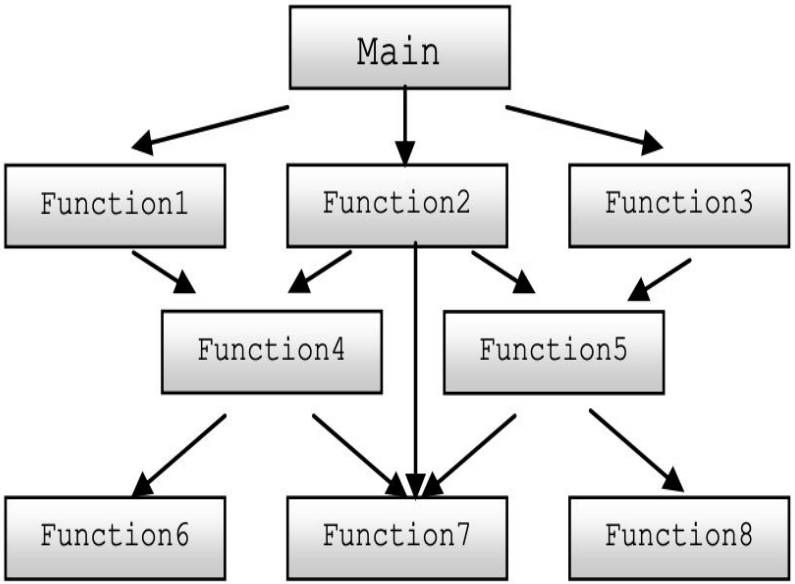
Object Oriented Programming

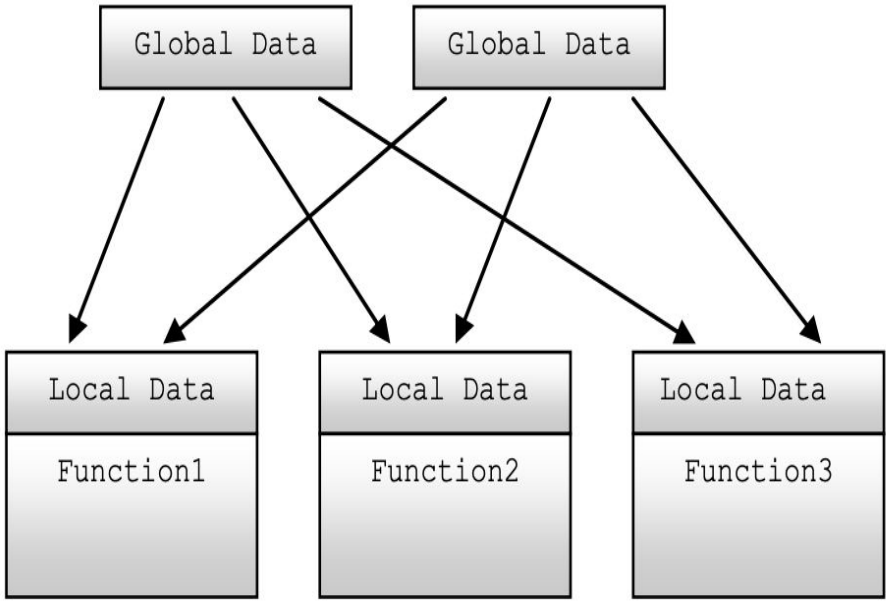
Chapter 1

Structured Programming

- Given problem is divided into sub problems and each of these sub problems can be further divided and so on till each of these sub problems can be easily be coded

- Each sub problem coded separately is called a procedure or function
- Procedures and functions provide main importance in structured programming but not data Many data items are placed as global, accessible by all functions
- Global data are more vulnerable to an inadvertent change by a function


Main
Function1
Function2
Function3
Function4
Function5
Function6
Function7
Function8


Global Data
Global Data
Local Data
Function1
Local Data
Function2
Local Data
Function3

Emphasis is on doing things i.e. algorithms and not on data. Data is given second-class status while coding. Any change in a data type being used needs to be made to all the functions using it . Time consuming.

In large programs it is very difficult to identify what data is used by which function.

Revising an external data structure means revising all functions that access the data .

Employs Top-Down approach i.e. first identify main function then other sub functions and then others. It does not model the real world problems because functions are action- oriented.

Object Oriented Programming

Invented to remove flaws encountered in procedural approach.

Solves problem in terms of objects used rather than procedures or functions for solving it.

Gives primary importance to data being used instead of operations performed on the data.

The combination of data and its associated functions is known as an object. .

Programs are divided into objects not into functions.

The close match between objects in the programming sense and objects in the real world helps us in thinking in terms of objects rather than functions that helps us to design programs easily.

We can access the data only thru the functions associated with the object (protection).

New data and functions can be easily added whenever necessary.

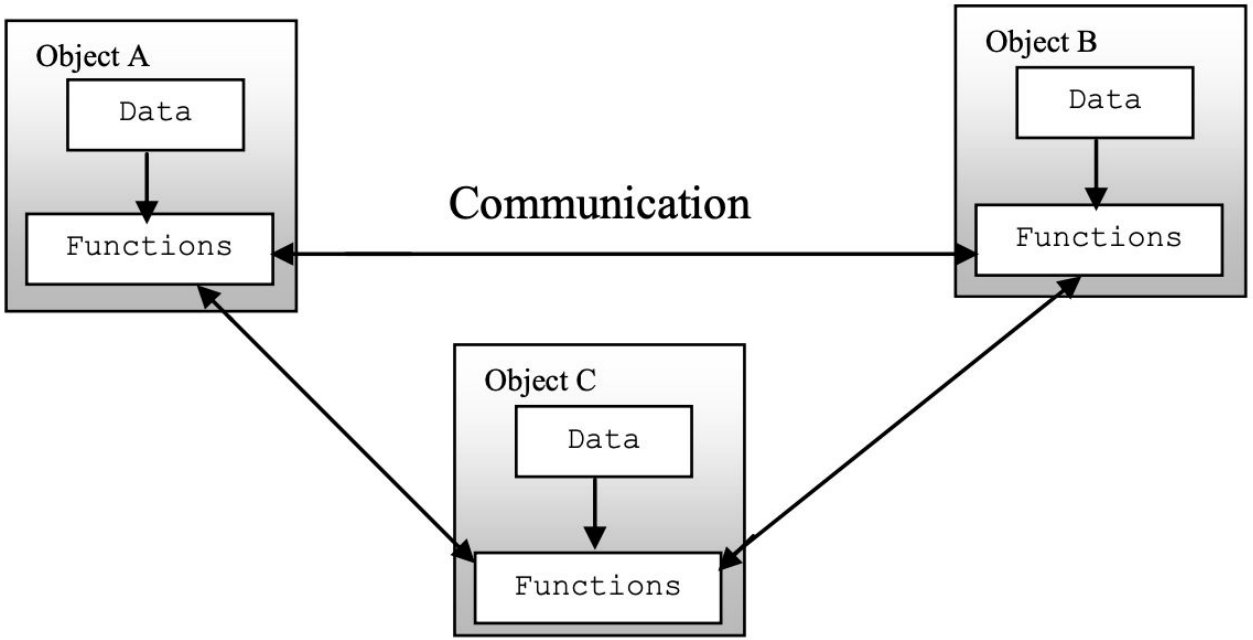
Object is considered to be a partitioned area of computer memory that stores data and set of operations that can access that data.

Follows Bottom-Up approach, in program design.

“An approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand”

Identify objects in the system first and then identify their collaboration or interaction to meet the goal of the system

In an Object Oriented System, the overall goal of the system is achieved thru the collaboration of objects



Object Oriented as a new paradigm

Paradigm: Any example, model, standard [Latin paradigma], overall strategy or approach to doing things
Programming Paradigms:

The model of developing software that describes the structure of computation

Imperative Programming: Pascal, C Logic Programming: Prolog, Lisp Functional Programming: FP, Haskell

“A programming paradigm is a way of conceptualizing what it means to perform computation and how tasks to be carried out on a computer should be structured and organized.”

The style of problem solving embodied in Object Oriented technique is frequently the method used to address problems in everyday life

Computer novices are often able to grasp the basic ideas of OOP easily whereas computer literate people are often blocked by their own preconceptions.

Alan Kay (promoter of Object Oriented Paradigm, designer of Small Talk) found teaching Small Talk to children as being lot easier than to computer professionals

A way of viewing the world - Agents, Responsibility, Messages and Methods

Solving problem mimicking the real world scenario. Real world analogy:

Sending flowers to a friend named Elise for her birthday Mechanism used:

Find an appropriate agent (namely Flo)

Pass her a message containing the request (kind and no. of flowers + address of friend) for delivery of flowers

It is the responsibility of Flo now to satisfy the request

Flo uses some method, some algorithm or set of operations to do this How the florist satisfies the request is not any of our concern.

Action is initiated in OOP by transmission of a message to an agent (an object) responsible for 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 agent 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 perform some method to satisfy the request.

e.g. while shopping you make a request to the shopkeeper, you don‟t show him how to get the things for you

```
shopkeeper.giveMeJeans(color,length,waist,style); agent message arguments
```

Information Hiding

Client sending request need not know the actual means by which the request will be honored Message Passing

1. There is a designated receiver for that message 1.
2. Late binding between the message (function 2. name) and the code fragment (method) used for response
3. Interpretation of the message is dependent on the receiver and can vary with different receivers.

Procedure Call

No designated receiver

Early binding (compile time) of name to code fragment

Behavior is described in terms of responsibilities, in OOP

No designated receiver

Early binding (compile time) of name to code fragment

For a particular request for action we can expect only the desired outcome i.e. a light bulb glows upon request (if switch is tuned on) but it doesn‟t make sound as the responsibility given to the light bulb is to glow (its behavior), it doesn‟t know how to make sound.

Responsibility permits greater independence between objects or agents (as each one can bear the responsibility without interference from others) e.g. a light bulb doesn‟t need others help from other objects to glow, it is functionally independent and capable.

Alan Kay's Description of Object Oriented Programming

i. Everything is an object

ii. Objects perform computation by making requests of each other thru the passing of messages

iii. Every object has it sown memory, which consists of other objects

iv. Every object is an instance of class. A class groups similar objects

v. The class is the repository of behavior associates with an object (objects of a class perform same actions)

Classes are organized into singly rooted tree structure called an inheritance hierarchy

Basic Concepts of OOP

i. Objects

Basic runtime entities that may represent a person, a place, a bank account etc

Programming problem is analyzed in terms of objects so they must be chosen such that they match closely with the real world objects

Objects take up space in memory and have an associated address like a structure in C

Objects interact by sending messages to one another during program execution

e.g. customer object requesting for the bank balance from account object,
`account.getBalance(customerID);`

Each object has its name, data and code to manipulate the data (functions). In other words, each object is defined by its identity, state and behavior

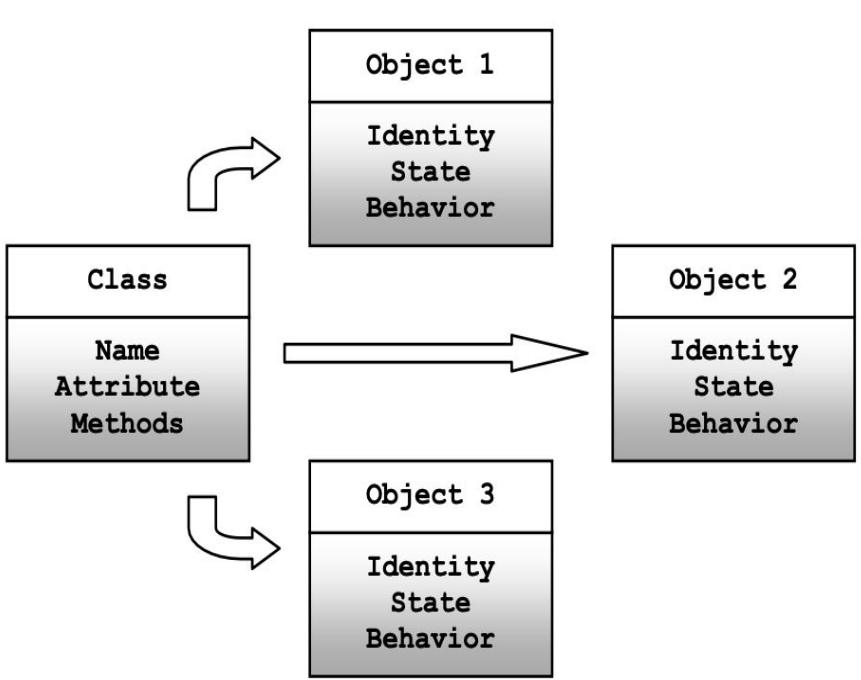
Basic Concepts of OOP

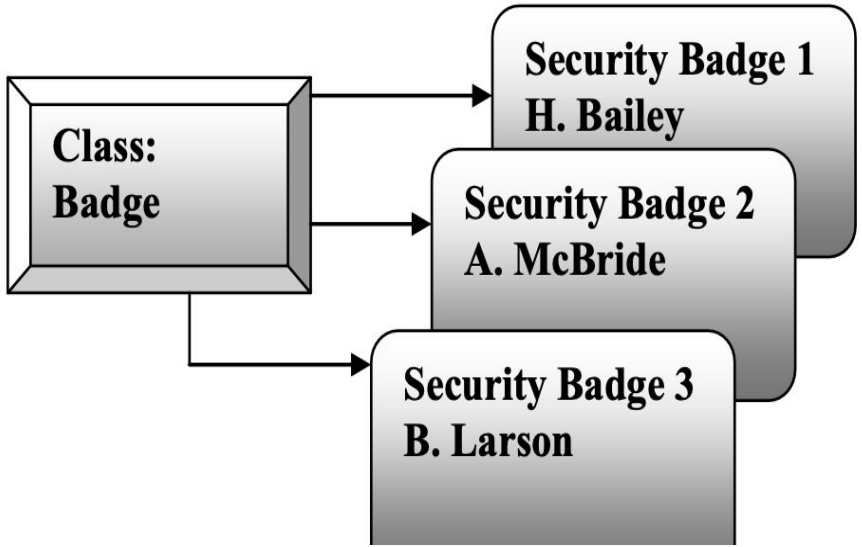
Identity: Each object can be identified by a unique name What it is called. e.g. Bajaj fan

State: Each object maintains its state in terms of variables What it is. e.g. Bajaj fan is in off state or on state

Behavior: Each object implements its behavior with methods (associated functions) What it does. e.g. Bajaj Fan can turn itself on or turn off upon request

Object	Identity	State	Behavior
Dog	Johnny	white, Doberman, hungry	sleeping, barking, eating
Pen	Parker	full, empty, nib out, nib in	write, refill, leak down
Cup	Coffee Cup	white , hot, full	can be filled, drunk from, carried
Fan	Khetan	white, big, off, on	turn on , turn off


Object 1
Identity
State
Behavior
Class
Name
Attribute
Methods
Object 2
Identity
State
Behavior
Object 3
Identity
State
Behavior


Class:
Badge
Security Badge 1
H. Bailey
Security Badge 2
A. McBride
Security Badge 3
B. Larson

In OOP objects are identified as

Tangible Things : physical objects e.g. car, bicycle, house, fan

Roles: of people or organization e.g. employee, account holder, employer

Incidents: Something happening at a particular point in time flight, transactions (deposit/withdrawal) Interaction: a link between objects e.g. electrical connection

Specification: a definition of a set of objects e.g. Description of a particular type of stock item

ii. Class

A collection of objects that share common properties and relationships

An OO concept that encapsulate the data and procedural abstractions (methods) required to describe the contents (state or attributes) and behavior of some real world entity

A class is an abstract data-type, which acts as a blue print or a prototype that defines the variables and methods common to all objects of a certain kind.

Objects are variables of the type class

A class is comprised of a set of attributes and behavior shared by similar type of objects
e.g. mango, orange apple are members of the class fruit

Selector Method

Read/ Get Method

Allows access to an attribute but does not allow that attribute to be changed

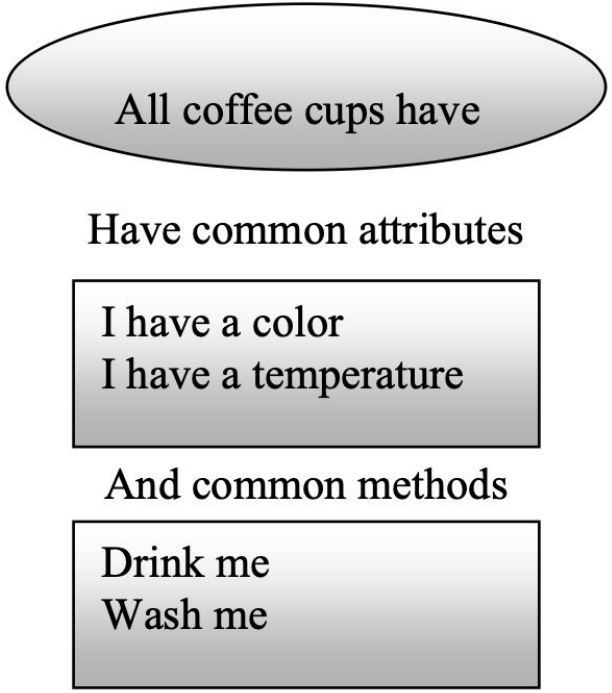
Analogous to functions (data is unaffected by executing a function)

Modifier Method

Write/Set Method

Allows attribute to be changed

Analogous to procedures (data is affected by executing a function)

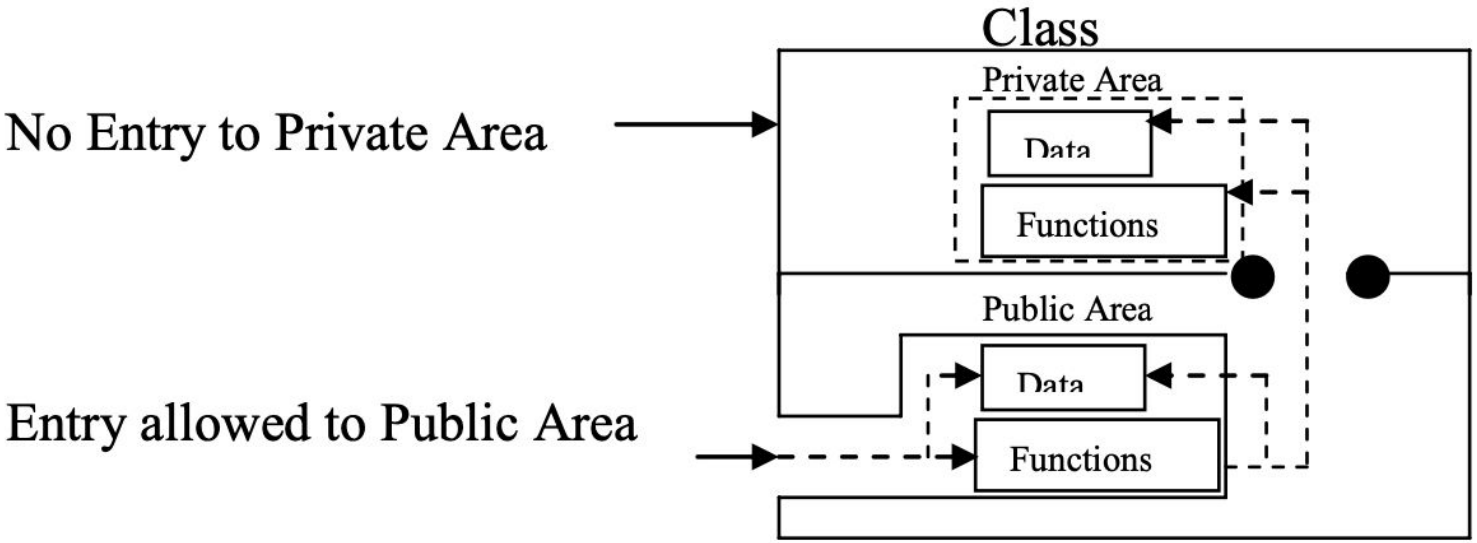


Data Abstraction and Encapsulation

The wrapping up of data and function into a single unit (class) is known as encapsulation.

The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.

This insulation of the data from direct access by the program is called data hiding or information hiding



The act of representing essential features without including the background details is known as abstraction.

e.g. we use a switch board for switching on a fan or bulb, we need not know about the internal fitting or back ground details

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight , cost and functions to operate on these attributes

Encapsulation is a way to implement data abstraction

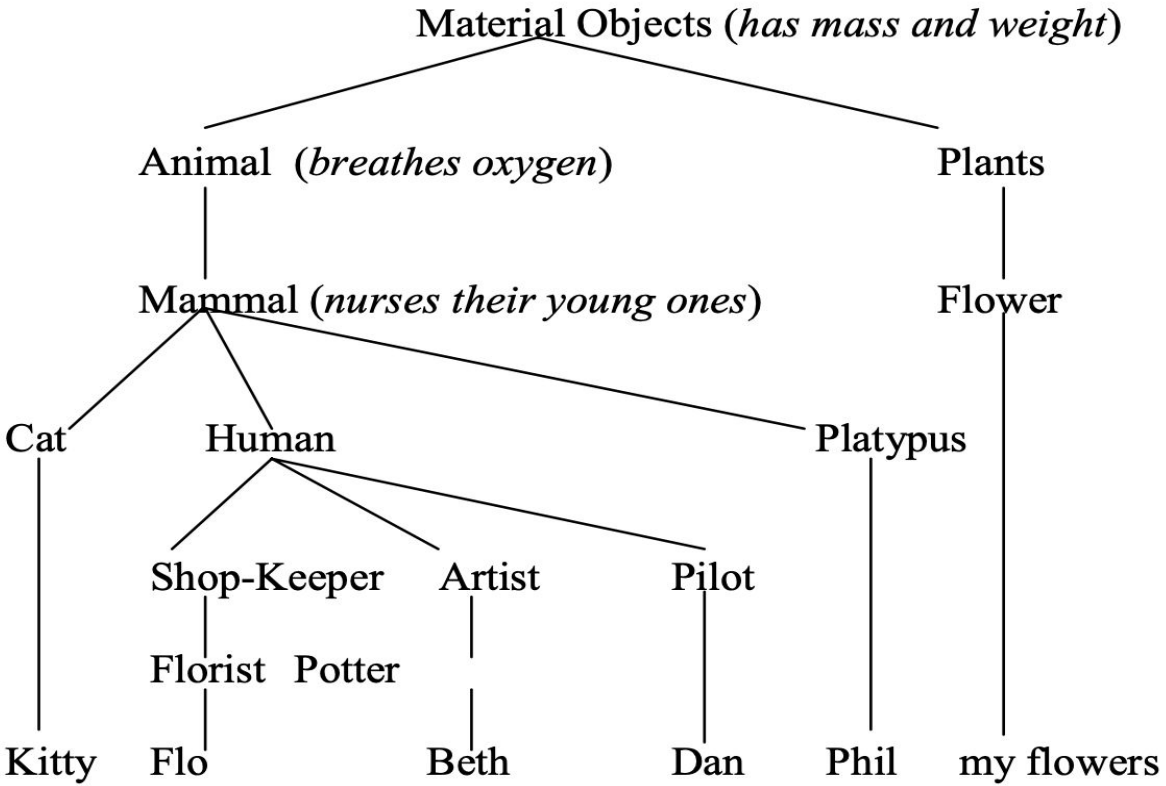
The attributes are called data members as they hold information

The functions that operate on these data are called member function or methods

Since classes use the concept of data abstraction , they are known as abstract data types (ADT)

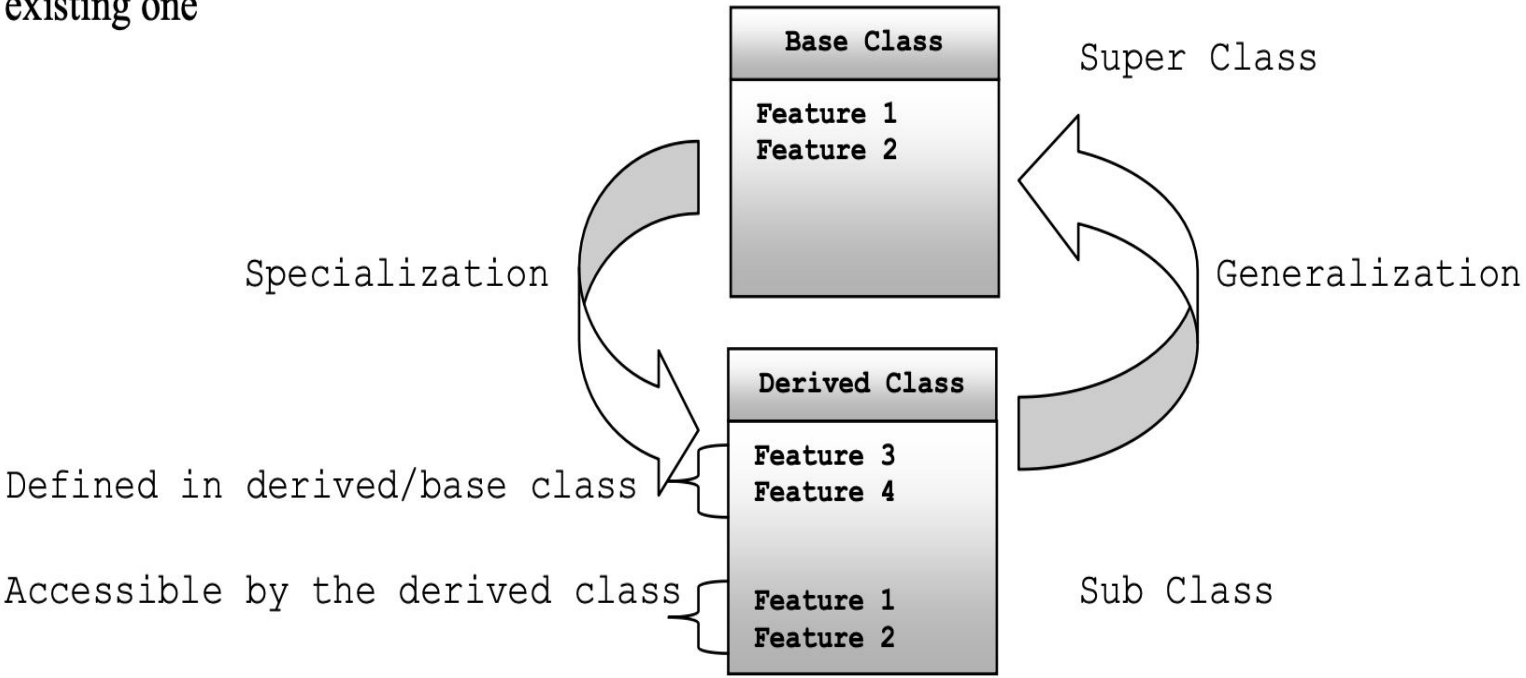
iii. Inheritance:

Inheritance is the process by which objects of one class acquire the properties of objects of another class It supports the concept of hierarchical classification



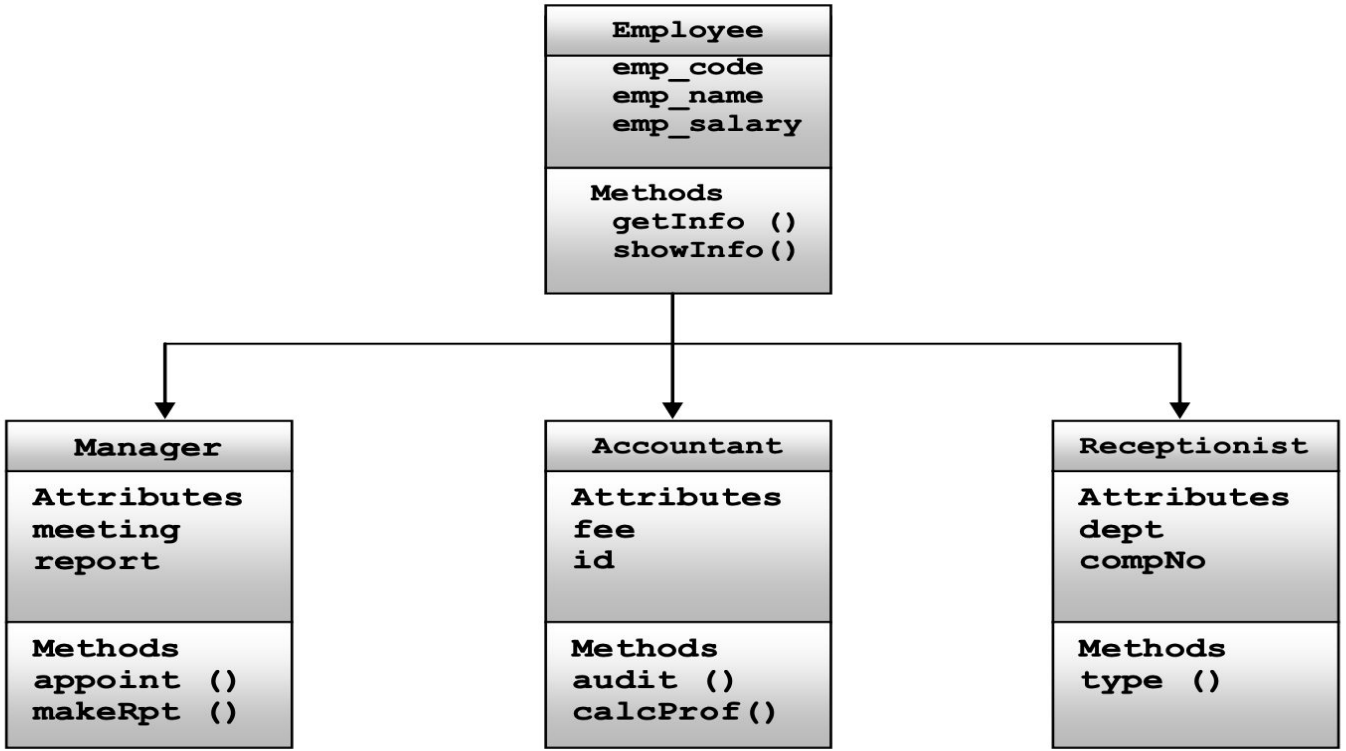
It provides the idea of ***reusability***

We can add additional features to an existing class without modifying it by deriving a new class from the existing one



The existing class is called the base class (generalization of the newly derived class) and the new class is called the derived class (specialization of the existing base class)

This new class will have the combined features of both the classes. In case of the employees of an organization, all employees share the common attributes emp_salary, emp_code, emp_name



iv. Polymorphism

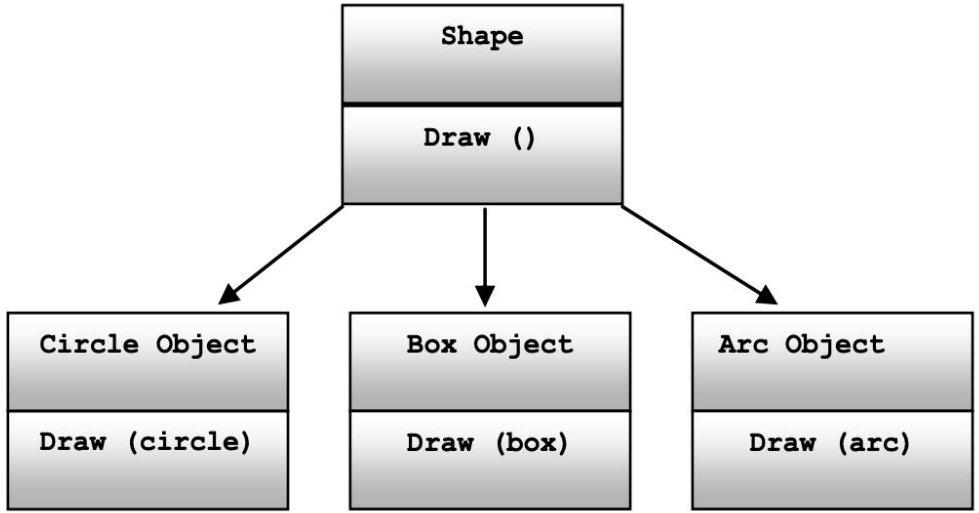
Poly means **many** and ***Morphos*** means **forms** , together it means the **ability to take more than one form**

Polymorphism means that it is possible to ***overload*** (*use to mean more than one thing*) the *symbols* (*operators and function names*) that we use in a program so that the **same symbol** can have different meanings in **different context**.

An operation may exhibit different behavior in different instances depending upon the types of data used in the operation

e.g. for the same request the Flo might respond in a different way (*sends flowers in a car*) than one of my friends (*he might use a motorcycle to drop those flowers*)

```
#include <iostream.h>
void display(char a[], char b[]){
  cout << strcat(a,b) ;
}
void display(int a, int b){
  cout << a+b ;
}
void main(){
 char a[]="Object", b[]="Oriented";
 int  c=99,d=100;
  display(c,d); display(a,b);
}
```



Similarly, operators +, -, = can also be used to mean more than one thing

The process of making an operator or function to exhibit different behavior in different instances is known as operator or function overloading respectively.

v. Dynamic Binding

Binding means linking a procedure call to the code to be executed in response to the call

Static Binding (Early Binding): If the binding of a function call to the function code is done during compile time it is known as static binding

Dynamic Binding(Late Binding): Means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.

In the above example, every object will inherit the draw() method from their parent class Shape However, its algorithm is unique to each other as the draw procedure will be redefined in each class that defined that object

At run time the code matching the object under current reference will be called.

vi. Message Passing

Objects communicate with each other thru sending and receiving messages much the same way as people pass messages to one another

The difference is in case of objects the message needs to be more precise/specific

In case of people, we ask a driver to drive faster

In case of objects the object making the above request also needs to specify the amount In case of a message passing, there is a desired recipient of the message.

Computation as Simulation

In discrete event driven simulation the user creates computer models of various elements of the simulation, describes how they will interact with one another and sets them moving.

This is almost identical to Object Oriented Programming in which the user describes what the various entities in the program are, how will they interact with one another and finally set them in motion.

Thus OOP we have the view that computation is simulation.

Coping with Complexity The non-linear behavior of complexity

A task that would take one programmer two months to complete could not be accomplished by two programmers working for one month.

The reason for this non-linear behavior is complexity which are

1. the interconnection between software components are complicated

2. a large amount of information has to be communicated among various members of the programming team

Abstraction

Abstraction is used to control the complexity

- Abstraction means showing only essential attributes and hiding unnecessary information
- It is hiding unwanted details and showing only most essential information

Abstraction Mechanism

1. Procedure and function
2. Block Scope
3. Modules
4. Abstract Data Type
5. Object

Abstraction Mechanism

1. Function

-one of the first abstraction mechanism used in programming language

- function allows to write a set of code to perform a task and when ever you need to perform that task, we can simply call the function rather than writing same code multiple times

-Libraries of function gave the first possibility of information hiding where one programmer could write a set of procedures which could be used by many others.

-The other programmer would only know the necessary interface and not know the exact details of implementation.

- A function abstraction works by allowing the programmer to call a function written by the another programmer without necessarily understanding how it is implemented.

1. Procedure and function

```
int a,b;
void input() {
 ……
}
void add()
{
……
}
```

Block Scope

Block scope determines the visibility or accessibility of the block

When we use { }, it is a block.

So, if we declare variables in the block scope, it means those variables exist only within the corresponding block

Nesting of function

• In nesting of function, one function is nested within another function

```
void add(int x,int y) {
 …
}
void input() {
int a,b;
add(a,b);
}
```

Modules

Large programs can be broken into modules

- The programmers can think about the implementation of a piece of a program without full knowledge of the rest of the programs
- The rest of the programs needs to be understood only abstractly

David Parnas’ principle for modules are

i. The development module must be provided with all information needed to complete the module and nothing more.

ii. The developer of a module must provide the intended user with all the information to use the module correctly and nothing more.

Abstract Data type

Abstract data type is user defined data type or class whose properties is defined by set of values and set of operations

- ADT helps us to group data members and member functions
- It can decide which data members will be visible to which parts of programs
- It prevents user from directly accessing the private members

Object

Objects provide abstraction by hiding internal implementation details

- We just need to know which methods of the object are available to call and which input parameters we need to pass do specific operation
- We don’t need to understand how the method is implemented and what kinds of operations the method has to perform to create the expected result
- It has message passing feature