JAVA @11

Introduction to OOPs

Objective

After completing this session you will be able to understand,

- Introduction to OOPs
- OOPs concept – Encapsulation
- OOPs concept – Inheritance
- OOPs concept - Polymorphism

What is a Procedural Language

Procedural programming language helps the programmers to develop applications as series of instructions to achieve the desired functionality.

Disadvantages:

- Not easy to maintain for larger applications.
- Not Flexible for changes.
- Complex to reuse a common programming logic.
- Programmers find it complex to understand the program and fix bugs.

What is OOPs?

The disadvantages of procedural language gave birth to Object Oriented Programming Language (OOPs)

Programmers develop application by breaking them into small objects.

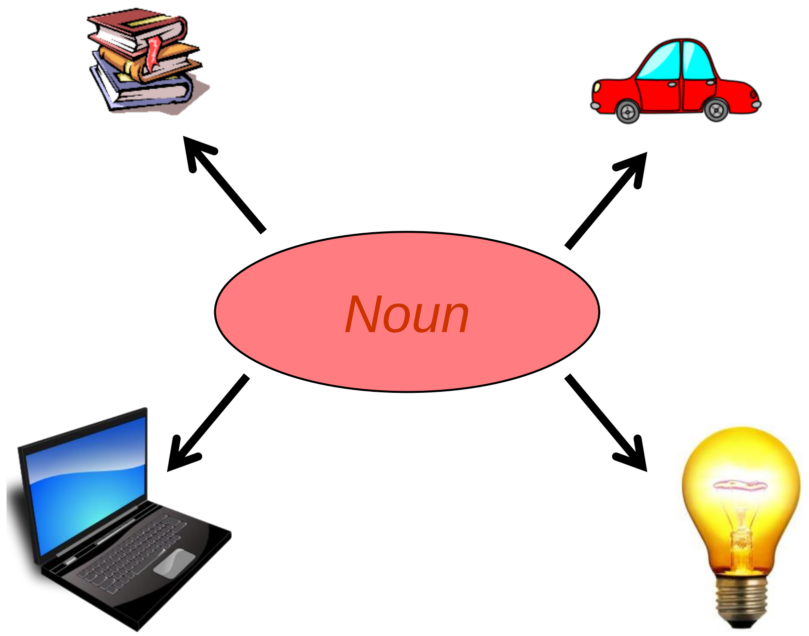
Advantages:

- The code is easy to maintain as the objects are self contained, cohesive units developed for specific functionalities.
- The objects can be reused to perform similar logics.
- The system is flexible to change.

What is Noun?

Definition: As a simple definition, a noun is any word that describes a person, place, or thing.

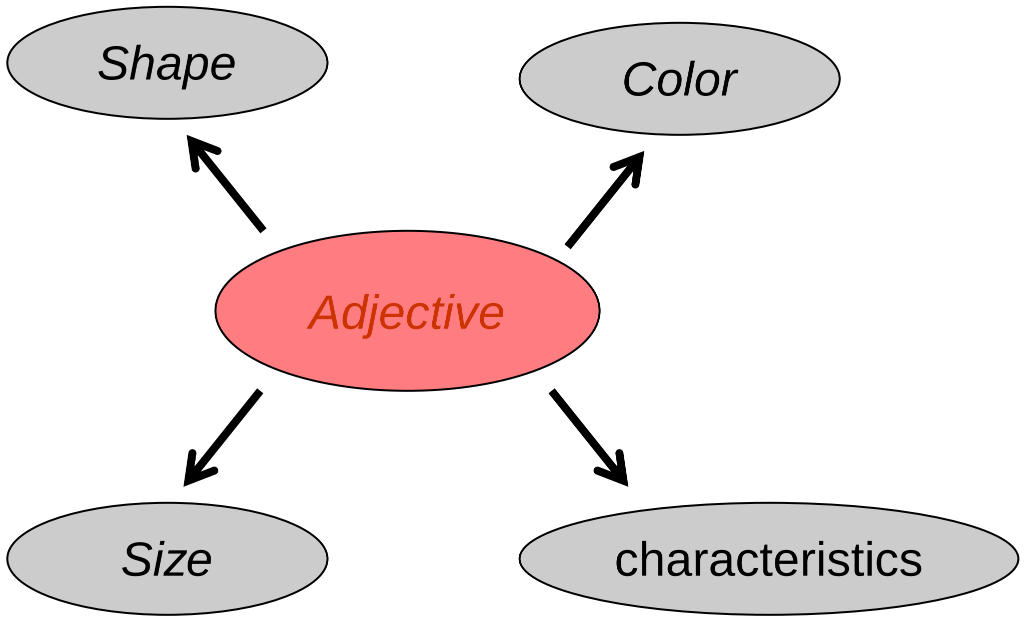
Example: The Man read a Book.



What is a Adjective?

Definition: Adjectives are describing words.

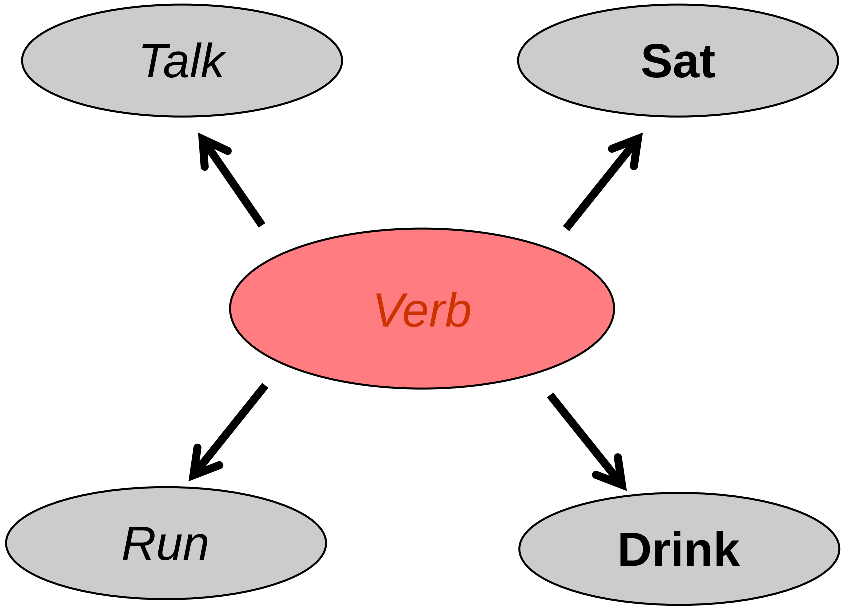
Example: The Man read a Java Book.



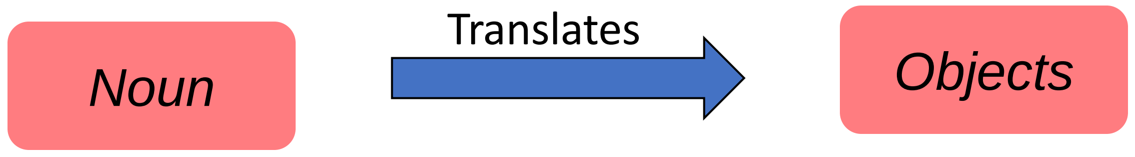
What is a verb

Definition: A verb is a word used primarily to indicate a type of action.

Example: The Man read a Java Book.



Analog to Software world

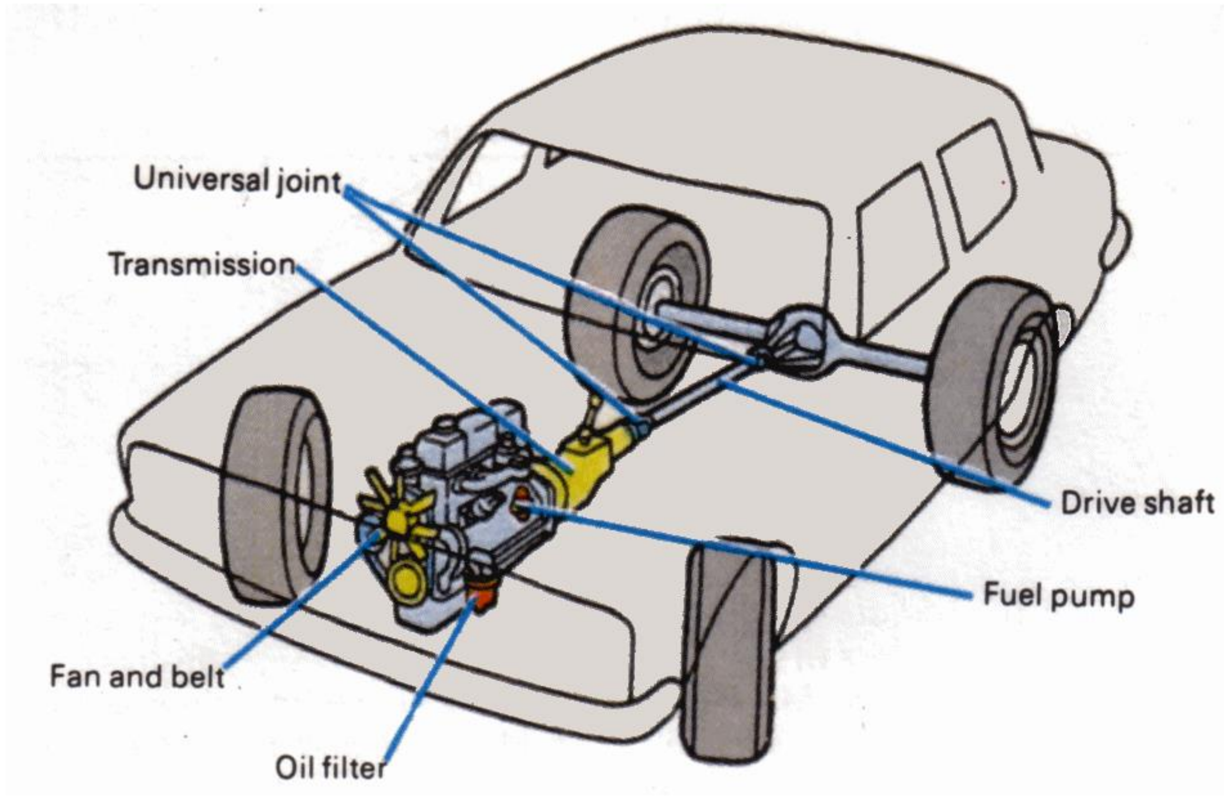


Nouns in real world translates into objects in software paradigm.

Real world Object System

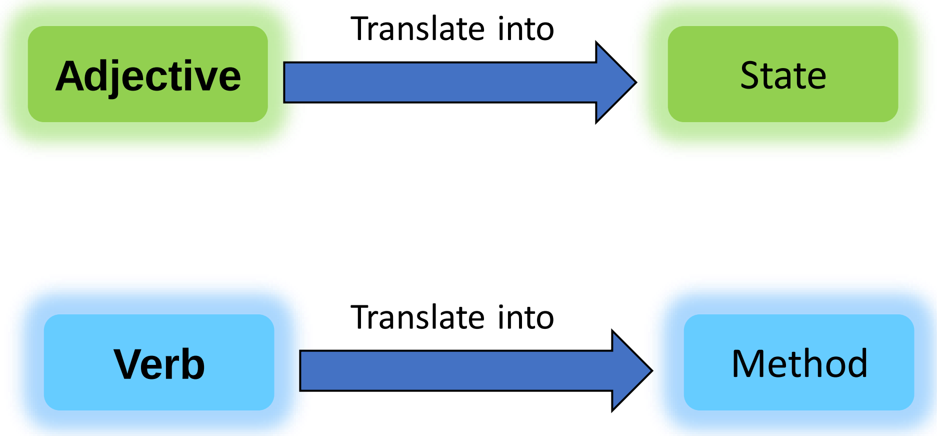
In real world all the entities are made of smaller objects, for example, a car is manufactured by assembling smaller objects like

- Engine
- Tyres
- Fan
- Steering Wheel



Characteristics of Object

Objects has two characteristics,



- **State** – State of the object.
- **Behavior** – Behavior of the object, it changes the state of the object.

Characteristics of Objects Explained with an example

Taking the car example,

- **State –** The speed of the car, fuel indicator which represents the state of the car.

- **Behavior -** Operations such as accelerate the speed, start/stop the car and apply brake are behaviors of the car.

The behavior “*accelerate the speed*” changes the car’s state “*speed*”.

Software System

Software system are collection software objects which interoperate to achieve a desired functionality.

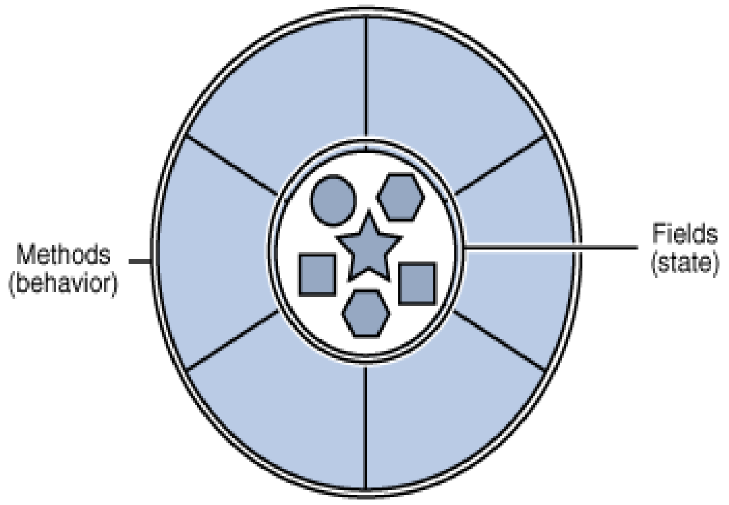
What is a Software Object?

Software objects are conceptually similar to real-world objects, they too consist of state and related behavior.

Software Object

Software objects have the same characteristics,

- **State** - Stored in fields.
- **Behavior** – Exposed through *methods*. Methods changes the state of the objects.



A software example

Lets take a net banking application where user can,

- Check account status.
- Check account balance.
- Transfer amount.

The system will be represented with a “*Account*” object.

Account Objects Characteristics:

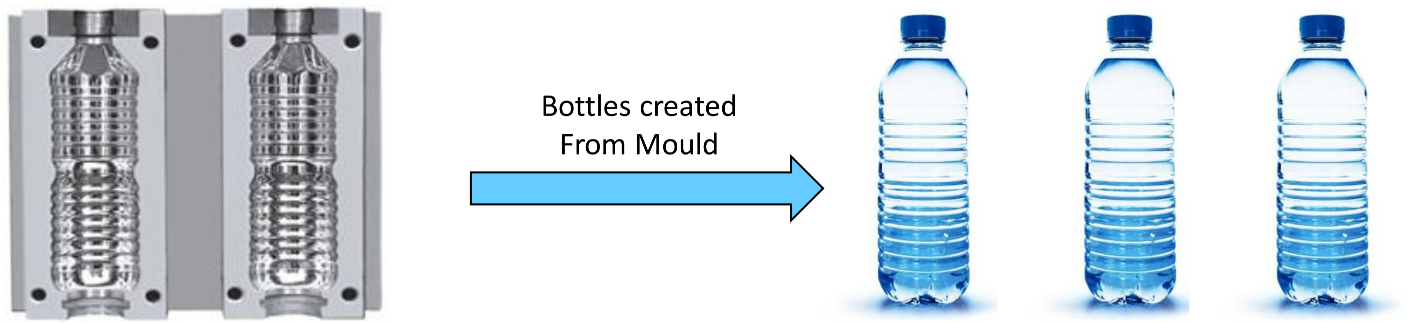
Methods (Behaviors)	Fields (States)
Check Account Status	**Account Status**
Transfer Amount **Check Account Balance**	**Account Balance**

What is Class?

What Is a Class?

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

Class defines the states and behavior of an object.

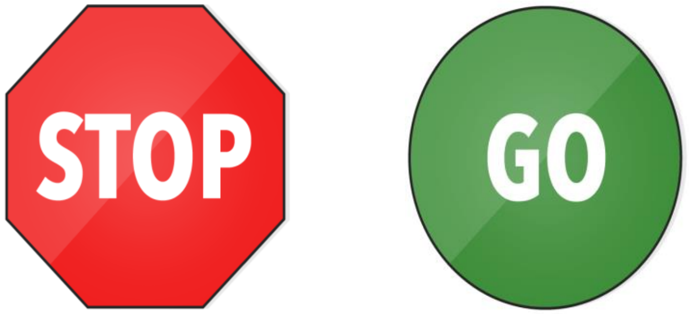


Bottle Mould

Bottles

The mold here refers to the "*Class*", the bottles created out of the mold are "*Objects*"

Time To Reflect



Trainees to reflect the following topics before proceeding.

- What is OOPs?
- What are Classes?
- What are Objects?
- What are the characteristics of objects? Explain with the Car example?

OOPS Concepts

The following are the OOPS concepts,

- Encapsulation
- Inheritance
- Polymorphism

We will learn about each in the next slides.!!

Encapsulation

What is Encapsulation?

Encapsulation is a process that allows selective hiding of properties and methods in a class.

Example #1:

The speed of the car is controlled by the method “accelerateSpeed”

The state “speed” is hidden from direct access and can be only controlled using the accelerateSpeed().

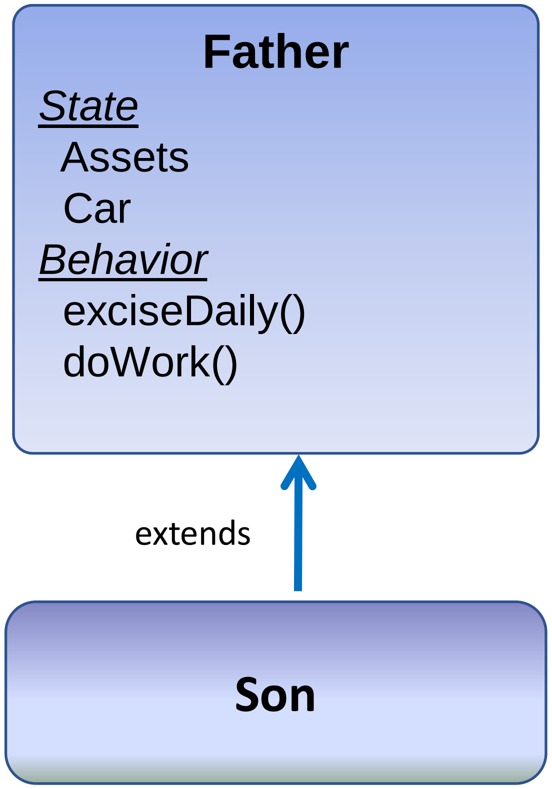
Example #2:

In the net banking application the “accountBalance” is hidden and can be accessed only through the “accountTransfer” method.

Inheritance

What is a Inheritance?

Object Oriented Programming allows classes to inherit the state and behavior from the classes.



Father is a parent class

Example:

The father is called the "*super*" class and the son is called the "*sub*" class

Son inherits the assets, car and the behavior exciseDaily(), doWork() from his father.

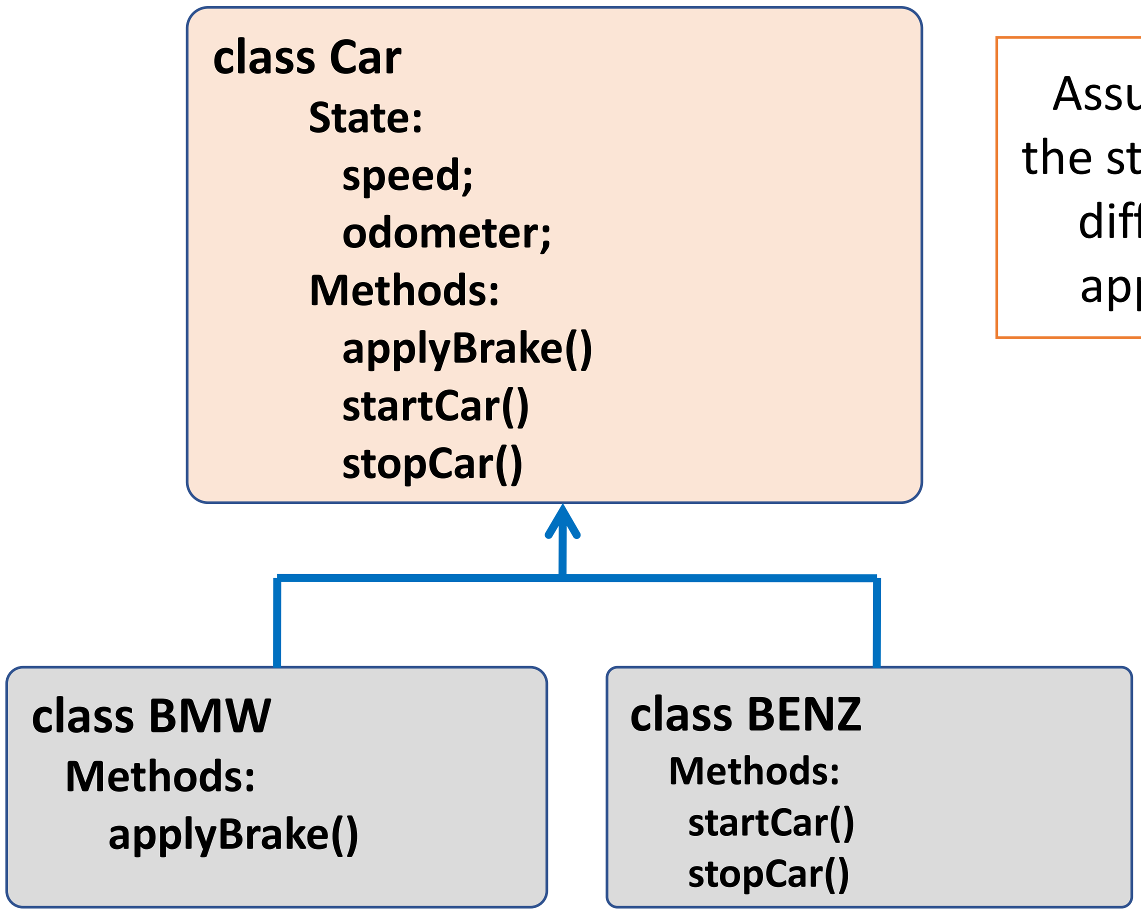
More of Inheritance

- The inheriting class contains all the attributes and behaviours of its parent class.
- Ideally the methods which needs to be reused are placed in the parent class.
- The inheriting class can override the definition of existing methods by providing its own implementation.

Overriding Example:

In the previous example if the son needs to work differently he can override the doWork() and provide a different implementation.

Overriding Example



Assume that the Benz car the start and stop method is different and BMW the apply Brake is different

applyBrake() is overridden

Start/stop method is overridden

Polymorphism

What is Polymorphism?

The dictionary definition of ***polymorphism*** refers to a principle in biology in which an organism or species can have many different forms or states.

Polymorphism is the ability to take more than one form. It is extensively used in implementing inheritance.

Polymorphism Types

Polymorphism Types:

Method Overloading - Two versions of the same method available in the same class.

Method Overriding - Methods of a subclass override the methods of a super class.

Dynamic Method Binding - At run time the interpreter will find out which objects methods needs to be executed.

Method Overloading

Method Overloading - Two versions of the same method available in the same class.

This is done by either changing the input parameter or return type.

Lets assume there is a object *"Dog"* with method *"makeSound".* Assume that the dog can make more than one sound based on whether it is normal or injured or other external factor. Now same method will have two implementations.

```
Implementation #1
makeSound(){
    //default parameter
    Bark woof woof
}
```

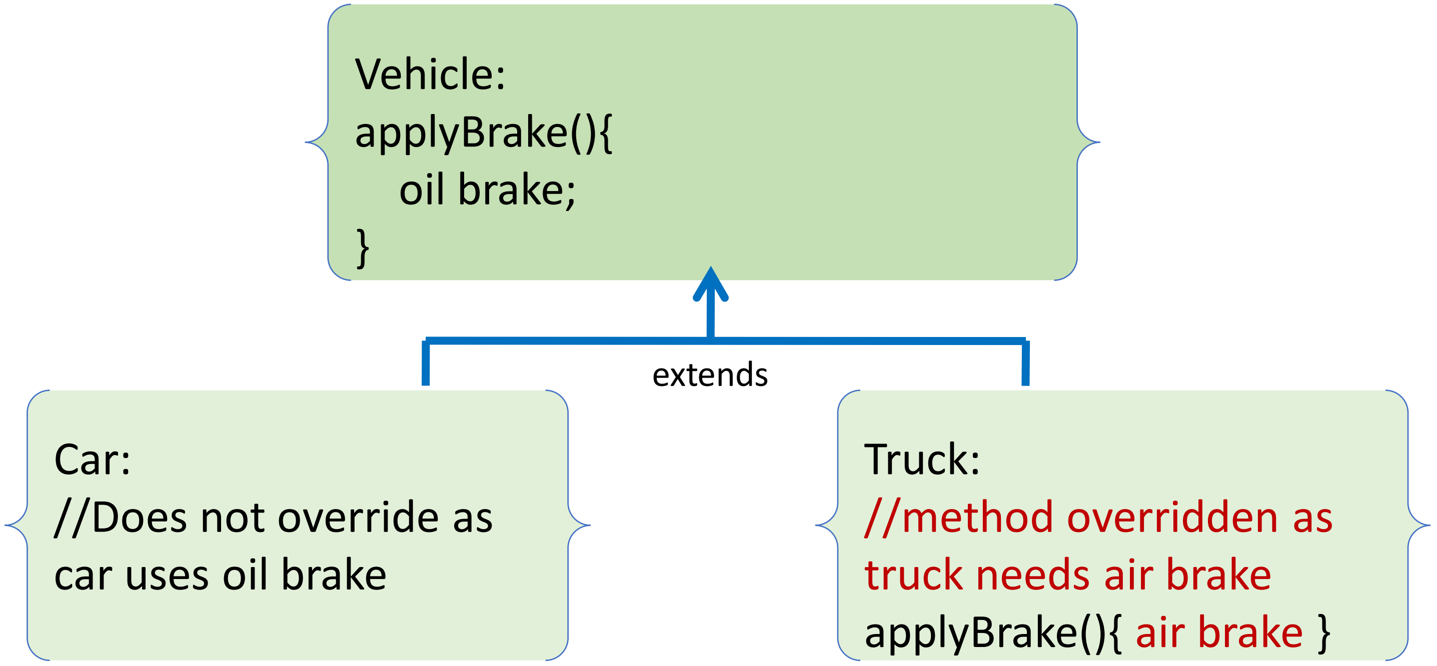
```
Implementation #2
makeSound(injured){
    //Added input parameter
    Make a whining sound
}
```

Method Overriding

Method Overriding - Methods of a subclass override the methods of a super class.

This is done by redefining the super class method in sub class.

Lets assume there are objects *"Truck"* & *"Car"* which extends *"Vehicle"*


Vehicle:
applyBrake(){
oil brake;
}
extends
Car:
//Does not override as car uses oil brake
Truck:
//method overridden as truck needs air brake
applyBrake(){ air brake }

Dynamic Method Binding

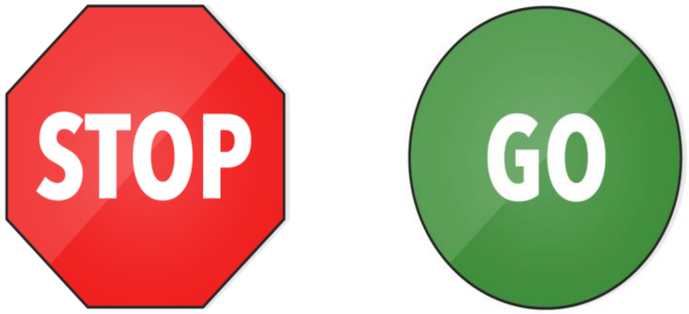
Dynamic Method or Late Binding - At run time the interpreter will find out which objects method needs to be executed.

The method being invoked is based on the type of the object and not based on the reference.

Taking the same truck example, based on what objects is created in run time either the oil/air brake is applied.

You will learn more about overriding and dynamic binding during the inheritance and polymorphism sessions.

Time To Reflect



Trainees to reflect the following topics before proceeding.

- What are the OOPs concepts?
- What is Encapsulation? Explain with a example
- What is Inheritance? Explain with a example
- What is overriding? Explain with a example
- What are the types of overriding?

Thank you

You have successfully completed

Introduction to OOPs