10.x — Chapter 10 comprehensive quiz

BY ALEX ON SEPTEMBER 12TH, 2016 | LAST MODIFIED BY ALEX ON JULY 22ND, 2017

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	Yes
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).
Hide Solution

Composition -- The animal type and name don't have a use outside of the Animal.

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.
Hide Solution

Dependency -- The text editor class is using the File object for the task of saving to disk.

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.
Hide Solution

Aggregation -- When the Items are associated with the Adventurer, the Adventurer has-them. A sword being used by an adventurer can't be used by anybody else at that time. But the Adventurer doesn't manage the items existences.

1d) A Programmer uses a Computer to watch cat videos on the internet.
Hide Solution

Association -- The programmer and Computer are otherwise unrelated, except when the Programmer is using the Computer to watch kittens ride around on top of Roombas.

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

Aggregation -- The computer has a CPU, but does not manage its existence.

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

composition

11.1 -- Introduction to inheritance

Index

10.7 -- std::initializer_list

14 comments to 10.x — Chapter 10 comprehensive quiz

Ran