Chapter-1
Thinking Object Oriented

Introduction to programming Concept 2
Procedural-Oriented Programming(POP) 2
Features of POP 2
Disadvantages of POP 3
Object-oriented Programming (OOP) 3
Features of OOP 3
Benefits of OOP 4
Disadvantages of OOP 4
POP vs OOP 4
Basic concept / Principles / Characteristics/Features of Object Oriented Programming 5
Object and Class 5
Encapsulation 5
Data Abstraction 6
Inheritance 6
Polymorphism 6
Message Passing 7
OOP as a new Paradigm 7
A Way of viewing world Agent(Object) 8
Agents and Communities 8
Messages & Methods 9
Responsibilities 9
Classes & Instances 9
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 a given class use the same method in response to similar messages 9
Class Hierarchies-Inheritance 10
Computation as Simulation 11
Traditional Model/Imperative Programming 11
Object Oriented Model 11
Coping with Complexity/Dealing with complexity 11
Non-Linear Behaviors of Complexity 11
Types of Classes 12

Introduction to programming Concept

- Since the invention of computer, many programming approaches have been tried.
- These techniques include modular programming, top-down programming, bottom-up programming and structured programming.
- The primary motivation in each has been the concern to handle **the increasing complexity** of programs that are reliable and maintainable.

Procedural-Oriented Programming(POP)

- A program in Procedural Language is a list of instructions each statement in the language tells the computer to do something involving – reading, calculating, writing output.
- A program in a procedural language is a list of instructions.
- This technique is only suitable for medium sized software applications.\
- Conventional programming, using HLL e.g. COBOL, FORTRAN, C etc. is commonly known as procedural programming.
- When program become larger, a single list of instructions become awkward so it is broken into smaller units called functions (also called as subroutine, procedure or subprogram).
- The primary focus of procedural oriented programming is on functions rather than data.
- The subroutines do not let code duplication.
- Procedure oriented programming basically consists of writing a list of instructions for computer to follow, and organize these instructions into groups known as functions.
- The idea of breaking a program into functions can be further extended by grouping a number of functions together into a larger entity called a module. Example: string.h has a bunch of functions for working with strings
- In procedural approach,
  - A program is a list of instruction.
  - When program in PL become larger, they are divided into functions (subroutines, sub-programs, procedures).
  - Functions are grouped into modules.

A typical program structure of Procedural programming is shown in figure below

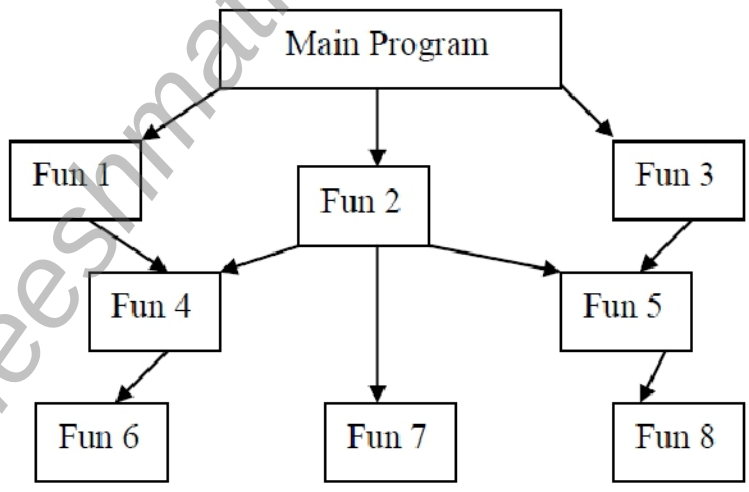


Fig: Structure of POP

Features of POP

- Emphasis is on doing thing (algorithm/procedure).
- Large programs are divided into smaller programs known as functions.
- Most of the functions share global variable
- Data move openly around the system from function to function.
- Employee **top-down approach** in program design.

Disadvantages of POP

- It emphasis on doing things.
- Since every function has complete access to the global variables, the new programmer can corrupt the data accidentally by creating function.
- It is often difficult to design because the components function and data structure do not model the real world as data and function are arranged separately.
- It is difficult to create new data types with procedural languages.
- Most Procedural languages are not usually extensible and hence procedural programs are more complex to write and maintain.

Object-oriented Programming (OOP)

- OOP is an approach to programming paradigm that attempts to eliminate some of the drawbacks of conventional programming methods by incorporating the best of structured programming features with several powerful new concepts.
- 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, C++, Python, C#, Visual Basic .NET and Java etc.

The general structure of OOP is shown in the figure below.

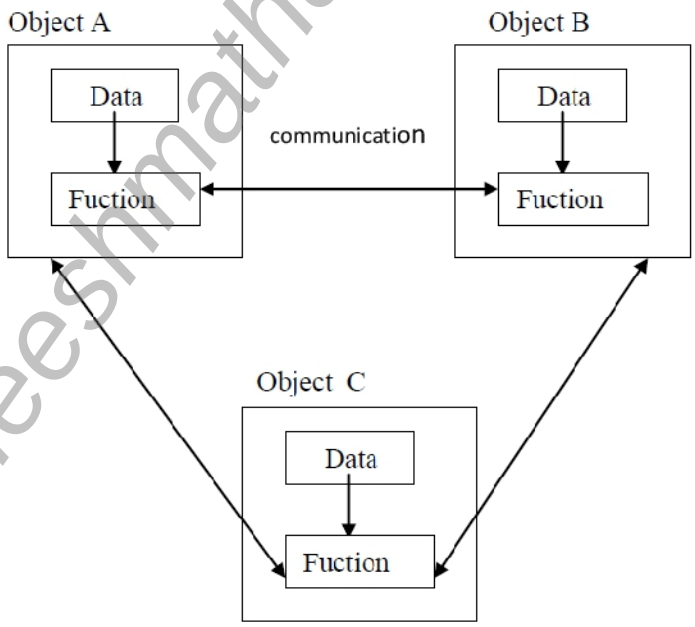


Fig: Organization of data and function in OOP

Features of OOP

- Emphasis is on data rather than procedures.
- Programs are divided into objects.
- Data structures are designed such that they characterize the objects.

- Functions & data are tied together in the data structures so that data abstraction is introduced.
- Data is hidden & can't be accessed by external functions.
- Object can communicate with each other through function.
- New data & functions can be easily added.
- Follows **Bottom up** approach.

Benefits of OOP

- Making the use of inheritance, redundant code is eliminated and the existing class is extended.
- Through data hiding, programmer can build secure program.
- It is possible to have multiple instances of an object to co-exist without any interference.
- System can be easily upgraded from small to large systems.
- Software complexity can be easily managed.
- Message passing technique for communication between objects makes the interface description with external system much simpler.
- Code reusability is much easier than conventional programming languages.
- It is possible to partition the work in a project based on objects.

Disadvantages of OOP

- Compiler and runtime overhead. Object oriented program required greater processing overhead – demands more resources.
- Re-orientation of software developer to object-oriented thinking.
- Requires the mastery in software engineering and programming methodology.
- Benefits only in long run while managing large software projects.
- The message passing between many objects in a complex application can be difficult to trace& debug.

POP vs OOP

Procedure Oriented Programming	Object Oriented Programming
Emphasis is given on procedures.	Emphasis is given on data.
Programs are divided into functions.	Programs are divided into objects.
Follow top-down approach of program design.	Follow bottom-up approach of program design.
Generally data cannot be hidden.	Data can be hidden, so that non-member function cannot access them.
It does not model the real world problem perfectly	It models the real world problem very well.
Data move from function to function	Data and function are tied together. Only related function can access them.
Maintaining and enhancing code is still difficult.	Maintaining and enhancing code is easy.
Code reusability is still difficult	Code reusability is easy in compare to procedure oriented approach.
Examples: FORTRAN, COBOL, Pascal, C	Examples: C++, JAVA, Smalltalk

It doesn't have any Access Specifier	It has three access specifiers named public, private and protected.

Programming Example:

1. WAP to find the area of rectangle using
   i. POP in c
   ii. Structure in C
   iii. POP in c++
   iv. Structure in C++
   v. OOP in c++
2. WAP to find the sum and difference of two number using Procedural Approach using C, procedural approach using C++ and object oriented approach using C++.
3. Create a class Student with two-member function to input and display roll and name. In main, create two objects to test the class.
4. WAP to input temperature in Celsius and convert it into Fahrenheit using concept of OOP.

$$C/5 = (F-32)/9$$

Basic concept / Principles / Characteristics/Features of Object Oriented Programming

Following are the basic principles of OOP

1. Object and Class
2. Data Abstraction and Encapsulation
3. Inheritance
4. Polymorphism
5. Message passing

Object and Class

- Object is an instance of a class i.e. variable of class type.
- Object are the basic runtime entities in an object oriented system.
- Objects contain data and function to manipulate the data. when a program is executed, the objects interact by sending message to one another.
- Generally, program objects are chosen such that they match closely with real world objects.

- A class is a framework that specifies what data and what functions will be included in objects of that class. It serves as a plan or blueprint from which individual objects are created.
- A Class is the collection of objects of similar type.
- Defining class doesn't create an object but class is the description of object's attributes and behaviors.
- For example: mango, apple and orange are members of class Fruit. so if we create a class Fruit then mango, apple, orange can be the object of Fruit.

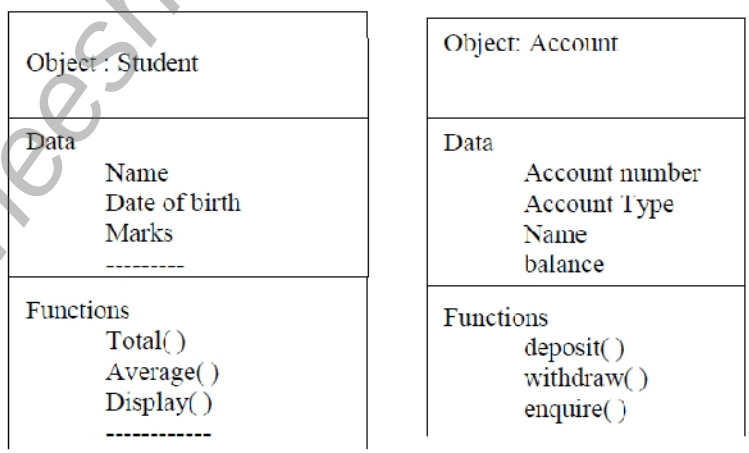


Fig: Representation of Object

Encapsulation

- Encapsulation is the process of wrapping or combining the data (called fields or attributes) and functions (called methods or behaviors) into a single framework called class.
- Encapsulation helps preventing the modification of data from outside the class by properly assigning the access privilege to the data inside the class.

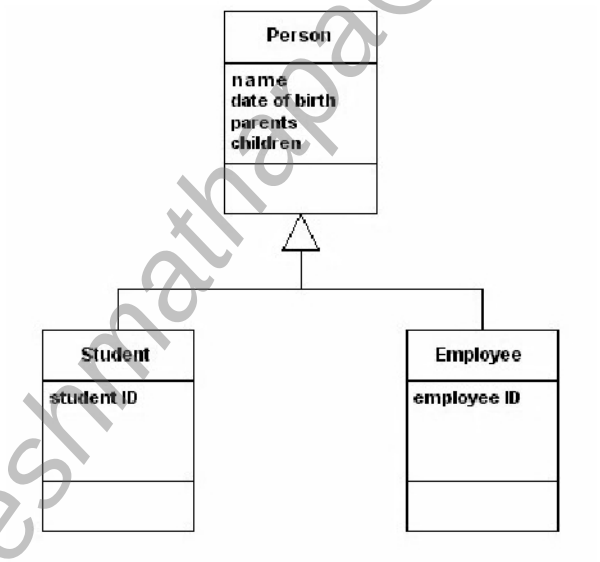
- So **the term data hiding** is possible due to the concept of encapsulation, since the data are hidden from the outside world, so that it is safe from accidental alteration.

Data Abstraction

- Abstraction refers to the act of representing essential features without including the complex background details or explanations.
- Classes uses the concept of abstraction and are defined as a list of abstract attributes and functions. Since classes uses the concept of data abstraction, they are known as abstract data types (ADT).
- Data abstraction is one of the most essential and important features of object-oriented programming in C++.
- Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation thereby decreasing complexity and increasing efficiency.
- Consider a **real-life example of a man driving a car**. The man only knows that pressing the accelerator will increase the speed of the car or applying brakes will stop the car but he does not know how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc. in the car. This is what abstraction is.

Inheritance

- Inheritance is the process by which objects of one class acquire the characteristics of object of another class.
- In OOP, the concept of inheritance provides the idea of reusability.
- We can use additional features to an existing class without modifying it.
- This is possible by deriving a new class (derived class) from the existing one (base class).
- This process of deriving a new class from the existing base class is called inheritance.
- It supports the concept of hierarchical classification.
- It allows the extension and reuse of existing code without having to rewrite the code.



- In the above figure, Person is base class while Student and Employee are derived form Person so they are referred to as derived class

Polymorphism

- Polymorphism means the quality of having more than one form.
- The representation of different behaviors using the same name is called polymorphism.
- However, the behavior depends upon the attribute the name holds at particular moment.
- Example of polymorphism in OOP is operator overloading, function overloading.

Example1:
Operator symbol '+' is used for arithmetic operation between two numbers, however by overloading same operator '+' it can be used for different purpose like concatenation of strings. So the process of making an operator to exhibits different behaviors in different instances is called **operator overloading**.

Example 2:

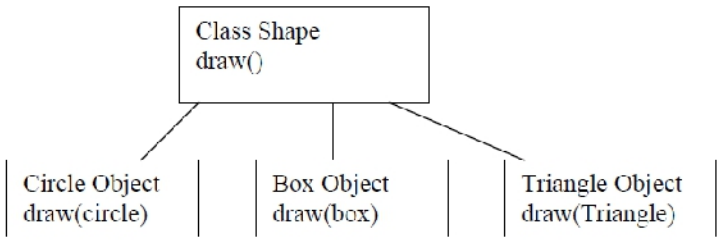


Fig: Polymorphism

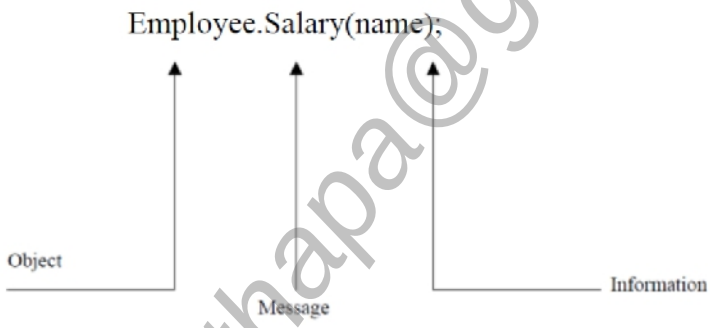
So in the above figure, a single function name is used to handle different number and different type of arguments. So using a single function name to perform different types of task is known as **function overloading**.

Message Passing

An object-oriented program consists of set of objects that communicate with each other. Object communicates with one another by sending and receiving information. The concept of message passing makes it easier to talk about building systems that directly model or simulate their real-world counterparts.

A Message for an object is a request for execution of a procedure, and therefore will invoke a function (procedure) in the receiving object that generates the desired results. Message passing involves specifying the name of object, the name of the function (message) and the information to be sent.

Example:



OOP as a new Paradigm

- Since the invention of computer, many programming approaches have been tried.
- These techniques include modular programming, top-down programming, bottom-up programming and structured programming.
- The primary motivation in each has been the concern to handle **the increasing complexity** of programs that are **reliable and maintainable.**
- The main necessity behind **inventing object oriented approach is to remove the drawback encountered in the procedural approach**. Procedural approach was only suitable for software development in small scale as it only focus on steps, face difficulty in separating data structures from functions that manipulated the data, reuse the existing code and maintenance.
- To build today's complex software it is just not enough to put together sequence of programming statements and set of procedures and modules, we need to incorporate sound programming techniques like OOP that can model the real world problem and are **easy to comprehend, implement and modify**
- The OOP programming paradigm holds data tightly rather than allowing it to move freely around the system. It ties the data to the function that operates on it and hides and protects it from accidental updates by external functions.
- Object oriented programming paradigm allows decomposition of the system into the number of entities called objects and then ties properties and function to these objects.
- Object-oriented programming paradigm methods enable us to create a set of objects that work together to produce software that is better understandable and models their problem domains than produced using traditional techniques.

- The software produced using object-oriented programming paradigm is easier to adapt to the changing requirements, easier to maintain, create modules of functionality, promote greater design, be more robust, and perform desired work efficiently.
- This newer technology also promises greeter programmer productivity, better quality of software and lesser maintenance cost.
- So OOP is a new programming paradigm of organizing and developing programs by incorporating best of structure programming features with several powerful new concepts. The word paradigm originally meant example, or model.

A Way of viewing world Agent(Object)

- To be clear on ideas of object-oriented programming, let us consider 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.
- Suppose I wish to **send flowers** to a friend (say Sally) who lives in a city many **miles away.** Because of the distance, there is no possibility of my picking the flowers and carrying them to her door myself. Nevertheless, sending her the flowers is an easy enough task; I just go down to my **local florist** (**say Flora**), tell her the **variety** and **quantity of** flowers I wish to send and give her **Sally's address**, and I can be assured the flowers will be delivered expediently and automatically without worrying all about how the flower is actually sent.

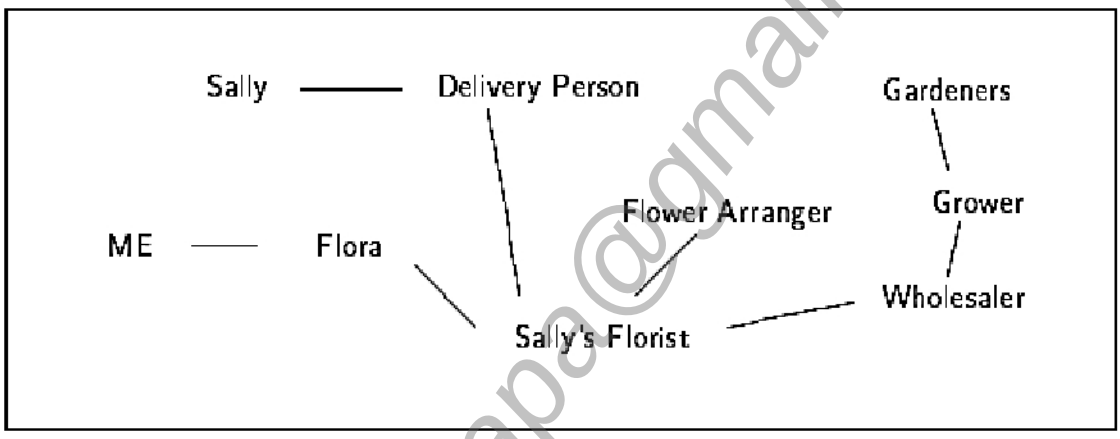


Fig: The community of Agents Helping Me

Note: Above situation can be models in OOP as

Class: Florist
Data Member: Address, Quantity and Variety
Function Members:
SetData() : responsible initialize sally's address, variety of flower and quantity to send
Send() : responsible for sending flower
Object: Flora i.e. it is responsibility of object flora to deliver flowers. Also there are several community of objects helping me as shown in above figure

Agents and Communities

- The mechanism I can use to solve my problem was first to find an appropriate **agent or Object** (namely, Flora) and to pass to her a **message** containing my request. So, it is the **responsibility** of Flora to satisfy my request. There is some method (some algorithm or set of operations) used by Flora to do this. I do not need to know the particular method she will use to satisfy my request; indeed, often I do not want to know the details. This information is usually hidden from my inspection i.e. **Information Hiding.** If I investigated however, **I might discover that there can be community of agents** to complete a task. Here, Flora delivers a slightly different message **to another florist** in my friend's city (i.e. **Sally's Florist**). That florist, in turn, perhaps has a subordinate who makes the **flower arrangement.** That florist then passes the flowers, along with yet another message, to a **delivery person,** and so on. Earlier, the florist in Sally's city had obtained her flowers from a flower wholesaler who, in turn, had interactions with the flower growers, each of whom had to manage a team of gardener
- **So, in 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 the community.**

- **In above case Flora is one of the agents in community that provide service of delivering flower to my friend Sally.**

Messages & Methods

- To solve this problem of delivering flower to my friend Sally, I can call agent/object Flora for delivering flower to my friend sally. After that there will be series of message passing and actions taken by various agents (Flora will communicate with sally's florist, sally's florist will arrange flower) until my flowers is delivered to my friend Sally as shown in figure above.
- **Action is initiated in object-oriented programming by the transmission of a message to an agent (an object) responsible for the action.** The message 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 message**, it accepts **the responsibility to carry out the indicated action**. In response to a message, the receiver will carry out the action by executing some **method/procedure** to satisfy the request.
- Also we need to notice the important principle of **information hiding** in regard to message passing i.e., the client sending the request need not know the actual means by which the request will be honored.

Message Passing Vs Procedure Call

- *In both cases, there is a set of well-defined steps that will be initiated following the request. But, there are two important distinctions.*
- *The first is that in a message there is a **designated receiver for that message**; the receiver is some object to which the message is sent. In a procedure call, there is no designated receiver.*
- *The second is that **the interpretation of the message** (that is, the method used to respond to the message) is **dependent on the receiver and can vary with different receivers**. I can give a message to my wife Elizabeth, for example, and she will understand it and a satisfactory outcome will be produced (that is, flowers will be delivered to my friend). However, the method Elizabeth uses to satisfy the request (in all likelihood, simply passing the request on to Flora) will be different from that used by Flora in response to the same request. If I ask Kenneth, my dentist, to send flowers to my friend, he may not have a method for solving that problem. If he understands the request at all, he will probably issue an appropriate error diagnostic.*

Responsibilities

- A fundamental concept in object-oriented programming is to describe **behavior in terms of responsibilities**.
- The behavior of an object is defined by its methods, which are the functions and subroutines defined within the object class. Without class methods, a class would simply be a structure.
- Methods determine what type of functionality a class has, how it modifies its data, and its overall behavior.
- In above example, my request for action indicates only the desired outcome (deliver flowers for my friend Sally). So, if Flora accepts my request, its' here responsibility to carry out the indication action by executing some method/procedures to satisfy the request. Flora is free to pursue any technique that achieves the desired objective i.e. to deliver flower to my friend, and is not hampered by interference on my part.
- By discussing a problem in terms of responsibilities we increase the level of **abstraction**. This permits greater independence between objects, a critical factor in solving complex problems.
- Also, it is possible to have multiple instances of an object with some responsibility to co-exist without any interference. For example, in the above problem, we can use new agent say Flora_new to satisfy my request of delivering flower for my friend Sally. Flora and Flora_new being object of same class Florist now can co-exist parallelly and both are responsible for delivering flower to my friend Sally without interfering each other. Responsibility implies non-interference.

Classes & Instances

- In above Scenario flora is a florist so we can use florist to represent the category (or class) of all florists. Which means Flora 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 a given class use the same method in response to similar messages

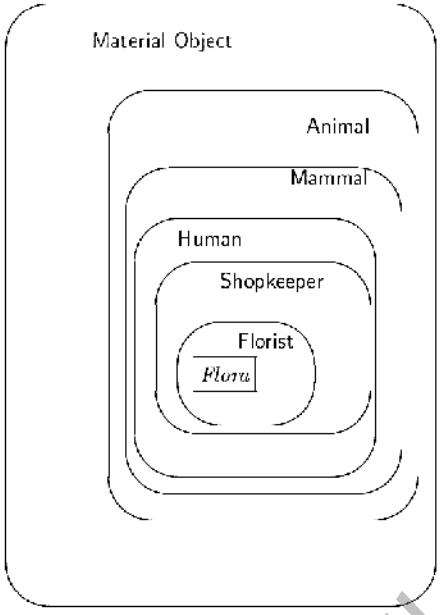
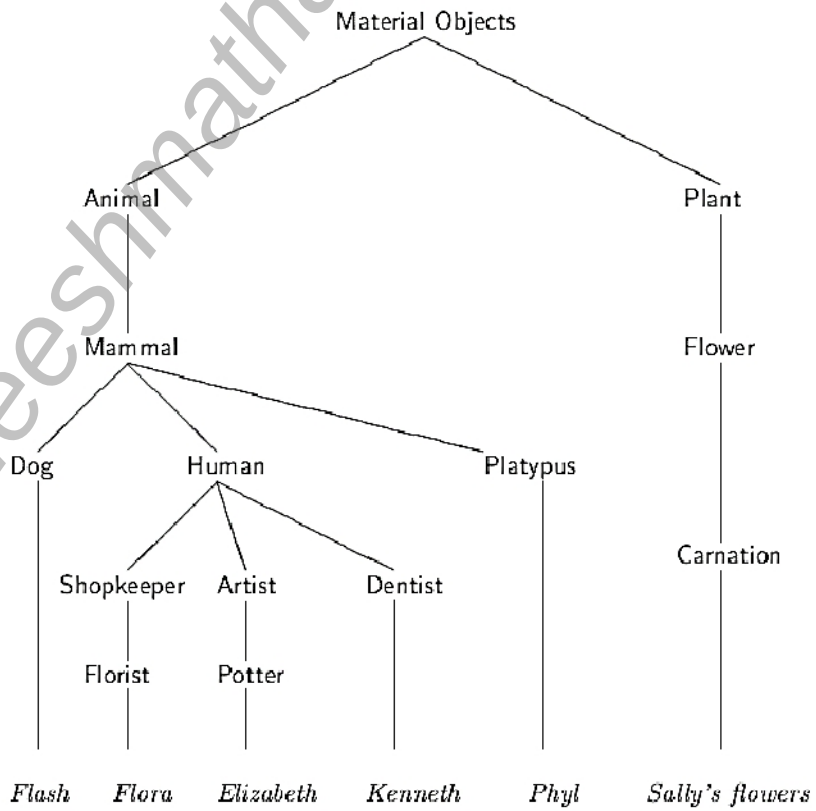


Fig: Categories Surrounding Flora

Class Hierarchies-Inheritance

- Flora-not necessarily because she is a florist, but because she is a shopkeeper
- One way to think about how I have organized my knowledge of Flora is in terms of a hierarchy of categories as shown in figure below.
- Category Florist is a more **specialized form of the category** Shopkeeper. All Florist are Shopkeeper but All Shopkeeper may not be Florist. So Florist holds isa relationship with Shopkeeper. So, any knowledge I have of shopkeeper is also true of Florists and hence of Flora.
- Flora is a Florist, but Florist is a specialized form of Shopkeeper. Furthermore, a Shopkeeper is also a Human; so I know, for example, that Flora is probably bipedal. A Human is a Mammal and a Mammal is an Animal, and an Animal is a Material Object (therefore it has mass and weight)



- In OOP, Classes can be organized into a hierarchical inheritance structure. A child class (or subclass) will inherit attributes from a parent class higher in the tree.

Computation as Simulation

- Object Oriented paradigm is a powerful technique when it comes to simulating real world problem via computer program using concept of class and objects in real life.

Traditional Model/Imperative Programming

- Traditional programming, also known as imperative programming, is a model where we provide the computer with a set number of instructions to follow to accomplish the provided goal.
- In traditional model, computer is viewed as 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.
- Behavior of a computer executing a program is a process-state or pigeon-hole model
- By examining the values in the slots, we 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 however, real world problem solving is difficult using traditional model. While this has the advantage of being easy to learn and understand on a smaller scale, as the scale increases, the sheer volume can make reasoning through the code and understanding everything pretty tough, which can lead to unforeseen problems in the codebase.

Object Oriented Model

- In object Oriented Model, we never mentioned memory addresses or variables or assignments or any of the conventional programming terms.
- Instead, we spoke of objects, messages, and responsibility for some action
- This model is process of creating a host of helpers(Objects) that forms a community and assists the programmer in the solution of a problem (Like in Flower example)
- This view of programming as creating a universe is in many ways similar to a style of computer simulation called "discrete event-driven simulation"
- Object oriented program is also similar to event driven simulation in which the user creates computer models of the various elements (entities) of the simulation, describes how they will interact with one another, and sets them moving.

Coping with Complexity/Dealing with complexity

- At early stages of computer programing, most of the programs were simple and written in assembly language by single individual.
- As program become more and more complex, programmer have difficulties in remembering all information needed to develop and debug all software.
- As problem become more and more complex, even the best programmer cannot perform the task by himself.
- There will be team/group of programmer working together to solve complex problem.

Non-Linear Behaviors of Complexity

- As programming projects became larger, an interesting phenomenon was observed.
- A task that would take one programmer two months to perform could not be accomplished by two programmers working for one month. Instead, the task will be more complicated as Computer programming is an intangible brain work. This can be treated as nonlinear behavior.
- Men and months are interchangeable commodities only when a task can be partitioned among many workers *with no communication among them.* This is true of reaping wheat or picking cotton, but it is not even approximately true in case of computer programming.
- In Fred Brooks' memorable phrase, "the bearing of a child takes nine months, no matter how many women are assigned to the task"
- The reason for this nonlinear behavior was complexity in particular, the interconnections between software components were complicated, and large amounts of information had to be communicated among various members of the programming team. Interconnectedness means the dependence of one portion of code on another section of code.

- The added burden of communication is made up of two parts: training and intercommunication. Each worker must be trained in the technology, the goals of the effort, the overall strategy, and the plan of work. This training cannot be partitioned, so this part of the work varies linearly with the number of workers

Types of Classes

- **Data managers classes :** Sometimes called data or state, are classes with the principal responsibility of maintaining data or state information. For example, in an abstraction of a problem, a major task for the class is simply to maintain. The data values that describe state information. Data managers are often recognizable as the nouns in a problem description and are usually the fundamental building blocks of a design.
- **Data sinks or data sources classes :** These are entities that generate data such as a random no. generator. Unlike a data manager, a data sink or data sources does not hold the data for any period, but generates it on demand.
- **View or observer class :** It an essential portion of most applications, display the information on a output devices such as terminal screen.
- **Facilitator or helper classes :** These are entities that maintain little or no state information themselves but assist with complex tasks. For example, in displaying an image we use the services of a facilitator class that handles the drawing of lines and text on the display device. In object-oriented languages like C++, Java etc: class describes a no. of possible run-time objects with same structure which are the instances of the class.

Assignment 1

1. Explain class as ADT with suitable example.
2. How is message different from procedure call?
3. In what way is object oriented program like a simulation?
4. Explain Abstraction mechanism technique in c++ with suitable example.

References:

1. E Balagurusamy, Object Oriented Programming with C++
2. Object-Oriented Programming in C++, Fourth Edition, Robert Lafore