# Interaction Between Class Objects#

By now, we've learned all we need to know about the definition and the behavior of a class. The concepts of inheritance and polymorphism taught us how to create dependent classes out of a base class. While inheritance represents a relationship between classes, there are situations where there are relationships between objects.

The next step for us is to use different class objects to create the design of an application. This means that objects of independent classes would have to find a way to interact with each other.
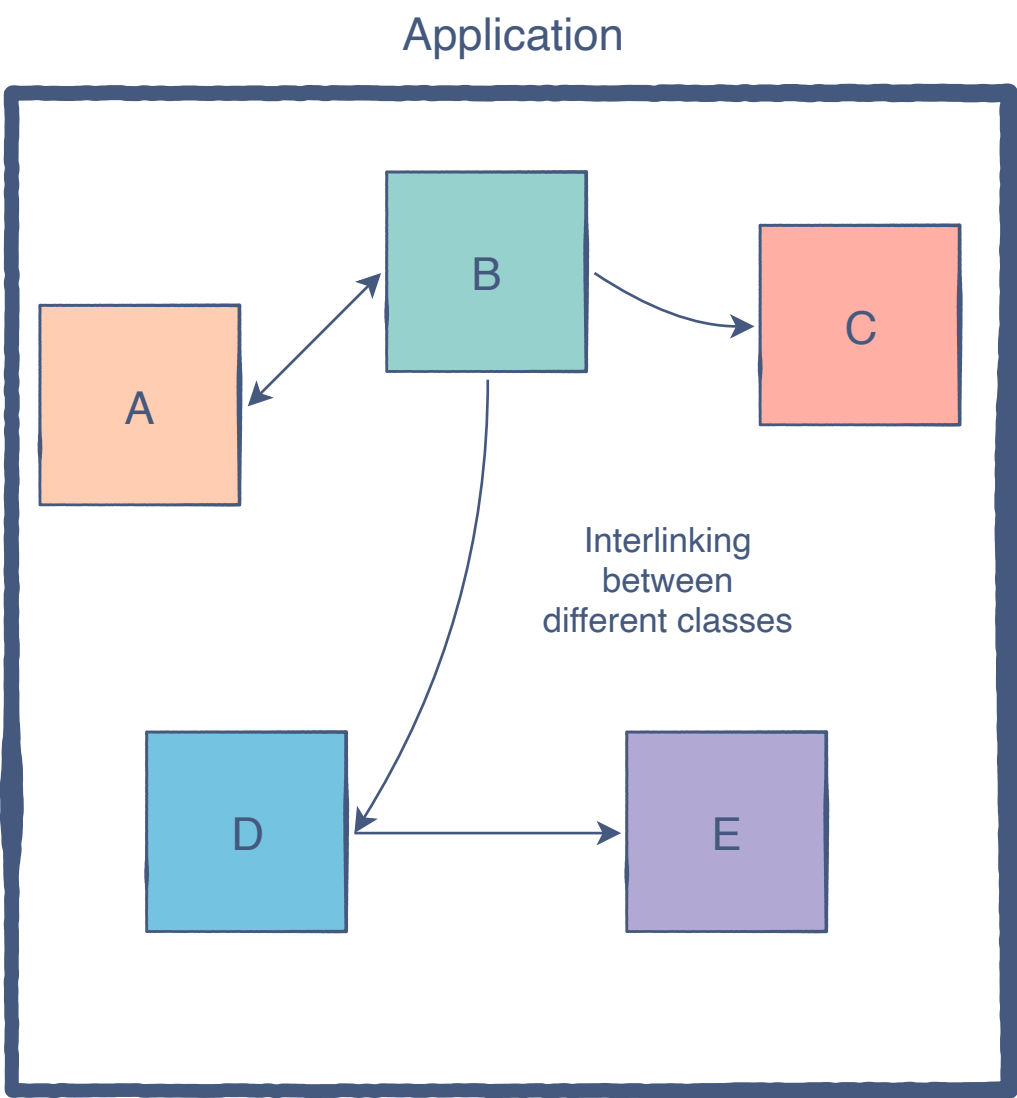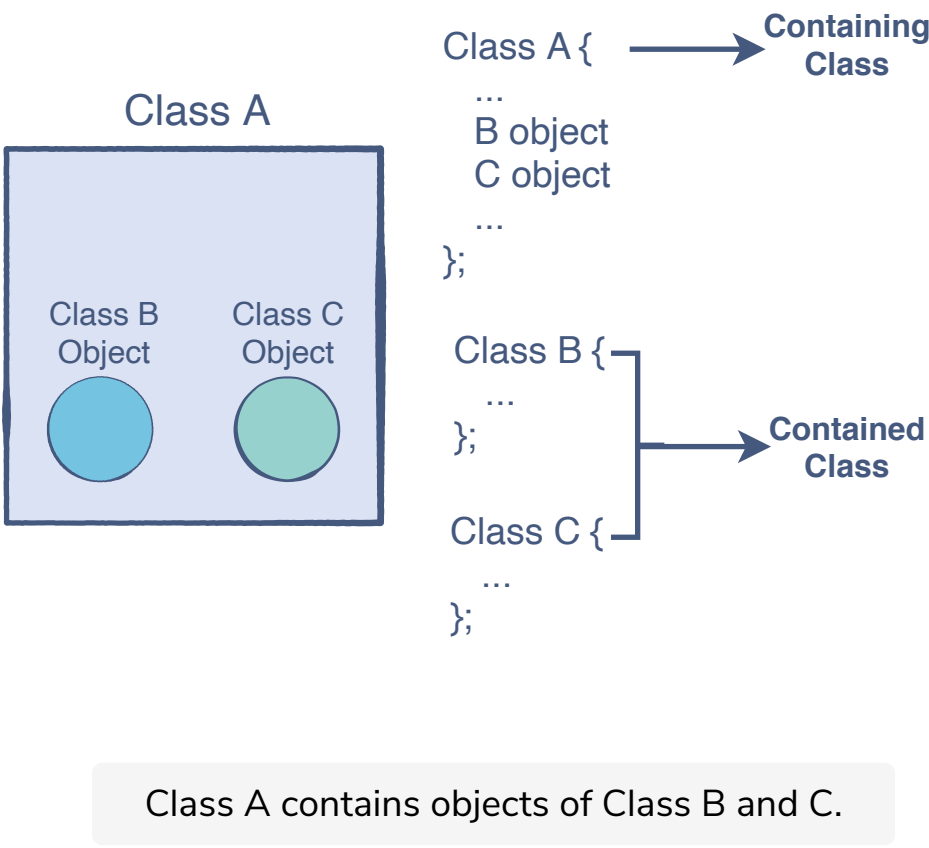

Relationships Between Classes#

There are three commonly used class relationships we need to know for now. We have studied the **IS A** relation in the Inheritance chapter. We'll study the other two below:

Part-of

In this relationship, one class object is a **component** of another class. Given two classes, *class A* and *class B*, they are in a **part-of** relationship if the object of class A is a part of class B, or vice-versa.

An instance of the component class(es) can only be created inside the containing class. In the example to the right, *class B* and *class C* have their own implementations, but their objects are **part-of** the implementation of *Class A* and are only created once a class A object is created. Hence, part-of is a dependent relationship.

Class A contains objects of Class B and C.

Has-a

The object of class exists independent of A.

This is a slightly less concrete relationship between two classes. Class A and class B have a **has-a** relationship if one or both need the other's object to perform an operation, but both class objects can exist independently of each other.

This implies that a class **has-a** reference to an object of the other class but does not decide the lifetime of the other class's referenced object.
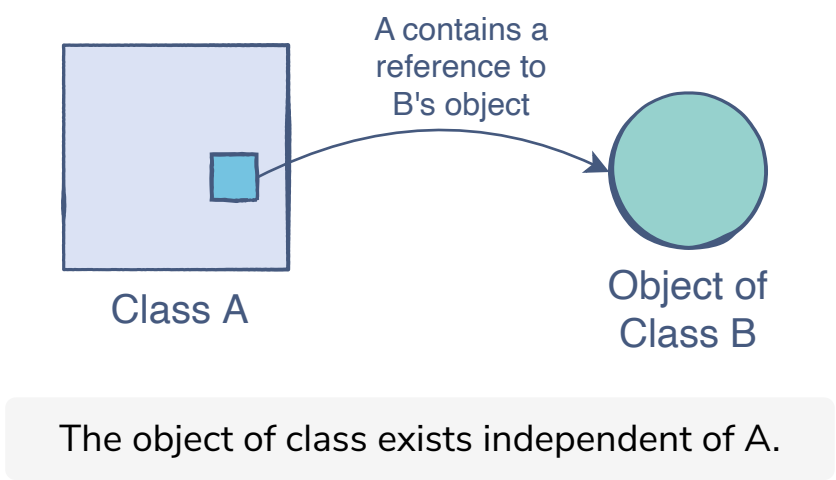
Association#

In object-oriented programming, **association** is the common term used for both the **has-a** and **part-of** relationships but is not limited to these. When we say that two objects are in an association relationship, this is a generic statement which means that we don't worry about the lifetime dependency between the objects. In the next couple of lessons, we will dive into the specialized forms of association: **Aggregation** and **Composition**.