Aggregation in the EMPCONT Program

Let's rearrange the EMPMULT program to use aggregation instead of inheritance. In EMPMULT the `manager` and `scientist` classes are derived from the `employee` and `student` classes using the inheritance relationship. In our new program, EMPCONT, the `manager` and `scientist` classes contain instances of the `employee` and `student` classes as attributes. This aggregation relationship is shown in Figure 9.12.

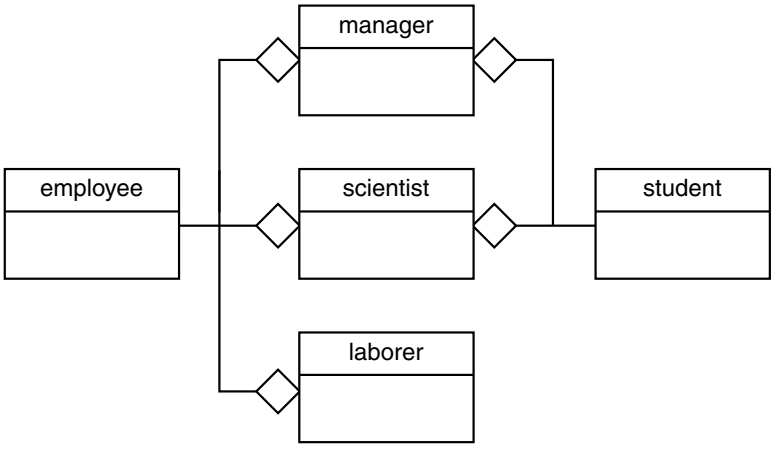

FIGURE 9.12
UML class diagram for EMPCONT.

The following miniprogram shows these relationships in a different way:

```
class student
   {};
class employee
   {};
class manager
   {
   student stu;    // stu is an object of class student
   employee emp;   // emp is an object of class employee
   };
class scientist
   {
   student stu;    // stu is an object of class student
   employee emp;   // emp is an object of class employee
   };
class laborer
   {
   employee emp;   // emp is an object of class employee
   };
```

Here's the full-scale listing for EMPCONT:

```
// empcont.cpp
// containership with employees and degrees
#include <iostream>
#include <string>
using namespace std;
////////////////////////////////////////////////////////////
class student                 //educational background
   {
   private:
      string school;          //name of school or university
      string degree;          //highest degree earned
   public:
      void getedu()
         {
         cout << "   Enter name of school or university: ";
         cin >> school;
         cout << "   Enter highest degree earned \n";
         cout << "   (Highschool, Bachelor's, Master's, PhD): ";
         cin >> degree;
         }
      void putedu() const
         {
         cout << "\n   School or university: " << school;
         cout << "\n   Highest degree earned: " << degree;
         }
   };
////////////////////////////////////////////////////////////
class employee
   {
   private:
      string name;            //employee name
      unsigned long number;   //employee number
   public:
      void getdata()
         {
         cout << "\n   Enter last name: "; cin >> name;
         cout << "   Enter number: ";      cin >> number;
         }
      void putdata() const
         {
         cout << "\n   Name: " << name;
         cout << "\n   Number: " << number;
         }
   };
////////////////////////////////////////////////////////////
class manager                 //management
```

```
   {
   private:
      string title;           //"vice-president" etc.
      double dues;            //golf club dues
      employee emp;           //object of class employee
      student stu;            //object of class student
   public:
      void getdata()
         {
         emp.getdata();
         cout << "   Enter title: ";          cin >> title;
         cout << "   Enter golf club dues: "; cin >> dues;
         stu.getedu();
         }
      void putdata() const
         {
         emp.putdata();
         cout << "\n   Title: " << title;
         cout << "\n   Golf club dues: " << dues;
         stu.putedu();
         }
   };
////////////////////////////////////////////////////////////////
class scientist               //scientist
   {
   private:
      int pubs;               //number of publications
      employee emp;           //object of class employee
      student stu;            //object of class student
   public:
      void getdata()
         {
         emp.getdata();
         cout << "   Enter number of pubs: "; cin >> pubs;
         stu.getedu();
         }
      void putdata() const
         {
         emp.putdata();
         cout << "\n   Number of publications: " << pubs;
         stu.putedu();
         }
   };
////////////////////////////////////////////////////////////////
class laborer                 //laborer
   {
```

```
   private:
      employee emp;           //object of class employee
   public:
      void getdata()
         { emp.getdata(); }
      void putdata() const
         { emp.putdata(); }
   };
////////////////////////////////////////////////////////////////
int main()
   {
   manager m1;
   scientist s1, s2;
   laborer l1;

   cout << endl;
   cout << "\nEnter data for manager 1";    //get data for
   m1.getdata();                            //several employees

   cout << "\nEnter data for scientist 1";
   s1.getdata();

   cout << "\nEnter data for scientist 2";
   s2.getdata();

   cout << "\nEnter data for laborer 1";
   l1.getdata();

   cout << "\nData on manager 1";           //display data for
   m1.putdata();                            //several employees

   cout << "\nData on scientist 1";
   s1.putdata();

   cout << "\nData on scientist 2";
   s2.putdata();

   cout << "\nData on laborer 1";
   l1.putdata();
   cout << endl;
   return 0;
   }
```

The `student` and `employee` classes are the same in EMPCONT as they were in EMPMULT, but they are related in a different way to the `manager` and `scientist` classes.

Composition: A Stronger Aggregation

Composition is a stronger form of aggregation. It has all the characteristics of aggregation, plus two more:

- The part may belong to only one whole.
- The lifetime of the part is the same as the lifetime of the whole.

A car is composed of doors (among other things). The doors can't belong to some other car, and they are born and die along with the car. A room is composed of a floor, ceiling, and walls. While aggregation is a "has a" relationship, composition is a "consists of" relationship.

In UML diagrams, composition is shown in the same way as aggregation, except that the diamond-shaped arrowhead is solid instead of open. This is shown in Figure 9.13.

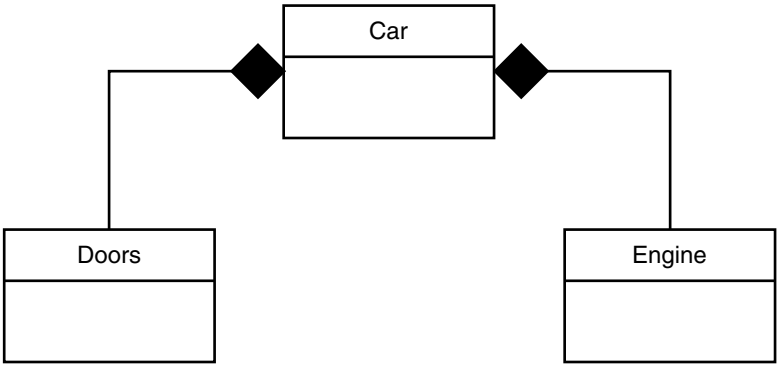

FIGURE 9.13
UML class diagram showing composition.

Even a single object can be related to a class by composition. In a car there is only one engine.

Inheritance and Program Development

The program-development process, as practiced for decades by programmers everywhere, is being fundamentally altered by object-oriented programming. This is due not only to the use of classes in OOP but to inheritance as well. Let's see how this comes about.

Programmer A creates a class. Perhaps it's something like the `Distance` class, with a complete set of member functions for arithmetic operations on a user-defined data type.