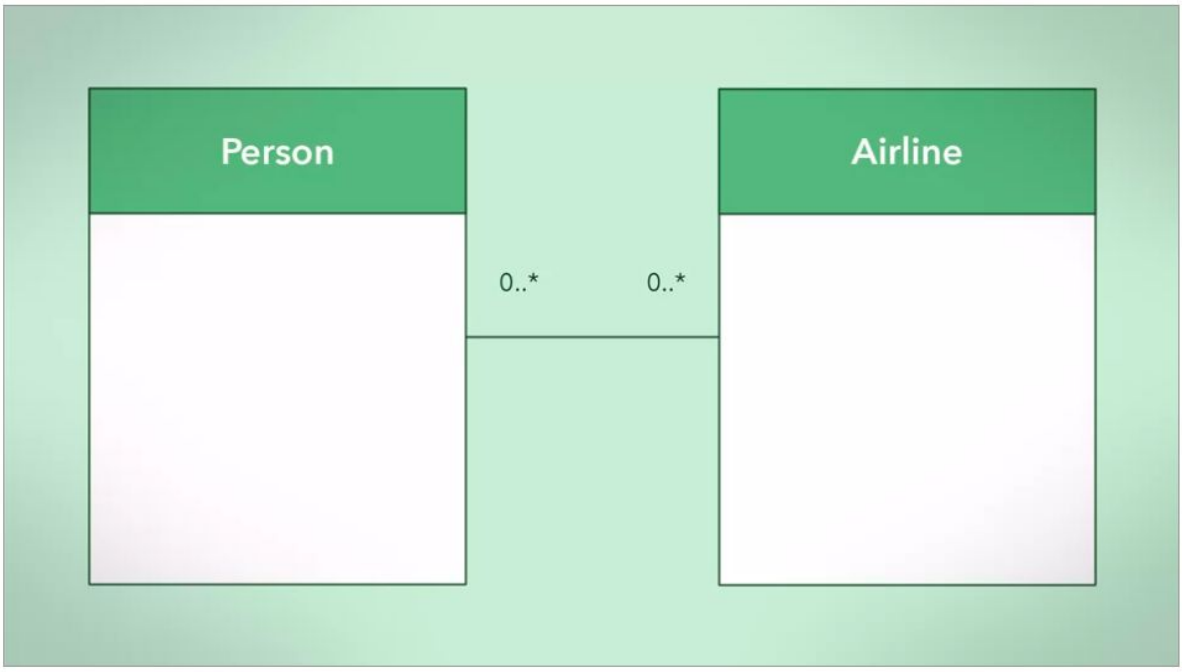
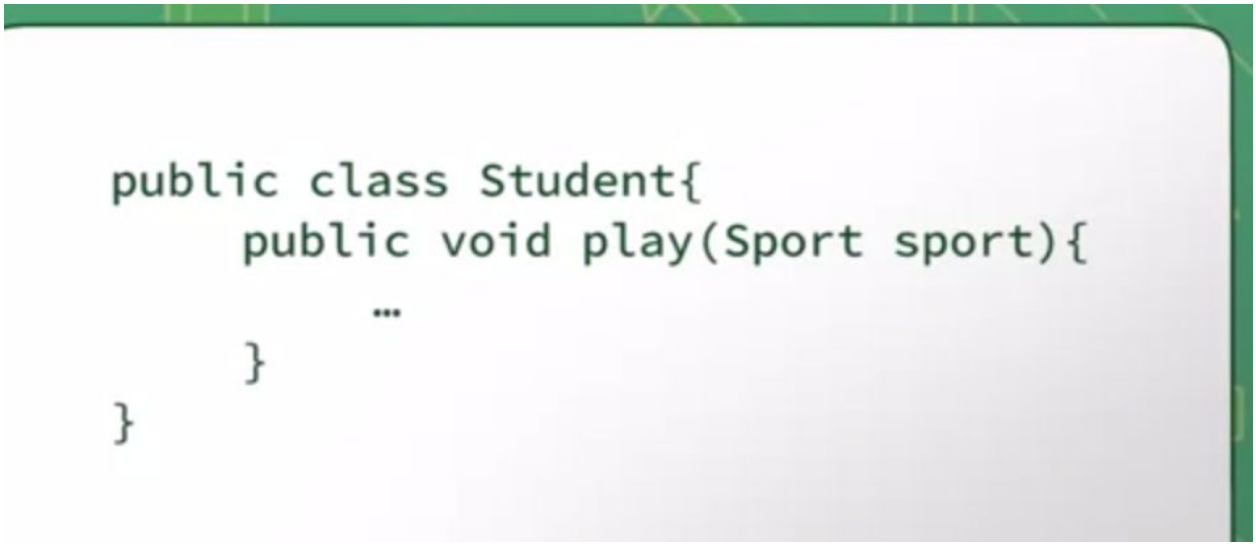
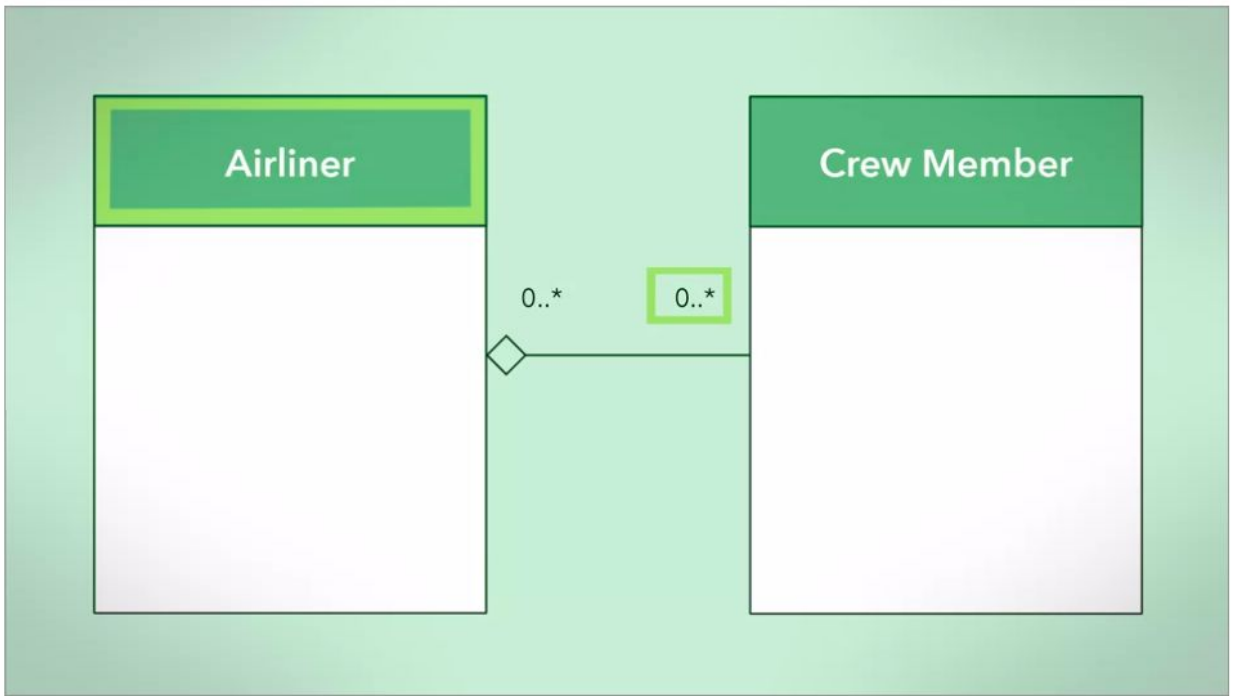
Decomposition: combining different things with different functionalities together to form something.

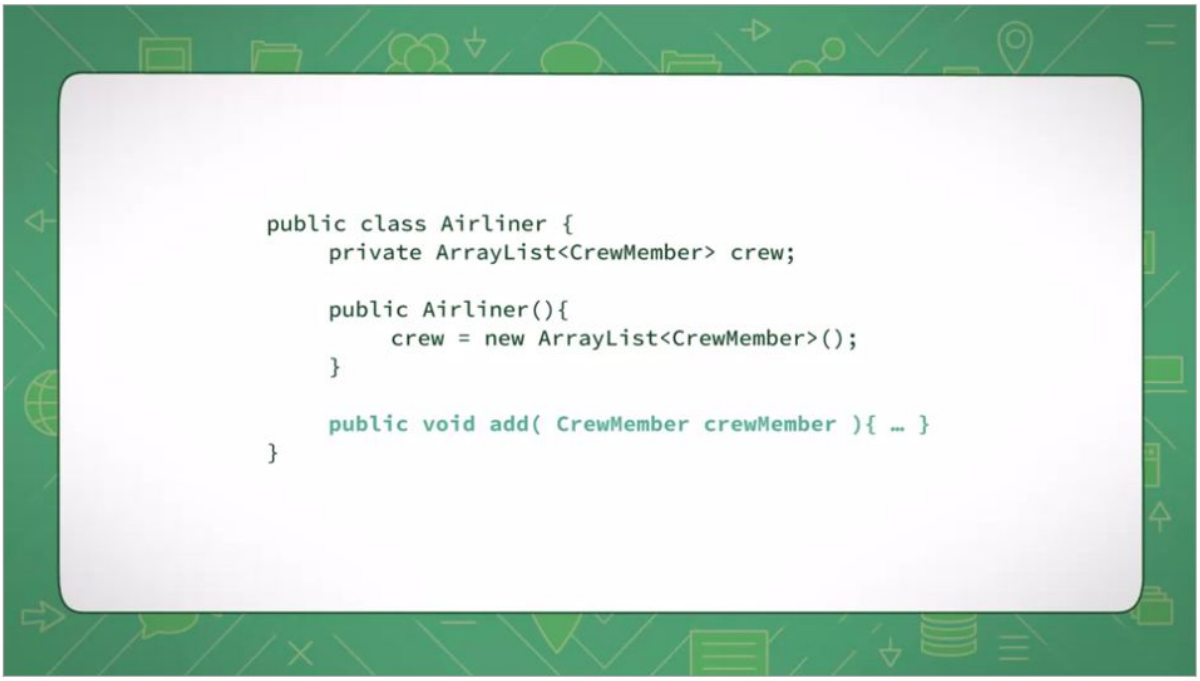
1. There are three types of decomposition:
   A. **Association:** is some relationship. This means that there is a loose relationship between two objects. These objects may interact with each other for some time. For example, an object of a class may use services/methods provided by object of another class. This is like the relationship between person and airline. A person does not generally own an airline, but can interact with one. An airline can also interact with many person objects. There are some persons and some airlines, neither is dependent on the other.





B. **Aggregation:** Aggregation is a has-a relationship where a whole has parts that belong to it. There may be sharing of parts among the wholes in this relationship. The has-a relationship from the whole to the parts is considered weak. What this means is although parts can belong to the wholes, they can also exist independently.



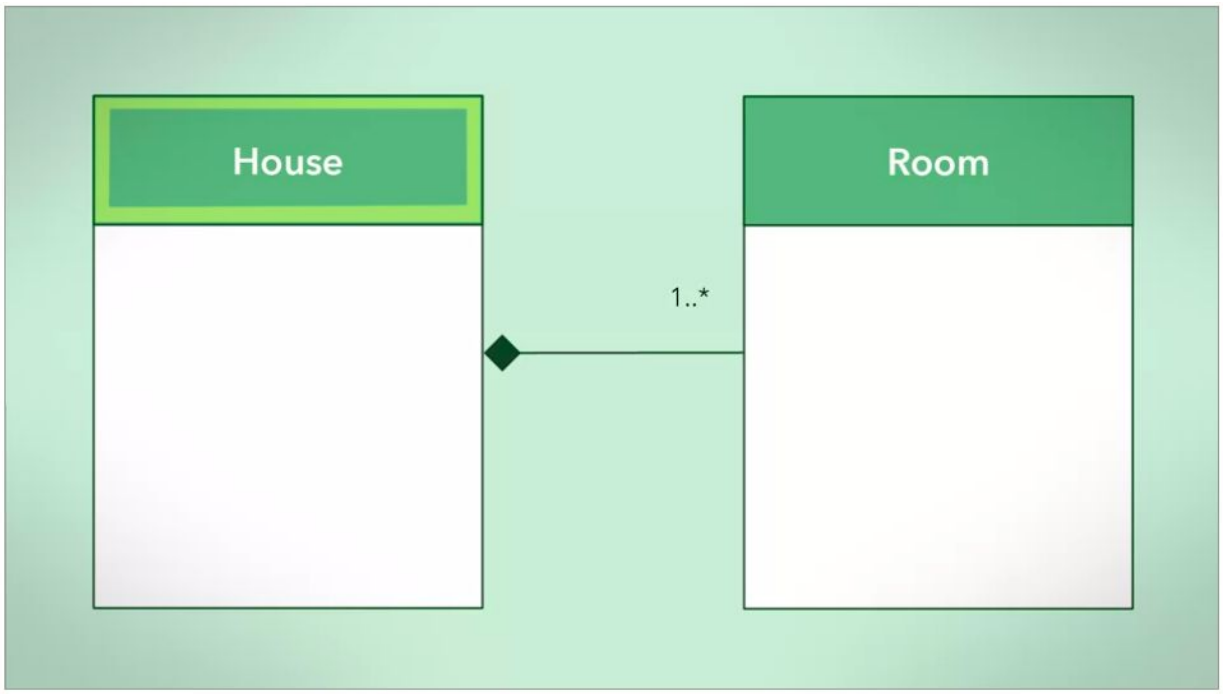


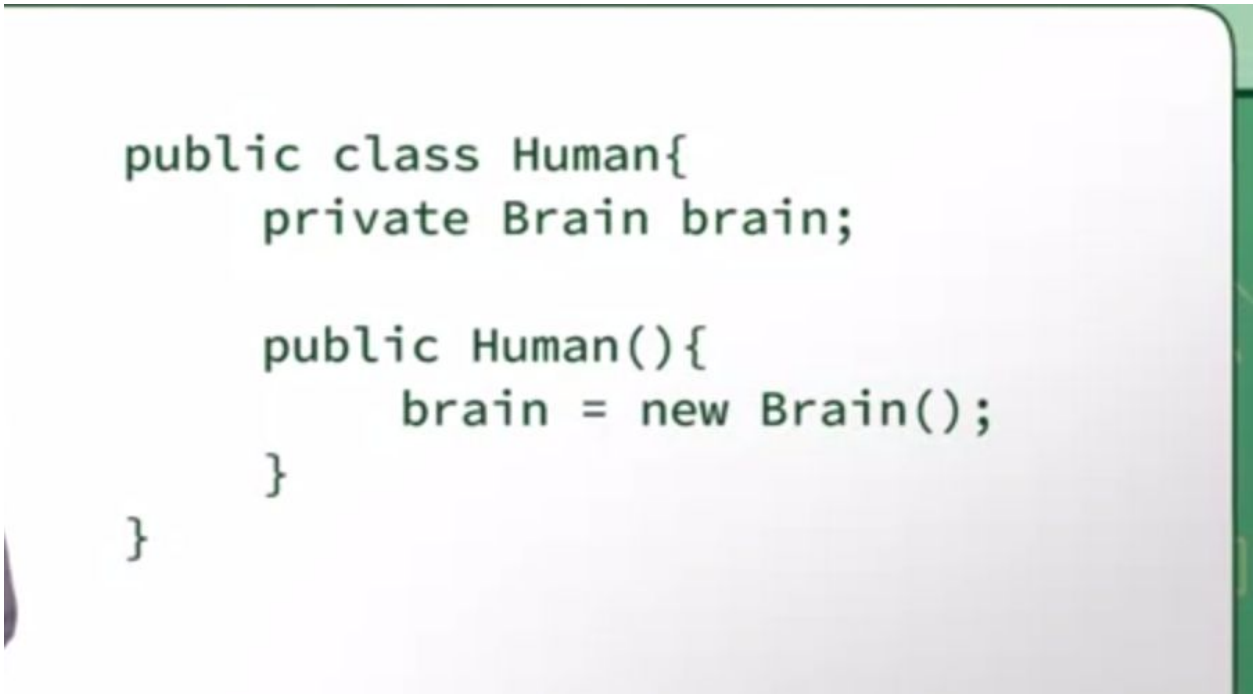
```
public class Airliner {
     private ArrayList<CrewMember> crew;

     public Airliner(){
          crew = new ArrayList<CrewMember>();
     }

     public void add( CrewMember crewMember ){ … }
}
```

C. **Composition:** is an exclusive containment of parts, otherwise known as a strong has-a relationship. What this means is that the whole cannot exist without its parts. If loses any of its parts, the whole ceases to exist. If the whole is destroyed, then all of its parts are destroyed too. Usually, you can only access the parts through its whole. Contained parts are exclusive to the whole.





```
public class Human{
    private Brain brain;

    public Human(){
        brain = new Brain();
    }
}
```