Object Oriented Concepts

Mohammed Ghouse

OBJECTIVES

Introduction to OOP Concepts

LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

- Learn the basics of Object Oriented Programming
- Describe Classes and objects
- Describe Inheritance, Encapsulation, Aggregation, Polymorphism
- Distinguish between Procedural and Object Oriented Programming

CONCEPT

Objects and Classes

PROCEDURAL PROGRAMMING

- Programming paradigm derived from Structural programming
- Emphasizes modular programming
- Emphasis on Algorithms and Procedures
- Procedures contains steps of an Algorithm

Languages : C, Fortran, BASIC, COBOL

NEED FOR OBJECT ORIENTED APPROACH

- Challenges in building complex business applications
  - Integration of modules/Applications
  - Extensibility of existing code
  - High level of Flexibility and Illusion of Simplicity

- Features needed to meet these challenges
  - Modularity, Extensibility, Reusability, Interoperability, Security

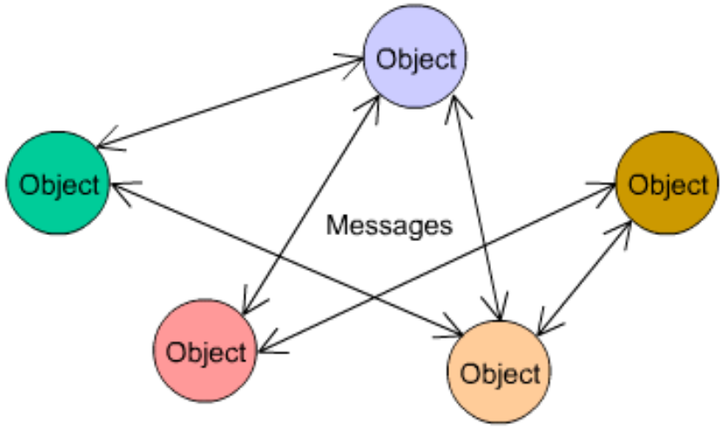
- Object oriented approach helps address these challenges easily

OBJECT ORIENTED PROGRAMMING

- Programming paradigm based on the concept of Modelling real world objects
- Groups data and related functionalities in an object

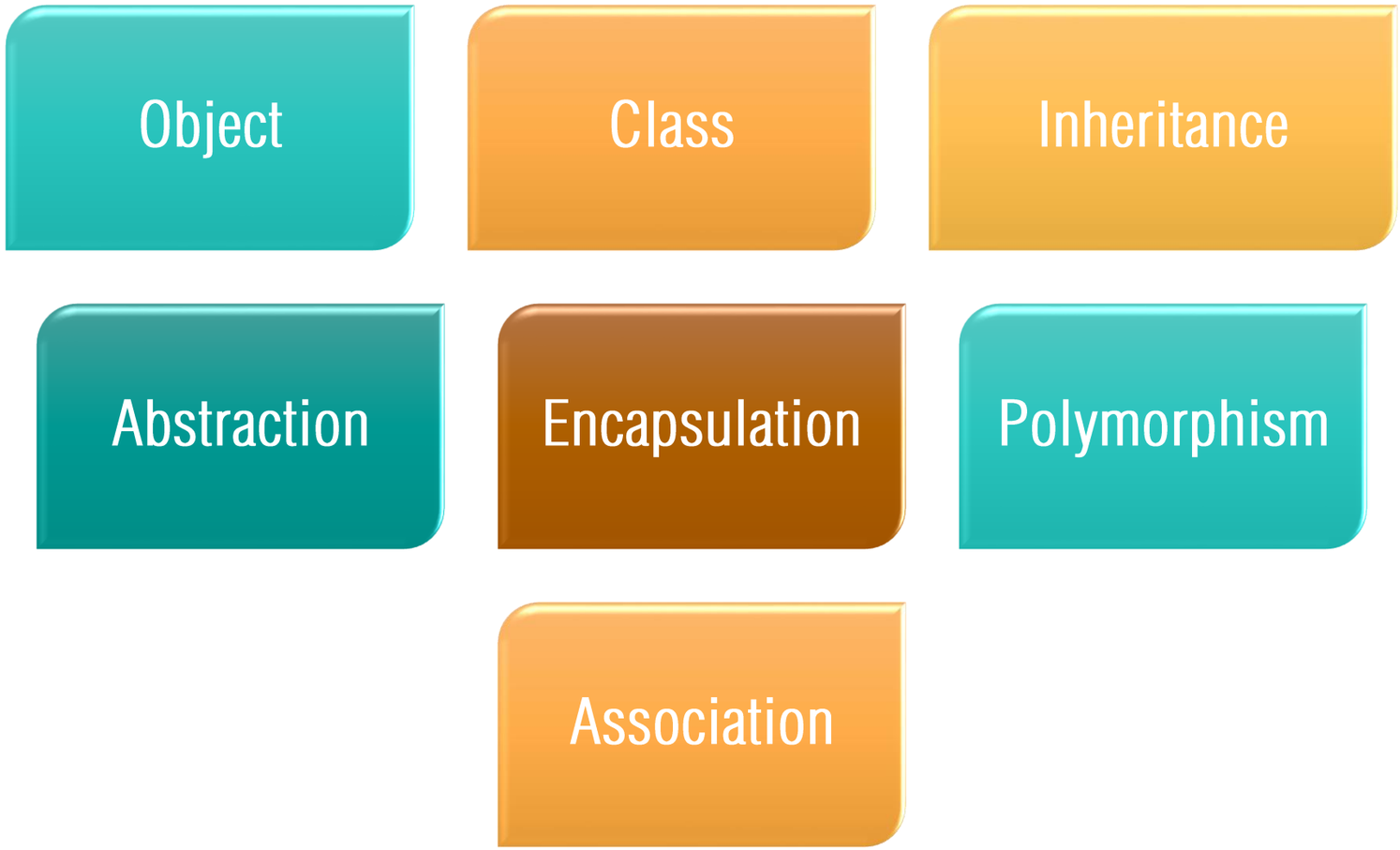
- Object oriented application
  - Is made of multiple type of objects
  - Objects communicate by passing messages and consuming services of other objects

- Emphasizes on
  - Reducing complexity
  - Increasing code reusability and maintainability
  - Providing extensibility



Interaction of objects via message passing

Fundamental OOP Concepts



OBJECT

- Conceptually similar to real world objects
- Have state and behaviour
- State represents the information which object stores about itself
  - a.k.a. attribute, property or fields
- Behaviour is the functionality which the object exposes to the external world
  - a.k.a. methods, functions
- Every object has unique identity

OBJECT

State (attributes)

Behaviour (functions)

NokiaLumia:MobilePhone
IMEI : 1234 Brand : Nokia Model : lumia screenSize : 4.5”
makeCall sendSMS playVideo capturePhoto

John:Employee
empNo : 1 empName : John empAddress : Jayanagar empSalary : 20000.00
setName setAddress setSalary

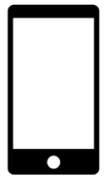
Cathy:Employee
empNo : 2 empName : Cathy empAddress : HSR empSalary : 25000.00
setName setAddress setSalary

CLASS

- Blueprint / Template used to define and create objects
- Defines State and Behaviour of the object
  - attributes and methods that belong to a category of objects
- Class is said to be **instantiated** when a object of that class is created
  - Object is an instance of a class
- Each Object has separate memory to store its attribute values

Synonymous to Factory

CLASS AND OBJECTS



MobilePhone
IMEI Brand Model screenSize
makeCall sendSMS playVideo capturePhoto

Noka Lumia

Asus Zenfone

Samsung Galaxy

Employee
empNo empName empAddress empSalary
setName setAddress setSalary

John

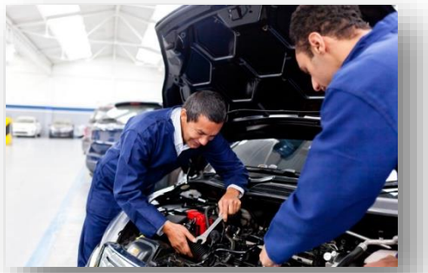
Cathy

Charles

Each Object of a class have the same attributes but different values and separate memory location

ABSTRACTION

- Identifying essential details and suppressing non-essential details from the perspective of the user of the system
- Emphasizes on details that are significant to the user
- Provides defined conceptual boundaries

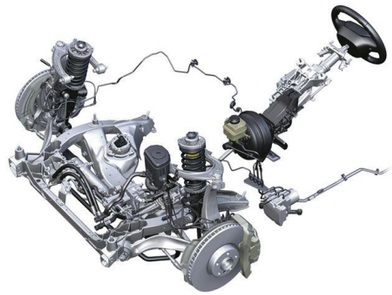


Car
steer( ) start( ) stop( ) accelerate( ) slowdown( )

Car
checkFuel( ) checkBrakes( ) rechargeBattery( ) doSomeRepair( ) tuneEngine( )

ENCAPSULATION

- Mechanism to restrict access to some components of an object
- Hiding internal details and providing simple and essential interface
- Ensures that object can be used without having to know how it works
- Class is like a Container/Capsule which encapsulates methods and data to provide intended functionality



Car internal parts and functions

Car Encapsulated

ENCAPSULATION

Account Object
accountNumber
101
balance
1000

- Scenario
  - In the Account object shown, balance variable is accessible without any restriction
  - Lets withdraw Rs 1500 from the Account Object.
  - The balance now becomes -500, which should not be allowed

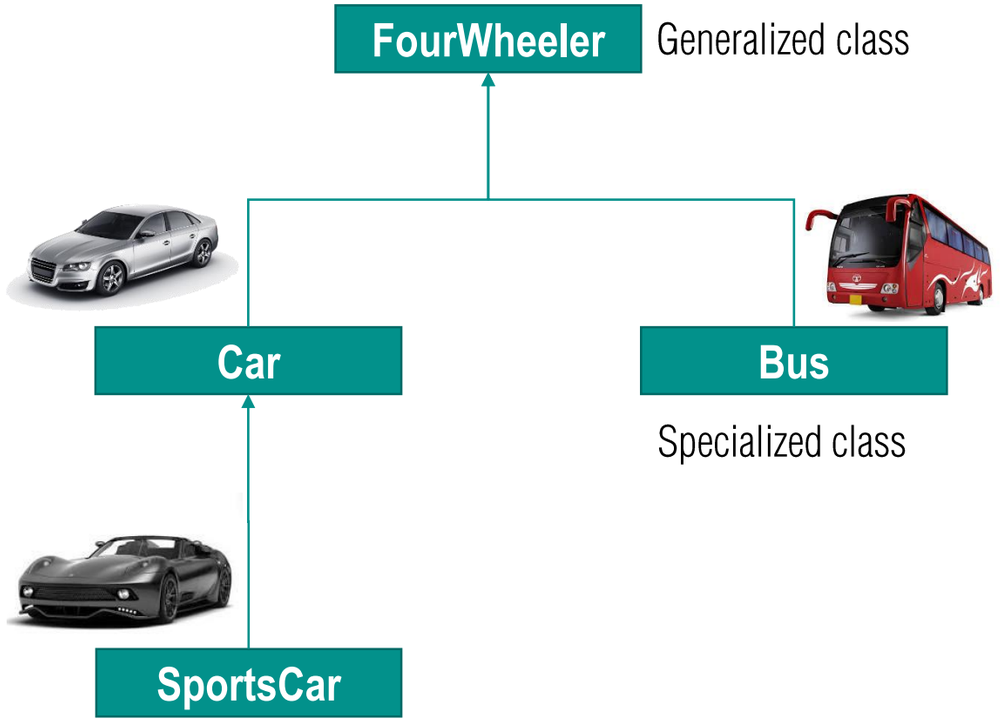
- Encapsulate the Account Class
  - Restrict access to balance variable
  - Provide a withdraw method which is accessible by all other classes
  - withdraw method can have logic to reduce the balance only if sufficient balance is available

Encapsulation
- Restrict access to the attributes
- Provide methods for getting and setting the values of the attributes (Getter and Setters)
- Restrict access to functions which are internal to the object

  Ex. Account number generation in Account Class

INHERITANCE

- Represents the hierarchy between two classes having IS-A relationship
- Specialized class(Sub Class) **IS A** subtype of Generalized class (Super Class)
- Sub classes are derived from Super class
- Any object of Sub class is also considered as an object of Super class



Ex:

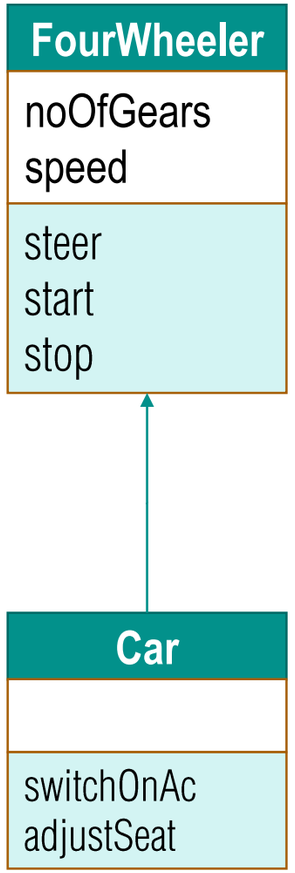
SportsCar IS A type of Car

Car IS A type of FourWheeler

INHERITANCE

- Super class defines the common attribute and behaviour for a hierarchy of classes
- Sub classes automatically inherit attributes and behaviour of Super classes

- Advantages
  - Emphasizes on code-reusability
  - Allows independent extensions of the original software



AGGREGATION

- One object is composed of other objects
- HAS-A relationship
- Whole-part relationship
  - "Whole" is called owning/composite object
  - "part" is called contained/constituent object
  - ex :
    - Car has Engine, Tyres, etc.
    - Account has Transactions
- Advantages
  - Emphasizes on code-reusability
  - Combines simple objects into more complex ones

Composite Object

Constituent Object

POLYMORPHISM

- Ability of an object/operation to behave differently in different situations
- Single object to be seen as having many types

```
Example
        Ferrari is-a sports car
        Ferrari is a car
        Ferrari is a Four wheeler
        Ferrari is a Vehicle
```

PROCEDURAL PROGRAMMING VS OBJECT ORIENTED PROGRAMMING

Procedural Programming	Object Oriented Programming
Emphasis on Algorithms and Procedures. Focus is on steps required and their order to produce desired outcome.	Models real world in terms of Objects. Decomposing problem in to smaller discrete pieces called objects
Data and Procedures are separate in a given module	Related Data and Functions are bundled in to an object
Reuse depends on programmer	Motivates Reuse and helps decrease redundancy
Top down process followed for program design	Bottom-up process followed for program design

When is a Class is said to be instantiated ?

- When attributes and behaviour are defined in a class
- When access to all the attributes of a class is restricted
- When the object of the class is created
- None

Procedural programming paradigm uses

- Top down approach
- Bottom up approach

Each Object of a given class have the same attributes but different values and separate memory location

- TRUE
- FALSE

Hiding internal implementations and providing simple interfaces to achieve the intended functionality is called as

- ❑ Inheritance
- ❑ Aggregation
- ❑ Polymorphism
- ❑ Encapsulation

Which of the following concept is used to provide extension to existing application

- ❑ Inheritance
- ❑ Aggregation
- ❑ Polymorphism
- ❑ Abstraction

References

- Refer following demo videos on EduNxt
  - M2L1L1_Procedural_Vs_Object_oriented_programming
  - M2L1L2_Class_vs_Object
  - M2L2L1_Major_and_minor_elements_of_OOP
  - M2L2L2_Abstraction
  - M2L2L3_Encapsulation
  - M2L2L4_Aggregation
  - M2L2L5_Inheritance
  - M2L2L6_Polymorphism

SUMMARY

Introduction to OOP

SUMMARY

In this lesson, you've learned to:

- Describe objects and classes
- Explain fundamental concepts of OOP
- Distinguish between Procedural and Object Oriented Programming