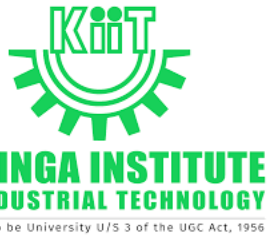
OOPs Concepts

Dr. Rajesh Kumar Panda

KIIT Deemed to be University



- Difference between POP and OOP
- Object
- Class
- Encapsulation
- Data Hiding
- Data Abstraction
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

- Genericity
- Delegation/Containership
- Persistence
- Association
- Aggregation
- Composition
- Generalization
- Specialization

Difference Between POP and OOP

Procedural Oriented Programming	Object Oriented Programming
• In procedural programming, program is divided into small parts called ***functions***.	• In object oriented programming, program is divided into small parts called ***objects***.
• Procedural programming follows ***top down approach***.	• Object oriented programming follows ***bottom up approach***.
• There is no access specifier in procedural programming.	• Object oriented programming have access specifiers like private, public, protected etc.
• Adding new data and function is not easy.	• Adding new data and function is easy.
• Procedural programming does not have any proper way for hiding data so it is ***less secure***.	• Object oriented programming provides data hiding so it is ***more secure***.
• In procedural programming, overloading is not possible.	• Overloading is possible in object oriented programming.
• In procedural programming, function is more important than data.	• In object oriented programming, data is more important than function.
• Procedural programming is based on ***unreal world***.	• Object oriented programming is based on ***real world***.
Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

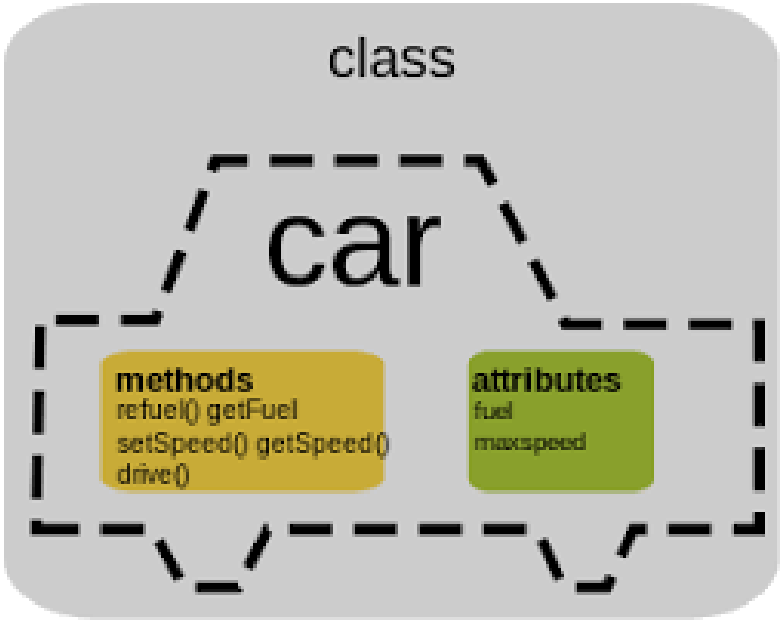
Object

- In general object is a real world entity which has properties and behaviours.
- Object is a run time entity of an object oriented system.
- Object is a variable of class type.
- Object is partitioned area of memory containing data and function together.
- Object is an Instance of a Class.

Class

- Class is a collection of similar types of objects.
- Class is an user-defined data type, which holds its own data members and member functions.
- Class is like a blueprint for an object.
- Common properties and behaviours of similar types of objects are define inside a Class.

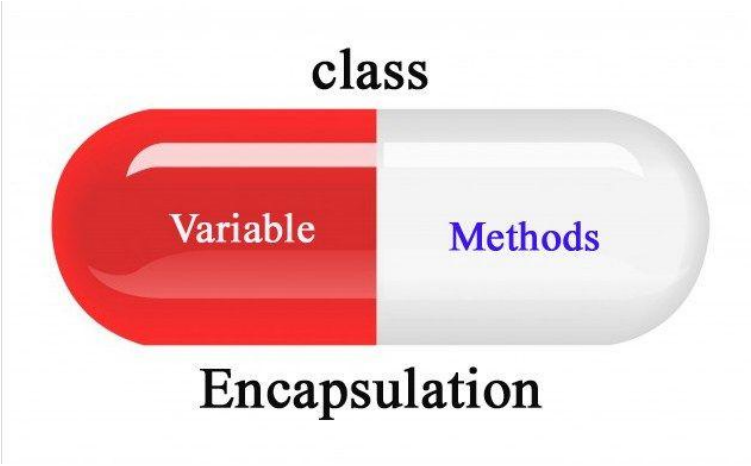
Class & Object



Objects

Encapsulation

- Wrapping up of data and function together.
- Mechanism of combining data and function into a single unit so that data can be accessed by the function inside the unit.
- Encapsulation also lead to data hiding.



Data Hiding

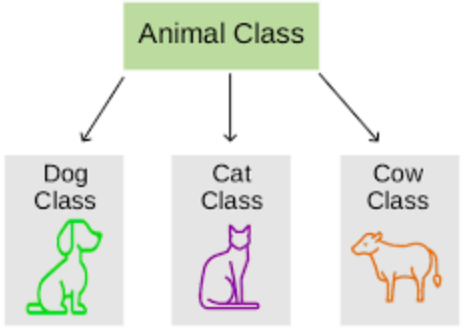
- Due to encapsulation data can only be accessed by the function inside the class. This insulation of data from being accessed by outside function is known as Data Hiding.
- Anything declared as private can be accessed through the member functions.

Data Abstraction

- Abstraction refers to implementing something without including the details about it.
- Class use this concept by abstracting some or all properties and behaviours of an Object so the Class is also known as Abstract Data Type(ADT).
- It hides unnecessary properties and behaviours of Objects.

Inheritance

- It is the ability to derive new classes from existing classes
- New class is known as child/derived class and old class is known as parent/base class.
- Derived class can inherit properties and behaviours of base class.
- It is used to represent IS_A or KIND_OF relationship between classes.



- Main benefit of Inheritance is reusability.

Polymorphism

- Polymorphism is the ability to take more than one form.
- We can define more than one function with same name called function overloading. One operator can be redefined to exhibit different task on different data types called operator overloading. These are different forms of Polymorphism.
- Real life example of polymorphism, a person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee.

Dynamic Binding

- Binding refers to linking of function call with function definition.
- Linking of function call with function definition at compile time is known as Static Binding or early binding.
- Linking of function call with function definition at runtime is known as Dynamic Binding or late binding.
- Dynamic binding uses Objects to resolve binding at Runtime.

Message Passing

- Two Objects can communicate with each other through functions that is called message passing.

Objectname . FunctionName (Argument List)

Genericity

- Genericity is the generic programming approach in which we can define classes and methods those can take data types as arguments and can work for variety of data types.
- It supports reusability of code.
- Example: Templates in C++ , Generic Classes and Methods in JAVA

Delegation/Containership

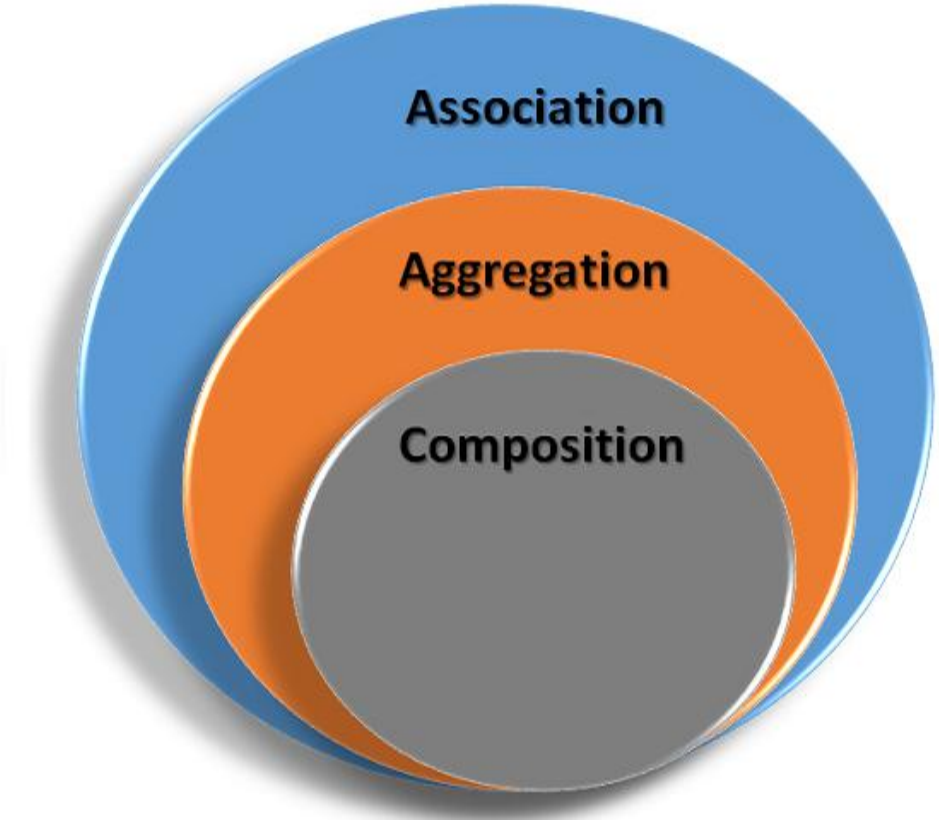
- One class can have objects of other classes as its member.
- This is another way of accessing properties and behaviour of other objects.
- It is used to establish HAS_A relationship between classes.
- Delegation is the passing of some tasks to other objects.
- Example: Car has Wheels, Engine.

Persistence

- Persistence is the ability to store the current state of an object in secondary memory or files.
- We can store the objects permanently in files or database.

Association

Association is a relationship between two separate classes and the relationship can be one to one, One to many, many to one and many to many. It joins two entirely separate entities.



ASSOCIATION

- Association is a relationship between two objects.
- Objects might not be completely dependent on each other.
- One-to-many, many-to-one, many-to-many all these words define an association between objects
- *Example:* A Student and a Faculty are having an association.

Aggregation

- Aggregation is a special form of association which is a unidirectional one way relationship between classes (or entities).

  Example: Wallet and Money classes.

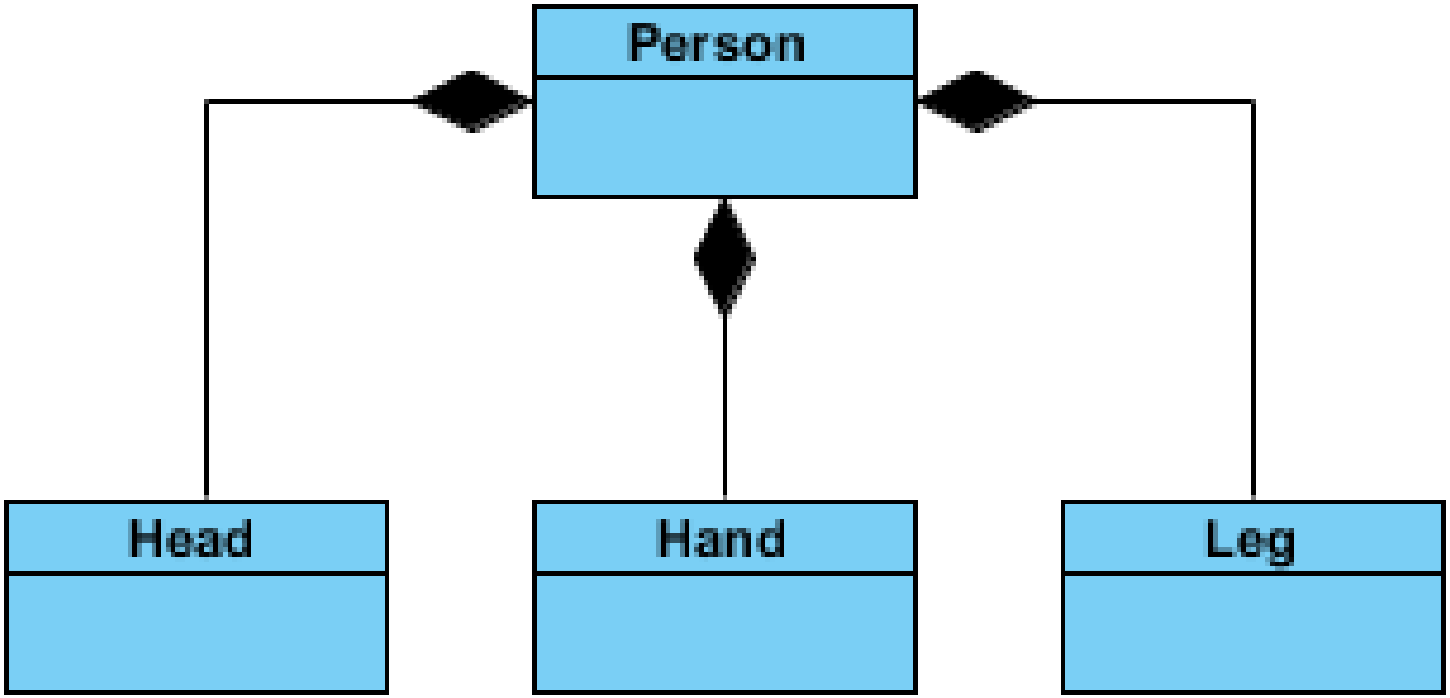
  Wallet has Money but money doesn't need to have Wallet necessarily so its a one directional relationship. In this relationship both the entries can survive if other one ends. In our example if Wallet class is not present, it does not mean that the Money class cannot exist.

Composition

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

  Example: Human and Heart.

  A human needs heart to live and a heart needs a Human body to survive. In other words when the classes (entities) are dependent on each other and their life span are same (if one dies then another one too) then its a composition. Heart class has no sense if Human class is not present.


Person
Head
Hand
Leg

Generalization vs Specialization

- **Generalization** is a mechanism for combining similar classes of objects into a single, more general class. Generalization identifies commonalities among a set of entities. The commonality may be of attributes, behavior, or both. In other words, a superclass has the most general attributes, operations, and relationships that may be shared with subclasses. A subclass may have more specialized attributes and operations.
- **Specialization** is the reverse process of Generalization means creating new sub-classes from an existing class.

Example

- a Bank Account is of two types - Savings Account and Credit Card Account. Savings Account and Credit Card Account inherit the common/ generalized properties like Account Number, Account Balance, etc. from a Bank Account and also have their specialized properties like unsettled payment etc.