Classes, Objects in C++

Nothing can have value without being an object of utility.

~ Karl Marx

Fundamental Idea - **Data Abstraction**

- **Data Abstraction** - Separation of interface and implementation
- **Interface** -Operations that users of the Class can execute
- **Implementation** :
    - Data members
    - The bodies of the functions that constitute the interface,
    - Functions needed to define the class that are not intended for general use

Fundamental Idea - **Encapsulation**

- Separating Class Interface and Implementation
- Hiding the Implenetation Details from the Users.
- Users of Class can user the interface, but have no access to the implementation.

Abstract Data Type

- A class that uses data abstraction and encapsulation defines an abstract data type.

- Class designer worries about the implementation of the Class.

- Programmers who use the class need not know how the typeworks, but think abstractly about what the type does.

First CPP Class - Define class and member variables and functions

```
class Person
{
        public:   // Access Specifier
        string name; //Data Member of type String
        int age;        // Data Member of type int
    // Member function with no return value
        void setName(string);
        void setAge(int);
}; //Notice the ; to end the class definition
```

Working with Class Functions

```
void Person::setName(string userName)
{
            name = userName;
}
// define the setAge() function in the Person class
void Person::setAge(int userAge)
{
                        age = userAge;
}
```

Instance and define the Object

```
int main(void)
{
        Person bob;
        bob.setName("Bob");
        bob.setAge(27);
        cout << bob.name << " is " << bob.age << " years old." << endl;
}
```

Dissecting the Person Class

- Keyword “**class**” followed by name of the Class defines the Classname.
- Syntax to define Class: Class <ClassName>. E.g. in our case, we defined our class as class Person.
- Every class’s body is enclosed in a pair of left and right braces ({ and }).
- The class definition terminates with a semicolon

Public: Access Specifier

- keyword public, an access specifier.
- In our Person class, we had defined data members and functions inside the public access specifier.
- This indicates that the members of this class are “available to the public”, i.e.it can be called by the other functions in the program (such as main), and by other functions of the class (if there exists any)
- Always followed by :

Functions definition

- Person class contains two functions, **void setName(string) and void setAge(int)**.
- Both these methods are part of the class Person, which is available to other function.
- The function defined here does not return anything.
- To access the function, we need to use the <return type> <className>::<functionname>
- :: is known as scope resolution operator
- E,g, we are using setName function as **void Person::setName(string userName)**
- The functions here are taking parameters, which will be defined when calling the function.
- To call a member function of a class, we need to create the object of that class.

Creating an instance of Class

- Inside the main() function, the Statement Person bob would create an object named bob.

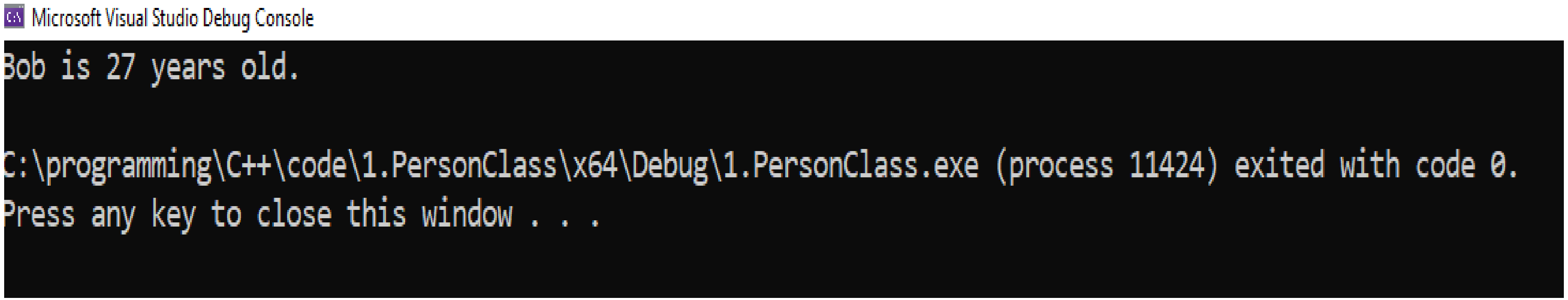
- Object is an instance of the class.

- Once an object is created, it has access to all the members of that class

- To access the members, we use . operator, for e.g. to access the function, we are using bob.setAge(25)

Final Output

Running this program should generate the output as follows:



```
Bob is 27 years old.

C:\programming\C++\code\1.PersonClass\x64\Debug\1.PersonClass.exe (process 11424) exited with code 0.
Press any key to close this window . . .
```

Creating another instance

- In the same class, we can create another object named john.

- john has access to all the methods of the Person, however, john and bob are not related to each other.

- This is the philosophy of the C++ programming. Every object will have access to the members of the class, but those objects are not related to each other.

- Person is an Abstract Data Type, as it has encapsulation of it's data and function, and also, hides the implementation details of the user.

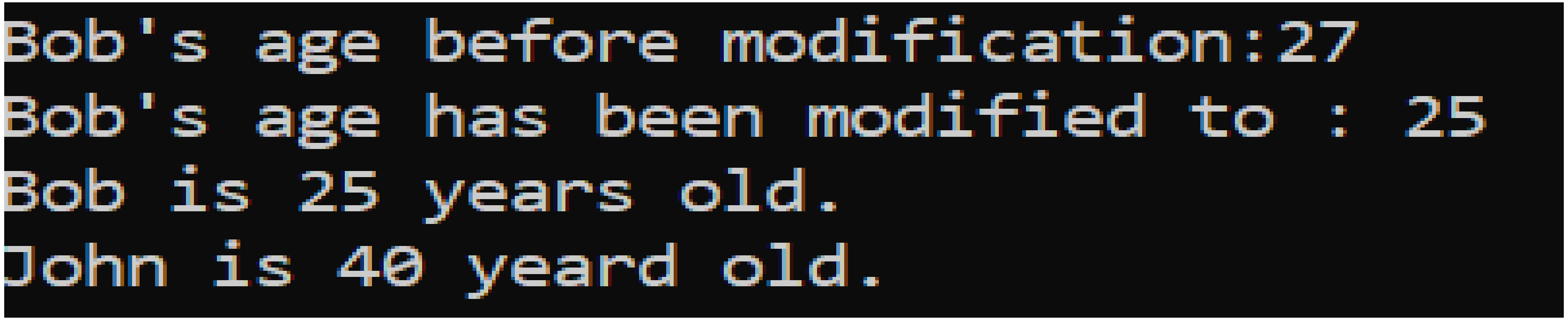
Access Specifier-Relook

- In our Person class, we have defined our data members as public.
- This has a potential pit fall.
- The following lines of code would make this statement clear:

```
            Person bob;

            bob.setName("Bob");

            bob.setAge(27);

            bob.age = 25;  // Note how the age for bob has been
        modified

          // display bob's age

      cout << bob.name << " is " << bob.age << " years old." << endl;
```

Access Specifier - Relook

- Guess the output of the Person.cpp program, after the above modification has been done?



```
Bob's age before modification:27
Bob's age has been modified to : 25
Bob is 25 years old.
John is 40 yeard old.
```

- The reason for such behavior is because data members have been declared as public, and it is available to all other functions. Therefore, user can modify the data members also. This is against the principle of Encapsulation, as data members must have scope of private in C++.

Private Access Specifier and Data Hiding

- As a rule of thumb, in C++, data members of class are declared as private.

- Variables or functions declared after access specifier private (and before the next access specifier if there is one) are accessible only to member functions of the class for which they're declared

- The default access for class members is private so all members after the class header and before the first access specifier (if there are any) are private.

Private Access Specifier and Data Hiding

- Declaring data members with access specifier private is known as data hiding
- In our example, when we do the following modifications:

```
class Person
      {
          // Private data members
          Private:
              string name;
              int age;
          }
```

Datamember name and age is encapsulated, and can be accessed only by the member functions of the class.

Private Access Specifier and Data Hiding

```
public:
        void setName(string);
        void setAge(int);
        // We need to add a function to get name and age of Person
        // As the data members are not availabe public
    // Notice that these two functions have return types
        string getName();
        int getAge();
};
```

Private Access Specifier and Data Hiding

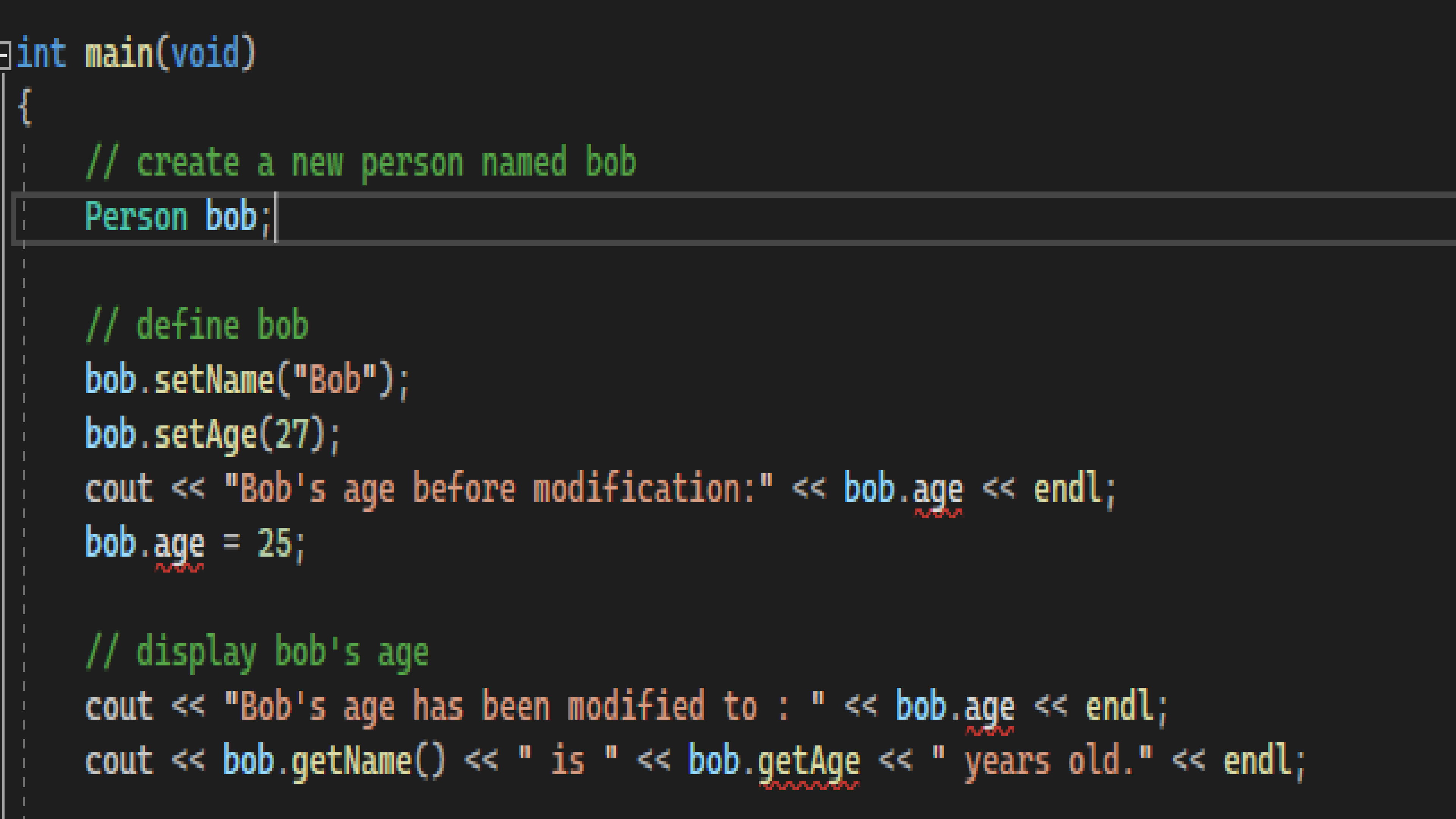
- In our example, only setter and getter member functions have access to the data members, and can manipulate them.

- By using getAge and getName, we are going to return corresponding name and age of the Person's object.

```
string Person::getName()
{
    return name;
}
```

Private Access Specifier- Data Hiding

```
int Person::getAge()
{
        return age;
}
```



```cpp
int main(void)
{
    // create a new person named bob
    Person bob;

    // define bob
    bob.setName("Bob");
    bob.setAge(27);
    cout << "Bob's age before modification:" << bob.age << endl;
    bob.age = 25;

    // display bob's age
    cout << "Bob's age has been modified to : " << bob.age << endl;
    cout << bob.getName() << " is " << bob.getAge << " years old." << endl;
```

- Compiler will not allow us to access the private members, and any attempt to run this faulty program would result in an error.
- Now, users cannot modify the data members of a class, and is strictly private.
- To run program correctly, modify the code as:

```
int main(void)
{
    Person bob;
    bob.setName("Bob");
    bob.setAge(27);
  cout << bob.getName() << " is " << bob.getAge() << " years old." <<
endl;
}
```

Constructors

- As an exercise, let's try to execute the program, Person.cpp, without initializing the variables in the class instantiation.

- In this case, compiler would execute the program successfully, however, the result would not be what we would like to have.

- The name variable would be a blank value, while age would contain some garbage values.

Constructors - What are they?

- C++ automatically calls a constructor for each object that's created, which helps to ensure that objects are initialized properly before they're used in a program.

- If a class does not explicitly include constructors, the compiler provides a default constructor with no parameters.

- For data members that are objects of other classes, the default constructor implicitly calls each data member's default constructor to ensure that the data member is initialized properly.

Constructor - Definition

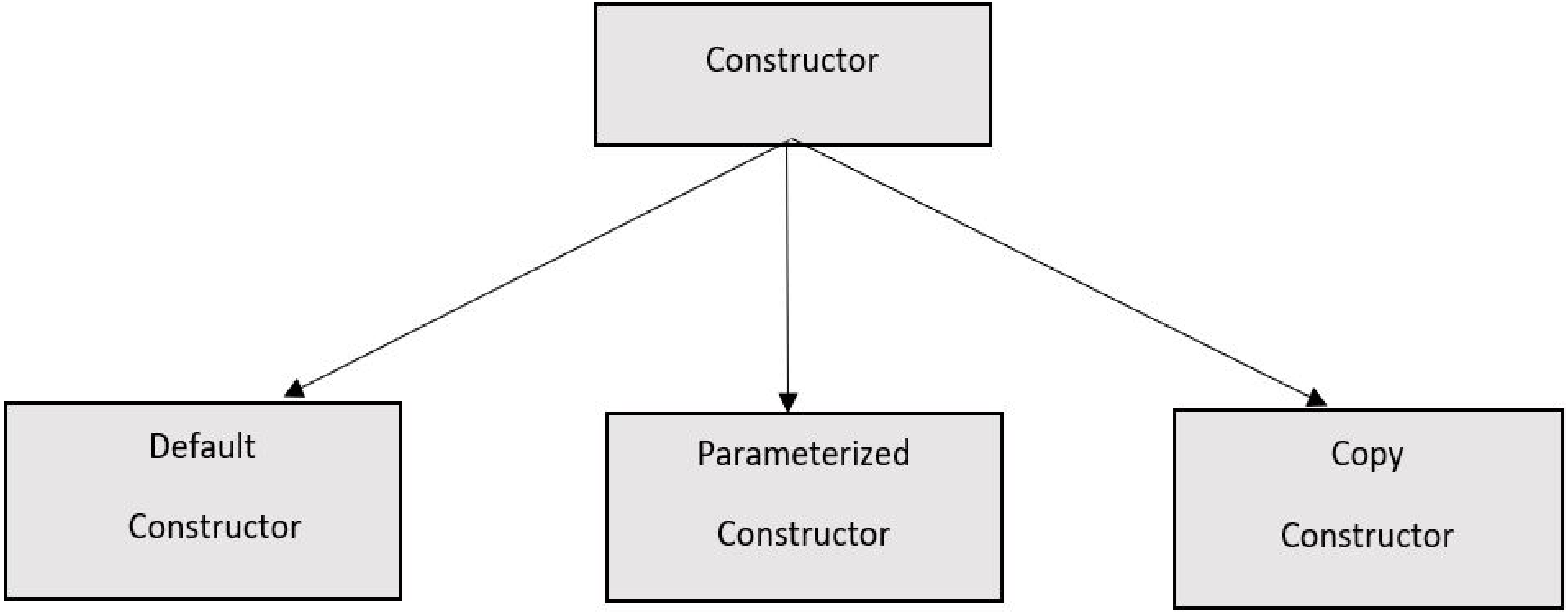
- A constructor is a special member function that must be defined with the same **so that the compiler can distinguish it from the class's other member functions.**

- An important difference between constructor and other member functions is that **constructor cannot return any value and cannot specify a return type, not even void.**

- Constructors are normally declared public.

- If a constructor is not declared inside a class, C++ automatically provides a default constructor.

Types of Constructors



Default Consturctor - Implicit

- Constructors **without any argument** is known as **Default Constructor.**
- C++ compiler implicitly creates a default constructor in a class, if that class does not have user defined constructor.
- Default Constructor does not initialize the class data members, but does call the default constructor for each data member that's an object of another class
- An uninitialized variable contains an undefined ("garbage") value.

Default Constructor - Explicit

- A Constructor can be defined explicitly, by the user.
- If such constructor does not take any parameter, then it's considered as default constructor.
- In Person class, the default constructor can be created explicitly as:

```
Class Person
{
        Person(); // no argument
}
Person::Person()
{
                // If variables are not initialized, then they will have garbage values for integer types and blank value for string type
}
```

Parametrized Constructor

- Constructor defined with at least one argument.

- In this case, compiler does not create any implicit default constructor for that class.

- C++11 can force to issue a default constructor, even if we have created a constructo with an argument.

- The number of arguments passed inside a parameterized constructor must be initialized, when the object is created.

REVIEW AND PRACTICE EXERCISES

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) 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.

Answers) a) class. b) .h. c) type, name. d) data member. e) access specifier. f) void. g) getline. h) scope resolution operator (::). i) #include.

True or False

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 prototype 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.

True or False

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) Any source-code file that contains int main() can be used to execute a program.

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

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

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

a) (Default Constructor) What's a default constructor? How are an object's data members initialized if a class has only an implicitly defined default constructor?

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

Programming Exercises (Refer C++ Dietel and Dietel chapter 3)

1> (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.

2> (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. A constructor that receives multiple arguments is defined with the form:

ClassName(TypeName1 parameterName1, TypeName2 parameterName2, ...)

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.

Bonus Exercise (Dietel and Dietel 3.16) Chapter 3]

3.16 (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) (www.americanheart.org/presenter.jhtml?identifier=4736), 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 is 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.] (On the next slide)

Create a class called HeartRates. The class attributes should include the person’s first name, last name and date of birth (consisting of separate attributes for the month, day and year of birth). Your class should have a constructor that receives this data as parameters. For each attribute provide set and get functions. The class also should include a function *getAge* that calculates and returns the person’s age (in years), a function *getMaxiumumHeartRate* that calculates and returns the person’s maximum heart rate and a function *getTargetHeartRate* that calculates and returns the person’s target heart rate. Since you do not yet know how to obtain the current date from the computer, function *getAge* should prompt the user to enter the current month, day and year before calculating the person’s age. Write an application that prompts for the person’s information, instantiates an object of class HeartRates and prints the information from that object—including the person’s first name, last name and date of birth—then calculates and prints the person’s age in (years), maximum heart rate and target-heart-rate range.