10.x — Chapter 10 comprehensive quiz

BY ALEX ON SEPTEMBER 12TH, 2016 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

In this chapter, we learned about some different kinds of relationships between two objects.

Summary

The process of building complex objects from simpler ones is called **object composition**. There are two types of object composition: composition, and aggregation.

Composition exists when a member of a class has a part-of relationship with the class. In a composition relationship, the class manages the existence of the members. To qualify as a **composition**, an object and a part must have the following relationship:

- The part (member) is part of the object (class)
- The part (member) can only belong to one object (class) at a time
- The part (member) has its existence managed by the object (class)
- The part (member) does not know about the existence of the object (class)

Compositions are typically implemented via normal member variables, or by pointers where the class manages all the memory allocation and deallocation. If you can implement a class as a composition, you should implement a class as a composition.

Aggregations exists when a class has a has-a relationship with the member. In an aggregation relationship, the class does not manage the existence of the members. To qualify as an **aggregation**, an object and its parts must have the following relationship:

- The part (member) is part of the object (class)
- The part (member) can belong to more than one object (class) at a time
- The part (member) does *not* have its existence managed by the object (class)
- The part (member) does not know about the existence of the object (class)

Aggregations are typically implemented via pointer or reference.

Associations are a looser type of relationship, where the class uses-an otherwise unrelated object. To qualify as an **association**, an object and an associated object must have the following relationship:

- The associated object (member) is otherwise unrelated to the object (class)
- The associated object (member) can belong to more than one object (class) at a time
- The associated object (member) does *not* have its existence managed by the object (class)
- The associated object (member) may or may not know about the existence of the object (class)

Associations may be implemented via pointer or reference, or by a more indirect means (such as holding the index or key of the associated object).

In a **dependency**, one class uses another class to perform a task. The dependent class typically is not a member of the class using it, but rather is temporarily created, used, and then destroyed, or passed into a member function from an external source.

In a **container class** one class provides a container to hold multiple objects of another type. A **value container** is a composition that stores copies of the objects it is holding. A **reference container** is an aggregation that stores pointers or references to objects that live outside the container.

std::initializer_list can be used to implement constructors, assignment operators, and other functions that accept a list initialization parameter. std::initailizer_list lives in the <initializer_list> header.

Property\Type	Composition	Aggregation	Association	Dependency
Relationship type	Whole/part	Whole/part	Otherwise unrelated	Otherwise unrelated
Members can belong to multiple classes	No	Yes	Yes	Yes
Members existence managed by class	Yes	No	No	No
Directionality	Unidirectional	Unidirectional	Unidirectional or bidirectional	Unidirectional
Relationship verb	Part-of	Has-a	Uses-a	Depends-on

Quiz time

This chapter is pretty straightforward and a little more abstract than the previous ones, so this quiz will be short and to the point.

1) What type of relationship (composition, aggregation, association, or dependency) do the following describe?
1a) An Animal class that contains an animal type (enum) and name (string).
Show Solution

1b) A text editor class with a save() function that takes a File object. The save() function writes the contents of the editor to disk.
Show Solution

1c) An Adventurer class that can carry various kinds of Items, such as swords, wands, potions, or spellbooks. These Items can be dropped and picked up by other Adventurers.
Show Solution

1d) A Classroom class hold Students.
Show Solution

1e) A Computer class that contains a CPU class. The CPU can be removed from the Computer and tested on its own.
Show Solution

2) Select one: If you can design a class using (composition, aggregation, association, or dependency), then you should.
Show Solution

11.1 -- Introduction to inheritance

Index

10.7 -- std::initializer_list

C++ TUTORIAL | PRINT THIS POST

31 comments to 10.x — Chapter 10 comprehensive quiz

Ryan
September 8, 2019 at 1:07 pm · Reply

I remember seeing a nice hangman code on this site and decided to have a go at it myself. At first I used char to store my words and it was a clock-full of syntax and problems. Is it possible to show an example of class that supports for a char to function for checking each letter.

What can I improve on this code?

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <fstream>
#include <cstring>
#include <random>
#include <ctime>

enum BodyPart : int
{
    ROPE = 0,
    HEAD,
    NECK,
    LEFT_ARM,
    BODY,
    RIGHT_ARM,
    LEFT_LEG,
    RIGHT_LEG
};

struct DrawingPart
{
    BodyPart DrawPart;
    std::string ASCII;
    bool ToShow;
};

DrawingPart StickFigure[7] = {
  {BodyPart::ROPE, "|            |           |\n", false},
  {BodyPart::HEAD, "|            O           |\n", false},
  {BodyPart::LEFT_ARM, "|           /", false},
  {BodyPart::RIGHT_ARM, "|\\          |\n", false},
  {BodyPart::BODY, "|            |           |\n", false},
  {BodyPart::LEFT_LEG, "|           /", false},
  {BodyPart::RIGHT_LEG, " \\          |\n", false}
};

void DrawHangman()
{
    for (int i{ 0 }; i < sizeof(StickFigure) / sizeof(StickFigure[0]); ++i)
    {
        if (StickFigure[i].ToShow != false)
            std::cout << StickFigure[i].ASCII;
    }
}

class WordMap
{
private:
    std::vector<std::string> words;
public:
    WordMap()
    {
        std::ifstream wordStream("wordlist.txt");
        std::string line;
        while (std::getline(wordStream, line))
        {
            if (line.size() > 0)
                words.push_back(line);
        }

    }
    std::string getRandomWord()
    {
        std::mt19937 generator{ static_cast<std::mt19937::result_type>(std::time(nullptr))
        std::uniform_int_distribution<int> rand(0, words.size());
        return words[rand(generator)];
    }
};

class Game
{
    const int maxWrongGuesses{ 8 };
    const std::string word;
    int wrongLetters;
    std::string progress;
    std::string guessedLetters;
public:
    Game(const std::string& word)
        : word{ word }, wrongLetters{ 0 }, progress(word.size(), '_'), guessedLetters{ ""
    {
    }

    ~Game()
    {
        if (wrongLetters == maxWrongGuesses)
            std::cout << "\nYou've been hanged\n";
        else
            std::cout << "\nYou guessed the correct word!\n";
        std::cout << "The word was " << word << "\n";
    }

    bool playGame()
    {
        system("cls");
        DrawHangman();
        if (StickFigure[2].ToShow == true)
            std::cout << "\n";
        std::cout << "Guessed Letters: " << guessedLetters << "\n";
        std::cout << progress;

        char guess = getGuess();
        IsContains(guess);
        return ((wrongLetters < maxWrongGuesses) && (progress != word));
    }
private:
    char getGuess()
    {
        char guess;
        std::cout << "\nEnter letter: ";
        std::cin >> guess;
        while (guessedLetters.find(guess) != std::string::npos)
        {
            std::cout << "You have guessed this letter.\n";
            std::cout << "Enter another letter: ";
            std::cin >> guess;
        }
        guessedLetters += guess;
        return guess;
    }

    void IsContains(char& guess)
    {
        if (word.find(guess) != std::string::npos)
        {
            for (int i{ 0 }; i < word.size(); ++i)
            {
                if (word[i] == guess)
                    progress[i] = guess;
            }
        }
        else
        {
            StickFigure[wrongLetters].ToShow = true;
            ++wrongLetters;
        }
    }
};

int main()
{
    WordMap word;
    const std::string wordToGuess = word.getRandomWord();
    Game game(wordToGuess);

    while (game.playGame())
    {
    }
    return 0;
}
```

nascardriver
September 9, 2019 at 2:53 am · Reply

- Initialize your variables with brace initializers.
- Limit your lines to 80 characters in length for better readability.
- Don't compare booleans to false/true.
- Use `std::size` instead of manual array size calculation.
- You're not using `BodyPart` for anything.
- Line 42: `if (StickFigure[i].ToShow)`
- You're using the same name style for variables, function, and types. This will lead to confusion.
- Don't use `system`, it doesn't work on other platforms.
- The loop in line 41 should stop once it found a part that doesn't get drawn.
- Seed random number generators only once. Seeding them multiple times resets the sequence and can cause the same "random" numbers.
- Use `constexpr` for compile-time constants.
- Initialize empty strings with empty curly braces. The `const char*` constructor is slower.
- Constructors and destructors are for construction and cleanup only. The compiler is allowed to omit them, even if it changes your programs behavior.
- Use single quotation marks for characters.
- Inconsistent name style.
- Line 147+: `while (game.playGame());`
- Pass fundamental types by value.

Try to incorporate these comments in your code to make sure you understand them. Feel free to ask if you have any questions.

I assume you mean char arrays. You can do it with them, but it'll be a lot more work because you have to manage the memory yourself. Use `std::string` and `std::string_view` unless you have a reason to use char arrays.

Ryan
December 12, 2019 at 6:47 am · Reply

Revisiting this again and I wish I hadn't stopped after chapter 14. I had just forgotten everything I learnt in C++. I am revisiting everything since the last two weeks, trying to absorb everything and understand everything to my fullest. I made a promise this time, I will finish the course and try to learn as much as I can and become a better computer scientist. Thanks nascarddriver for answering my questions along the way.

nascardriver
December 12, 2019 at 7:20 am · Reply

Good to see you're giving it another try. It took me several shots too before I switched to C++.
Was there anything in specific that turned you off last time?

Ryan
December 12, 2019 at 8:41 am · Reply

No. Just went through depression. The error handling stuff was pretty boring though i.e not much action was going on.

However not long after, I luckily got a 1 week shadowing at a really big tech company in London's financial district. Dad knew a friend who knew a friend who knew a cousin lol.

Everyone there was super busy and didn't actually help me much since my knowledge is limited (i'm still in HighSchool).
You know what surprised me, is the amount of money the developers make there and only one guy in the dev team knew how to code in c++. What a travesty.
However I got to write a shitty program that could read their log files (or something, don't know what its called)and decrypt it and then verify it's authenticity. I would use this site as reference, mostly Chapter 18, to read each line to push_back() into a vector. And rearranging the strings a bit. Rest was some complicated maths.

Anyway I've been taking the step forward now to advance my knowledge. Seeing ppl achieve great things in CompSci, makes me strive to work harder.

Matt
January 9, 2020 at 6:18 am · Reply

I figured I'd respond to this since it seems you're interested in feedback. First, this is more of a hobby than anything for me, my employment has nothing to do with coding. Last time I worked on these tutorials I made it to chapter 9 then burned out. Chapter 9 was unique in that it felt...boring? Compared to previous chapters where we had learned so much new functionality that had major and instantly tangible impacts on what it was possible to code, chapter 9 just felt like it dragged on. Granted, the functionality overloads give us is huge, but you were interested in what made me take a break and that's where I lost interest last time. I don't think making overloads exciting is really possible though, ha.

TL;DR - I think it was a combination of having gone through 8 chapters in a few weeks and simply burning out, and chapter 9 being very syntax heavy with very little "fun" compared to previous chapters (eg: coding blackjack!)

Regards!

nascardriver
January 9, 2020 at 6:48 am · Reply

Thanks for your feedback!
We're thinking about adding tags to lessons indicating how important they are (or to whom they are important). The lessons on operator overloading could be marked as "optional", "read later", or similar. This would allow them to be skipped for the time being and read them later on demand. Do you think this could prevent chained dull lessons from getting too boring?

> very little "fun" compared to previous chapters
I'm going to have to disappoint you. The further you get into the tutorials, the less exciting the examples are. Most of the topics in the later chapters would require too much code to make the examples realistic and useful. We're trying to stay on topic, realistic examples would be mostly off-topic, except for a tiny part where the new feature is used.
My common recommendation of having a small personal project solves the issue of dull examples/quizzes. When you learn something, you can apply it to your project, that's better than every example.
Don't let this demotivate you too much. Once you finished chapter 15 you're pretty much done. The rest are repetitions and references, you don't need those unless you have nothing else to do.

Anastasia
September 8, 2019 at 11:58 am · Reply

I don't understand the difference between 1b (a text editor and a file) and 1d (a programmer and a computer)... I'd say they are both dependencies.

The TextEditor class needs a File object, so it can pass it to the save() function. Otherwise the save() function wouldn't work. This means some functionality of the TextEditor class depends on the File class.

The Programmer class needs a Computer object to pass in to the watchCatVideos() function. Without a Computer object it won't work. Some functionality of the Programmer class depends on the Computer class.

Is the difference in how essential a dependency is for the overall functionality? I mean if a Text editor can't save anything it's almost pointless, while a programmer should be able to survive a day or two without cat videos... Or it's more abstract than that and I just didn't get it? :/

Alex
September 10, 2019 at 11:54 am · Reply

I... agree with you. I've replaced that question with another one. Thanks for the feedback.

Anastasia
September 10, 2019 at 12:23 pm · Reply

Thank you, I think the new one illustrates association much better (though I liked the programmer watching cat videos, even if their relationships with computers weren't clear lol).

But there's a little bug - the 1e question is hidden under 1d solution now.

Alex
September 16, 2019 at 12:28 pm · Reply

Fixed. Thanks! I'm a little sad to see the humor go too, but this it's better to be accurate than funny.

Vir1oN
July 9, 2019 at 4:28 am · Reply

I think there`s a word missing in the last sentence of the quiz "2) Select one: If you can design a class using (composition, aggregation, association, or dependency), then you should." "Should use" or "should choose" may sound better.

sallyx
June 2, 2019 at 4:21 am · Reply

Members existence managed by class -> Dependency -> YES?
A text editor class with a save() function that takes a File object. -> How does the text editor manage File ojbect existence?

Alex
June 11, 2019 at 2:13 pm · Reply

A dependency does not have its existence managed by the class. The save function is passed the File from an external source.

Atas
July 11, 2019 at 5:03 am · Reply

I think sallyx is referring to the table where Dependecy has a 'Yes' in 'Members existence managed by class?' A typo, pehaps?

Alex
July 13, 2019 at 8:03 pm · Reply

Oh, yes. Lesson updated. Thanks for pointing that out!

Ran
March 29, 2018 at 10:40 am · Reply

Unfortunately, I had a hard time on this quiz.

b
Association. Test editor use a save() function to save a file.

Wrong.
Why wrong. This is not a use relation?

d
Dependency. It is similar to the function std::cout.

Wrong.
Why wrong? In this case, the Program is similar to a normal function;
A Computer can be viewed as a function (similar to std::cout). So,
the relation could be described as Dependency?

nascardriver
March 29, 2018 at 11:16 am · Reply

Hi Ran!

> I had a hard time on this quiz
That's not a problem. You don't need to know how things are called in order to understand them. As long as you understood lesson 10.6 and 10.7 you'll be fine (Unless you're learning c++ for school/university, then you'll need to know this too).

Alex
March 31, 2018 at 3:18 pm · Reply

b) Typically the existence of the File class would be managed by the text editor, so this fits better with a dependency.
d) The computer and programmer likely exist independently of each other (one doesn't necessarily manage the other).

Dani
November 22, 2017 at 10:55 am · Reply

Alex, why 1d is an association and not a dependency?
As I see it, the computer is not a part of the programmer, but something he uses to see the cat videos (like std::string would use std::cout to print on screen).
What am I missing?

Alex
November 24, 2017 at 3:39 pm · Reply

It could be either. If the Programmer has a ViewCatVideos() function that takes a Computer parameter, then the Computer is a dependency.

But association really a better fit here. More likely the Programmer class has an associated Computer member pointing to the Computer being used for all of the tasks the Programmer is likely to do, so the caller doesn't have to pass in the associated Computer for every member function. This would be an association.

Anastasia
September 8, 2019 at 12:04 pm · Reply

Ah, I think I understand better now...
But in case Programmer has-a computer (a pointer to a computer object) which it uses for several different functions, wouldn't it be an aggregation? Although a computer is not a part of a programmer... Or can it be considered as such (in an abstract sense)?

edit: And yes, it says in the answer that the Programmer and the Computer are otherwise unrelated. But for example in 1c question - An Adventurer and a Sword would be otherwise unrelated too, as I see it...
I'm sorry for being so dumb, but this Programmer confuses me :(

Akshay Chavan
July 22, 2017 at 2:48 am · Reply

Hi Alex,

In the line "To quality as an association, and object and an associated object must have the following relationship", shouldn't it be "qualify" and "an object" ?

Martin
April 23, 2017 at 11:32 pm · Reply

To quality as a composition/aggregation/association should be "to qualify as a..."

"a composition that stories copies" should be
"a composition that stores copies"

Alex
April 24, 2017 at 7:35 pm · Reply

Thanks, fixed!

Mohammad
April 21, 2017 at 3:50 pm · Reply

Hello

about question 1d, only one person can use the computer at a time, doesn't that mean its not an association?

Alex
April 23, 2017 at 9:08 am · Reply

'Nope. The associated object (member) *can* belong to more than one object (class) at a time. Note the word "can" -- that doesn't mean it has to.

Mohammad
April 24, 2017 at 4:19 pm · Reply

oops lol, i need to be more careful and not misunderstand things.
Thanks for clarifying, and the awesome summary

EGa
November 1, 2016 at 6:01 pm · Reply

How to make composition in C++ though?

LuatNT
September 13, 2016 at 8:31 pm · Reply

@Alex Could you please explain more detail about "1c" case? I think that the answer can be aggregation, am I wrong?

Alex
September 14, 2016 at 11:22 am · Reply

You're right, my mistake. I'll update the answer.