9

Classes: A Deeper Look

Objectives

In this chapter you'll:

- Engineer a class to separate its interface from its implementation and encourage reuse.
- Access class members via an object's name or a reference using the dot (`.`) operator.
- Access class members via a pointer to an object using the arrow (`->`) operator.
- Use destructors to perform "termination housekeeping."
- Learn the order of constructor and destructor calls.
- Learn about the dangers of returning a reference or a pointer to `private` data.
- Assign the data members of one object to those of another object.
- Create objects composed of other objects.
- Use `friend` functions and learn how to declare `friend` classes.
- Use the `this` pointer in a member function to access a non-`static` class member.
- Use `static` data members and member functions.

Self-Review Exercises

9.1 Fill in the blanks in each of the following:

a) Class members are accessed via the _____ operator in conjunction with the name of an object (or reference to an object) of the class or via the _____ operator in conjunction with a pointer to an object of the class.

ANS: dot (`.`), arrow (`->`).

b) Class members specified as _____ are accessible only to member functions of the class and `friend`s of the class.

ANS: `private`.

c) _____ class members are accessible anywhere an object of the class is in scope.

ANS: `public`.

d) _____ can be used to assign an object of a class to another object of the same class.

ANS: Default memberwise assignment (performed by the assignment operator).

e) A nonmember function must be declared by the class as a(n) _____ of a class to have access to that class's `private` data members.

ANS: `friend`.

f) A constant object must be _____; it cannot be modified after it's created.

ANS: initialized.

g) A(n) _____ data member represents classwide information.

ANS: `static`.

h) An object's non-`static` member functions have access to a "self pointer" to the object called the _____ pointer.

ANS: `this`.

i) Keyword _____ specifies that an object or variable is not modifiable.

ANS: `const`.

j) If a member initializer is not provided for a member object of a class, the object's _____ is called.

ANS: default constructor.

k) A member function should be `static` if it does not access _____ class members.

ANS: non-`static`.

l) Member objects are constructed _____ their enclosing class object.

ANS: before.

m) When a member function is defined outside the class definition, the function header must include the class name and the _____, followed by the function name to "tie" the member function to the class definition.

ANS: `::` scope resolution operator.

9.2 Find the error(s) in each of the following and explain how to correct it (them):

a) Assume the following prototype is declared in class `Time`:

```
void ~Time(int);
```

ANS: *Error:* Destructors are not allowed to return values (or even specify a return type) or take arguments.
Correction: Remove the return type `void` and the parameter `int` from the declaration.

b) Assume the following prototype is declared in class `Employee`:

```
int Employee(string, string);
```

ANS: *Error:* Constructors are not allowed to return values.
Correction: Remove the return type `int` from the declaration.

c) The following is a definition of class `Example`:

```
class Example {
public:
   Example(int y = 10) : data(y) { }

   int getIncrementedData() const {
      return ++data;
   }

   static int getCount() {
      cout << "Data is " << data << endl;
      return count;
   }
private:
   int data;
   static int count;
};
```

ANS: *Error:* The class definition for `Example` has two errors. The first occurs in function `getIncrementedData`. The function is declared `const`, but it modifies the object.
Correction: To correct the first error, remove the `const` keyword from the definition of `getIncrementedData`. [*Note:* It would also be appropriate to rename this member function, as *get* functions are typically `const` member functions.]
Error: The second error occurs in function `getCount`. This function is declared `static`, so it's not allowed to access any non-`static` class member (i.e., data).
Correction: To correct the second error, remove the output line from the `getCount` definition.

Exercises

9.3 ***(Scope Resolution Operator)*** What's the purpose of the scope resolution operator?

ANS: The scope resolution operator is used to specify the class to which a function belongs. It resolves the ambiguity caused by multiple classes having member functions of the same name. It also associates a member function in a `.cpp` file with a class definition in a `.h` file.

9.16 ***(Friendship)*** Explain the notion of friendship. Explain the negative aspects of friendship as described in the text.

ANS: Functions that are declared as `friend`s of a class have access to that class's `private` and `protected` members. Some people in the object-oriented programming community prefer not to use `friend` functions. Such people believe friendship corrupts information hiding and weakens the value of the object-oriented design approach, because `friend` functions can directly access a class's implementation details that are supposed to be hidden.

9.17 ***(Constructor Overloading)*** Can a `Time` class definition that includes *both* of the following constructors:

```
Time(int h = 0, int m = 0, int s = 0);
Time();
```

be used to default construct a `Time` object? If not, explain why.

ANS: No. There is ambiguity between the two constructors. When a call is made to the default constructor, the compiler cannot determine which one to use because both constructors can be called with no arguments.

9.18 *(Constructors and Destructors)* What happens when a return type, even `void`, is specified for a constructor or destructor?

ANS: A compilation error occurs. You cannot specify a return type for a constructor or destructor.