Section 3.5 Software Engineering with Set and Get Member Functions

- Through the use of *set* and *get* member functions, you can validate attempted modifications to `private` data and control how that data is presented to the caller.
- A client (p. 88) of a class is any other code that calls the class's member functions.
- Any client code can see a `public` data member and do whatever it wanted with it, including setting it to an invalid value.
- *Set* functions can be programmed to validate their arguments and reject any attempts to *set* the data to bad values.
- A *get* function can present the data to a client in a different form.
- Tightly controlling the access to and presentation of `private` data can greatly reduce errors, while increasing the usability, robustness and security of your programs.

Section 3.6.1 Data Member `balance`

- You can initialize fundamental-type data members in their declarations. This is known as an **in-class initializer** (p. 90) and was introduced in C++11.

Section 3.6.2 Two-Parameter Constructor with Validation

- A constructor can perform **validation** (p. 91) or **validity checking** (p. 91) before modifying a data member.

Section 3.6.3 `deposit` *Member Function with Validation*

- A *set* function can perform validity checking before modifying a data member.

Self-Review Exercises

3.1 Fill in the blanks in each of the following:

a) Every class definition contains the keyword _____ followed immediately by the class's name.
b) A class definition is typically stored in a file with the _____ filename extension.
c) Each parameter in a function header specifies both a(n) _____ and a(n) _____.
d) When each object of a class maintains its own version of an attribute, the variable that represents the attribute is also known as a(n) _____.
e) Keyword `public` is a(n) _____.
f) Return type _____ indicates that a function will perform a task but will not return any information when it completes its task.
g) Function _____ from the `<string>` library reads characters until a newline character is encountered, then copies those characters into the specified `string`.
h) Any file that uses a class can include the class's header via a(n) _____ preprocessing directive.

3.2 State whether each of the following is *true* or *false*. If *false*, explain why.

a) By convention, function names begin with a capital letter and all subsequent words in the name begin with a capital letter.
b) Empty parentheses following a function name in a function definition indicate that the function does not require any parameters to perform its task.
c) Data members or member functions declared with access specifier `private` are accessible to member functions of the class in which they're declared.
d) Variables declared in the body of a particular member function are known as data members and can be used in all member functions of the class.
e) Every function's body is delimited by left and right braces ({ and }).

f) The types of arguments in a function call must be consistent with the types of the corresponding parameters in the function's parameter list.

3.3 What is the difference between a local variable and a data member?

3.4 Explain the purpose of a function parameter. What's the difference between a parameter and an argument?

Answers to Self-Review Exercises

3.1 a) `class`. b) `.h`. c) type, name. d) data member. e) access specifier. f) `void`. g) `getline`. h) `#include`.

3.2 a) False. Function names begin with a lowercase letter and all subsequent words in the name begin with a capital letter. b) True. c) True. d) False. Such variables are local variables and can be used only in the member function in which they're declared. e) True. f) True.

3.3 A local variable is declared in the body of a function and can be used only from its declaration to the closing brace of the block in which it's declared. A data member is declared in a class, but not in the body of any of the class's member functions. Every object of a class has each of the class's data members. Data members are accessible to all member functions of the class.

3.4 A parameter represents additional information that a function requires to perform its task. Each parameter required by a function is specified in the function header. An argument is the value supplied in the function call. When the function is called, the argument value is passed into the function parameter so that the function can perform its task.

Exercises

3.5 *(Default Constructor)* What's a default constructor? How are an object's data members initialized if a class has only a default constructor defined by the compiler?

3.6 *(Data Members)* Explain the purpose of a data member.

3.7 *(Using a Class Without a `using` Directive)* Explain how a program could use class `string` without inserting a `using` directive.

3.8 *(Set and Get Functions)* Explain why a class might provide a *set* function and a *get* function for a data member.

3.9 *(Modified `Account` Class)* Modify class `Account` (Fig. 3.8) to provide a member function called `withdraw` that withdraws money from an `Account`. Ensure that the withdrawal amount does not exceed the `Account`'s balance. If it does, the balance should be left unchanged and the member function should display a message indicating `"Withdrawal amount exceeded account balance."` Modify class `AccountTest` (Fig. 3.9) to test member function `withdraw`.

3.10 *(`Invoice` Class)* Create a class called `Invoice` that a hardware store might use to represent an invoice for an item sold at the store. An `Invoice` should include four data members—a part number (type `string`), a part description (type `string`), a quantity of the item being purchased (type `int`) and a price per item (type `int`). Your class should have a constructor that initializes the four data members. Provide a *set* and a *get* function for each data member. In addition, provide a member function named `getInvoiceAmount` that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as an `int` value. If the quantity is not positive, it should be set to `0`. If the price per item is not positive, it should be set to `0`. Write a test program that demonstrates class `Invoice`'s capabilities.

3.11 *(`Employee` Class)* Create a class called `Employee` that includes three pieces of information as data members—a first name (type `string`), a last name (type `string`) and a monthly salary (type

int). Your class should have a constructor that initializes the three data members. Provide a *set* and a *get* function for each data member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee's capabilities. Create two Employee objects and display each object's *yearly* salary. Then give each Employee a 10 percent raise and display each Employee's yearly salary again.

3.12 ***(Date Class)*** Create a class called Date that includes three pieces of information as data members—a month (type int), a day (type int) and a year (type int). Your class should have a constructor with three parameters that uses the parameters to initialize the three data members. For the purpose of this exercise, assume that the values provided for the year and day are correct, but ensure that the month value is in the range 1–12; if it isn't, set the month to 1. Provide a *set* and a *get* function for each data member. Provide a member function displayDate that displays the month, day and year separated by forward slashes (/). Write a test program that demonstrates class Date's capabilities.

3.13 ***(Removing Duplicated Code in the main Function)*** In Fig. 3.9, the main function contains six statements (lines 14–15, 16–17, 26–27, 28–29, 37–38 and 39–40) that each display an Account object's name and balance. Study these statements and you'll notice that they differ only in the Account object being manipulated—account1 or account2. In this exercise, you'll define a new displayAccount function that contains *one* copy of that output statement. The member function's parameter will be an Account object and the member function will output the object's name and balance. You'll then replace the six duplicated statements in main with calls to displayAccount, passing as an argument the specific Account object to output.

Modify Fig. 3.9 to define the following displayAccount function *after* the using directive and *before* main:

```
void displayAccount(Account accountToDisplay) {
   // place the statement that displays
   // accountToDisplay's name and balance here
}
```

Replace the comment in the member function's body with a statement that displays accountToDisplay's name and balance.

Once you've completed displayAccount's declaration, modify main to replace the statements that display each Account's name and balance with calls to displayAccount of the form:

```
displayAccount(nameOfAccountObject);
```

In each call, the argument should be the account1 or account2 object, as appropriate. Then, test the updated program to ensure that it produces the same output as shown in Fig. 3.9.

3.14 ***(C++11 List Initializers)*** Write a statement that uses list initialization to initialize an object of class Account which provides a constructor that receives an unsigned int, two strings and a double to initialize the accountNumber, firstName, lastName and balance data members of a new object of the class.

Making a Difference

3.15 ***(Target-Heart-Rate Calculator)*** While exercising, you can use a heart-rate monitor to see that your heart rate stays within a safe range suggested by your trainers and doctors. According to the American Heart Association (AHA) (http://bit.ly/AHATargetHeartRates), the formula for calculating your *maximum heart rate* in beats per minute is 220 minus your age in years. Your *target heart rate* is a range that's 50–85% of your maximum heart rate. [*Note:* These formulas are estimates provided by the AHA. Maximum and target heart rates may vary based on the health, fitness and gender of the individual. **Always consult a physician or qualified health-care professional before beginning or modifying an exercise program.**] Create a class called HeartRates. The class attributes