

00

Basic

Concepts



Problems with Procedural Languages

- ❖ Data does not have an owner.
- ❖ Difficult to maintain data integrity.
- ❖ Functions are building blocks.
- ❖ Many functions can modify a given block of data.
- ❖ Difficult to trace bug sources when data is corrupted.



What is Object?

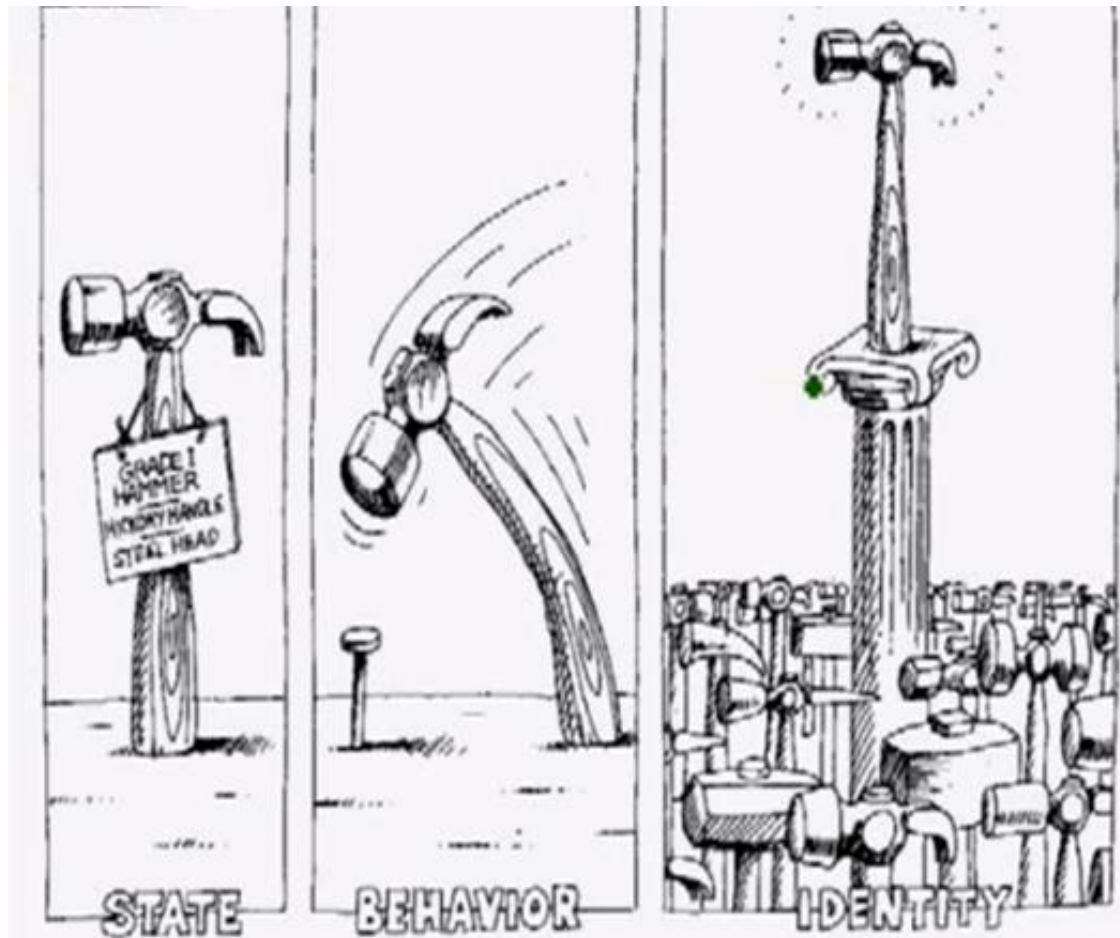
- ❖ An object has **state**, exhibits some well defined **behaviour**, and has a unique **identity**.

State

data members
fields
properties

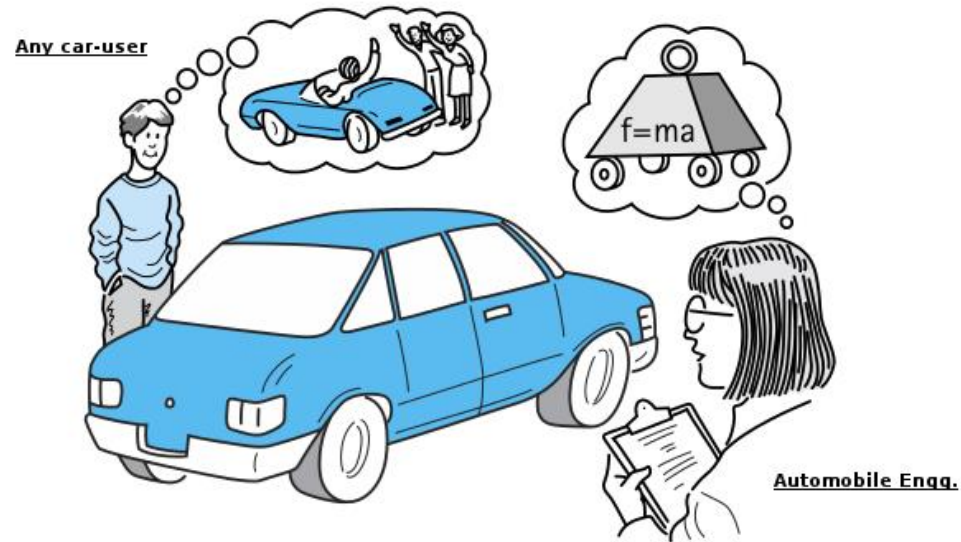
Behavior

member functions
methods



Abstraction - Modeling

- ❖ **Abstraction** focuses upon the **essential** characteristics of some object, relative to the perspective of the viewer.

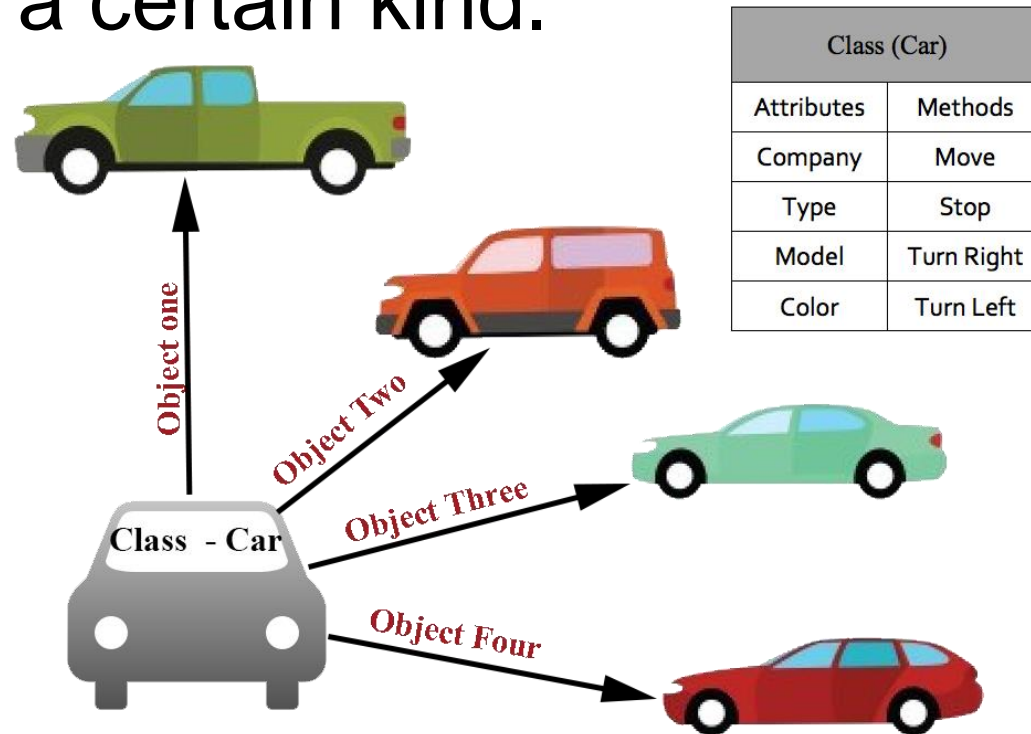


An abstraction includes the essential details relative to the perspective of the viewer

Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

What is **Class**?

- A **class** represents a set of objects that share common structure and a common behavior.
- A **class** is a **blueprint** or **prototype** that defines the variables and methods common to all objects of a certain kind.



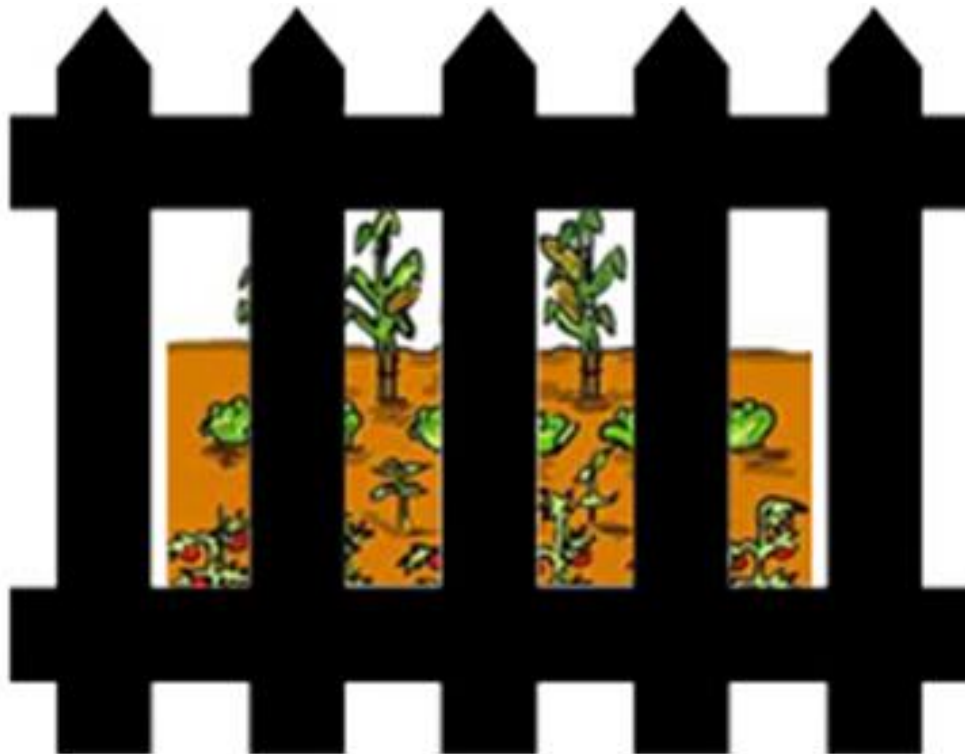
Class Access

PROBLEM: You have a garden and it is public. Anyone can take the properties of the garden when they want.



Class Access cont.

SOLUTION? Put a high fence around my garden, now it is safe! But waite, **I can no longer access my own garden.**



Class Access cont.

SOLUTION: Hire a private guard and give him rules on who is able to access the garden. Anyone wanting to use the garden must get permission from guard. garden is now safe and accessible.



Class Access cont.

Setters and Getters to Safeguard Data

Data

Set Property
Get Property

Outside
Requester

Private
Property



Setter & Getter



Requester



Initialization of Objects

What if garden had weeds from the beginning?



- ❖ **Constructors** ensure correct initialization of all data. They are automatically called at the time of object creation.
- ❖ **Destructors** on the other hand ensure the de allocation of resources before an object dies or goes out of scope.



Lifecycle of an Object

> Born Healthy

Using **constructors**

> Lives safely

Using **setters and getters**

> Dies cleanly

Using **destructors**

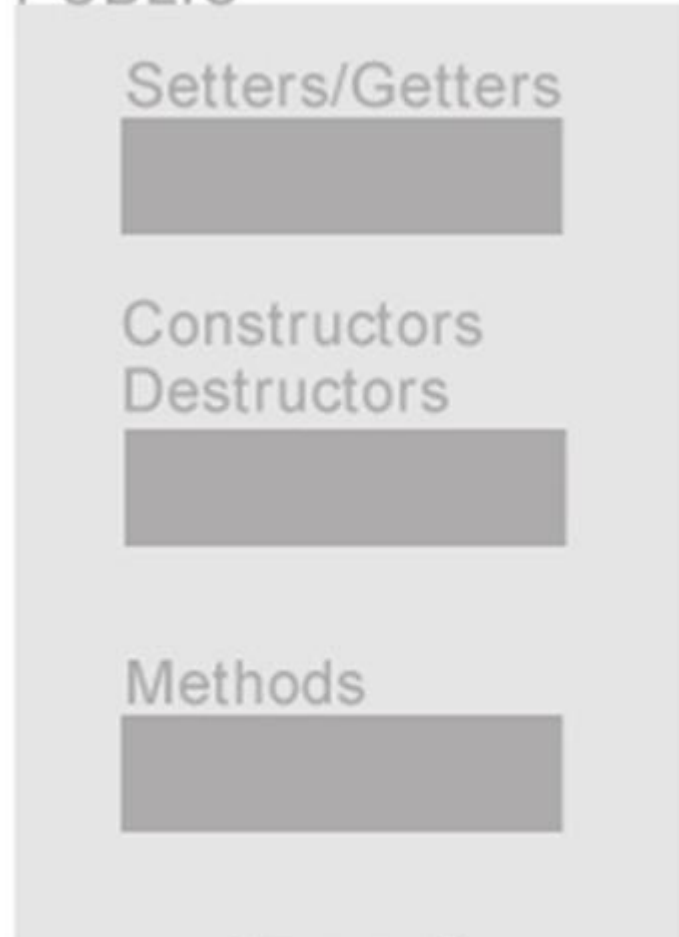


Anatomy of a Class

PRIVATE



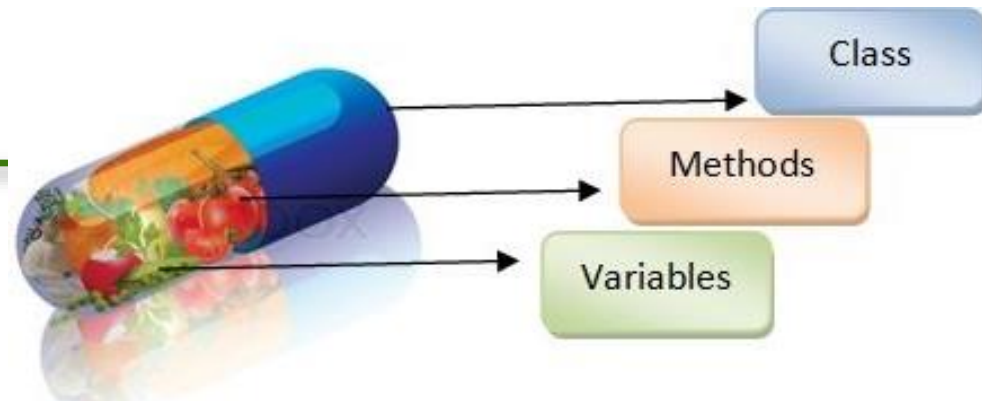
PUBLIC



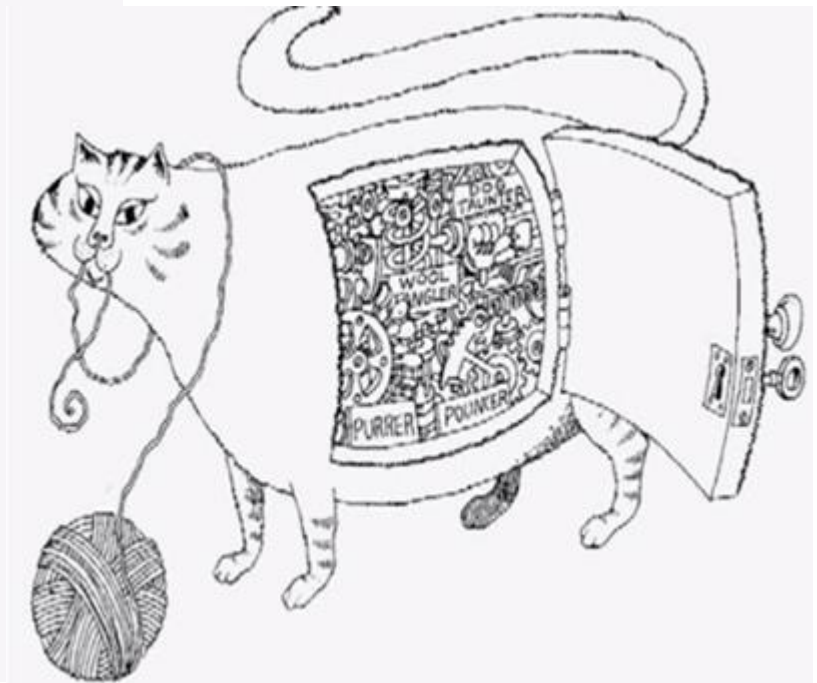
Public Interface Of Class



Encapsulation



- ❑ FIRST LAW OF OOP: Data must be hidden, i.e., **PRIVATE**
- ❑ Read access through read functions
- ❑ Write access through write functions
- ❑ For every piece of data, 4 possibilities
 - >> read and write allowed
 - >> read only
 - >> write only
 - >> no access



Encapsulation



- ❖ Encapsulation is used to hide unimportant implementation details from other objects.
- ❖ In real world
 - When you want to change gears on your car:
 - You don't need to know how the gear mechanism works.
 - You just need to know which lever to move.



Encapsulation cont.



- ❖ In software programs:
 - You don't need to know how a class is implemented.
 - You just need to know which methods to invoke.
 - Thus, the implementation details can change at any time without affecting other parts of the program.



Inheritance

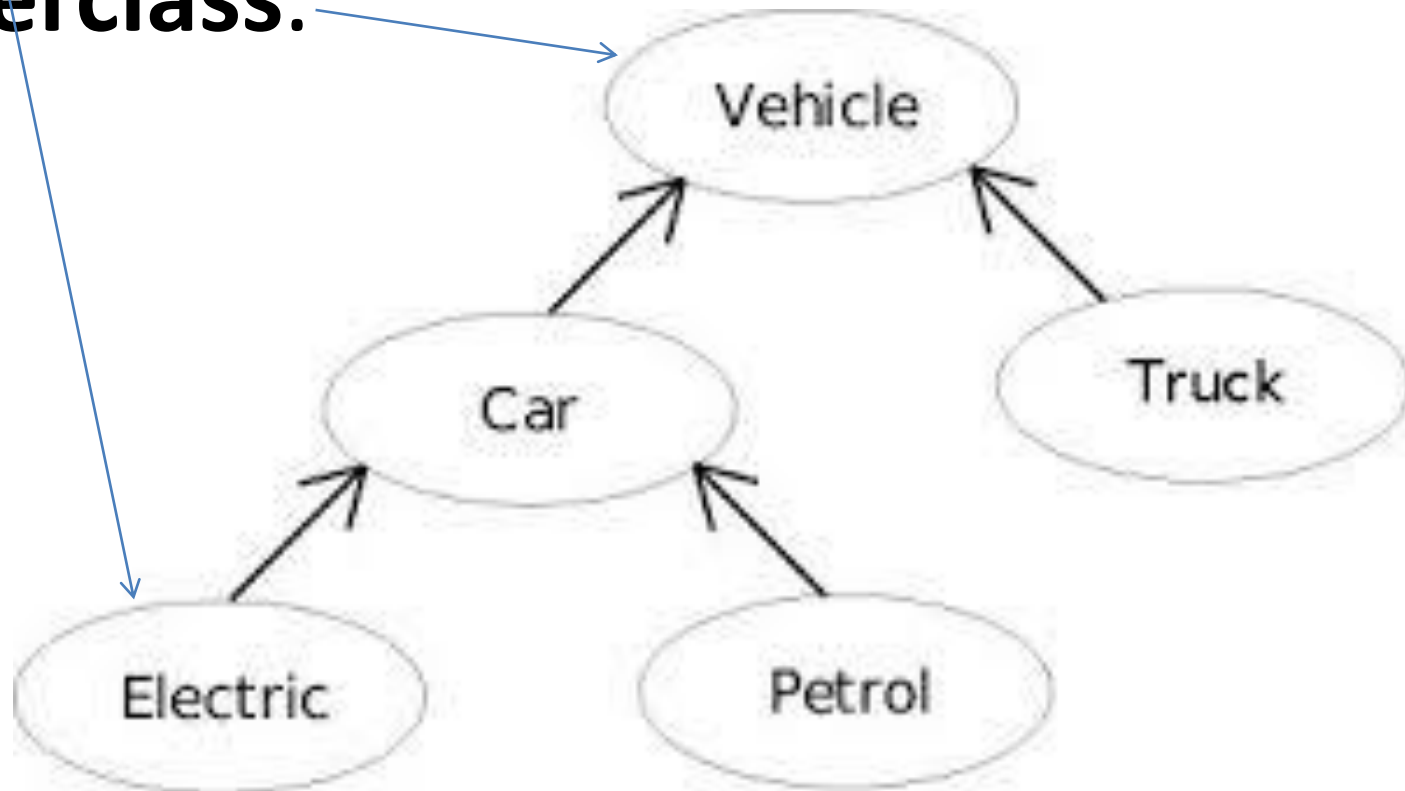


- ❖ **Extending** the functionality of a class or
- ❖ **Specializing** the functionality of the class.



Inheritance cont.

- ❖ **Subclasses:** a subclass may inherit the structure and behaviour of its **superclass.**



Multiple Inheritance

- ❖ One class have more than one superclass.

Horse



Eagle



Flying Horse

Copyright © Ren Leishan · <http://ToonClips.com/2008>



Multiple Inheritance cont.

❖ Ambiguity in multiple inheritance:



Horse

eat()

Eagle

eat()



Flying Horse

eat()?



Polymorphism

- ❖ **Polymorphism** refers to the ability of an object to provide different behaviours (use different implementations) depending on its own nature. Specifically, depending on its position in the class hierarchy.

`drawShape (class Shape)`

