**Inheritance (IS-A) vs. Composition (HAS-A) Relationship**

--One of the advantages of an Object-Oriented programming language is code reuse.

--There are two ways we can do code reuse either by the implementation of inheritance (IS-A relationship), or object composition (HAS-A relationship).

**IS-A Relationship:**

In object-oriented programming, the concept of IS-A is a totally based on Inheritance, which can be of two types Class Inheritance or Interface Inheritance. It is just like saying "A is a B type of thing". For example, Apple is a Fruit, Car is a Vehicle etc. Inheritance is uni-directional. For example, House is a Building. But Building is not a House.

It is a key point to note that you can easily identify the IS-A relationship. Wherever you see an extends keyword or implements keyword in a class declaration, then this class is said to have IS-A relationship.

**HAS-A Relationship:**

Composition(HAS-A) simply mean the use of instance variables that are references to other objects. For example Pulsar has Engine, or House has Bathroom.

Let’s understand these concepts with an example of Bike class.

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Lets write the code together.

## **What is Composition?**

**Composition** is a method of wrapping the simple objects or data types into a single unit. It is a type of association used to represent the relationship between two classes.

Composition is considered as a strong association type. This is because, in composition, the parent owns the child entity, so the child entity cannot exist without its parent entity. Thus, in composition, the child entity does not have its own life cycle. We cannot directly or independently access a child entity. In the UML diagram, composition is denoted by a filled diamond.

## **What is Aggregation?**

**Aggregation** is another type of association that is used to represent the relationship between two classes. Aggregation is different from an ordinary composition because it gives information about a collection, and not about a mixture.

Aggregation does not imply any ownership on the child entity. In aggregation, the parent and the child entities maintain 'Has-A' relationship but both can also exist independently. We can use the parent and child entities independently. Any modification in the parent entity will not impact the child entity or vice versa.

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

An association is represented with the arrow you can see in front of you. However in our case the customer class is kind of a container for the bank account class which is called the contained in this case, and the BankAccount class is also a container but for the SubAcnt which in this case is contained, this is refered to as a has-a relationship or directional association, also known as an “aggregation”, which in our case is what we have in both cases, we can say that the customer has-a or several bak accounts, or the bank account has-a or several sub-accounts.

If we delete our Customer class, the BankAccount class can still be used by our system, it will some way still be useful, But what about sub-account class? Can we say the same about it? If we delete all bank account classes from our application. The customer class may still be useful in some way to the application, but this SubAcnt. Class. Nothing it is useless. And when the contained object cannot exist without its container this is called composition, and

The customer has-a or several bank accounts, or the bank account has-a or several sub-accounts.

**Problem Statement**

We need to design a Racing Car that can be used in a game and the Car must beb able to be started, run, and stopped.

**Requirements**

1. The car must have an Engine and Tires.
2. The car must have 4 tires.
3. The car must have 1 Engine.
4. The engine must be able to started and stopped.
5. Before the Engine cam be started all Tires must have 32 psi.
6. The car must be. Able to started, running between 1 and 60 mph, stopped and also be able to be restarted.

**Relationships:**

If Object A Has-A relationship to Object B then it must be included as a member variable in Object A.

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Final:**

UML class diagram of your complete model.

You can start, stop and drive the car.

Your turn: Finish the UML diagram and Java code.