**Chapter 1: Introduction**

**1.1 Introduction to OOAD**

* The object oriented paradigm took its shape from the initial concept of a new programming approach, while the interest in design and analysis methods came much later.
* The first object-oriented language was simula (simulation of the real system) that was developed in 1960 by researcher at the Norwegian computing center.
* It’s a structured method for analyzing, designing a system by applying the object-orientated concepts, and develop a set of graphical system models during the development life cycle of the software.
* In 1970, Alan kay and his research group at Xerox PARK created a personal computer named Dynabook and the first object-oriented programming language (OOPL).
* In the 1980s, Grady Booch published a paper titled Object Oriented Design that mainly presented a design for the programming language, Ada. In the ensuing editions, he extended his ideas to a complete object–oriented design method.
* In the 1990s, Coad incorporated behavioral ideas to object-oriented methods.

**1.1.1 Object Oriented Analysis (OOA)**

* Object–Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system’s object model, which comprises of interacting objects.
* The main difference between object-oriented analysis and other forms of analysis is that in object-oriented approach, requirements are organized around objects, which integrate both data and functions. They are modelled after real-world objects that the system interacts with. In traditional analysis methodologies, the two aspects - functions and data - are considered separately.

Grady Booch has defined OOA as, ***“Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain”*.**

The primary tasks in object-oriented analysis (OOA) are −

* Identifying objects
* Organizing the objects by creating object model diagram
* Defining the internals of the objects, or object attributes
* Defining the behavior of the objects, i.e., object actions
* Describing how the objects interact

The common models used in OOA are use cases and object models.

**1.1.2 Object-Oriented Design (OOD)**

* Object–Oriented Design (OOD) involves implementation of the conceptual model produced during object-oriented analysis.
* In OOD, concepts in the analysis model, which are technology−independent, are mapped onto implementing classes, constraints are identified and interfaces are designed, resulting in a model for the solution domain, i.e., a detailed description of how the system is to be built on concrete technologies.

**The implementation details generally include −**

* Restructuring the class data (if necessary),
* Implementation of methods, i.e., internal data structures and algorithms,
* Implementation of control, and
* Implementation of associations.

Grady Booch has defined object-oriented design as ***“a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design”*.**

**1.1.3 Object Oriented Programming (OOP)**

* Object-oriented programming (OOP) is a programming paradigm based upon objects (having both data and methods) that aims to incorporate the advantages of modularity and reusability.
* Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

**The important features of object–oriented programming are −**

* Bottom–up approach in program design
* Programs organized around objects, grouped in classes
* Focus on data with methods to operate upon object’s data
* Interaction between objects through functions
* Reusability of design through creation of new classes by adding features to existing classes

Some examples of object-oriented programming languages are C++, Java, Smalltalk, Delphi, C#, Perl, Python, Ruby, and PHP.

Grady Booch has defined object–oriented programming as ***“a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships”*.**

**1.2 Basic Concept of OOAD**

1. **Object**

* An object is a real-world element in an object–oriented environment that may have a physical or a conceptual existence.
* Each object has –
  + Identity that distinguishes it from other objects in the system.
  + State that determines the characteristic properties of an object as well as the values of the properties that the object holds.
  + Behavior that represents externally visible activities performed by an object in terms of changes in its state.
* Objects can be modelled according to the needs of the application. An object may have a physical existence, like a customer, a car, etc.; or an intangible conceptual existence, like a project, a process, etc.

1. **Class**

* A class represents a collection of objects having some characteristic properties that exhibit common behavior.
* It gives the blueprint or description of the objects that can be created from it.
* Creation of an object as a member of a class is called instantiation. Thus, object is an instance of a class.

**The constituents of a class are −**

* + A set of attributes for the objects that are to be instantiated from the class. Generally, different objects of a class have some difference in the values of the attributes. Attributes are often referred as class data.
  + A set of operations that portray the behavior of the objects of the class. Operations are also referred as functions or methods.

1. **Encapsulation**

* Encapsulation is the process of binding both attributes and methods together within a class.
* Through Encapsulation, the internal details of a class can be hidden from outside. It permits the elements of the class to be accessed from outside only through the interface provided by the class.

1. **Abstraction**

* It is a way or mechanism to represent complex reality using simplified model.
* It could be also defined as a way to capture only those details about object that are relevant to the current perspective.
* Data abstractions required that the behavior of the data objects were completely characterized by the set of operations.
* Classical example is defined of stack cluster using only Push and POP operations.

1. **Polymorphism**

* Polymorphism is ability to apply different meaning (semantic, implementation) to the same symbol (message, operation) in different contexts.
* When context is defined at compile time, it is called static or compile –time polymorphism. When context is defined during program execution, it is dynamic or run time polymorphism.
  + For example: sum: = x+y
* In this example “+” is polymorphism operation which could be used with different types of operands –integer, real, string, complex, vector, etc.

1. **Hierarchy**

* **The subsystem layer:** Representation of each of the subsystem

That Enables the software achieve its customer defined requirements.

* **The Class Layer:** The class hierarchies (generalization)

& representation of objects.

* **The Message Layer:** The design details of communication of each object with its collaborates (external & internal interfaces)
* **The responsibilities Layer:** Data structure and algorithm design for all attributes and operations.

1. **Modularity:**

* Modularity refers to the concept of making multiple modules first and then linking and combining them to form a complete system.
* Modularity enables re-usability and minimizes duplications.
* “Modularity is the degree to which a system’s components are made up of relatively independent components or parts which can be combined.”

1. **Object Interaction**

* Interaction object illustrate how object interact via message.
* They are used for dynamic object modeling.
* There are two common types:
  + Sequence &
  + Communication interaction diagram