Module 1: INTRODUCTION

WHAT IS OBJECT ORIENTATION?

Definition: OO means that we organize software as a collection of discrete objects (that incorporate both data structure and behavior).

There are four **aspects(characteristics)** required by an OO approach

- Identity.
- Classification.
- Inheritance.
- Polymorphism.

Identity:

Identity means that data is quantized into discrete, distinguishable entities called objects.

E.g. for objects: personal computer, bicycle, queen in chess etc.

- Objects can be concrete (such as a file in a file system) or conceptual (such as scheduling policy in a multiprocessing OS).
- Each object has its own inherent identity. (i.e two objects are distinct even if all their attribute values are identical).
- In programming languages, an object is referenced by a unique handle.

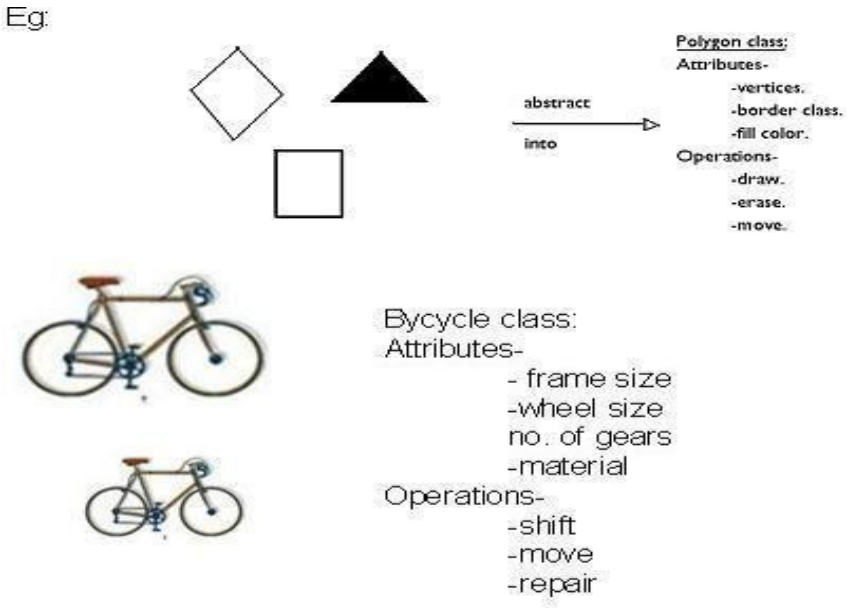
Classification:

Classification means that objects with the same data structure (attribute) and behavior (operations) are grouped into a class.

E.g. paragraph, monitor, chess piece.

- Each object is said to be an instance of its class.

Fig below shows objects and classes: Each class describes a possibly infinite set of individual objects.



Inheritance:

It is the sharing of attributes and operations (features) among classes based on a hierarchical relationship. A super class has general information that sub classes refine and elaborate.

E.g. Scrolling window and fixed window are sub classes of window.

Polymorphism:

Polymorphism means that the same operation may behave differently for different classes.

For E.g. move operation behaves differently for a pawn than for the queen in a chess game.

Note: An *operation* is a procedure/transformation that an object performs or is subjected to. An implementation of an operation by a specific class is called a *method*.

WHAT IS OO DEVELOPMENT?

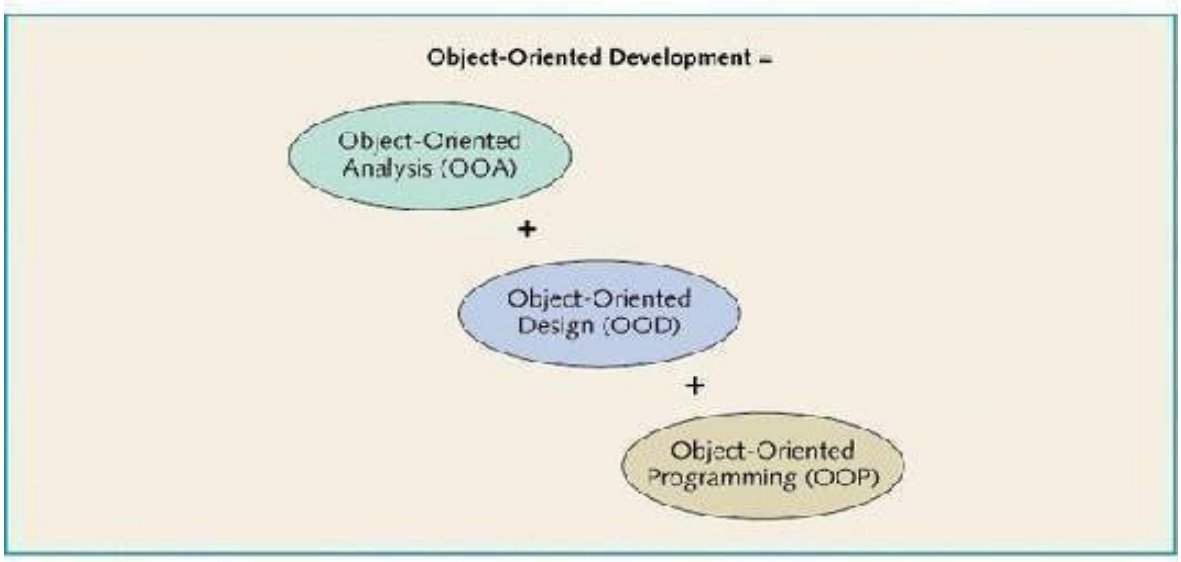


Figure 1-3 Object-oriented development

Development refers to the software life cycle: Analysis, Design and Implementation. The essence of OO Development is the *identification* and *organization* of application concepts, rather than their final representation in a programming language. It's a conceptual process independent of programming languages. OO development is fundamentally a way of thinking and not a programming technique.

OO methodology

Here we present a process for OO development and a graphical notation forrepresenting OO concepts. The process consists of building a model of an application and then adding details to it during design.

The methodology has the following stages

System conception: Software development begins with business analysis orusers conceiving an application and formulating tentative requirements.

Analysis: The analyst scrutinizes and rigorously restates the requirements from the system

conception by constructing models. The analysis model is a concise,precise abstraction of what the desired system must do, not how it will be done.

The analysis model has two parts-

Domain Model- a description of real world objects reflected within the system.

Application Model- a description of parts of the application system itself that are visible to the user.

E.g. In case of stock broker application-

Domain objects may include- stock, bond, trade & commission.

Application objects might control the execution of trades and present the results.

System Design: The development teams devise a high-level strategy- The System Architecture- for solving the application problem. The system designer should decide what performance characteristics to optimize, chose a strategy of attacking the problem, and make tentative resource allocations.

Class Design: The class designer adds details to the analysis model in accordance with the system design strategy. His focus is the data structures and algorithms needed to implement each class.

Implementation: Implementers translate the classes and relationships developed during class design into a particular programming language, database or hardware. During implementation, it is important to follow good softwareengineering practice.

Three models

We use three kinds of models to describe a system from different view points.

1. **Class Model—**for the objects in the system & their relationships. It describes the static structure of the objects in the system and theirrelationships.
   Class model contains class diagrams- a graph whose nodes are classes and arcs are relationships among the classes.
2. **State model—**for the life history of objects.
   It describes the aspects of an object that change over time. It specifies and implements control with state diagrams-a graph whose nodes are states and whose arcs are transition between states caused by events.
3. **Interaction Model**—for the interaction among objects.
   It describes how the objects in the system co-operate to achieve broader results. This model starts with use cases that are then elaborated with sequence and activity diagrams.
   Use case – focuses on functionality of a system – i.e what a system does for users.
   Sequence diagrams – shows the object that interact and the time sequence of their interactions.
   Activity diagrams – elaborates important processing steps.

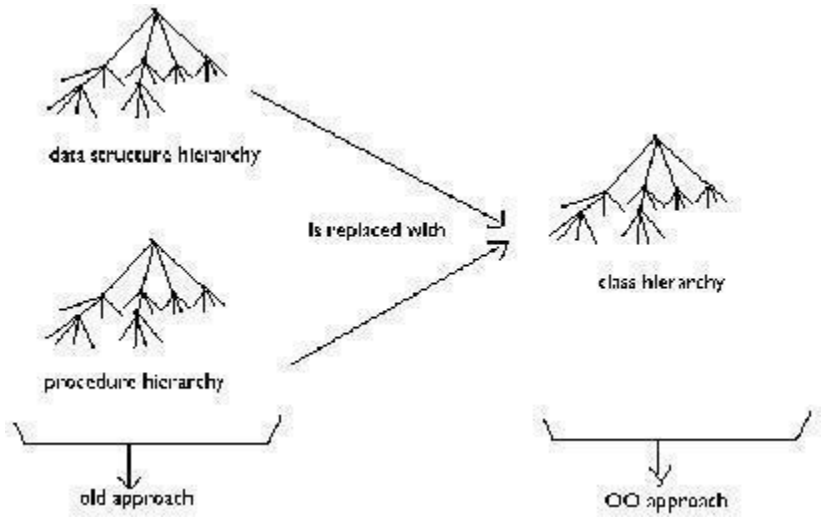
THEMES

Several themes pervade OO technology. Few are –

1. **Abstraction**
   Abstraction lets you focus on essential aspects of an application while ignoring details i.e

focusing on what an object is and does, before deciding how to implement it.
It's the most important skill required for OO development.

2. **Encapsulation (information hiding)**
   It separates the external aspects of an object (that are accessible to other objects) from the internal implementation details (that are hidden from other objects)
   Encapsulation prevents portions of a program from becoming so interdependent that a small change has massive ripple effects.
3. **Combining data and behavior**
   Caller of an operation need not consider how many implementations exist. In OO system the data structure hierarchy matches the operation inheritancehierarchy (fig).



4. **Sharing**
   OO techniques provide sharing at different levels. Inheritance of both data structure and behavior lets sub classes sharecommon code.
   OO development not only lets you share information within an application,but also offers the prospect of reusing designs and code on future projects.
5. **Emphasis on the essence of an object**
   OO development places a greater emphasis on data structure and a lesser emphasis on procedure structure than functional-decomposition methodologies.
6. **Synergy**
   Identity, classification, polymorphism and inheritance characterize OO languages.
   Each of these concepts can be used in isolation, but together they complement each other synergistically.

MODELLING AS A DESIGN TECHNIQUE

Note: A model is an abstraction of something for the purpose of understanding itbefore building it.

MODELLING

Designers build many kinds of models for various purposes before constructing things.

Models serve several purposes–

1. **Testing a physical entity before building it:** Medieval built scale models of Gothic Cathedrals to test the forces on the structures. Engineers test scale models of airplanes, cars and boats to improve their dynamics.
2. **Communication with customers:** Architects and product designers build models to show their customers (note: mock-ups are demonstration products that imitate some of the external behavior of a system).
3. **Visualization:** Storyboards of movies, TV shows and advertisements let writers see how their ideas flow.
4. **Reduction of complexity:** Models reduce complexity to understand directlyby separating out a small number of important things to do with at a time.

ABSTRACTION

Abstraction is the selective examination of certain aspects of a problem.

The goal of abstraction is to isolate those aspects that are important for some purpose and suppress those aspects that are unimportant.

THE THREE MODELS

1. **Class Model**: represents the static, structural, "data" aspects of a system.

It describes the structure of objects in a system- their identity, theirrelationships to other objects, their attributes, and their operations.

Goal in constructing class model is to capture those concepts from the realworld that are important to an application.

Class diagrams express the class model.

2. **State Model**: represents the temporal, behavioral, "control" aspects of asystem.

State model describes those aspects of objects concerned with time and the sequencing of operations – events that mark changes, states that define the context for events, and the organization of events and states.

State diagram express the state model.

Each state diagram shows the state and event sequences permitted in a system for one class of objects.

State diagram refer to the other models.

Actions and events in a state diagram become operations on objects in the class model. References between state diagrams become interactions in the interaction model.

3. **Interaction model** – represents the collaboration of individual objects, the "interaction"

aspects of a system.

Interaction model describes interactions between objects – how individualobjects collaborate to achieve the behavior of the system as a whole.

The state and interaction models describe different aspects of behavior, andyou need both to describe behavior fully.

Use cases, sequence diagrams and activity diagrams document the interaction model.