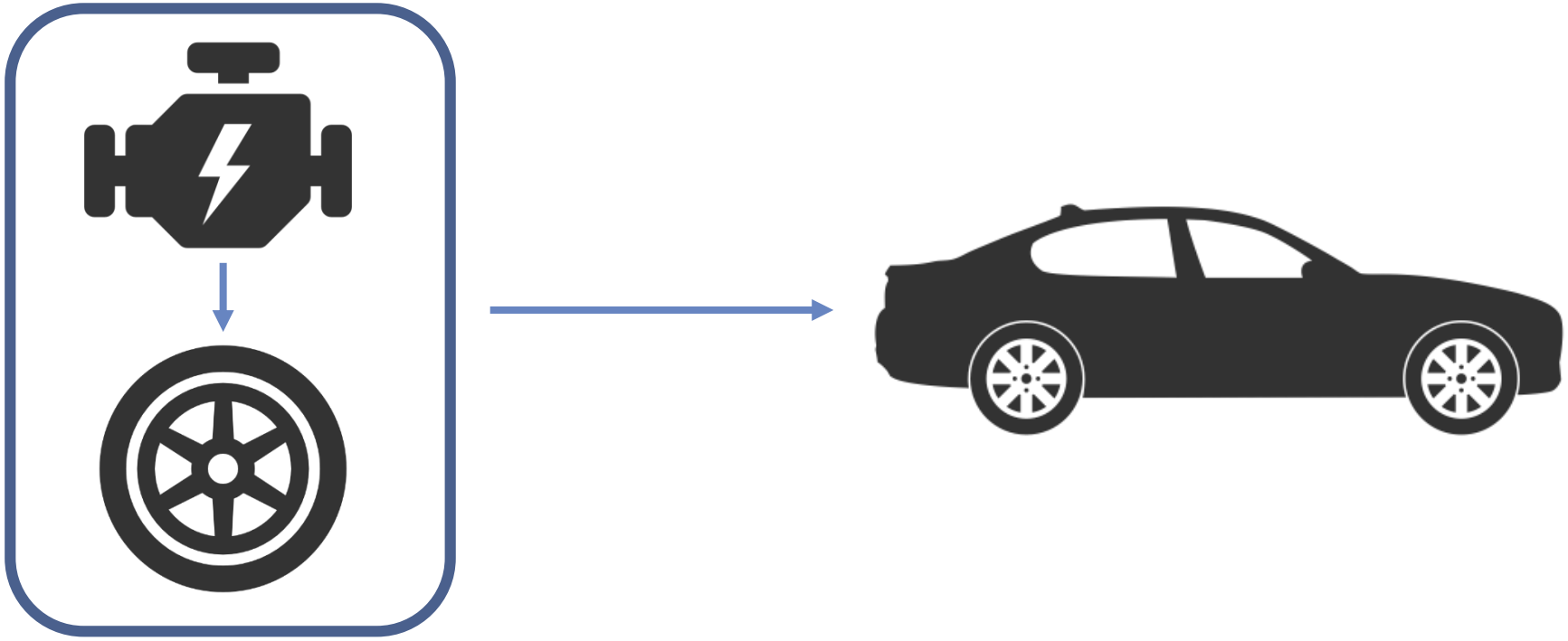
OBJECT ORIENTED PROGRAMMING REITERATION

OUTLINE

1. WHAT IS OBJECT-ORIENTED PROGRAMMING ?
2. WHAT IS A CLASS ?
3. WHAT IS AN OBJECT ?
4. OBJECT ORIENTED PROGRAMMING PRINCIPLES
5. RELASHIONSHIPS BETWEEN OBJECTS

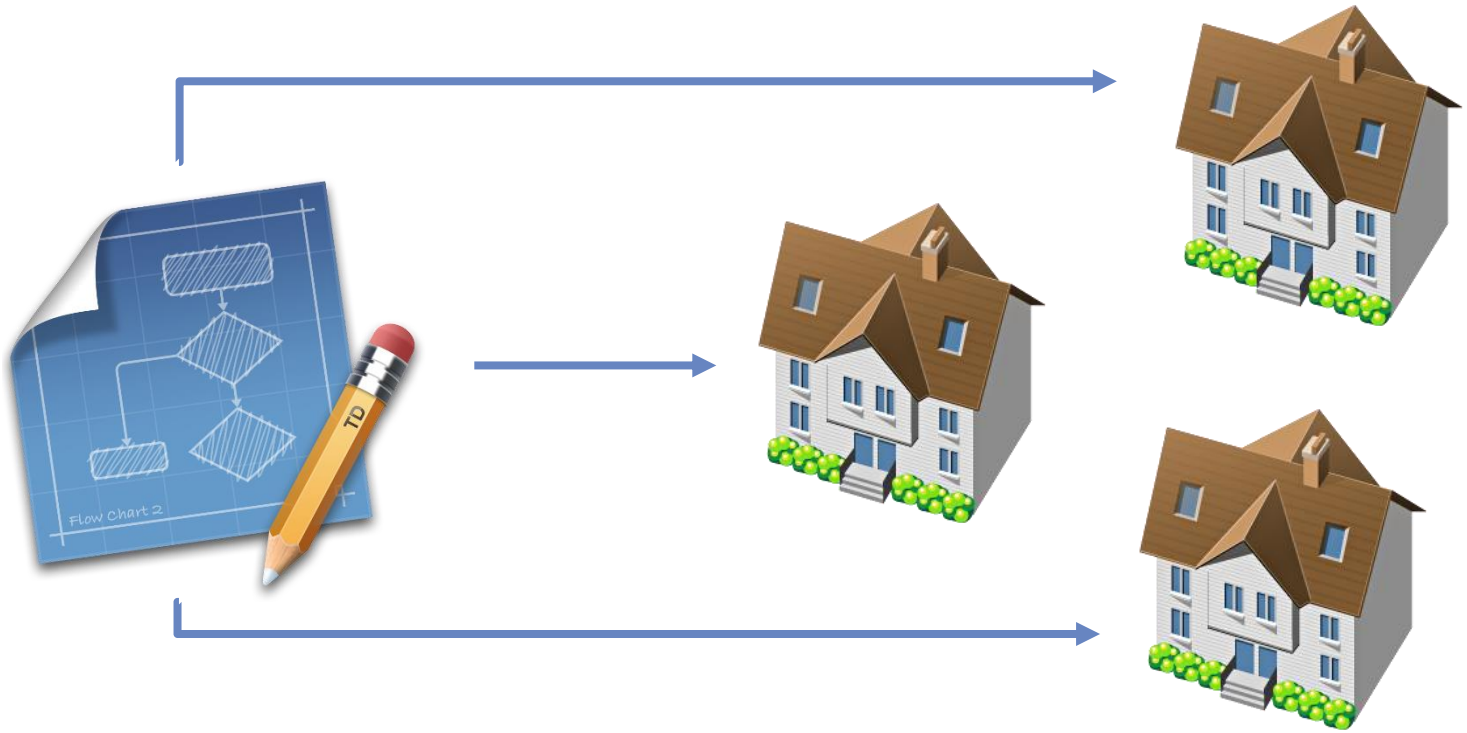
1) WHAT IS OBJECT ORIENTED PROGRAMMING ?

- Is a programming paradigm based on the concept of ***objects*** and how they interact with each other.



2) WHAT IS A CLASS ?

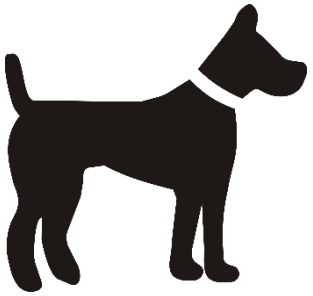
- A class is a blue print from which individual objects are created. A class can contain fields and methods to describe the behavior of an object.



3) WHAT IS AN OBJECT ?

- An object is an instance of a class. Objects are characterized by states (fields) and behaviors (methods).

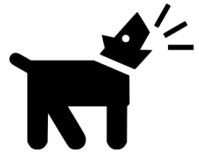
STATES



Name: **Max**

Color: **Black**

BEHAVIORS



Barking

Playing

4) OBJECT-ORIENTED PROGRAMMING PRINCIPLES

ABSTRACTION

- Abstraction is the concept of taking some object from the real world, and converting it to programming terms relative to the perspective of the viewer.

FIRST NAME, LAST NAME

EMAIL

PASSWORD

SHIPPING ADDRESS

E-COMMERCE USER

4) OBJECT-ORIENTED PROGRAMMING PRINCIPLES

ENCAPSULATION

- Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.
- In encapsulation the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class, therefore it is also known as ***data hiding***.
- **To achieve encapsulation in Java**
  - Declare the variables of a class as **`private`**.
  - Provide **`public`** setter and getter methods to modify and view the variables values.

```
public class Dog {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

4) OBJECT-ORIENTED PROGRAMMING PRINCIPLES

INHERITANCE

- Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another.
- With the use of inheritance the information is made manageable in a hierarchical order.
- The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

Name
Color
Can sleep

Name
Color
Nickname
Can sleep
Can play

```
public class Dog {

    protected String name;
    protected String color;

    public void sleep() {
        System.out.println("I am sleeping");
    }

public class Puppy extends Dog {

    private String nickName;

    public void play() {
        System.out.println("I am playing");
    }
```

5) RELASHIONSHIPS BETWEEN OBJECTS

ASSOCIATION

- Association establish relationship between two **classes** through their **objects**.
- The relationship can be one-to-one, one-to-many, many-to-one and many-to-many.
- Association is a relationship where all object have their own lifecycle and there is no owner.

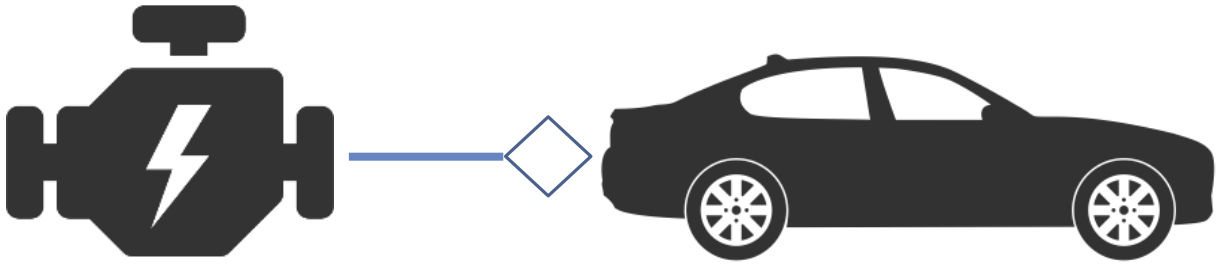
EXAMPLE

- Multiple students can associate with a single teacher and a single student can associate with multiple teachers
- There is no ownership between the objects and both have their own lifecycle.
- **Both can be created and deleted independently.**

5) RELASHIONSHIPS BETWEEN OBJECTS

AGGREGATION

- **“HAS-A” relationship.**
- Aggregation is a specialized form of Association where the parent object doesn’t have sense without the child object.

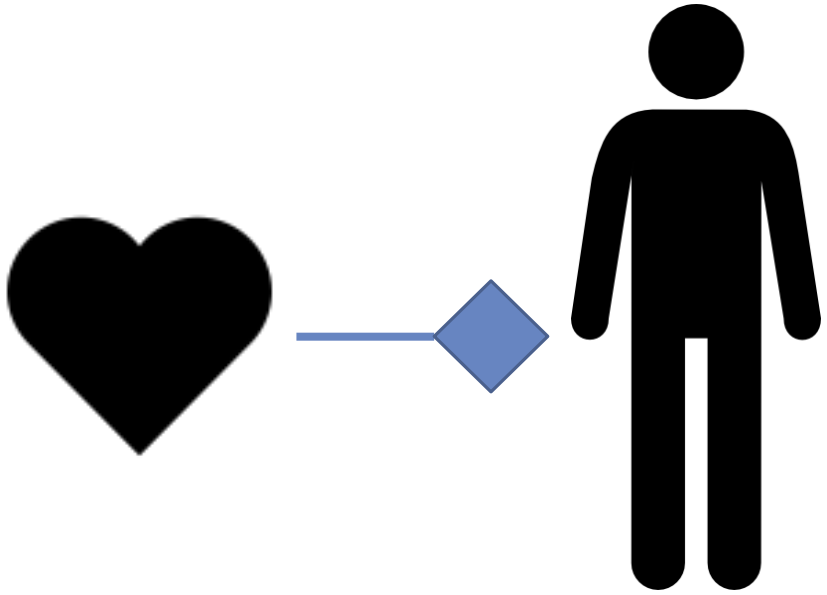


```
public class Engine {

    private int horsePower;

public class Car {

    private Engine engine;
```

5) RELASHIONSHIPS BETWEEN OBJECTS

COMPOSITION

- **“HAS-A” relationship.**
- Composition is a restricted form of Aggregation in which two entities (or you can say classes) are highly dependent on each other.

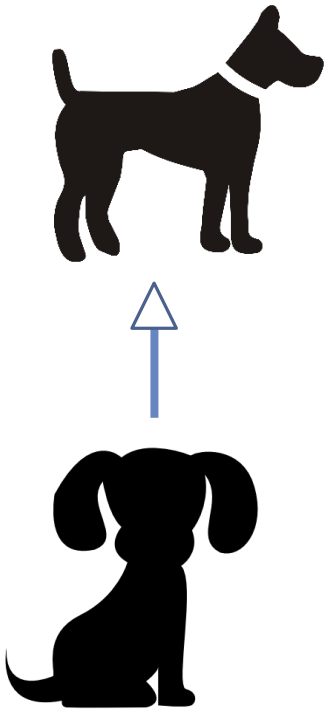


```
public class Heart {

    private int heartRate;

public class Human {

    /*
     * final will make sure that the heart
     * will be initialized
     */
    public final Heart heart = new Heart();
```

5) RELASHIONSHIPS BETWEEN OBJECTS

“IS-A” RELATIONSHIP

- Depends on inheritance.



```
public class Dog {

    protected String name;
    protected String color;

    public void sleep() {
        System.out.println("I am sleeping");
    }

  public class Puppy extends Dog {

      private String nickName;

      public void play() {
          System.out.println("I am playing");
      }
```