OOPs (Object Oriented Programming) Concepts

January, 2023

Objectives of OOPs Concept

- Purpose:
  - To understand software object, class and OOPs concepts
- Product:
  - To understand Object; Behavior and state of object
  - To understand Class
  - Understanding of OOPs concepts like – Polymorphism, Encapsulation, Inheritance.
- Process:
  - Theory Sessions along with a workshop on class and object identification
  - A review at the end of the session and a Quiz.

Table of Contents

Object

REAL-WORLD OBJECTS

Share two characteristics:

They all have ***state*** and ***behavior***.

Example – A vehicle have state (current gear, is moving, current speed) and behavior (changing gear, changing speed / accelerate, applying brakes).

SOFTWARE OBJECTS

Conceptually similar to real-world objects:

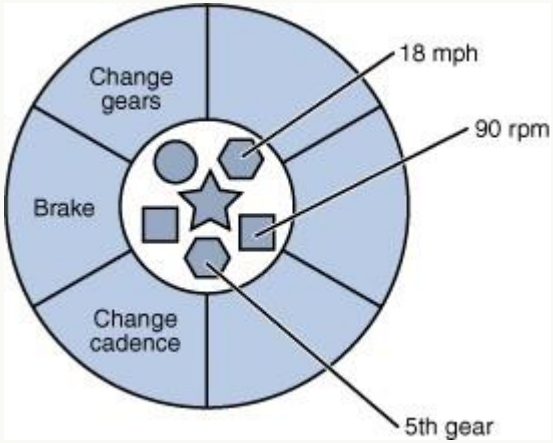
They too consist of state and related behavior.

State as ***fields***

(variables in some programming languages) and

Behavior as ***methods***

(functions in some programming languages)



Class

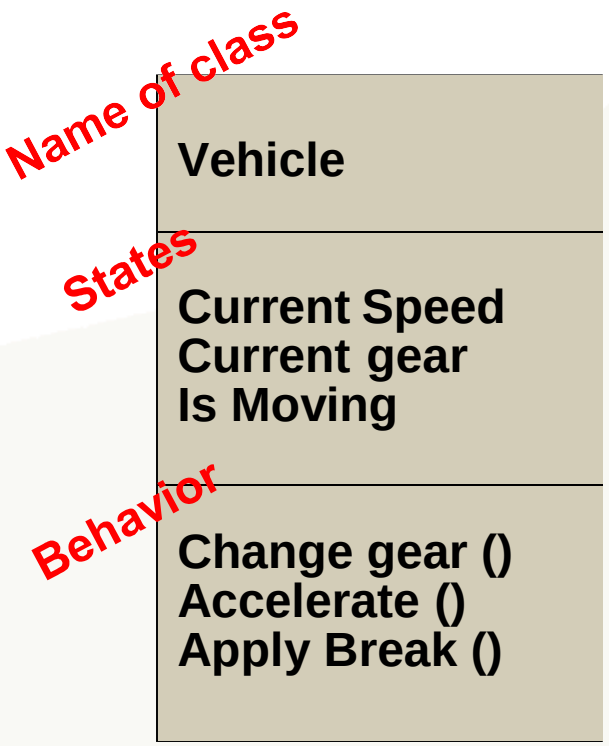
- In the real world, you'll often find many individual objects all of the same kind.

  (example - There may be thousands of other bicycles in existence, all of the same make and model.)

- Each object (bicycle) was built from the same set of blueprints and therefore contains the same components.

- In object-oriented terms, we say that your bicycle is an ***instance* of the *class of objects*** known as bicycles.

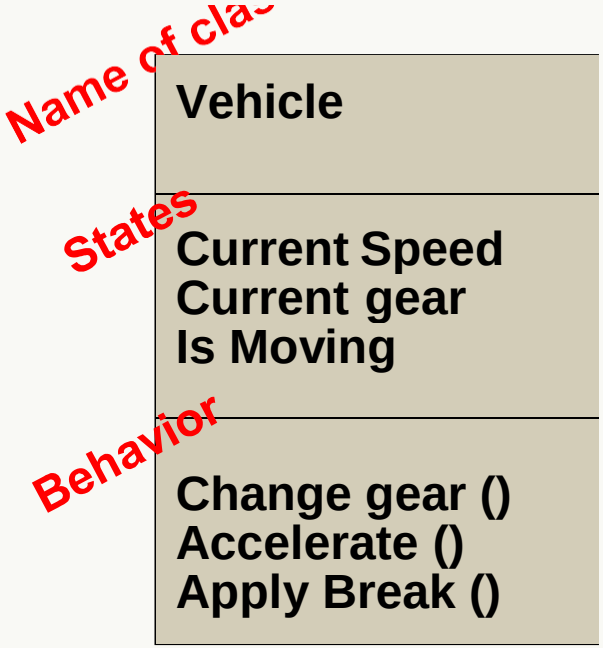
- A ***class* is the blueprint** from which individual objects are created.



Workshops

Workshop 1

- *Consider few real-world objects (like table, car, bottle, etc)*
- *For each object ask yourself two questions: "What possible states can this object be in?" and "What possible behavior can this object perform?"*
- *List down states and behavior of the objects*



Workshop 2

- *Create class diagram of object of workshop 1 (three part rectangular box - three part for its name, states, and behavior respectively)*

OOPs Concepts - Encapsulation

Hiding internal state and requiring all interaction to be performed through an object's methods is known as ***data encapsulation***

- a fundamental principle of object-oriented programming.

Example – Change Gear () behavior of vehicle (say car). Driver just know how to change gear, but he/she has no idea of how physically gear is getting changed inside the engine.

User (driver in this example) have access of this behavior but exact changing mechanism is hiding inside engine

OOPs Concepts - Inheritance

- Different kinds of objects often have a certain amount in common with each other.

  Example – car, bike, train all are vehicle and have common state (current speed, is Moving, etc) and common behavior (accelerate, apply break)

- Yet each also defines additional features that make them different

- We do not need to rewrite common field and functionality (state and behavior) in every class.

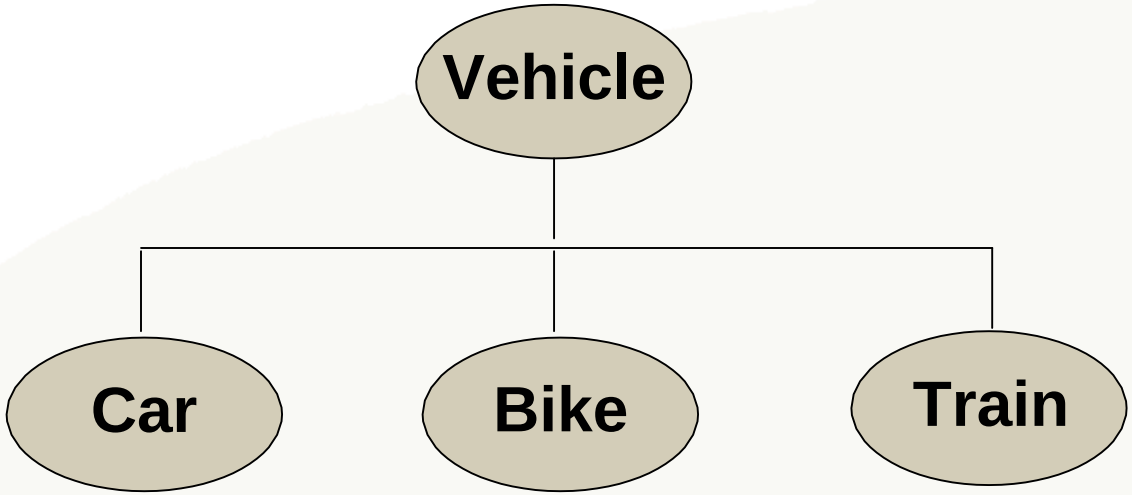
- Object-oriented programming allows classes to ***inherit*** commonly used state and behavior from other classes

- Inheriting class termed as sub class and parent class called super class

OOPs Concepts - Inheritance

- In the Java programming language, each class is allowed to have one direct super class,

- Each superclass has the potential for an unlimited number of *subclasses*:



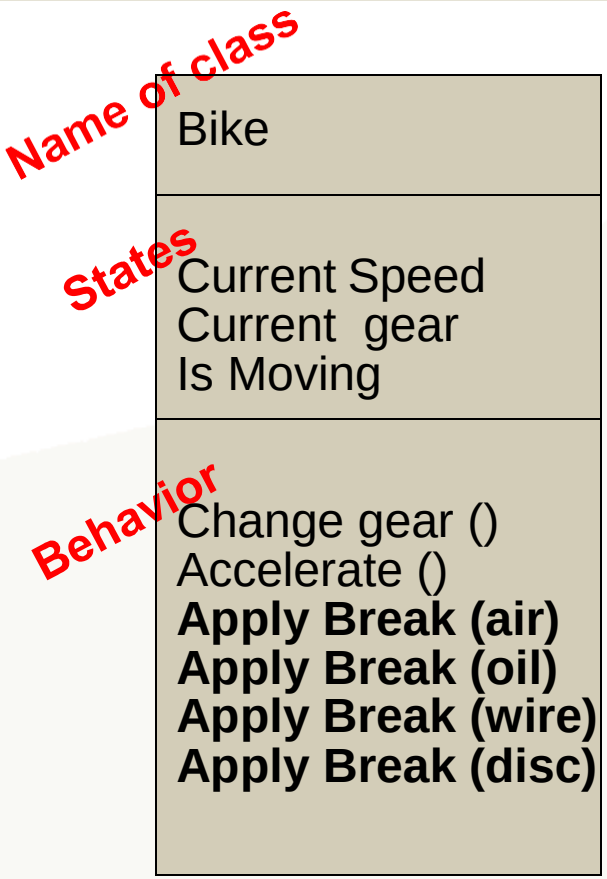
OOPs Concepts - Polymorphism

- Some functions (behavior) of an object can be achieved by different manner

  Example - a vehicle can have various types of break – air break, oil break, wire break, disc break, etc.

- Now the name of function should not change on how it is achieving, but different functions of achieving it should be there in object

- Object-oriented programming allows classes to ***overload*** its own function (behavior)



In Bike class; the apply break functionality is overloaded

OOPs Concepts – Polymorphism

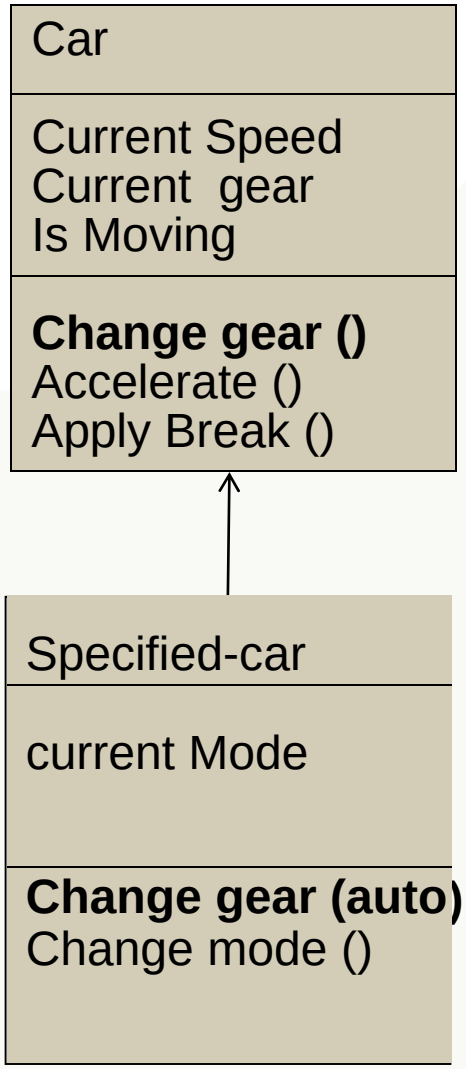
- Some functions (behavior) of an object, which it inherited from its super class may needs to modify

  Example - a specified-car class, which is sub class of car class, has inherited change gear functionality.

  Now, specified-car class is an automatic car, so change gear () should be automatic.

- Object-oriented programming allows classes to ***override*** function (behavior) of its super class

- Object of super class will have original method, while sub class will have overridden method



OOPs Concepts – Polymorphism

- Overloading and Overriding together called Polymorphism
- Overloading scenario arises for functionality (behavior) within same class
- Overriding case comes into picture in case of sub and super classes scenario.

OOPs Concepts –

To remember these OOPs concepts, remember PIE

P **P**olymorphism

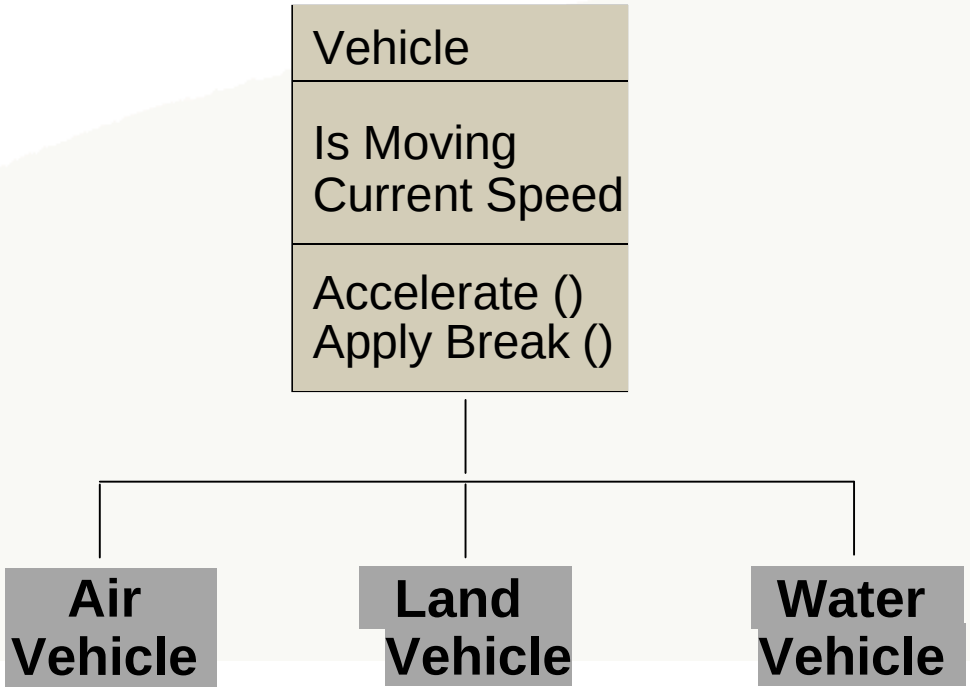
I **I**nheritance

E **E**ncapsulation

Workshops

Workshop 3

- *Create tree like structure of vehicle class and its sub classes (generalize to specialize objects).*
- *Vehicle class will be pre-created, with three sub classes – Land, Air and Water vehicle. Divide the class into 3 group, and each group will create structure for each sub class*



Advantage of OOPs

- **Modularity:**

  The source code for an object can be written and maintained independently of the source code for other objects. Once created, an object can be easily passed around inside the system.

- **Information-hiding:**

  By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.

- **Code re-use:**

  If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.

- **Pluggability and debugging ease:**

  If a particular object turns out to be problematic, you can simply remove it from your application and plug in a different object as its replacement.

Recap

Behavior

PIE

Overloading

Encapsulation

Instance of a class

State

Overriding

Software Object

Thank You For Your Time