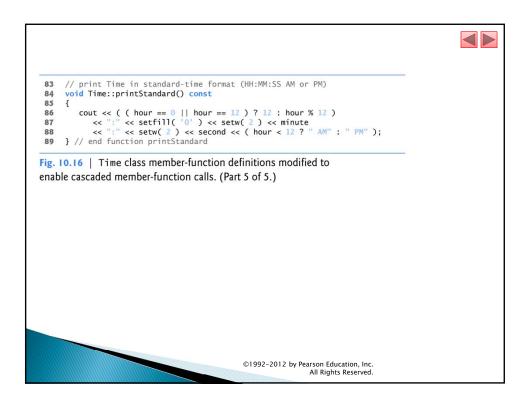


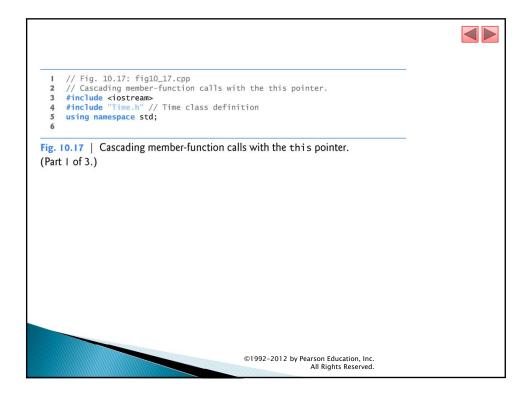
```
// Fig. 10.16: Time.cpp
       // Time class member-function definitions.
       #include <iostream>
       #include <iomanip>
#include "Time.h" // Time class definition
using namespace std;
       // constructor function to initialize private data;
// calls member function setTime to set variables;
// default values are 0 (see class definition)
  10
  11
       Time::Time( int hr, int min, int sec )
 12
       {
           setTime( hr, min, sec );
 13
      } // end Time constructor
 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
 21
            setSecond( s );
      return *this; // enables cascading
} // end function setTime
 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.
```



```
// set second value
Time &Time::setSecond( int s ) // note Time & return
 48
 49
 50
          if ( s >= 0 && s < 60 )
             second = s;
 51
 52
 53
              throw invalid_argument( "second must be 0-59" );
 54
55
     return *this; // enables cascading
} // end function setSecond
 56
 57
      // get hour value
int Time::getHour() const
 58
 59
 60
 61
          return hour;
     } // end function getHour
 62
Fig. 10.16 | Time class member-function definitions modified to
enable cascaded member-function calls. (Part 3 of 5.)
                                                      ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```







```
int main()
  8
        {
            Time t; // create Time object
 10
            // cascaded function calls
t.setHour( 18 ).setMinute( 30 ).setSecond( 22 );
 11
12
13
            // output time in universal and standard formats \mbox{cout} << \mbox{"Universal time: ";}
 14
15
 16
            t.printUniversal();
            cout << "\nStandard time: ";
t.printStandard();</pre>
 18
19
 20
 21
            cout << "\n\nNew standard time: ";</pre>
// cascaded function calls
t.setTime( 20, 20, 20 ).printStandard();
cout << endl;
// end main</pre>
Fig. 10.17 | Cascading member-function calls with the this pointer.
(Part 2 of 3.)
                                                                ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```

