CHAPTER 1
Thinking Object Oriented

Procedural Oriented Programming:

Conventional Programming, using high level languages such as COBOL (Common Business Oriented Language), FORTAN (Formula Translation) and C, is commonly known as procedural oriented programming (POP). In procedure-oriented approach, the problem is views as a sequence of things to be done such as reading, calculating and printing. POP basically consists of writing a list of instructions (or actions) for the computer to follow, and organizing these instructions into groups known as function. A typical program structure for procedural programming is shown in the figure below. The technique of hierarchical decomposition has been used to specify the task to be completed for solving a problem.

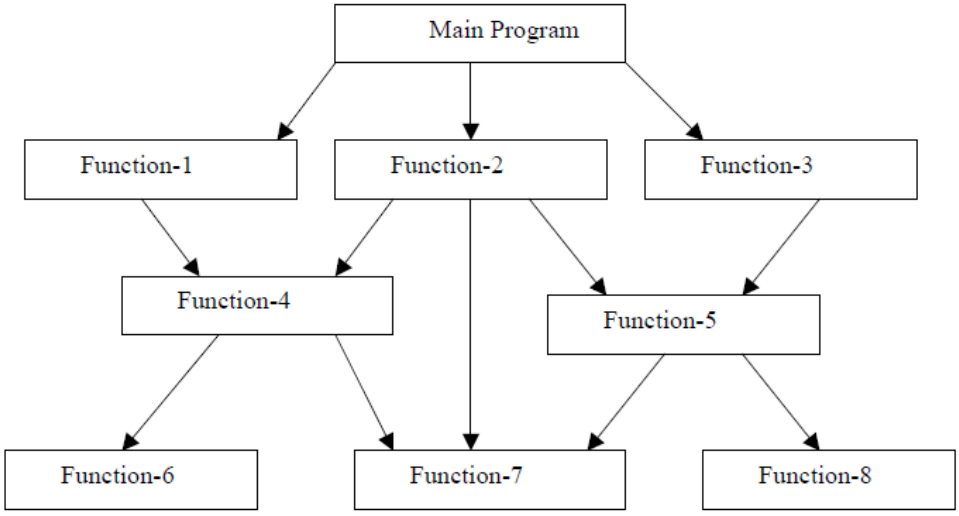
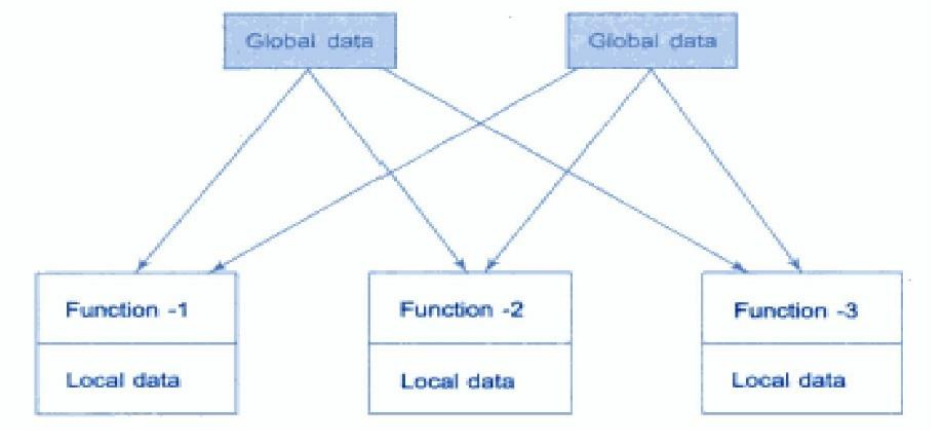


Fig. 1.2 Typical structure of procedural oriented programs

In a multi-function program, many important data items are placed as globally. So that they may be accessed by all the functions. Each function may have its own local data. The figure shown below shows the relationship of data and function in a procedure-oriented program.



Relationship of data and functions in procedural programming

Some features/characteristics of procedure-oriented programming are:

- Emphasis is on doing things (Algorithms).
- Large programs are divided into smaller programs called as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.
- Employs *top-down* approach in program design.

Limitations of Procedural Oriented Programming

- Procedural languages are difficult to relate with the real world objects.
- Procedural codes are very difficult to maintain, if the code grows larger.
- Procedural languages does not have automatic memory management .Hence, it makes the programmer to concern more about the memory management of the program.
- The data, which is used in procedural languages are exposed to the whole program. So, there is no security for the data.
- Creation of new data type is difficult. Different data types like complex numbers and two dimensional co-ordinates cannot be easily represented by POP.

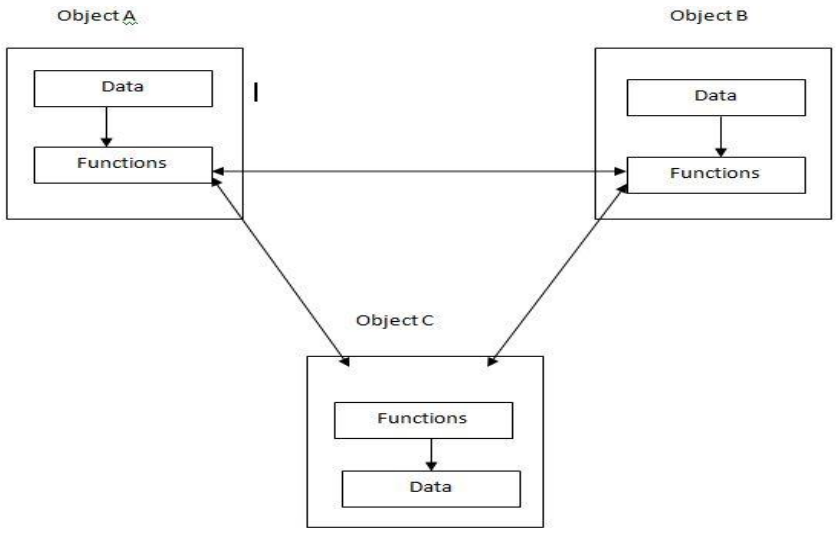
Object oriented programming Paradigm

(Paradigm = a typical example or pattern of something; a pattern or model.)

OOP can be considered as a programming paradigm and in the same time it is a set of software engineering tools which can be used to build more reliable and reusable systems.

Object Oriented Programming deals with or solves a problem by considering how we might go about handling a real-world situation and then ask how we could make the computer more closely model the techniques employed.

OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects. The organization of data and functions in the object-oriented programs shown in the figure:



The data of an object can be accessed only by the function associated with that object.

The fundamental idea behind object-oriented programming is to combine or encapsulate both data (or instance variables) and functions (or methods) that operate on that data into a single unit. This unit is called an object. The data is hidden, so it is safe from accidental alteration. An object's functions typically provide the only way

to access its data. In order to access the data in an object, we should know exactly what functions interact with it. No other functions can access the data. Hence OOP focuses on data portion rather than the process of solving the problem.

An object-oriented program typically consists of a number of objects, which communicate with each other by calling one another's functions. This is called sending a message to the object. This kind of relation is provided with the help of communication between two objects and this communication is done through information called message. In addition, object-oriented programming supports encapsulation, abstraction, inheritance, and polymorphism to write programs efficiently. Examples of object-oriented languages include Simula, Smalltalk, C++, Python, C#, Visual Basic .NET and Java etc.

Features of Object oriented programming

Some features of object oriented programming are:
• Emphasis is on data rather than procedure.
• Programs are divided into what are called objects.
• Data structures are designed such that they characterize the objects.
• Functions that operate on the data of an object are tied together in the data structure.
• Data is hidden and cannot be accessed be external functions.
• Objects may communicate with each other through functions.
• New data and functions can be easily added whenever necessary.
• Follows bottom-up approach in program design.

Advantages of Object-oriented Programming

- It is easy to model a real system as programming objects in OOP represents real objects.
  The objects are processed by their member data and functions. It is easy to analyze the
  user requirements.
- Elimination of redundant code due to inheritance, that is, we can use the same
  code in a base class by deriving a new class from it.

- Modularize the programs. Modular programs are easy to develop and can be distributed independently among different programmers.
- In OOP, data can be made private to a class such that only member functions of the class can access the data. This principle of data hiding helps the programmer to build a secure program that cannot be invaded by code in other part of the program.
- With the help of polymorphism, the same function or same operator can be used for
  different purposes. This helps to manage software complexity easily.
- Large problems can be reduced to smaller and more manageable problems. It is easy to
  partition the work in a project based on objects.
- Program complexity is low due to distinction of individual objects and their related data and functions.

Disadvantages of Object Oriented Programming

- Sometimes, the relation among the classes become artificial in nature.
- Designing a program in OOP concept is a little bit tricky.
- The programmer should have a proper planning before designing a program using OOP approach.
- Since everything is treated as objects in OOP, the programmers need proper skill such as design skills, programming skills, thinking in terms of objects etc.
- The size of programs developed with OOP is larger than the procedural approach.
- Since larger in size, that means more instruction to be executed, which results in the slower execution of programs.

Applications of Object-oriented Programming

1. Client-Server System
2. Object Oriented Database
3. Real Time Systems Design
4. Simulation and Modeling System
5. Hypertext, Hypermedia
6. Neural Networking and Parallel Programming
7. Decision Support and Office Automation Systems
8. CIM/CAD/CAM Systems

9. AI and Expert Systems

Differences between structured and Object oriented Programming Language

Structured Programming/procedural oriented Programming	Object oriented Programming
1. Large Programs are divided into smaller self- contained program segment known as functions.	1. Program are divided into entity called objects.
2. Focuses on process/logical structure and then data required for that process.	2. Object oriented Programming is designed which focuses on data.
3. Structured Programming follows top-down approach.	3. Object oriented Programming follows bottom-up approach.
4. Data and functions don't tie with each other.	4. Data and functions are tied together.
5. Structured programming is less secure as there is no way of data hiding.	5. Object oriented programming is more secure as having data hiding feature.
6. Structured programming can solve moderately complex programs.	6. Object oriented programming can solve any complex programs.
7. Provides less reusability and more function dependency.	7. Provide less function dependency and more reliability.
8. Data moves free around the system from one function to another.	8. Data is hidden and cannot accessed by external functions.
9. Eg.C, pascal ,ALGOL	9. Eg.C++,Java and C# (C sharp)

A way of viewing the world

Describe how object oriented programming models the real world problem with reference of agents, method, behavior and responsibilities. [PU: 2017 fall]

To illustrate the major idea of object oriented programming, Let us consider the real world scenario.

Suppose an individual named Chris wishes to send flowers to a friend named Robin, Who lives in another city. Because of the distance, Chris cannot simply pick the flowers and take them to Robin in person. Nevertheless, it is a task that is easily solved.

Chris simply walks to a nearby flower shop, run by a florist named Fred. Chris will tell Fred the kinds of flowers to send to Robin and the address to which they should be delivered .Chris can then can be assured that the flowers can be delivered expediently and automatically.

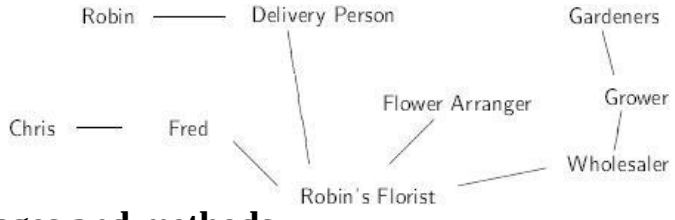
In above example,

- Fred is a **agent (objects)** to deliver flower**.**
- **Message** of wishes with specification of flowers is passed to florist (named Fred) who has **responsibility** to satisfy request.
- Fred uses some **method or algorithm or set of operations** to do this.
- There will community of agents to complete a task. Communication is established to all the concern mediators in a channel to deliver the flowers.
- How the florist satisfies the request is not Chris concern **(data hiding/information hiding).**

Agents and Communities

In above example Chris solved his problem with help of agent (object) Fred to deliver the
flower. **There will be community of agents to complete a task. In above example Fred will communicate with Robin's florist**.

An object oriented program is structured as a community of interacting agents called objects.
Each object has a role to play. Each object provides a service, or performs an action that is used by other members of community.



The community of agents helping delivery flowers

Messages and methods

Chris request Fred for delivering the flower to his friend Robin. This request lead to other requests, which lead to more requests, until the flowers ultimately reached Chris friend, Robin. We see therefore a members of this community interact with each other making requests.

Action is initiated in object-oriented programming by the transmission of a message to an agent (an object) responsible for the action. The messages encodes the request for an action and is accompanied by any additional information (arguments) needed to carry out the request. The receiver is the object to whom the message is sent. If the receiver accepts the messages, it accepts the responsibility to carry out the indicated action. In response to a message, the receiver will perform some method to satisfy the request.

Message Versus Procedure calls

In message passing

- There is a designated receiver for that message (the receiver is some object to which the message is sent).
- Interpretation of the message is dependent on the receiver and can vary with different receivers.
- There is a late binding between the message (function or procedure name) and the code fragment (method) used to respond the message.

In a procedure call

- There is no designated receiver.
- There is early (compile-time or link time) binding of name to code fragment in conventional procedure calls.

Responsibilities

A fundamental concept in object oriented programming is to describe behavior in term of responsibilities. Chris's Request for action indicates only the desired outcome (flowers send to Robin) .Fred is free to pursue any technique that achieves the desired objective and in doing so will not be hampered by interference from Chris.

Class and instances

In above scenario Fred is florist we can use florist to represent the category (or class) of all florists. Which means Fred is instance (object) of class Florist.

All objects are instances of a class. The method invoked by an object in response to a message is determined by the class of the receiver. All objects of given class use the same method in response to similar messages.

Summary of Object-oriented Concepts

Alan Kay, considered by some be the father of object-oriented programming, identified the following characteristics as fundamental to OOP.

1. Everything is an object.
2. Computation is performed by objects communicating with each other, requesting that other objects perform actions. Objects communicate by sending and receiving messages. A message is a request for action bundled with whatever arguments may be necessary to complete the task.
3. Each object has its own memory, which consists of other objects.
4. Every object is an instance of a class. A class simply represents a grouping of similar objects, such as integers or lists.
5. The class is the repository for behavior associated with an object. That is, all objects that are instances of the same class can perform the same actions.
6. Classes are organized into a singly rooted tree structure, called the inheritance hierarchy. Memory and behavior associated with instances of a class are automatically available to any class associated with a descendant in this tree structure.

Coping with Complexity

At early stages of computing most program were written in assembly language by a single individual.

As program becomes more complex, programmer found it difficult to remember all the information needed to know in order to develop or debug their software. such as

- Which values were contained in what registers?
- Did a new identifier name conflict with any previously defined name?
- What variables needed to be initialized before control could be transferred to another section of code?

The introduction of high level language,such as FORTAN,cobol and Algol solved some difficulties while simultaneously raising people expectations of what a computer could do.

Many software systems are complex not because they are large, but because they have many interconnections.

Interconnections make it difficult to understand pieces in isolation, or to carry them from one project to the next.

The inability to cleanly separate out components makes it difficult to divide a task between several programmers.

Complexity can only be managed by means of abstraction.

1. **Non-linear Behavior of complexity**

As programming projects became larger, an interesting phenomenon was observed that:
A task that would take one programmer 2 months to perform could not be accomplished by two programmers working for 1 month.

"The bearing of a child takes nine months, no matter how many women are assigned to the task".(refer to Freed Brook's book Mythical Man-Month)

This behavior of complexity is called non-linear behavior of complexity.

The reason for this nonlinear behavior

- The interconnections between software components were complicated
- Large amounts of information had to be communicated among various members of the programming team.

Conventional techniques brings about high degree of interconnectedness.
This means the dependence of one portion of code on another portion (coupling)
So, a complete understanding of what is going on requires a knowledge of both portion of code we are considering and the code that uses it .An individual section of code cannot be understood in isolation.

Abstraction Mechanism

Abstraction mechanism is use to control complexity.

It is the ability to encapsulate and isolate design and execution information.

Alternatively, Abstraction is a mechanism to displaying only essential information and hiding the details.

Making use of abstraction

a) Procedure and function:

Procedure and function are main two first abstraction mechanism to be widely used in programming language. They allowed task that were executed repeatedly to be collected in one place and reused rather than being duplicated several times. In addition, procedure gave the first possibility for information hiding.

A set of procedure written by one programmer can be used by many other programmers without knowing the exact details of implementation. They needed only the necessary interface.

b) Block Scoping:

Nesting of function (one function inside another) to solve problem associated with procedure and function, the idea of block and scoping was introduced which permits functions to be nested.

This is also not truly the solution. If a data area is shared by two or more procedure, it must still be declared in more general scope than procedure. For example:

```
int sum()
{
int a,b;
…….
……
int mul()
{
….
……
}
}
```

Partially solved the abstraction mechanism

c) Modules:

A module is basically a mechanism that allows the programmers to encapsulate a collection of procedures and data and supply both import and export statement to control visibility feature from outside modules.

Parnas principle for creation and use of modules:

•One must provide the intended user with all the information needed to use the module correctly and nothing more.

•One must provide the implement or with all the information needed to complete the module and nothing more.

d) Abstract data type (ADT):

- To solve instantiation, the problem of what to do, your application needed more than one instance of software abstraction. The key to solve this problem is ADT.
- ADT is simply programmer defined data type that can be manipulated in a manner similar to system provided data types.
- This supported both information hiding as well as creating many instance of new data type.

But message passing is not possible in ADT.

e) Objects- Messages, Inheritance and Polymorphism:

OOP added new ideas to the concept of ADT.

- **Message passing:** Activity is initiated by a request to a specific object, not by invoking of a function.
- **Inheritance** allows different data types to share the same code, leading to a reduction in code size and an increase in functionality.
- **Polymorphism** allows this shared code to be designed to fit the specific circumstances of individual data types.

Computation As Simulation

- Explain Computation as Simulation?

- In Case of Object oriented Programming, Explain how do we have the view that computation is simulation?[PU:2013 Spring]

OOP framework is different from the traditional conventional behavior of computer. Traditional model can be viewed as the computer is a data manager, following some pattern of instructions, wandering through memory, pulling values out of various memory transforming them in some manner, and pushing the results back into other memory.

The behavior of a computer executing a program is a process-state or pigeon-hole model. By examining the values in the slots, one can determine the state of the machine or the results produced by a computation. This model may be a more or less accurate picture of what takes place inside a computer. Real word problem solving is difficult in the **traditional model.**

In **Object Oriented Mode**l we speak of objects, messages, and responsibility for some action. We never mention memory addresses, variables, assignments, or any of the conventional
programming terms. This model is process of creating a host of helpers that forms a community and assists the programmer in the solution of a problem (Like in flower example).

The view of programming as creating a universe is in many ways similar to a style of
computer simulation called **"discrete event-driven simulation".**
In, in a discrete event-driven simulation the user creates computer models of the various
elements of the simulation, describes how they will interact with one another, and sets them
moving. This is almost identical to OOP in which user describes what various entities, object in program are, how will interact with one another and finally set them moving. **Thus in OOP, we have view that computation is simulation.**

Power of metaphor

- When programmer think about problems in terms of behavior and responsibilities of objects, they bring with them a wealth of intuition, ideas and understanding from their everyday experience.

- When envisioned as pigeon holes, mailboxes, or slots containing values, there is a little in the programmers background to provide insight into how should be structured.

Avoiding Infinite Regression

Object cannot always respond to a message by politely asking another object to perform some action. The result would be an infinite circle of request ,like two gentle man each politely waiting for the another to go first before entering a doorway. Thus to avoid infinite regression at a time, at least few objects need to perform some work besides passing request to another agent.

Varieties of Classes

Based on different forms of responsibility classes can be used for many different purposes and
be categorized as follows:

i) Data Manager (Data or State classes)

- Main responsibility is to maintain data or state information
- For example, in an abstraction of playing card, a major task for the class Card is simply maintain the data values that describe rank and suit.
- Often recognizable as nouns in problem description and are usually the fundamental building blocks of design.

ii) Data Sink or Data Sources

- These type of classes generate data, such as random number generator, or that accept data and then process them further, such as a class performing output to disk or file.
- These types of classes don't hold data for any period of time, but generates it on demand (for a data source) or processes it when called upon (for a data sink).

iii) View or Observer classes

- Most of the application is to display of information on an output device such as terminal screen.
- Because the code performing this activity is often complex, frequently modified and largely independent of the actual data being displayed, it is

good programming practice to isolate display behavior in classes other than those that maintain the data being displayed.

iv) Facilitator or Helper Class

- These are classes that maintain little or no state information themselves but assist in the execution of complex tasks.
- For example, in displaying a playing card image we use the services of facilitator class that handles the drawings of lines and text on the display device. Another facilitator class might help maintain linked list of cards.

Previous Board Exam Questions from this chapter

1) What is object orientation? Explain the difference between structured and Object oriented Programming approach.**[PU:2006 spring]**
2) Why OOP is known as a new paradigm? Illustrate with certain examples. **[PU:2005 fall]**
3) What is class? Explain the different types of classes.**[PU:2005 fall]**
4) Describe Object Oriented Programming as a new paradigm in Computer programming field.**[PU:2015 Spring]**
5) What makes OOP a new paradigm? Explain your answer with suitable points. **[PU:2010 fall]**
6) What influence is an object oriented approach said to have on software system design? What is your own opinion ?Justify through example.**[PU:2009 fall]**
7) Explain the advantages of object oriented paradigm.
8) What are the Critical issues that are to be considered while designing the large Programming? Why? **[PU:2009 spring]**
9) Why Object oriented Programming is Superior than Procedural-Oriented Programming. Explain.**[PU:2016 fall]**
10) What are the main features of Object Oriented Programming . **[PU:2013 fall]**
11) What are the mechanism of data abstraction? Explain the difference between structured and Object Oriented Programming Approach?**[PU:2013 fall]**
12) What is the significance of forming abstractions while designing an object oriented system? In case of object oriented Programming, Explain how do we have view that computation is simulation? **[PU:2013 spring]**
13) What makes OOP better than POP. Explain with features of OOP. **[PU:2014 fall]**

14) With the help of object oriented Programming, explain how can object oriented Programming cope in solving complex problem. Explain computation as simulation.
[PU: 2014 spring][PU: 2018 fall]

15) How does making use of abstraction help in designing of an object oriented System. Explain with an example.**[PU:2015 fall]**

16) What is the use of abstraction mechanism in C++?Explain with example.**[PU: 2019 fall]**

17) Describe how object oriented Programming models the real word object problem with reference of agents , method, behavior and responsibilities.**[PU:2017 fall]**

18) What are the shortcoming of procedural Programming? Explain the notation of "Everything is an object" in an object oriented programming. **[PU:2017 spring]**

19) Explain the encapsulation and data abstraction.

20) Write a short notes on:
- Abstraction **[PU:2006 spring]**
- Non-linear behavior of Complexity **[PU :2014 fall] [PU:2015 spring] [PU:2009 spring]**