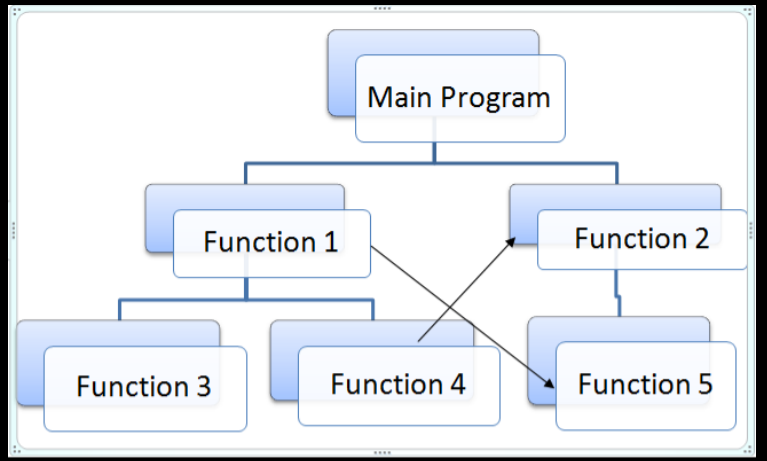
Unit - 1 Object Oriented Programming concepts

Introduction

Introduction

- CHANGE is one of the contrast characteristics in today's software industry.
- In programming many approaches have been tried like modular programming, top-down programming, bottom-up programming and structured programing to handle the increasing complexity, reliability and maintainability of program.
- Object-Oriented Programming is an approach to program organization and development, which attempts to eliminate some of the pitfalls of conventional programming methods.

Procedure Oriented Programming



Procedure-Oriented Programming

- In the procedure oriented approach, the problem is viewed as the sequence of things to be done such as reading, calculating and printing.
- The primary focus is on functions.

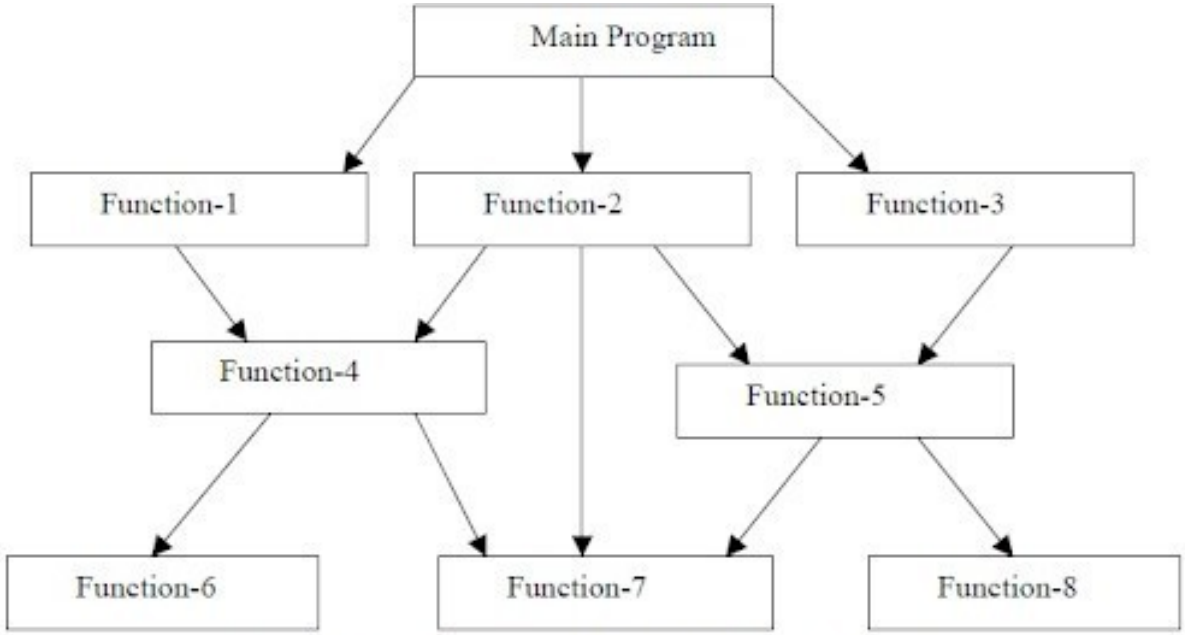


Fig. 1.2 Typical structure of procedural oriented programs

Procedure-Oriented Programming

- 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. 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.

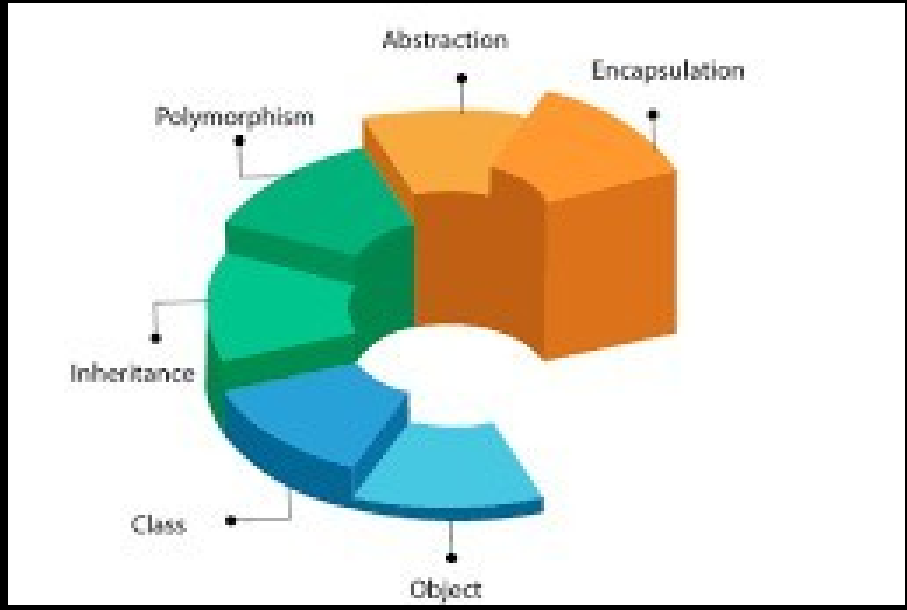
Procedure-Oriented Programming

Another serious concern 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:

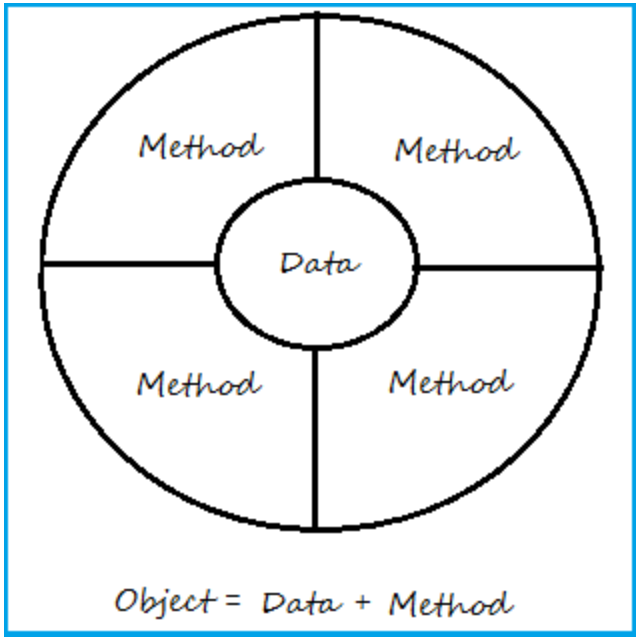
- Emphasis is on doing things (algorithms).
- Large programs are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.
- Employs top-down approach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

Object Oriented Paradigm



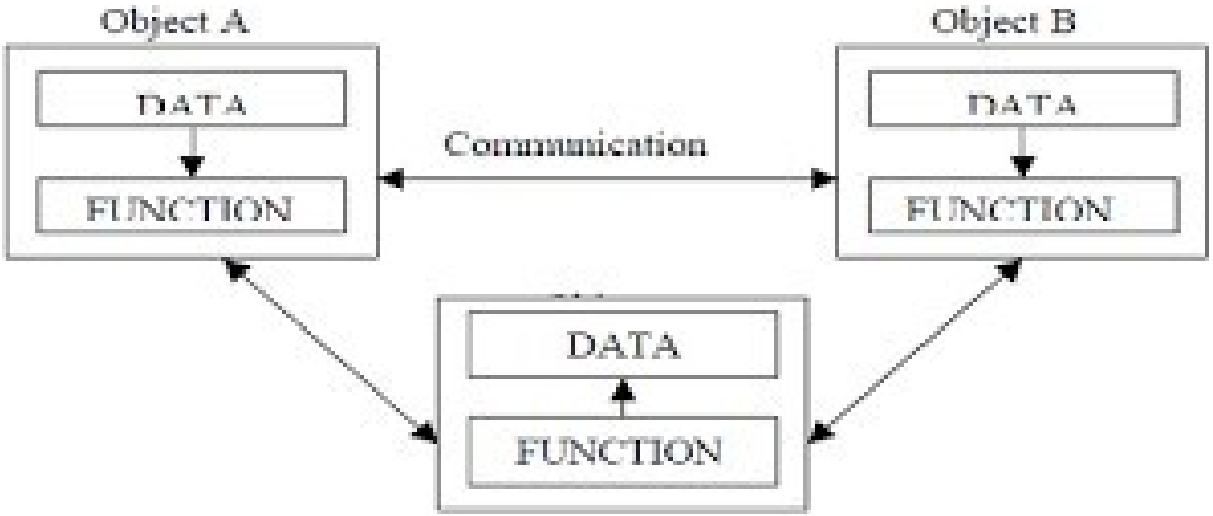
Object-Oriented Paradigm

- The major motivating factor in the invention of object-oriented approach is to remove some of the flaws 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.



Object-Oriented Paradigm

The data of an object can be accessed only by the function associated with that object. However, function of one object can access the function of other objects.



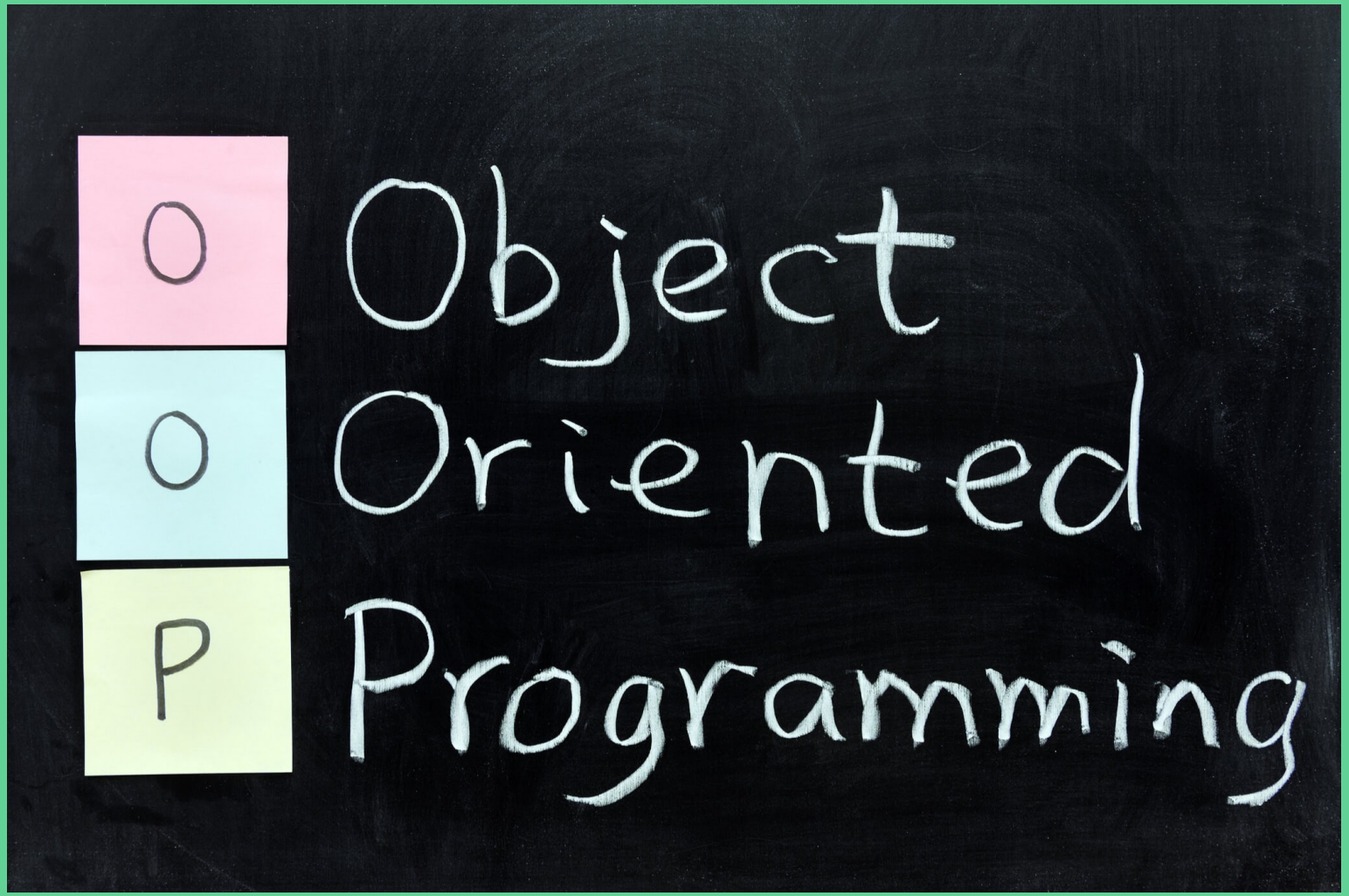
Organization of data and function in OOP

Object-Oriented Paradigm

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.


O
O
P
Object
Oriented
Programming

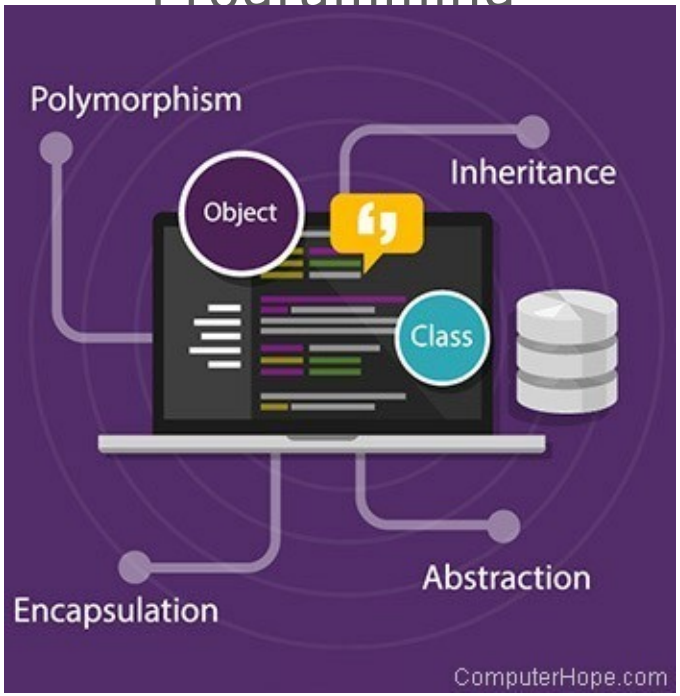
Object Oriented Programming

Object-oriented programming is an approach that provided 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.

This means that an object is considered to be a partitioned area of computer memory that stores data and a set of operations that can access the data.

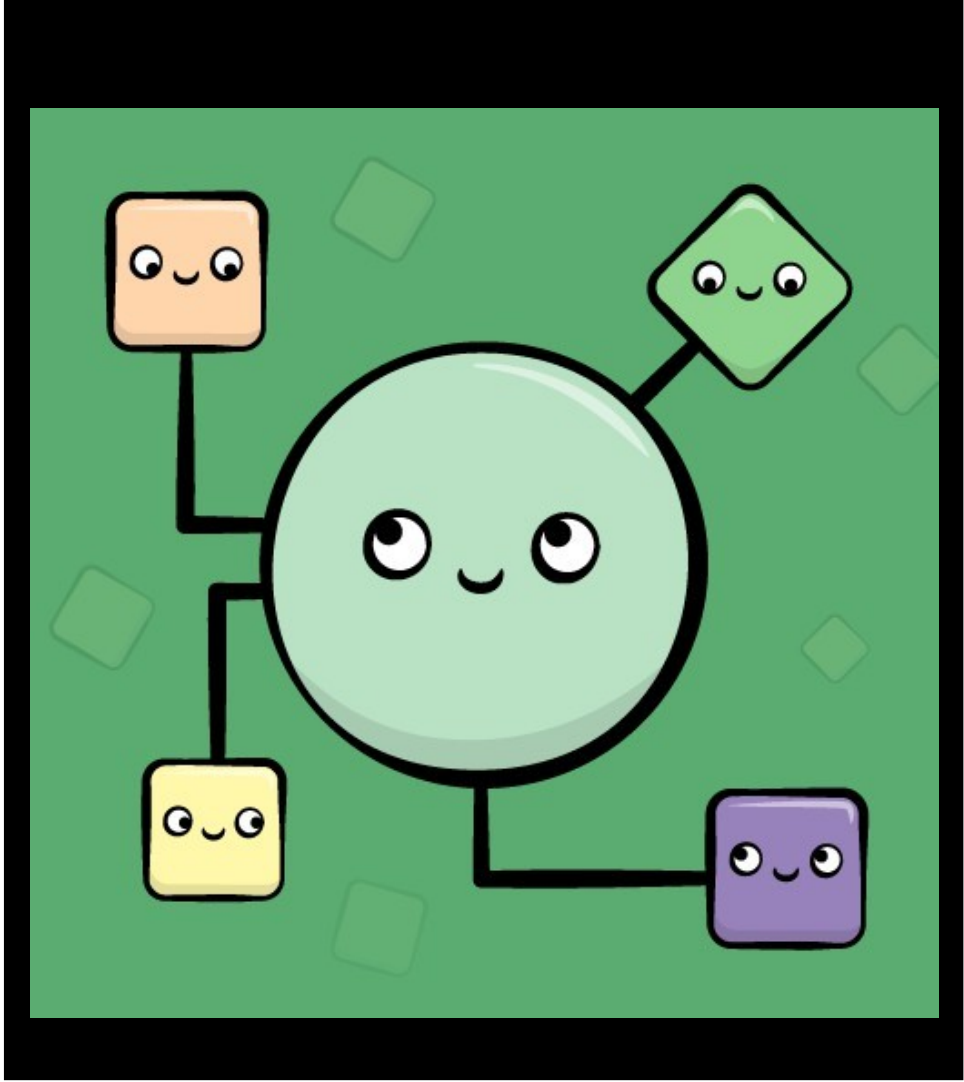
Basics Concepts

Object Oriented Programming



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

Objects



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.

Objects

- When a program is executed, the objects interact by sending messages to one another.
- For example, if “customer” and “account” are two object in a program, then the customer object may send a message to the 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.

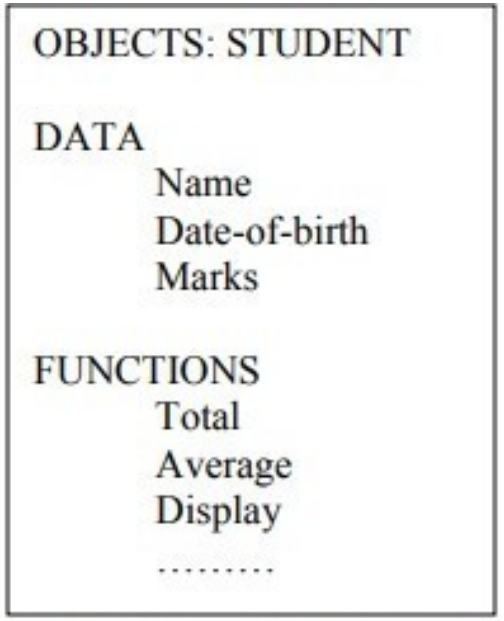


Fig. 1.5 representing an object

Objects

Object is a real world entity with specific features and behaviour. Objects in a program are generally used to model the real-world items.

- Object can be tangible like person, place, car, house and tree etc.
- It can be logical entity such as date of birth, job profile, bank account etc.

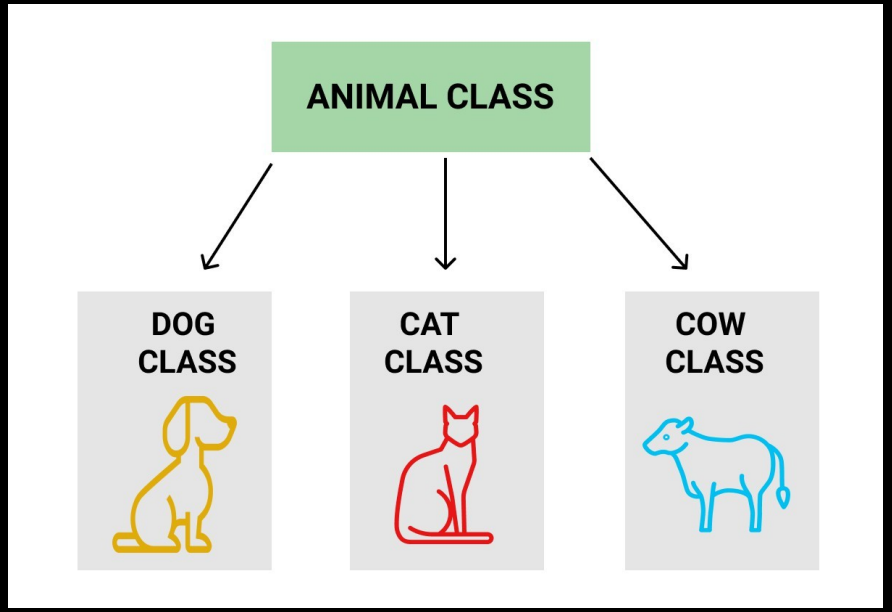
An object is defined with two major items

- Attributes of object, Examples name, colour, shape, size or other features
- Behaviour of object defined by operations or functions, Examples read, drive, display, update_balance etc.

Objects

Name	Attributes/ Data	Behaviour / Functions
Person	Name, age, DOB, phone	Walk(), sleep(), work(), work()
Car	Type, make, model, gear	Start(), stop(), setspeed(), setgear()
Time	Hours, min, sec	Set_time(), current_time(), get_time()

Classes



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 similar types.
- For examples, Mango, Apple and orange members of class fruit. Classes are user-defined that types and behave like the built-in types of a programming language.

Classes

The real world objects, such as television have many features and properties. For example every television is a complete unit in itself even if it is attached to some other device like DVD player it remains a television.

There is an interface like buttons and remote control to operate television. All televisions have some common properties and operations. Collection of similar objects is known as class. And instances of that class are called objects.

- Name of class – television
- Data – brand, model, type, volume_level, channel_number etc.
- Functions – switch_on(), switch_off(), change_channel(), set_volume() etc.
- Name of class – BankAccount
- Data –accountno,customername,balance,typeofaccount,mobnumber, debitcardstatus,
- Function – checkdetail(),deposit(),withdraw(),checkbalance(),changepin()

Classes

- Class is a collection of similar objects.
- Class defines all the properties (data and functions) of an object.
- Class provides a well-defined interface to use the functions of that class.
- A class must be complete and well-documented.
- The programming code of a class should be robust

Class	Object
user defined data type	variable of type class
group of similar objects	Instance of a class
Class defines properties i.e. data and function members	Objects hold actual values
Blue print of the object	Number of objects are created using class

Classes

Class – television with attributes as brand_name, model, size and colour.

Object1 - commonhall_tv with values as Sony, 2015, 40”, black

Object2- room_tv with values as Samsung, 2016, 32”, white

Object3 - kitchen_tv with values as Sony, 2014, 22”, blue

Data Abstraction and Encapsulation

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.

Data Abstraction and Encapsulation

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

Data Abstraction and Encapsulation

Abstraction means providing only the essential or relevant information and hiding the background or technical implementation details of something.

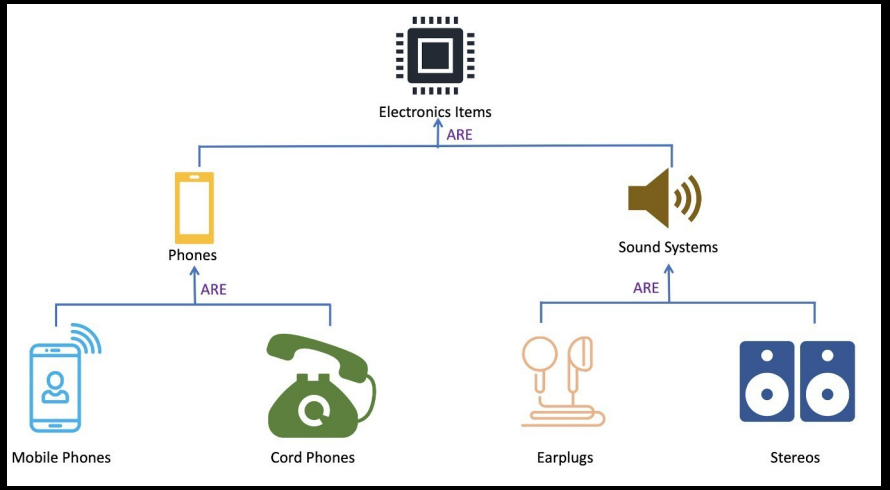
For example in the class TV

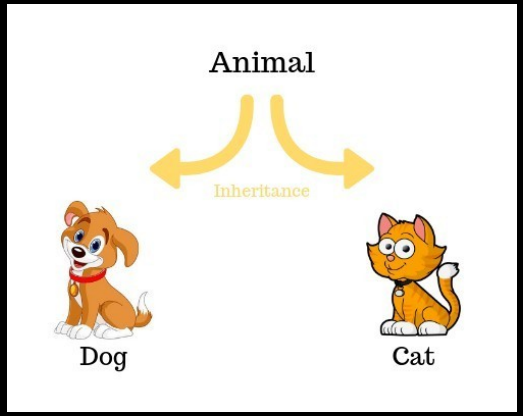
- Interface part is through remote buttons (i.e. visible part methods like Changechannel( ), Change-volume(), Power-on( ))
- Implementation part is mechanism behind these switches (i.e. hidden partmethods like Decode-signal( ), Display-picture( ))

Data Abstraction and Encapsulation

Abstraction	Encapsulation
Is about identifying necessary components required to build a system	Is about hiding the internal complexities of a system to make it user friendly
Abstraction takes place at design level as it focuses on what should be implemented in the system	Encapsulation happens at implementation level as it focuses on how it should be done or actually implemented
Abstraction is achieved through encapsulation	Encapsulation is achieved by hiding data and information in class

Inheritance

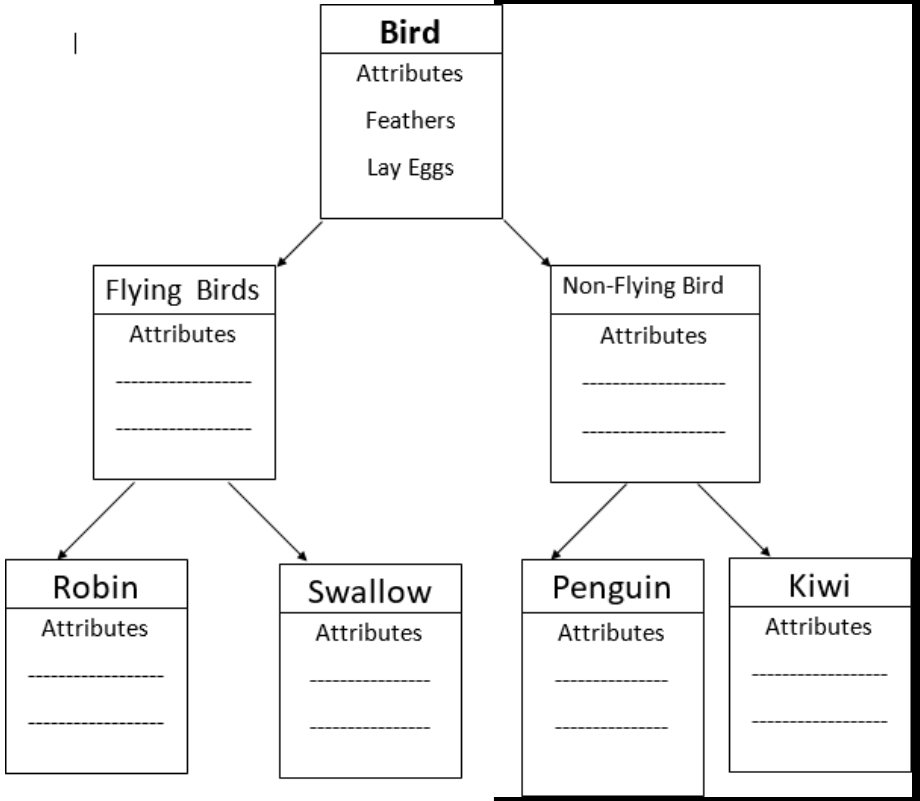




Inheritance

- 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.

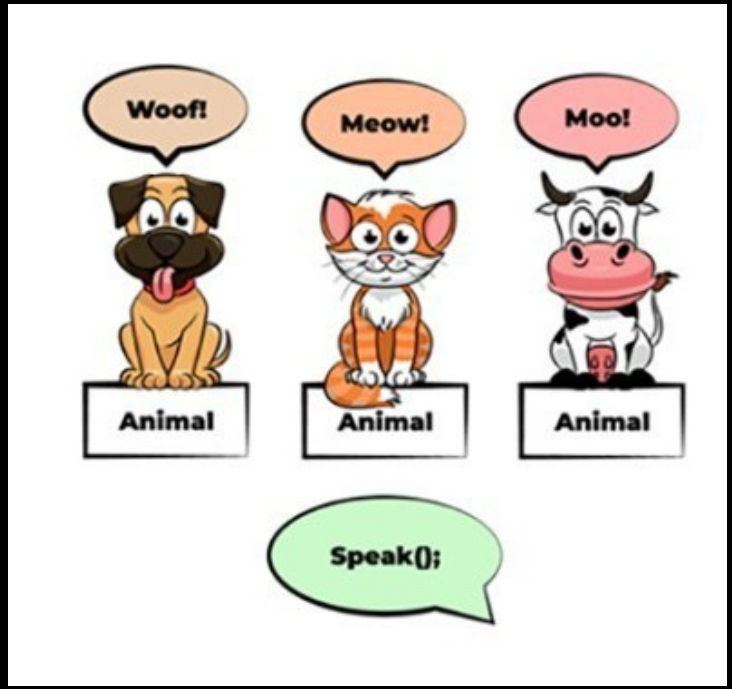
Inheritance



Inheritance

- 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 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

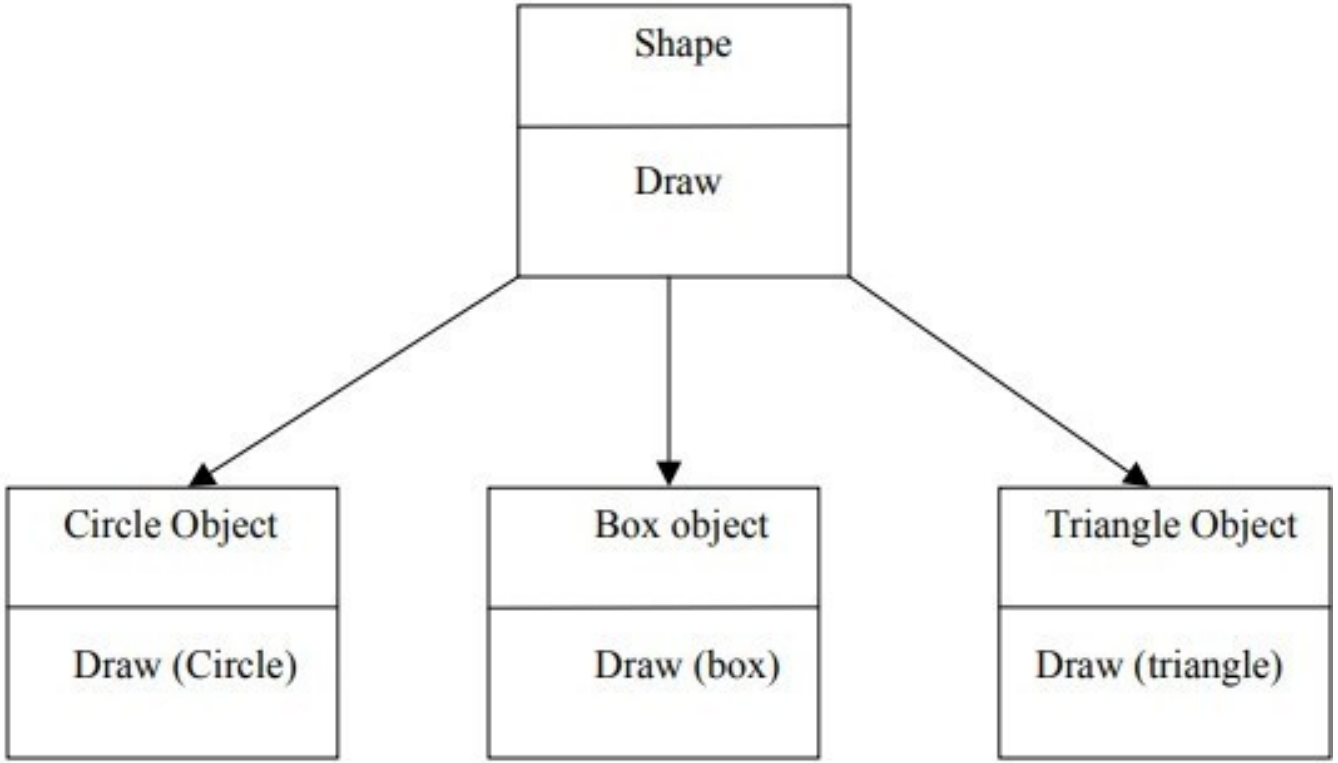


Polymorphism

- 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.

Polymorphism

Fig. 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.

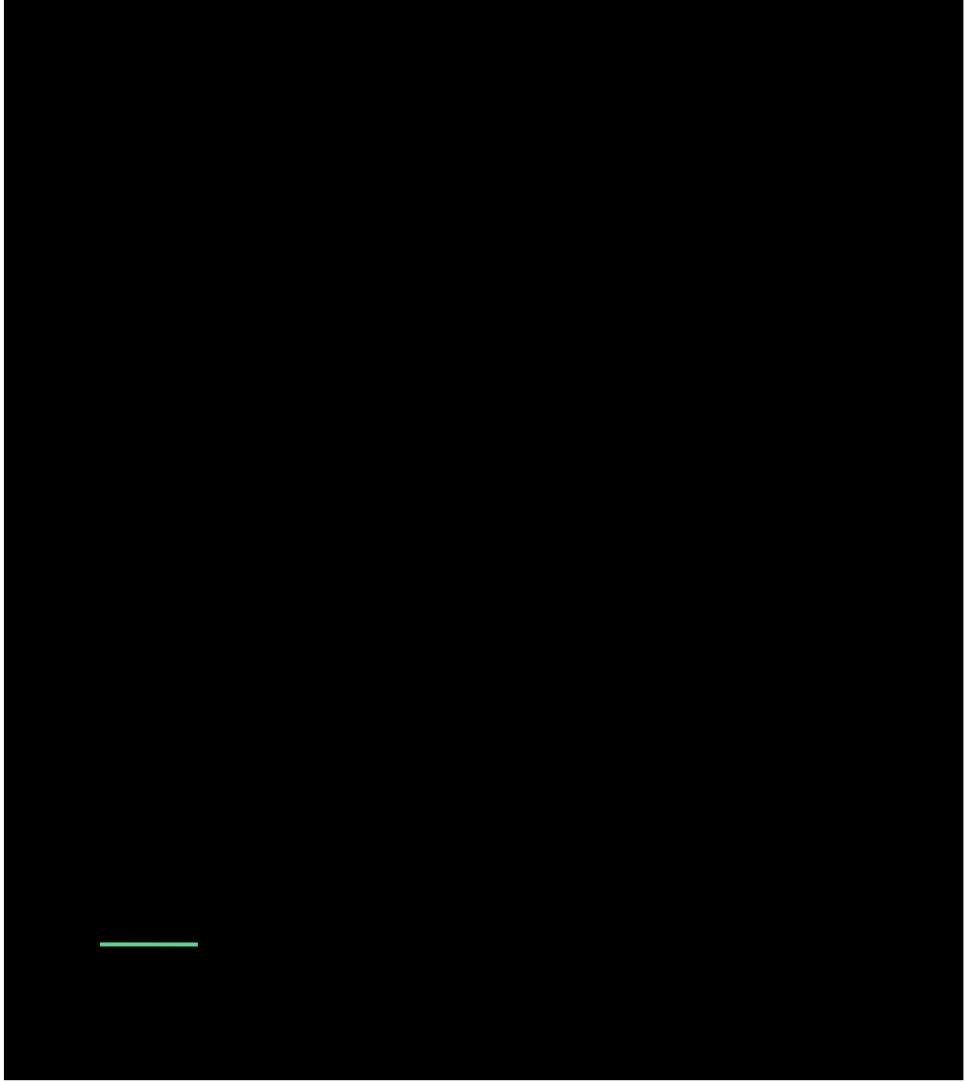


Polymorphism

- 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

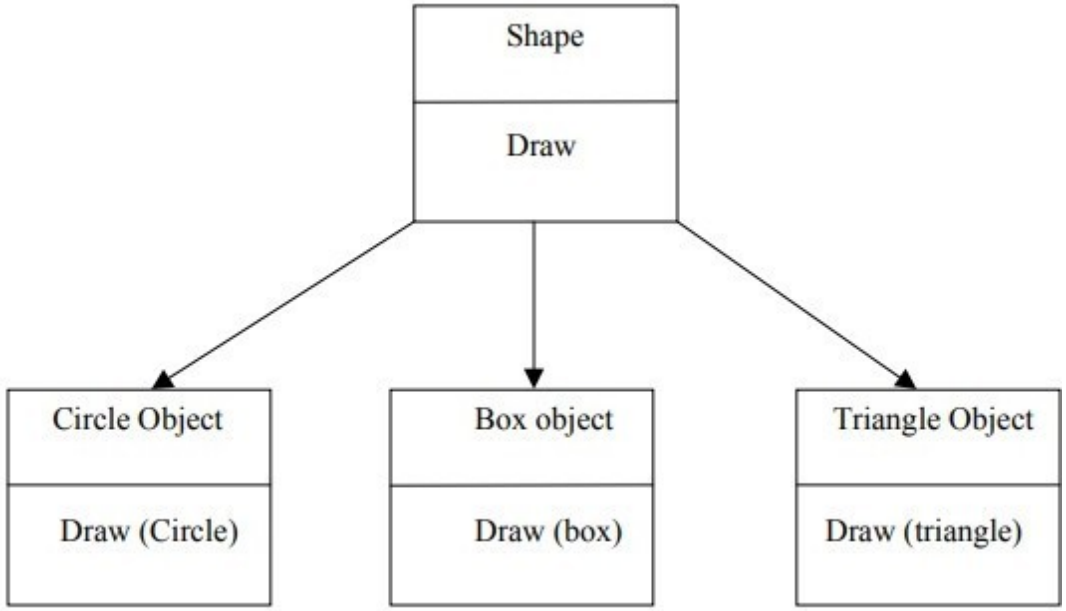


Dynamic Binding

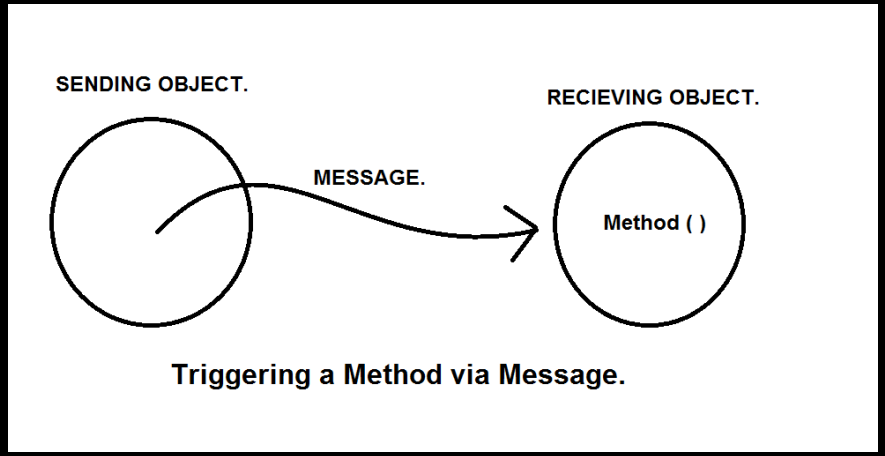
- Binding refers to the linking of a procedure call to the code to be executed in response to the call.
- Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time.
- It is associated with polymorphism and inheritance.
- A function call associated with a polymorphic reference depends on the dynamic type of that reference.

Dynamic Binding

- Consider the procedure "draw" in fig.
- 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



Triggering a Method via Message.

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,

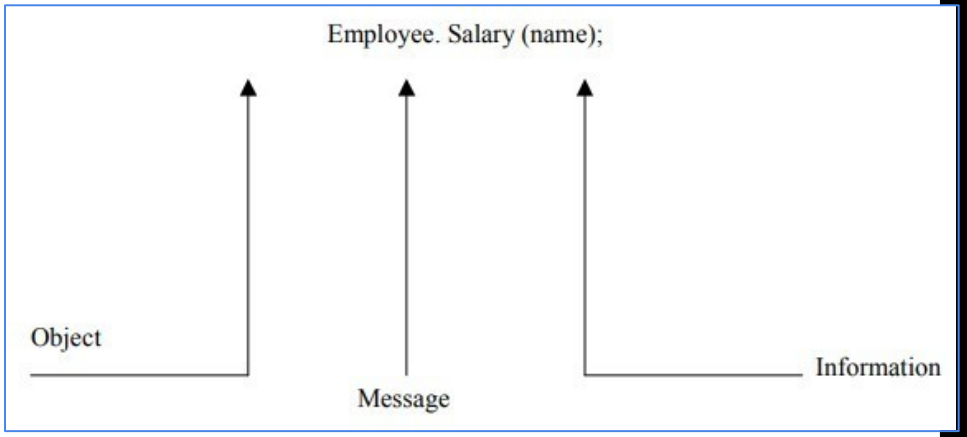
2. Creating objects from class definitions, and

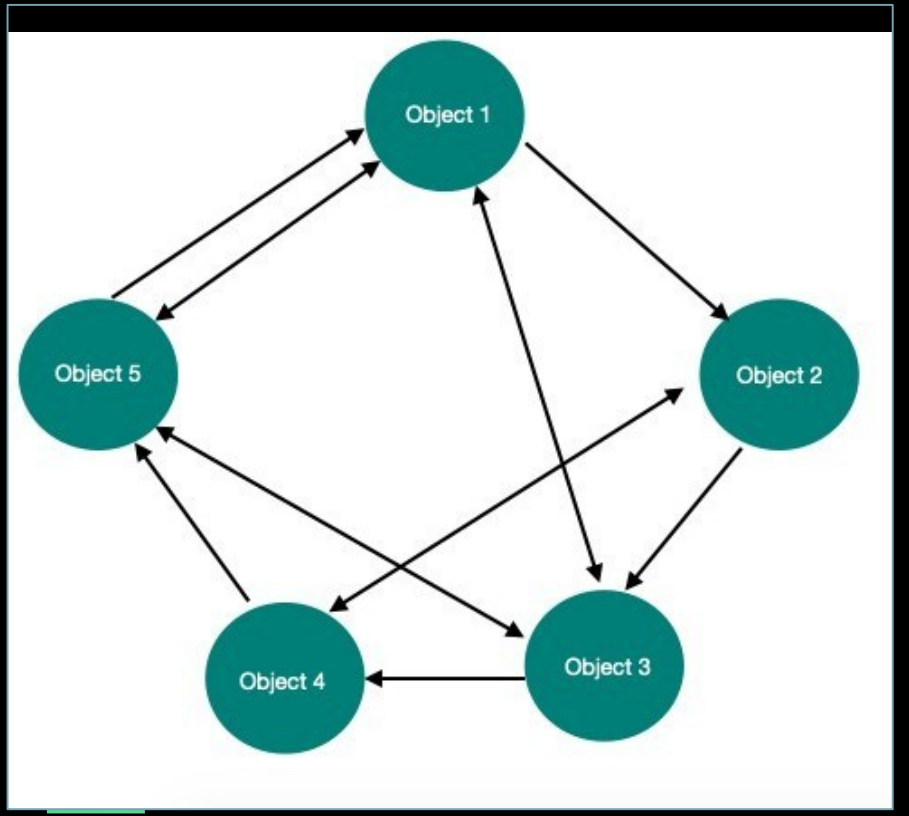
3. Establishing communication among objects.

Message Passing

- 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.

Message Passing





Benefits of OOP

Benefits of OOP

- 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 can not 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.

Benefits of OOP

- 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.

Benefits of OOP

- 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.

Benefits of OOP

- 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.

Object Oriented Languages

A language that is specially id designed to support the OOP concepts makes it easier to implement them.

The languages should support several of the OOP concepts to claim that they are object-oriented. Depending upon the features they support, they can be classified into the following two categories:

1. Object-based programming languages,
2. Object-oriented programming languages.

Object-based programming

Object-based programming is the style of programming that primarily supports encapsulation and object identity.

Major feature that are required for object based programming are:

- Data encapsulation
- Data hiding and access mechanisms
- Automatic initialization and clear-up of objects
- Operator overloading

Languages that support programming with objects are said to the objects-based programming languages. They do not support inheritance and dynamic binding. Ada is a typical object-based programming language.

Object-oriented programming

Object-oriented programming language incorporates all of object-based programming features along with two additional features, namely, inheritance and dynamic binding. Object-oriented programming can therefore be characterized by the following statements:

Object-based features + inheritance + dynamic binding

Application of OOP

Application of OOP

Hundreds of windowing systems have been developed, using the OOP techniques. Real-business system are often much more complex and contain many more objects with complicated attributes and method. OOP is useful in these types of application because it can simplify a complex problem. The promising areas of application of OOP include:

- Real-time system
- Simulation and modeling
- Object-oriented databases
- Hypertext, Hypermedia, and expertext
- AI and expert systems
- Neural networks and parallel programming
- Decision support and office automation systems
- CIM/CAM/CAD systems

Object-oriented technology is certainly going to change the way the software engineers think, analyze, design and implement future system.


THE
END