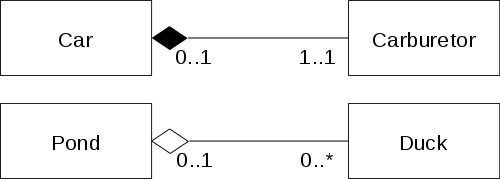
**What is the Composition and how it differs from Inheritance?**

**What is Inheritance?**

Inheritance is one of the most powerful tools in implementing code reusability in object-oriented programming. It refers to the functionality by which one object acquires the characteristics of one or more other objects. Inheritance in C++ means you can create classes that derive their attributes from existing classes. This means you specialize a class to create an is-a relationship between the classes which results in a strong coupling between the base and derived classes. Implementing inheritance promotes code reusability because new classes are created from existing classes. Class inheritance also makes it easier to modify the implementation being reused. But class inheritance has some downsides to it too. First, because inheritance is defined at compile-time, you cannot change the implementations inherited from parent classes at run-time.



**What is Composition?**

OOP provides yet another relationship between classes called composition, which is also known as a has-a relationship. If the features of one object need to be a part of another object, the relationship calls for composition. To compose a class from existing classes, an object of each class should be declared as the member of the new class. In simple words, using an object within another object is known as composition. Many a times, you might want to use an object as a field within another class. You use an object inside a class in composition. Unlike class inheritance, object composition is defined dynamically at run-time through objects acquiring references to other objects. Additionally, composition provides a better way to use an object without compromising the internal details of the object, that is where composition is useful.

**Difference between Composition and Inheritance**

**Approach**

While both inheritance and composition promote code reusability in object oriented system by establishing relationships between classes and they provide equivalent functionality in many ways, they use different approaches. With inheritance, you can create classes that derive their attributes from existing classes so while using inheritance to create a class, you can expand on an existing class. On the contrary, using an object within another object is known as composition. Object composition is an alternative to class inheritance. If the features of one object need to be a part of another object, the relationship calls for composition.

**Relationship**

In inheritance, you specialize a class to create an “is-a” relationship between the classes which results in a strong coupling between the base and derived classes. It enables a hierarchy of classes to be designed and the hierarchy begins with the most general class and moves to more specific classes. By implementing inheritance, member functions from one class become properties of another class without coding them explicitly within the class. In composition, you use an object inside a class and any requests to the object are forwarded to the object. The internal details are not exposed to each other in composition, so it is a “has-a” relationship.

**Implementation**

Class inheritance is defined at compile-time, so you cannot change the implementations inherited from parent classes at run-time. Because inheritance exposes a subclass to details of its parent’s implementation, it often breaks encapsulation. Any changes in the parent class will reflect in the subclass which can create problems when you try to reuse a subclass. Object composition, on the contrary, is defined dynamically at run-time through objects acquiring references to other objects. And because objects are accessed solely through their interfaces, it won’t break encapsulation. Any object can be replaced at run-time by another object as long as it has the same type.