Composition

In this lesson, you'll learn how to achieve composition in Java.

We'll cover the following

`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.

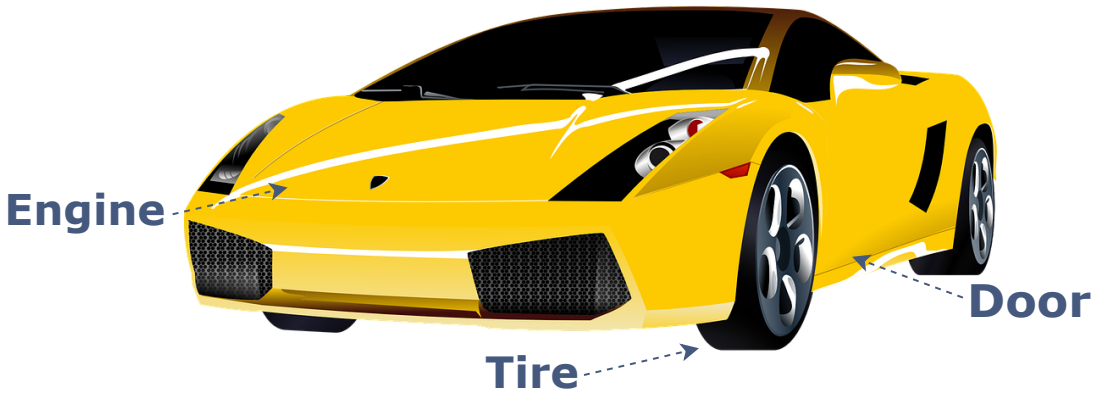
Example

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

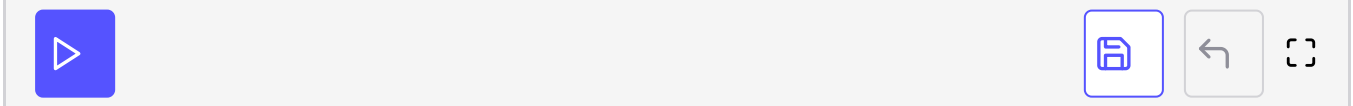
Let's look at the implementation of `Car` class for better understanding:

A car is composed of engine, tires and doors.




```
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;
  }

  public void tireDetails() {
    System.out.println("Number of tyres: " +  noOfTires);
  }
```

```
}


class Doors {

  private int noOfDoors;

  public Doors() {
    noOfDoors = 0;
  }

  public Doors(int nod) {
    noOfDoors = nod;
  }

  public void doorDetails() {
    System.out.println("Number of Doors: " + noOfDoors);
  }

}

class Car {

  private Engine eObj;
  private Tires tObj;
  private Doors dObj;
  private String color;

  public Car(String col, int cap, int nt, int nod) {
    this.eObj = new Engine(cap);;
    this.tObj = new Tires(nt);;
    this.dObj = new Doors(nod);

    color = col;
  }

  public void carDetail() {
    eObj.engineDetails();
    tObj.tireDetails();
    dObj.doorDetails();
    System.out.println("Car color: " + color);
  }

}

class Main {

  public static void main(String[] args) {
    Car cObj = new Car("Black", 1600, 4, 4);
    cObj.carDetail();
  }
}
```



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*.

Now, let's test your knowledge with a quick quiz!