`Composition` is the practice of creating other class objects in your class. In such a scenario, the class which creates the object of the other class is known as the *owner* and is responsible for the lifetime of that object.

Composition relationships are **Part-of** relationships where the *part* must constitute part of the whole object. We can achieve composition by adding smaller parts of other classes to make a complex unit.

So, what makes the composition so unique?

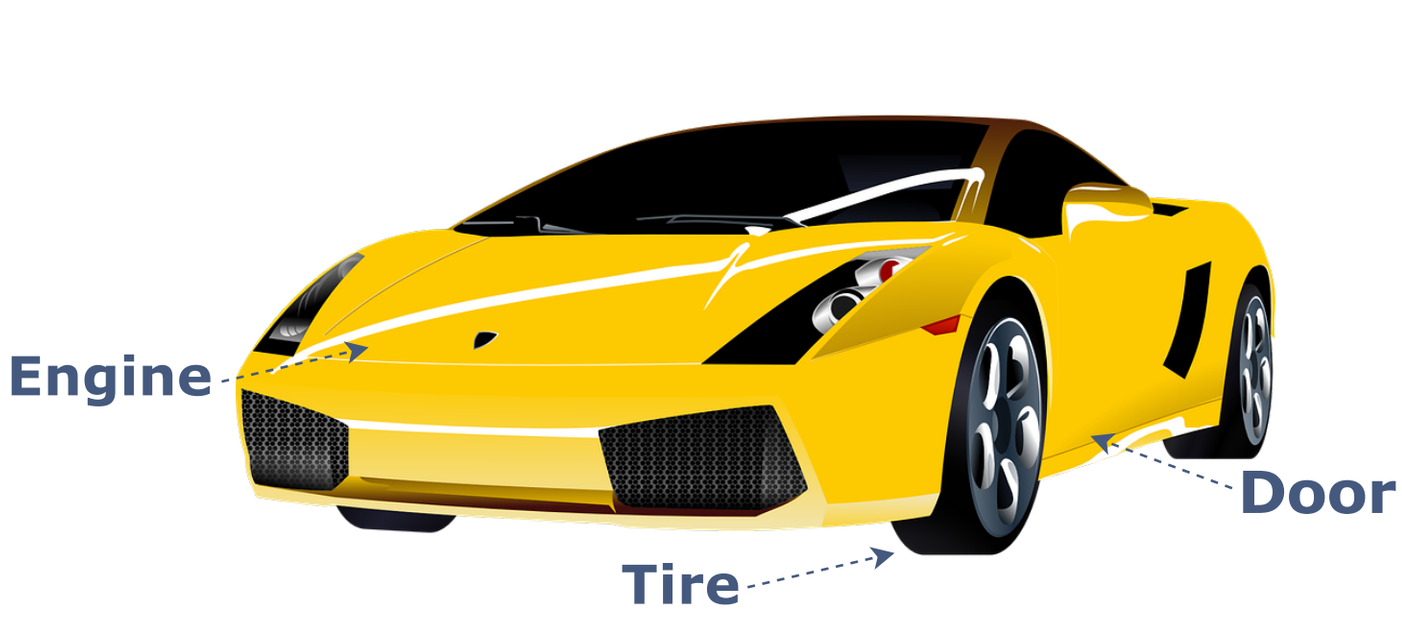
> In **composition**, the lifetime of the owned object depends on the lifetime of the owner.

Example#

A `car` is composed of an *engine*, *tires*, and *doors*. In this case, a `Car` owns these objects so a `Car` is an *Owner* class and `tires`, `doors` and `engine` classes are *Owned* classes.

Implementation#

Let's look at the implementation of `Car` class for better understanding:




```
class Engine {

  private int capacity;

  public Engine(){
    capacity = 0;
  }

  public Engine(int cap) {
    capacity = cap;
  }

  public void engineDetails() {
    System.out.println("Engine details: " + capacity);
  }

}

class Tires {

  private int noOfTires;

  public Tires() {
    noOfTires = 0;
  }

  public Tires(int nt) {
    noOfTires = nt;
```

Run Save Reset

We have created a `Car` class which contains the objects of `Engine`, `Tires` and `Doors` classes. The `Car` class is responsible for the lifetime of the owned objects, i.e., when the Car dies, so does the *tires*, *engine* and *doors*.