OBJECT ORIENTED PROGRAMMING LAB

Lab Manual # 14

Composition in C++

Instructor: Muhammad Abdullah Orakzai

Semester Fall, 2021

Course Code: CL1004

Fast National University of Computer and Emerging Sciences
Peshawar

Department of Computer Science

OBJECT ORIENTED PROGRAMMING LANGUAGE

Table of Contents

Introduction to Complex Objects and Composition

An object is a basic unit of Object-Oriented Programming and represents the real-life entities. **Complex objects are the objects that are built from smaller or a collection of objects. For example**, a mobile phone is made up of various objects like a camera, battery, screen, sensors, etc. In this article, we will understand the use and implementation of a complex object.

In object-oriented programming languages, object composition is used for objects that have a **"has-a"** relationship with each other. **For example**, a mobile has-a battery, has-a sensor, has-a screen, etc. Therefore, the complex object is called the whole or a parent object whereas a simpler object is often referred to as a child object. In this case, all the objects or components are the child objects which together make up the complex object(mobile).

Classes that have data members of built-in type work really well for simple classes. However, in real-world programming, the product or software comprises of many different smaller objects and classes. In the above example, the mobile phone consisted of various different objects that on a whole made up a complete mobile phone. Since each of those objects performs a different task, they all are maintained in different classes. Therefore, this concept of complex objects is being used in most of the real-world scenarios. The advantages of using this concept are:

- Each individual class can be simple and straightforward.
- One class can focus on performing one specific task and obtain one behavior.
- The class is easier to write, debug, understand, and usable by other programmers.
- While simpler classes can perform all the operations, the complex class can be designed to coordinate the data flow between the simpler classes.
- It lowers the overall complexity of the complex object because the main, task of the complex object would then be to delegate tasks to the sub-objects, who already know how to do them.

The most important advantage is if any changes have to be made in a child class, only the child class can be changed rather than changing the entire parent class.

- For example, if we want to change the battery class in the mobile object, with the help of composition, the changes are only made in the battery class and the entire mobile object works fine with the updated changes.

Scope of Use

Although there are no well-defined rules to state when a programmer must use the composition, as a rule of thumb, each class should be built to accomplish a single task. The task should be to either perform some part of manipulation or be responsible for coordinating other classes but cannot perform both tasks. This immensely increases the maintenance of the code and future updations because, when we wish to update a feature or an object, only the class pertaining to that specific functionality needs to be updated. And also, it's very easy to keep track of the errors in the program. For example:

```
// C++ program to implement a composition

// A point class
class Point {
private:
      int x;
      int y;
};

// Every location has a point.
// Every point has two coordinates.
// The above class's functionality
// is only to store the coordinates
// in two variables. Any functionality
// using the points is implemented
// here
class Location {
      Private:
      Point Source;
      Point Destination
};
```

In the classes given here, **Location** uses objects of class Point as its data members. Hence, Location is a complex class which uses a simple class Point. Let us have a look at the program that makes use of the composition.

Below is the implementation of a composite class:

Program Example # 01

```
// C++ program to implement a composite class
using namespace std;
#include <iostream>

// Class with a private parameter
// and the getters and setters
class One {

      // Private parameter
private:
      int num;

      // Public setter and getter
```

```
public:
      void set(int i)
      {
            num = i;
      }
      int get()
      {
            return num;
      }
};

// Another class
class Two {

      // Public method and the object
      // of the previous class
public:
      One O;
      void show()
      {
            cout << "\n Number = "
                  << O.get();
      }
};

// Driver code
int main()
{
      // Creating the object of
      // class Two
      Two T;

      // Perform some operation using
      // the object of One in this class
      T.O.set(100);
      T.show();
}
```

Output:

```
Number = 100
```

Explanation: Note that in the program, class Two has an object of class One. To access a member of One, we must use the object of Two as in T.O.set(100). Moreover, since num is a private member of One, we must use a public function to access its value.

C++ Composition

In real-life complex objects are often built from smaller and simpler objects. For example, a car is built using a metal frame, an engine some tires, a transmission system, a steering wheel, and a large number of other parts. A personal computer is built from a CPU, a motherboard, memory unit, input and output units, etc. even human beings are building from smaller parts such as a head, a body, legs, arms, and so on. This process of building complex objects from simpler ones is called c++ composition. It is also known as object composition.

So far, all of the classes that we have used had member variables that are built-in data type (e.g. int, float, double, char). While this is generally sufficient for designing and implementing small, simple classes, but it gradually becomes difficult to carry out from more complex classes, especially for those built from many sub-parts. In order to facilitate the building of complex classes from simpler ones, C++ allows us to do object C++ Composition in a very simple way by using classes as member variables in other classes.

A class can have one or more objects of other classes as members. A class is written in such a way that the object of another existing class becomes a member of the new class. this relationship between classes is known as C++ Composition. It is also known as containment, part-whole, or has-a relationship. A common form of software reusability is C++ Composition.

In C++ Composition, an object is a part of another object. The object that is a part of another object is known as a sub-object. When a C++ Composition is destroyed, then all of its subobjects are destroyed as well. Such as when a car is destroyed, then its motor, frame, and other parts are also destroyed with it. It has a do and die relationship.

Program Example # 02

How to use C++ Composition in programming

```
#include <iostream>
using namespace std;
class X
{
    private:
    int d;
    public:
    void set_value(int k)
    {
        d=k;

    }
    void show_sum(int n)
    {
        cout<<"sum of "<<d<<" and "<<n<<" = "<<d+n<<endl;
```

```
    }

};
class Y
{
    public:
    X a;
    void print_result()
    {
        a.show_sum(5);

    }

};

int main()
{
  Y b;
  b.a.set_value(20);
  b.a.show_sum(100);
  b.print_result();

}
/*
Output
sum of 20 and 100 = 120
sum of 20 and 5 = 25
*/
```

Programming Explanation:

- In this program, class X has one data member ‘d’ and two member functions ‘set_value()’ and ‘show_sum()’. The set_value() function is used to assign value to ‘d’. the show_sum() function uses an integer type parameter. It adds the value of parameter with the value of ‘d’ and displays the result on the screen.
- Another class Y is defined after the class x. the class Y has an object of class x that is the C++ Composition relationship between classes x and y. this class has its own member function print_result().
- In the main() function, an object ‘b’ of class y is created. The member function set_value() of object ‘a’ that is the sub-object of object ‘b’ is called by using two dot operators. One dot operator is used to access the member of the object ‘b’ that is object ‘a’, and second is used to access the member function set_value() of sub-object ‘a’ and ‘d’ is assigned a value 20.
- In the same way, the show_sum() member function is called by using two dot operators. The value 100 is also passed as a parameter. The member function print_result of object ‘b’ of class Y is also

called for execution. In the body of this function, the show_sum() function of object 'a' of class X is called for execution by passing value 5.

Program Example # 03

```
#include <iostream>
#include <string>

using namespace std;

class Birthday{

public:
    Birthday(int cmonth, int cday, int cyear){
        cmonth = month;
        cday = day;
        cyear = year;

    }
    void printDate(){
        cout<<month <<"/" <<day <<"/" <<year <<endl;

    }
private:
    int month;
    int day;
    int year;

};

class People{

public:
    People(string cname, Birthday cdateOfBirth)
    :name(cname),
    dateOfBirth(cdateOfBirth)
    {

    }
    void printInfo(){
        cout<<name <<" was born on: ";
        dateOfBirth.printDate();
    }

private:
```

```
    string name;
    Birthday dateOfBirth;

};


int main() {

    Birthday birthObject(7,9,97);
    People infoObject("Lenny the Cowboy", birthObject);
    infoObject.printInfo();

}
```

References

https://beginnersbook.com/2017/08/cpp-data-types/

http://www.cplusplus.com/doc/tutorial/basic_io/

https://www.w3schools.com/cpp/default.asp

https://www.javatpoint.com/cpp-tutorial

https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/?ref=lbp

https://www.programiz.com/

https://ecomputernotes.com/cpp/