HowToDoInJava

Association, Aggregation and Composition

Last Updated: January 25, 2022 By: Lokesh Gupta Java Object Oriented Programming Java OOP

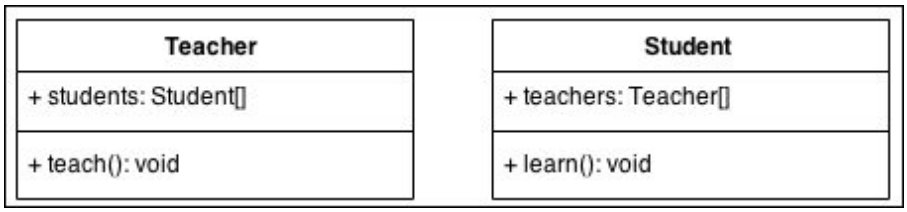
Association, aggregation and composition are three kind of relationships which classes can have in object oriented programming. Let's understand the difference between them.

Table of Contents

1. Association in Java

We call **association** those relationships whose **objects have an independent lifecycle** and where there is **no ownership between the objects**.

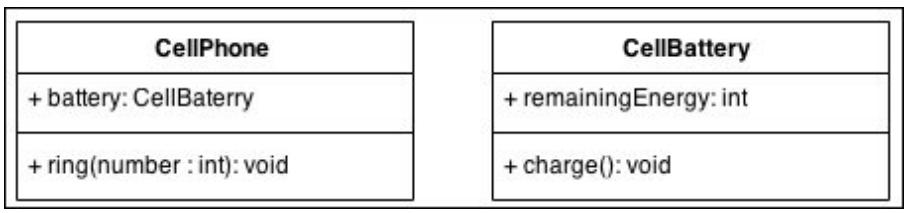
Let's take an example of a teacher and student. Multiple students can associate with a single teacher, and a single student can associate with multiple teachers, but both have their own lifecycles (both can be create and delete independently); so when a teacher leaves the school, we don't need to delete any students, and when a student leaves the school, we don't need to delete any teachers.



2. Aggregation in Java

We call **aggregation** those relationships whose **objects have an independent lifecycle, but there is ownership**, and child objects cannot belong to another parent object.

Let's take an example of a cell phone and a cell phone battery. A single battery can belong to a phone, but if the phone stops working, and we delete it from our database, the phone battery will not be deleted because it may still be functional. So in aggregation, while there is ownership, objects have their own lifecycle.



3. Composition in Java

We use the term **composition** to refer to relationships whose objects **don't have an independent lifecycle**, and *if the parent object is deleted, all child objects will also be deleted*.

Let's take an example of the relationship between questions and answers. Single questions can have multiple answers, and answers cannot belong to multiple questions. If we delete questions, answers will automatically be deleted.

4. Summary

Sometimes, it can be a complicated process to decide if we should use association, aggregation, or composition. This difficulty is caused in part because aggregation and composition are subsets of association, meaning they are specific cases of association.

Association, Aggregation and Composition Relationship

Drop me your questions in comments section.

Happy Learning !!

Recommended Reading:

1. Defining Hibernate Association Mappings
2. Guide to Abstraction
3. Interface vs Abstract Class in Java
4. Encapsulation vs Abstraction in Java
5. Guide to Polymorphism
6. Object Oriented Programming
7. Java Access Modifiers
8. Guide to Inheritance
9. Constructors in Java
0. Java Instance Initializer Blocks

Subscribe

* We do not spam !!

9 thoughts on “Association, Aggregation and Composition”

Mehdi Afifi

February 17, 2020 at 10:53 am

Hi, please write an article about inheritance vs composition vs aggregation and which one is better to use.

Reply

sreenath

December 29, 2019 at 12:53 pm

Not at all, clear man. just a theory, even Wikipedia would have some code examples. Without code examples, this post has no value.

Reply

Lokesh Gupta

December 29, 2019 at 2:47 pm

Thanks for the feedback. I will update the post.

Reply

Valerii Synenko

June 20, 2019 at 12:57 pm

Good explanation, but if it were some examples in the form of the code, it would be great.

Reply

Arpit Agrawal

September 26, 2017 at 11:15 am

I have one question – i.e. can we say that tight coupling exist for Composition? because if we delete the parent object then its collabrated objects gets deleted also so can we say that it is tight coupling?

One more – > Car has seats . Is this composition or Aggregation?

Reply

Rajeev

November 8, 2017 at 8:30 am

One more – > Car has seats . Is this composition or Aggregation?
To ans this question -> this is composition because without car seats can't be exist. In simple term car contains seats so no car no seats. cheers. 🙂

Reply

Zee

October 9, 2018 at 5:57 pm

But if the Car is spoiled we can delete the car and shift the seats to a new car . So it is Aggregation . I think Engine and Car is more of Composition

Reply

UkProgrammer

February 18, 2019 at 5:52 pm

yup right

Reply

Onur Mete Kaplan

May 15, 2019 at 10:59 pm

Actually No. It is a matter of implementation in code, rather than concept

If Engine is created by third party code and passed into Car, then both lifecycles are independent, and it is a aggregation (whole part relationship)

If Engine is created by Car class directly, then Engine's lifecycle is dependent on Car, and it is a composition (full ownership)

Reply

Leave a Comment

Name *

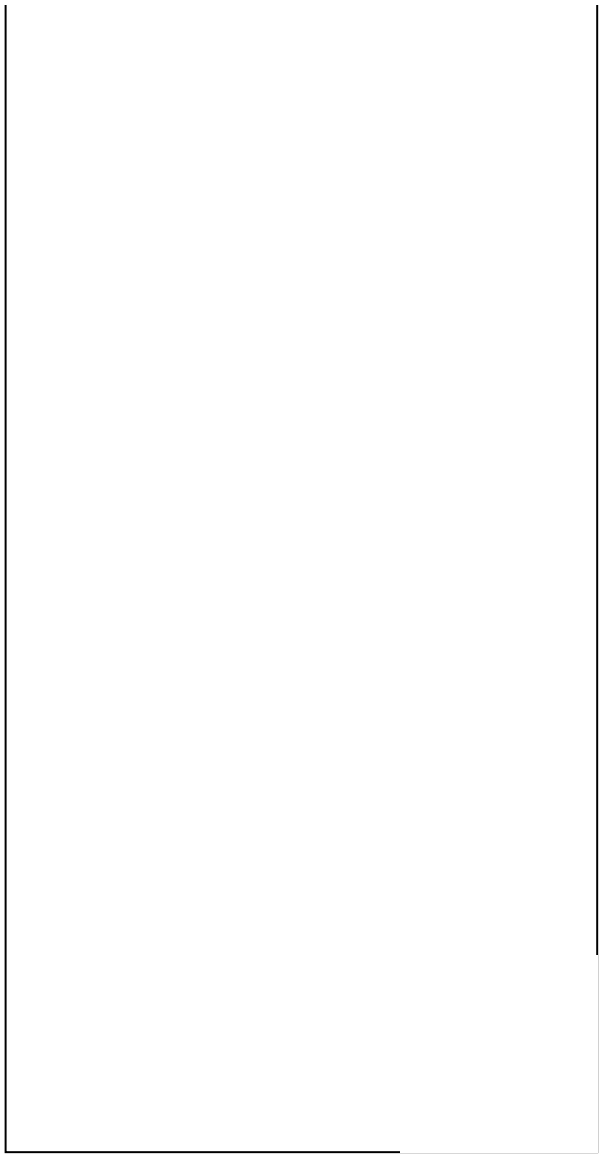
Email *

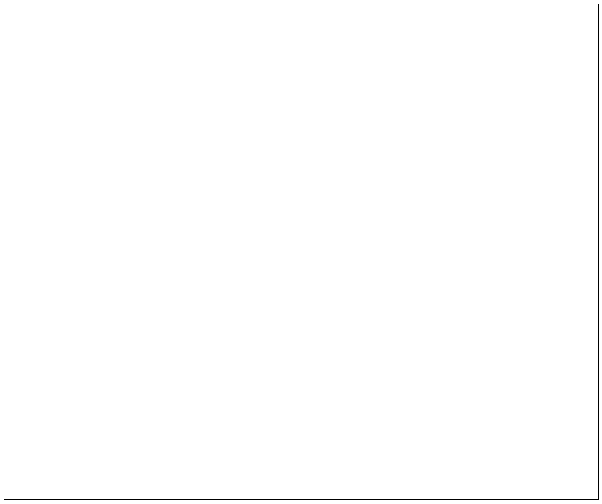
Website

Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search ...





HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial