* + Simple objects
    - object state(fields) are basic data types
  + as objects become more complex
    - encapsulated object state is complex, building more layers of abstraction
    - object state for complex objects are modeled by a combination of other objects
  + **Composition**
    - the individual parts that make up the whole are owned solely by the whole
      * otherwise, the parts don't have a reason for existing
      * the constructor usually creates all the fields itself and doesn't need the fields to be passed in; thus usually no setters and often no getters for these encapsulated objects

Ex:

**outside the class:**

Porsche myPorsche = new Porsche();

**within the class:**

public Porsche(){

Engine engine = new Engine();

Tire[] tires = new Tires();

}

* + **Aggregation**
    - the individual parts that make up the whole may also exist on their own
      * the parts can also exist as part of other objects
      * encapsulated objects are provided externally AKA you have to give the parts when you create an instance:
        + Ex:

**outside the class:**

Class class = new Class(teachersArray, studentsArray);

**within the class:**

public Class(Teacher[] teachers, Student[] students){

this.teachers = teachers;

this.students = students;

}

* + usually you mark a class as abstract if it is intended to be a parent class for subclasses that will provide specific implementations of methods
    - But, you can also mark a class abstract if you want to prevent anyone from creating an instance of the parent class directly
      * so a class can be abstract even if none of the methods are abstract
  + interfaces can also include static constants
  + for switch statements, if you want several cases to execute the same code, put them in order like so:

case FIRST:

case SECOND:

case THIRD:

super(park);

break;

* + **Delegation**
    - claiming an "is-a" relationshi with an interface but relying on another object to actually do the work
    - more common with composition, where internals are more hidden