VSIT | Vidyalankar School of Information Technology

NAAC Accredited College

Object Oriented Programming

MODULE-1: Object Oriented Methodology and Principles of OOPS

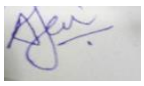
Dr. Amita Jain

VSIT | Vidyalankar School of Information Technology

NAAC Accredited College

Certificate

This is to certify that the e-book titled "OBJECT ORIENTED PROGRAMMING" comprises all elementary learning tools for a better understating of the relevant concepts. This e-book is comprehensively compiled as per the predefined eight parameters and guidelines.



Signature

Dr. Amita Jain

Assistant Professor

Department of IT

Date:18-11-2019

⚠ *DISCLAIMER: The information contained in this e-book is compiled and distributed for educational purposes only. This e-book has been designed to help learners understand relevant concepts with a more dynamic int erface. The compiler of this e-book and VidyalankarInstitute of Technology give full and due credit to the authors of the contents,developers and all websites from wherever information has been sourced. We acknowledge our gratitude towards the websites YouTube, Wikipedia, and Google search engine. No commercial benefits are being drawn from this project.*

Unit I ***Object Oriented Methodology and Principle of OOPS***

Contents:

- **Object Oriented Methodology:**
  Introduction, Advantages and Disadvantages of Procedure Oriented Languages, what is Object Oriented? What is Object Oriented Development? Object Oriented Themes, Benefits and Application of OOPS.

- **Principles of OOPS:** OOPS Paradigm, Basic Concepts of OOPS: Objects, Classes, Data Abstraction and Data Encapsulation, Inheritance, Polymorphism, Dynamic Binding, Message Passing

- Summary

- Question Bank

- MCQs

Recommended Books:

1. Object Oriented Analysis and Design , 3rd Edition 2012, Timothy Budd , TMH
2. Mastering C++ , 2nd Edition 2011, K R Venugopal, Rajkumar Buyya, T Ravishankar , Tata McGraw Hill
3. C++ for beginners, 2013, B. M. Hirwani, SPD
4. Effective Modern C++ , SPD, Scott Meyers
5. Object Oriented Programming with C++ , 4TH Edition , E. Balagurusamy , Tata McGraw Hill
6. Learning Python , 5th Edition 2013, Mark Lutz , O' Reilly
7. Mastering Object Oriented Python , 2014, Steven F. Lott , Pact Publishing

Prerequisites and Future Relevance

	Pre-requisites	Future Relevance			
Unit I	Sem. I	Sem. III	Sem. IV	Sem. V	Sem. VI
Object Oriented Methodology and Principle of OOPS	Imperative Programming	Python and Data Structures	Java	Project	Project

Unit I

What is Procedure Oriented Programming Approach?

In the procedure oriented approach, the problem is viewed as the sequence of things to be done such as reading, calculating and printing such as COBOL, fortran and c. The primary focus is on functions. A typical structure for procedural programming is shown in fig.1.1. The technique of hierarchical decomposition has been used to specify the tasks to be completed for solving a problem.

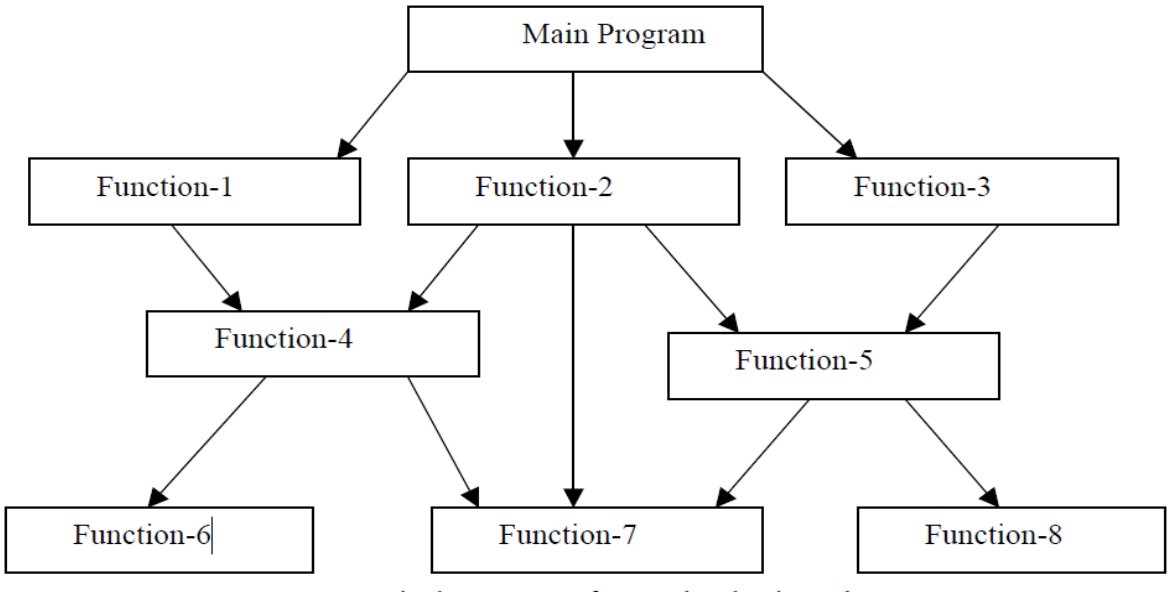


Fig. 1.1 Typical structure of procedural oriented programs

Procedure oriented programming basically consists of writing a list of instructions for the computer to follow, and organizing these instructions into groups known as *functions*. We normally use flowcharts to organize these actions and represent the flow of control from one action to another.

In a multi-function program, many important data items are placed as global so that they may be accessed by all the functions. Each function may have its own local data. Global data are more at risk to an unintentional change by a function. In a large program it is very difficult to identify what data is used by which function. In case we need to revise an external data structure, we also need to revise all functions that access the data. This provides an opportunity for bugs to creep in.

Another serious drawback with the procedural approach is that we do not model real world problems very well. This is because functions are action-oriented and do not really corresponding to the element of the problem.

Some Characteristics exhibited by procedure-oriented programming are:

- Emphasis is on doing things (algorithms).
- Large programs are divided into smaller programs known 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.

Object Oriented Methodology:

Introduction:

Object-Oriented Programming (OOP) is the term used to describe a programming approach based on **objects** and **classes**. The object-oriented paradigm allows us to organise software application as a collection of objects that consist of both data and behaviour. This is in contrast to conventional functional programming practice that only loosely connects data and behaviour.

Since 1980s the word 'object' has appeared in relation to programming languages, with almost all languages developed since 1990 having object-oriented features. It is widely accepted that object-oriented programming is the most important and powerful way of creating software applications.

The object-oriented programming approach encourages:

- Modularisation: where the application can be decomposed into modules.
- Software re-use: where an application can be composed from existing and new modules.

An object-oriented programming language generally supports seven main features:

- Objects
- Classes
- Data Abstraction and Data Encapsulation:
- Inheritance:
- Polymorphism:
- Dynamic Binding:
- Message Passing

Advantages and Disadvantages of Procedure Oriented Languages:

Advantages:

Simplicity: With procedural/imperative programming, you use conditionals and loops and functions and you avoid the complexity of OOP. No need to think about classes and inheritance and polymorphism and so on.

Less Storage: Another advantage is the reduced amount of storage during runtime. Once you start overloading methods and using interfaces, your code will need extra storage to track virtual method tables and other kinds of mapping. This needs to be stored somewhere within your project.

Disadvantages:

Difficult to visualize a very large program as a single concept: Programmers realize the limitations of POP when POP programs were used at a very large scale. Lengthy POP programs were used at a very large scale. Lengthy POP programs lose a ‘Global View’ and become very difficult to visualize as a single concept.

Emphasis is on doing things (algorithms): POP doesn't provide any mechanism for data abstraction to hide the data which are not required. It doesn't provide any mechanism to re-use the shared information that is available to all.

Data are not secured: Data can be accessed by any function, if it is declared globally. Hence it will be available in all the functions of that program. Any function can change the value of that variable.

What is Object Oriented?

Object Oriented Programming (OOP) is an approach to program organization and development that attempts to eliminate some of the pitfalls of conventional programming methods by incorporating the best of structured programming features with several powerful new concepts. It is a new way of organizing and developing programs and has nothing to do with any particular language. However, not all languages are suitable to implement the OOP concepts easily.

Some of the striking features of object-oriented programming are:

- Emphasis is on data rather than procedure.
- Programs are divided into what are known as 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 by external functions.
- New data and functions can be easily added whenever necessary.
- Follows bottom-up approach in program design.

The external interfaces are implemented by providing a set of methods (functions), each of which accepts and responds to a particular kind of messages as shown in the following figure 1.2.

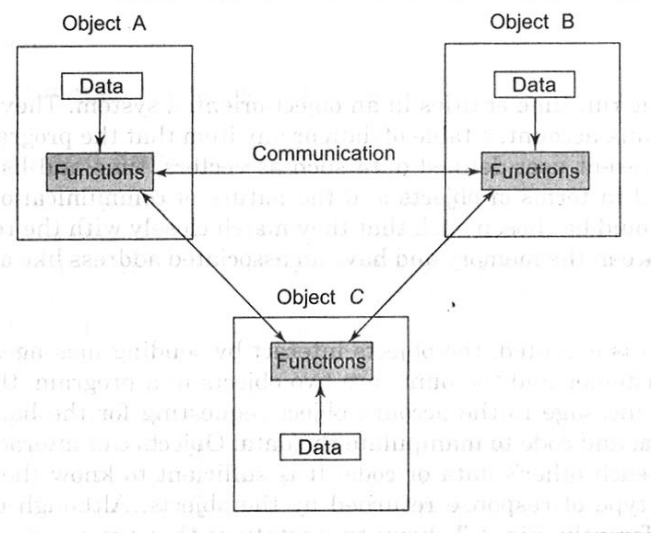


Fig 1.2 **:** Organization of data and functions in OOP.

What is Object Oriented Development?

In the structured programming languages, the data is defined as global and accessible to all the functions of a program without any restriction. It has reduces data security and integrity, since the entire data is

available to all functions and any function can change any data without any restrictions as shown in the following figure 1.3.

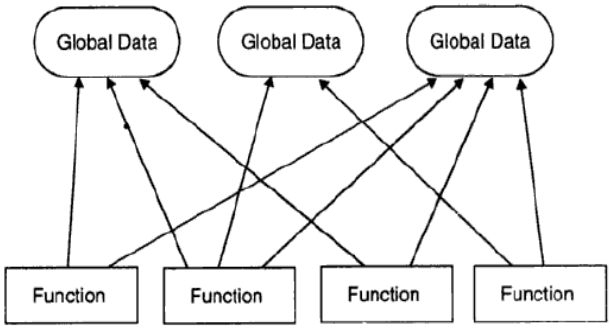


Fig 1.3: Function oriented paradigm

Unlike the traditional methodology, Object Oriented programming emphasizes on the data rather than the algorithm. In OOP, data is encapsulated with the associated functions that operate on it. And this capsule is called an object. In the OO approach, the problem is divided into objects, whereas in FO approach, the problem is divided into functions.

Unlike traditional languages, OO languages allow localization of data and code and restrict other objects from referring to its local region. OOP is centered around the concepts of objects, encapsulations, abstract data types, inheritance, polymorphism, message-based communication, etc. An OO language views the data and its associated set of functions as an object and treats this combination as a single entity. Thus, an object is visualized as a combination of data and functions which manipulate them.

During the execution of a program, the objects interact with each other by sending message and receiving responses. For instance, in a program to perform withdrawals from an account, a customer object can send a withdrawal message to a bank account object. An object communicating with other objects need not be aware of the internal working of the objects with which it interacts. This situation is analogous to operating a television receiver, a computer or an automobile; where one need not to know the internal operations since these machines provide the user with some system controls that hide the complexity of internal structure and working. Likewise an object can be manipulated through an interface that responds to a few messages. The object's internal structure is totally hidden from the user and this property is called data/information hiding or data encapsulation.

The external interfaces are implemented by providing a set of methods (functions), each of which accepts and responds to a particular kind of messages as shown in the following figure 1.4.

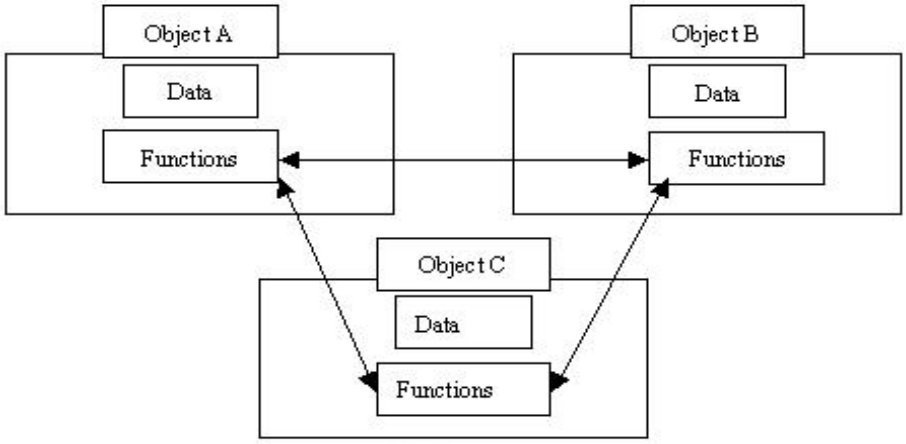


Fig. 1.4: Object oriented paradigm

The methods defined in an object's class are the same for all objects belonging to that class, but the data is unique for each object.

Object Oriented Themes:

Objects, as opposed to *structures*, incorporate both data and behavior. But there is more to that. Several themes underlie object-oriented technology. These are :

- **Abstraction**

  Abstraction consists of focusing on the essential, inherent aspects of an entity and ignoring its accidental aspects.

- **Classes and instances**

  Objects are instances of classes. A class is a definitive Description of a group of objects with similar properties and behaviors. Classes are abstract, objects are concrete. Objects are aware of their class identity.

- **Encapsulation**

  Encapsulation (also *information hiding*) consists of separating the external aspects of an object, which are accessible to other objects, from the internal details of the object, which are hidden from other objects.

  Objects have an *outside* (how they are seen or interact) and an *inside* (what they are.)

- **Polymorphism**

  Polymorphism is the kindred to the incorporation of data and behavior. From the outside, the same operation Solve or Draw applies to various objects (solvable or, respectively, drawable). The object user is not burdened with the implementation - the inside - of the object. The object knows how to do the job.

- **Inheritance**

  Inheritance is a mechanism for sharing similarities among classes while preserving their differences.

In the OO terminology, the individual pieces of data are known as *attributes* while the operations that objects carry out are known as *methods*.

Pluggability stems from the concepts of abstraction and encapsulation. The programming libraries on the market are nowadays *class libraries*. Classes know how to create their instances - objects. If one wants to modify class behaviour a little, there is inheritance and polymorphism. Of course, one can always design one's own classes from scratch.

How then information may know about itself and me? Here's one possibility. Implemented as an instance of a class Info, for example, information will know everything about itself. An object of the ComputerUser class will reside on my machine. The "outside" of the class - the public interface - will

be queried by the Info object for the necessary particulars. Authentication and protection will be built into the ComputerUser class.

Object oriented technology stresses specifying first what an object is, then how is it used. Implementation is the very last thing. At this point, I am absolutely unconcerned how the classes will be implemented.

Benefits of OOPS:

OOP offers several benefits to both the program designer and the user. Object-Orientation contributes to the solution of many problems associated with the development and quality of software products. The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost. The principal advantages are:

- Through inheritance, we can eliminate redundant code extend the use of existing
- Classes.
- We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.
- The principle of data hiding helps the programmer to build secure program that cannot be invaded by code in other parts of a programs.
- It is possible to have multiple instances of an object to co-exist without any interference.
- It is possible to map object in the problem domain to those in the program.
- It is easy to partition the work in a project based on objects.
- The data-centered design approach enables us to capture more detail of a model can implemental form.
- Object-oriented system can be easily upgraded from small to large system.
- Message passing techniques for communication between objects makes to interface descriptions with external systems much simpler.
- Software complexity can be easily managed.

While it is possible to incorporate all these features in an object-oriented system, their importance depends on the type of the project and the preference of the programmer. There are a number of issues that need to be tackled to reap some of the benefits stated above. For instance, object libraries must be available for reuse. The technology is still developing and current product may be superseded quickly. Strict controls and protocols need to be developed if reuse is not to be compromised.

Application of OOPS:

Some of the application areas where OOP has been used to develop software are listed below.

• **Simulations and Modeling:** Simulation is the technique of representing the real world entities with the help of a computer program. Simula-67 and Smalltalk are two object-oriented languages are designed for making simulations.

• **User-interface design:** Another popular application of OOP has been in the area of designing graphical user interfaces such as Windows. C++ is mainly used for developing user-interfaces.

• **Developing computer games:** OOP is also used for developing computer games such as Diablo, Startcraft and Warcraft **III.** These games offer virtual reality environments in which a number of objects interact with each other in complex ways to give the desired result.

• **Scripting:** In recent years, OOP has also been used for developing HTML, XHTML and XML documents for the Internet. Python, Ruby and Java are the scripting languages based on object-oriented principles which are used for scripting.

• **Object Databases:** These days OOP concepts have also been introduced in database systems to develop a new DBMS named object databases. These databases store the data directly in the form of objects. However, these databases are not as popular as the traditional RDBMS.

Some other areas of applications include office automation systems, decision support systems, Artificial Intelligence (AI) and expert systems, Neural networks and parallel programming, and Computer-Aided Design (CAD) systems.

Principles of OOPS:

OOPS Paradigm:

The major motivating factor in the invention of object-oriented approach is to remove some of the weaknesses encountered in the procedural approach. OOP treats data as a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the function that operate on it, and protects it from accidental modification from outside function. OOP allows decomposition of a problem into a number of entities called objects and then builds data and function around these objects. The organization of data and function in object-oriented programs is shown in fig.1.5. The data of an object can be accessed only by the function associated with that object.

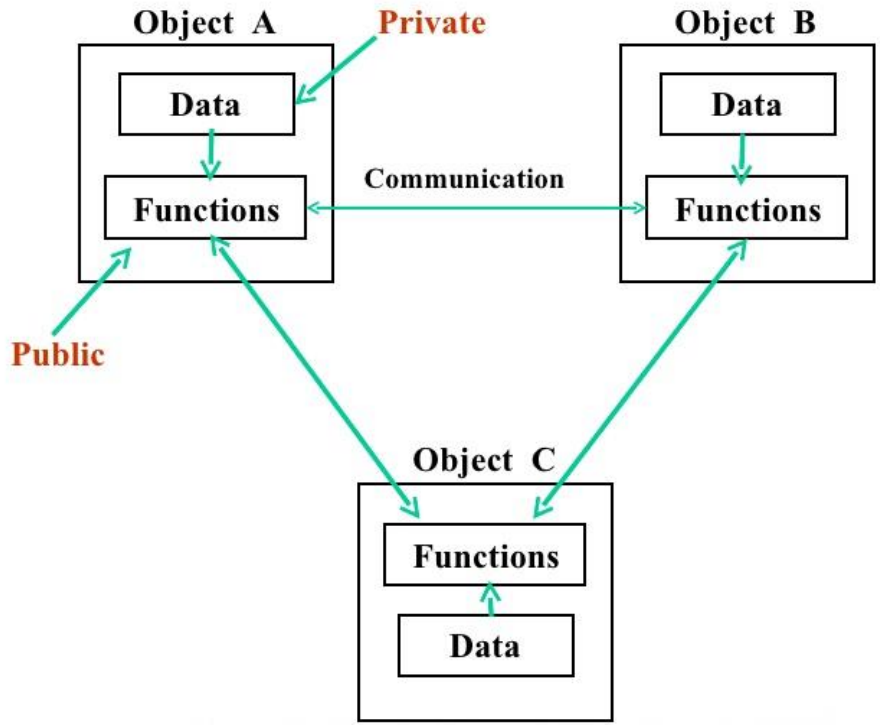


Fig. 1.5 The organization of data and function in object-oriented programs

Some of the features of object oriented programming are:

- Emphasis is on data rather than procedure.
- Programs are divided into what are known as objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are ties together in the data structure.
- Data is hidden and cannot be accessed by external function.

- Objects may communicate with each other through function.
- New data and functions can be easily added whenever necessary.
- Follows bottom up approach in program design.

Object-oriented programming is the most recent concept among programming paradigms and still means different things to different people.

Basic Concepts of OOPS:

It is necessary to understand some of the concepts used extensively in object-oriented programming. These include:

- Objects
- Classes
- Data abstraction and encapsulation
- Inheritance
- Polymorphism
- Dynamic binding
- Message passing

We shall discuss these concepts in some detail in this section.

Objects

Objects are the basic run time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. They may also represent user-defined data such as vectors, time and lists. Programming problem is analyzed in term of objects and the nature of communication between them. Program objects should be chosen such that they match closely with the real-world objects. Objects take up space in the memory and have an associated address like a record in Pascal, or a structure in c.

When a program is executed, the objects interact by sending messages to one another. For example, if "customer" and "account" are two objects in a program, then the customer object may send a message to the account object requesting for the bank balance. Each object contain data, and code to manipulate data. Objects can interact without having to know details of each other's data or code. It is a sufficient to know the type of message accepted, and the type of response returned by the objects. Although different author represent them differently fig 1.6 shows two notations that are popularly used in object-oriented analysis and design.

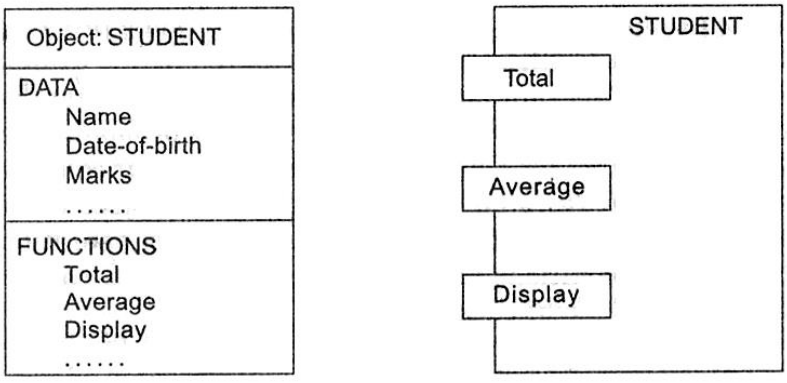


Fig. 1.6: Two ways of representing an object

Classes

We just mentioned that objects contain data, and code to manipulate that data. The entire set of data and code of an object can be made a user-defined data type with the help of class. In fact, objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created. A class is thus a

collection of objects of similar types. For example, Mango, Apple and orange are members of class fruit. Classes are user-defined that types and behave like the built-in types of a programming language. The syntax used to create an object is not different then the syntax used to create an integer object in C. If fruit has been defines as a class, then the statement

Fruit Mango;

creates an object **mango** belonging to the class **fruit.**

Data Abstraction and Encapsulation

The wrapping up of data and function into a single unit (called class) is known as *encapsulation.* Data and encapsulation is the most striking feature of a class. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. This insulation of the data from direct access by the program is called *data hiding or information hiding*.

Abstraction refers to the act of representing essential features without including the background details or explanation. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, wait, and cost, and function operate on these attributes. They encapsulate all the essential properties of the object that are to be created.

The attributes are sometimes called *data members* because they hold information. The functions that operate on these data are sometimes called *methods or member function*.

Inheritance

Inheritance is the process by which objects of one class acquired the properties of objects of another classes. It supports the concept of *hierarchical classification.* For example, the bird, 'robin' is a part of class 'flying bird' which is again a part of the class 'bird'. The principal behind this sort of division is that each derived class shares common characteristics with the class from which it is derived as illustrated in fig 1.7.

In OOP, the concept of inheritance provides the idea of *reusability*. This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined feature of both the classes. The real appeal and power of the inheritance mechanism is that it

Fig. 1.7 Property inheritances

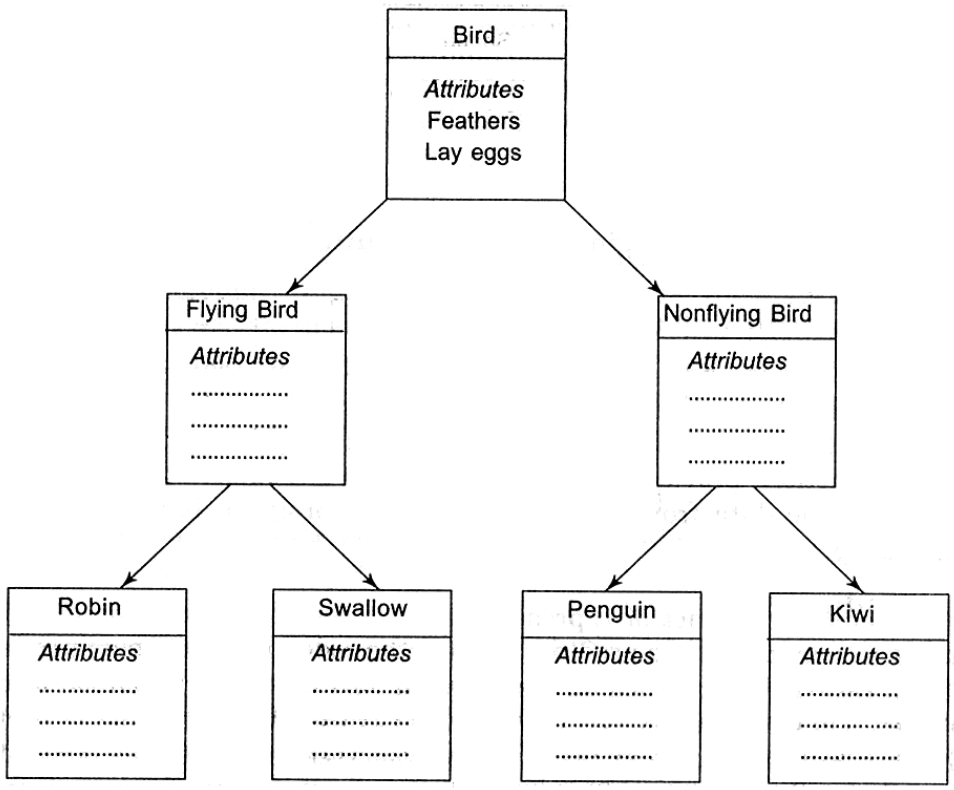


Fig. 1.7: Property inheritance

Allows the programmer to reuse a class i.e almost, but not exactly, what he wants, and to tailor the class in such a way that it does not introduced any undesirable side-effects into the rest of classes.

Polymorphism

Polymorphism is another important OOP concept. Polymorphism, a Greek term, means the ability to take more than on form. An operation may exhibit different behavior is different instances. The behavior depends upon the types of data used in the operation. For example, consider the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation. The process of making an operator to exhibit different behaviors in different instances is known as *operator overloading*.

Fig. 1.8 illustrates that a single function name can be used to handle different number and different types of argument. This is something similar to a particular word having several different meanings depending upon the context. Using a single function name to perform different type of task is known as *function overloading.*

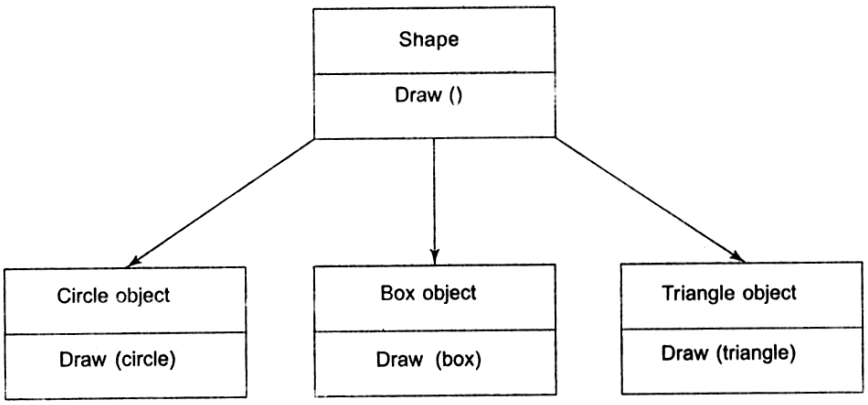


Fig. 1.8: Polymorphism

Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific action associated with each operation may differ. Polymorphism is extensively used in implementing inheritance.

Dynamic Binding

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic 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. A function call associated with a polymorphic reference depends on the dynamic type of that reference.

Consider the procedure "draw" in fig. 1.7. by inheritance, every object will have this procedure. Its algorithm is, however, unique to each object and so the draw procedure will be redefined in each class that defines the object. At run-time, the code matching the object under current reference will be called.

Message Passing

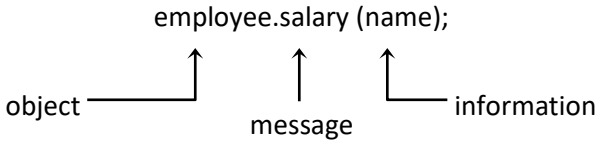
An object-oriented program consists of a set of objects that communicate with each other. The process of programming in an object-oriented language, involves the following basic steps:

1. Creating classes that define object and their behavior,
2. Creating objects from class definitions, and
3. Establishing communication among objects.

Objects communicate with one another by sending and receiving information much the same way as people pass messages to one another. 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:



Object has a life cycle. They can be created and destroyed. Communication with an object is feasible as long as it is alive.

Video Links :-

Source: https://www.youtube.com/watch?v=5SWKbS87p98

https://www.youtube.com/watch?v=xnh7ip5gpzc&list=PLfVsf4Bjg79DLA5K3GLbIwf3baNVFO2Lq

Summary

- The object-oriented programming approach encourages:

  - Modularisation: where the application can be decomposed into modules.
  - Software re-use: where an application can be composed from existing and new modules.

- Advantages of Procedure Oriented Languages: simplicity, less storage
  Disadvantages of Procedure Oriented Languages: Difficult to visualize a very large program as a single concept, Emphasis is on doing things (algorithms), Data are not secured

- Some of the striking features of object-oriented programming are:
  - Emphasis is on data rather than procedure.
  - Programs are divided into what are known as 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 by external functions.
  - New data and functions can be easily added whenever necessary.
  - Follows bottom-up approach in program design.

- In the structured programming languages, the data is defined as global and accessible to all the functions of a program without any restriction. It has reduces data security and integrity, since the entire data is available to all functions and any function can change any data without any restrictions

- An OO language views the Some of the application areas where OOP has been used to develop software are listed below.
  - Simulations and Modeling
  - User-interface design

- Developing computer games
- Scripting
- Object Databases

- It is necessary to understand some of the concepts used extensively in object-oriented programming. These include:
  - Objects
  - Classes
  - Data abstraction and encapsulation
  - Inheritance
  - Polymorphism
  - Dynamic binding
  - Message passing

Question Bank

1. What are the advantages and disadvantages of procedure oriented language?
2. Write a short note on Object Oriented Development.
3. List and explain benefits of object oriented programming.
4. List and explain applications of object oriented programming.
5. Explain Object Oriented Programming paradigm.
6. Describe the concept of classes and Objects in OOP languages
7. Describe the concept of object and Encapsulation in OOP languages.
8. Describe the concept of Inheritance and data abstraction in OOP languages.
9. Describe the concept of dynamic binding and message passing.
10. a) Write a program in C++ to find GCD, LCM of two numbers.
    b) Write a program in C++ to find factorial of a number.
    c) Write a program in C++ to find whether a number is even or odd.
11. a) Write a program in C++ to find prime numbers from 1 to 100.
    b) Write a program in C++ to display result sheet.

MCQs :-

1. **What will be the output of following program?**

```
#include<iostream.h>
void main()
{
float x;
x=(float)9/2;
cout<<x;
}
```

- 4.5
- 4.0
- 4
- 5

2. The term _____ means the ability to take many forms.

- Inheritance
- Polymorphism
- Member function
- Encapsulation

3. **If the variable count exceeds 100, a single statement that prints "Too many" is**

- if (count<100) cout << "Too many";
- if (count>100) cout >> "Too many";
- if (count>100) cout << "Too many";
- None of these.

4. The mechanism that binds code and data together and keeps them secure from outside world is known as

- Abstraction
- Inheritance
- Encapsulation
- Polymorphism

5. C++ was originally developed by

- Clocksin and Melish
- Donald E.Knuth
- Sir Richard Hadlee
- Bjarne Stroustrup

6. What is the output of the following code

```
char symbol[3]={'a','b','c'};
for (int index=0; index<3; index++)
    cout << symbol [index];
```

- a b c
- "abc"

- abc
- ‘abc’

7. The polymorphism can be characterized by the phrase

- One interface,multiple methods
- Multiple interfaces,one method
- One interface,one method
- None of the above