Object Oriented Programming- SE203

Content :

- Principles of object-oriented programming
- Beginning with C++
- Tokens, Expressions and Control Structures
- Functions in C++
- Classes and Objects
- Constructors and Destructors
- Operator Overloading and Type Conversions
- Inheritance: Extending Classes
- Pointers, Virtual Functions and Polymorphism
- Managing console I/O Operations
- Working With Files
- Templates
- Exception Handling

Chapter 1

Principles of Object Oriented Programming

Unit Structure

1.1 Software crisis

1.2 Software Evolution

1.3 POP (Procedure Oriented Programming)

1.4 OOP (Object Oriented Programming)

1.5 Basic concepts of OOP

- 1.5.1 Objects
- 1.5.2 Classes
- 1.5.3 Data Abstraction and Data Encapsulation
- 1.5.4 Inheritance
- 1.5.5 Polymorphism
- 1.5.6 Dynamic Binding
- 1.5.7 Message Passing

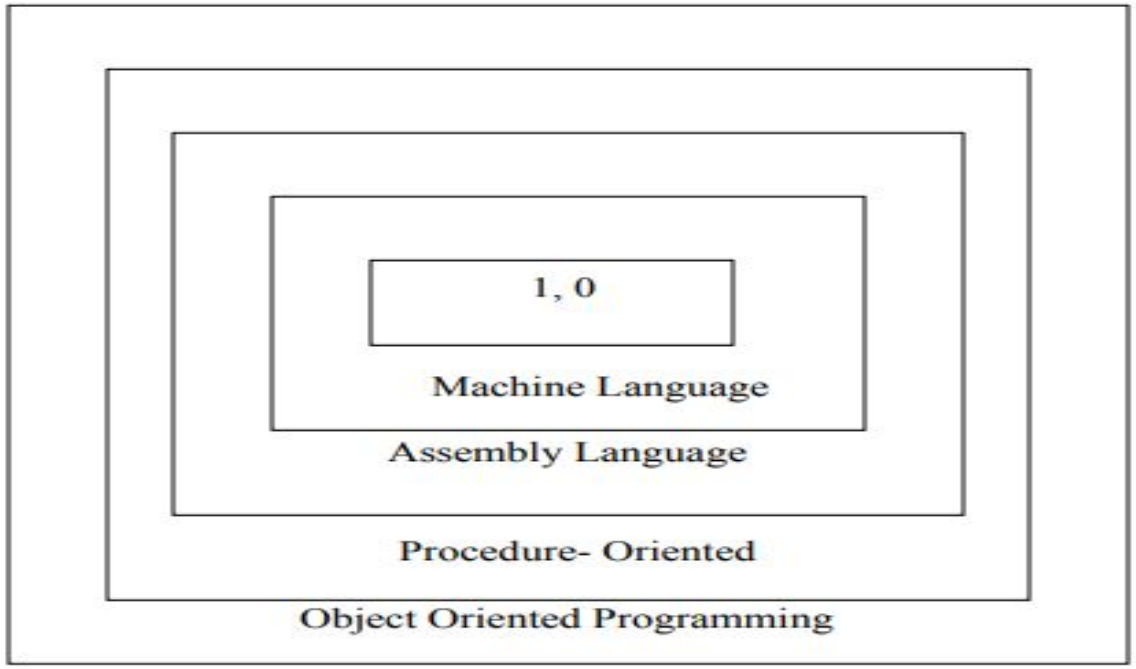
Software Crisis

- Developments in software technology continue to be dynamic.
- New tools and techniques are announced in quick succession.
- This has forced the software engineers and industry to continuously look for new approaches to software design and development, and they are becoming more and more critical in view of the increasing complexity of software systems as well as the highly competitive nature of the industry.
- These rapid advances appear to have created a situation of crisis within the industry.

- The following issued need to be addressed to face the crisis:
  - How to represent real-life entities of problems in system design?
  - How to design system with open interfaces?
  - How to ensure reusability and extensibility of modules?
  - How to develop modules that are tolerant of any changes in future?
  - How to improve software productivity and decrease software cost?
  - How to improve the quality of software?
  - How to manage time schedules?

Software Evolution

- Ernest Tello, A well known writer in the field of artificial intelligence, compared the evolution of software technology to the growth of the tree.
- Like a tree, the software evolution has had distinct phases “layers” of growth. These layers were building up one by one over the last five decades as shown in fig. 1.1, with each layer representing and improvement over the previous one.



- With the advent of languages such as c, structured programming became very popular and was the main technique of the 1980's.
- Structured programming was a powerful tool that enabled programmers to write moderately complex programs fairly easily.
- However, as the programs grew larger, even the structured approach failed to show the desired result in terms of bug-free, easy-to- maintain, and reusable programs.

- *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

Procedure Oriented Programming (POP)

- In POP approach, the problem is viewed as a 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.1.2.

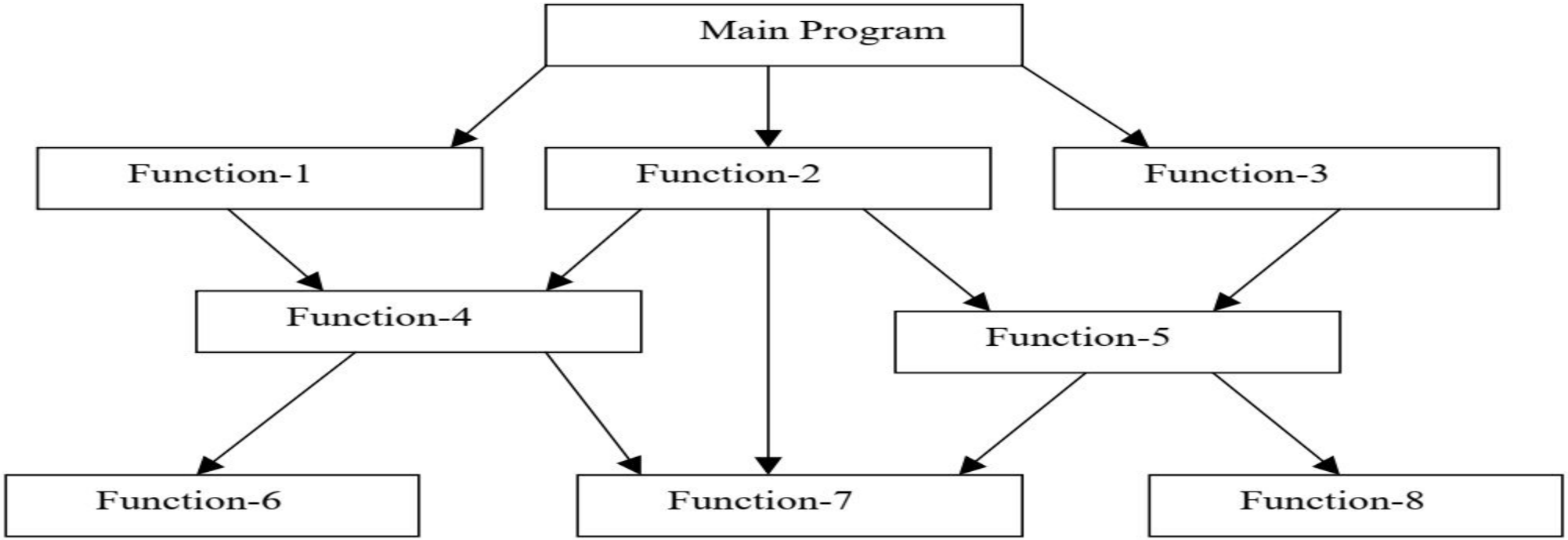
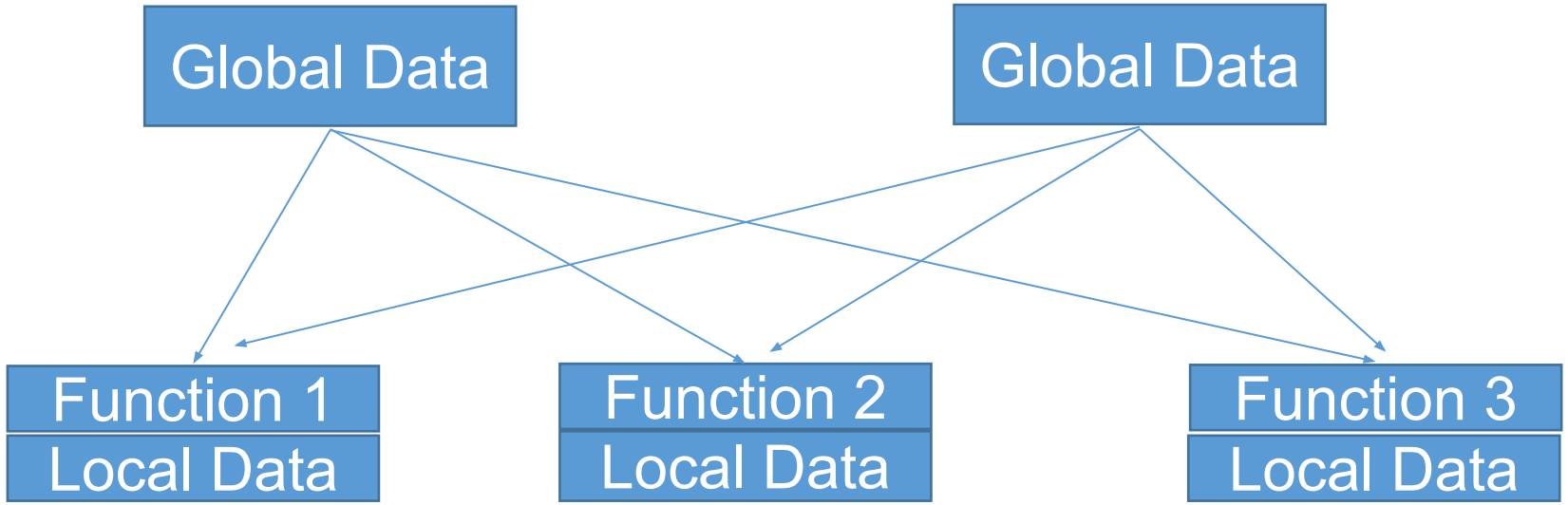


Fig. 1.2 Typical structure of procedural oriented programs

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.

Relationship of data and functions in POP

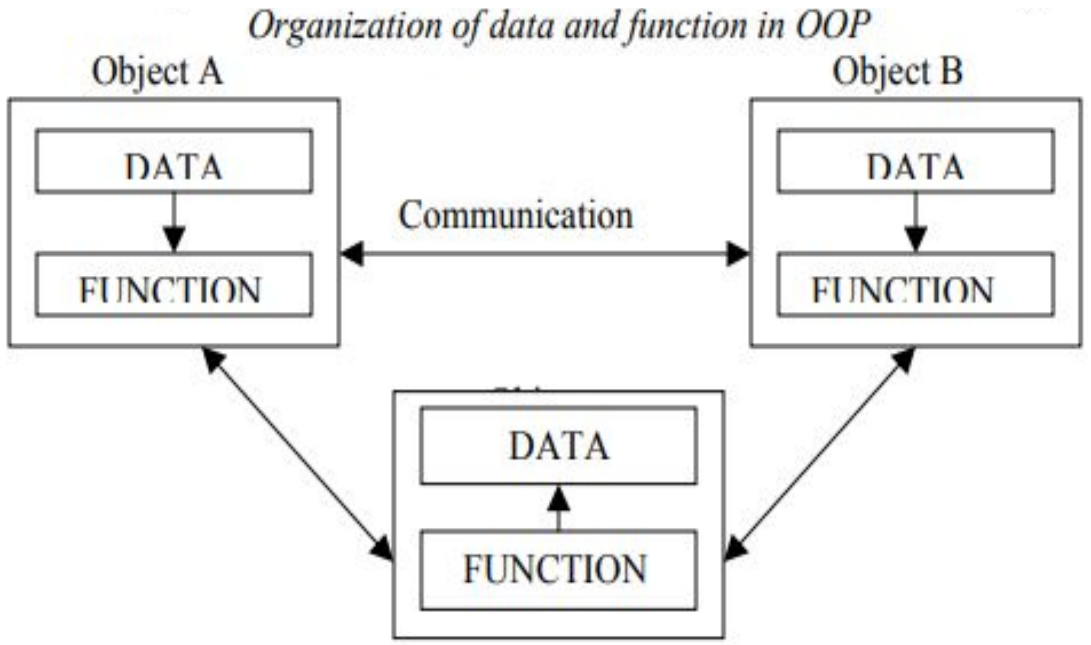


- In a multi-function program, many important data items are placed as global so that they may be accessed by all the functions. Each function may have its own local data.
- In a large program it is very difficult to identify what data is used by which function.
- In case we need to revise an external 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.
- Don’t have any proper way to hide the data, so it is less secure

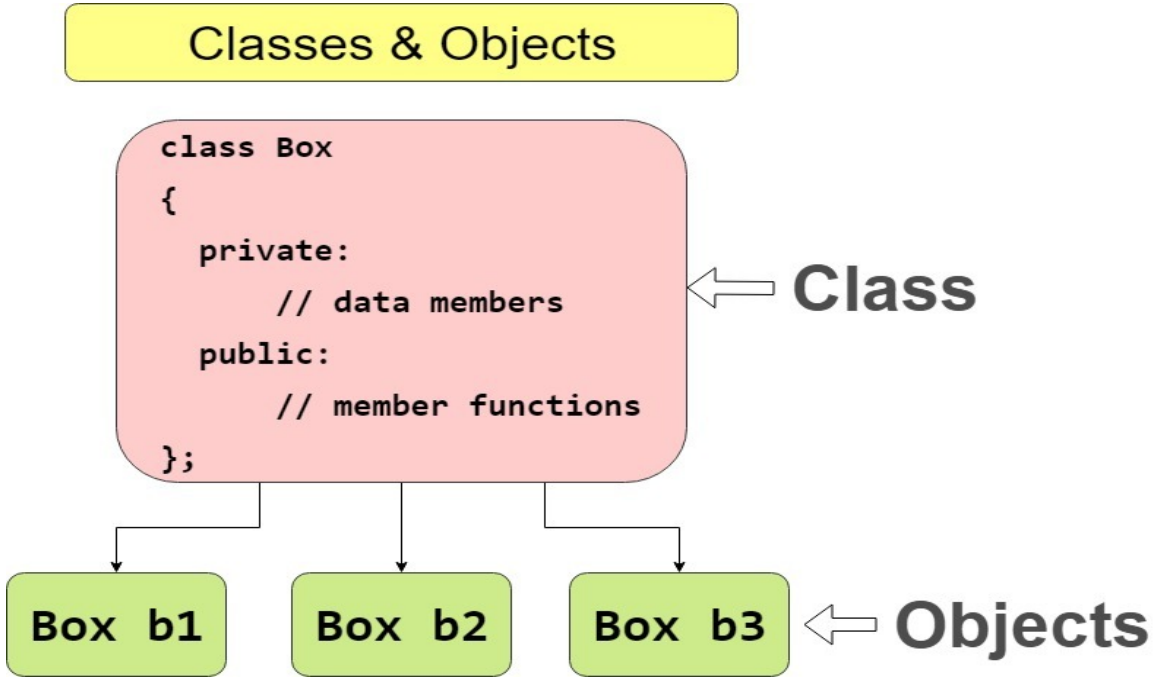
Object Oriented Programming

- OOP is introduced in order to overcome the flaws in procedural and traditional approach to programming as they were lacking in
  - Reusability of code
  - Maintainability of code
- Fundamental idea behind object-oriented language is to combine both data and the functions that operate on that data into a single unit.
- Such a unit is called an object.
-



- 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.



Features of object oriented programming are:

- Emphasis is on data rather than procedure.
- Programs are divided into entities known as objects.
- Objects may communicate with each other through function.
- New data and functions can be easily added whenever necessary.
- 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 function.
- Follows bottom up approach in program design.

Basic Concepts of Object Oriented Programming

1. Objects
2. Classes
3. Data abstraction
4. Data encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic binding
8. Message passing

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.

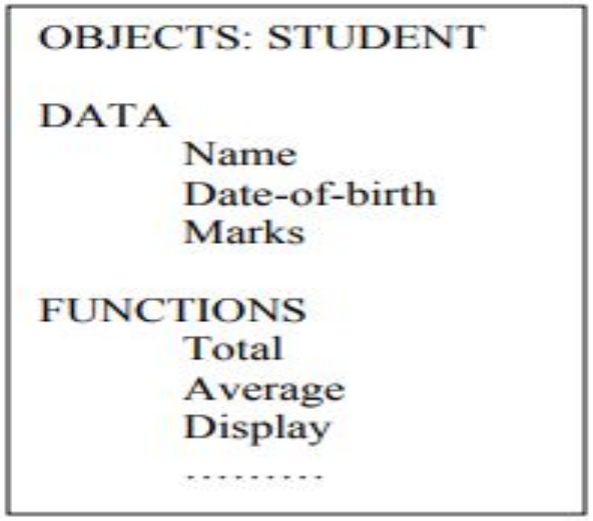


Fig. 1.5 representing an object

Classes

- A class is a collection of objects of similar types.
- The entire set of data and code of an object can be made a user-defined data type with the help of class.
- Once a class has been defined, we can create any number of objects belonging to that class.
- Syntax for creating object of the class is:

class-name object-name ;

- For examples, Mango, Apple and orange members of class fruit.
- If fruit has been defined as a class, then the statement

Fruit Mango;

Will create an object mango belonging to the class fruit.

Data Encapsulation

- The wrapping up of data members (variables) and member functions (methods) into a single unit (called class) is known as encapsulation.
- Data encapsulation is the first principle of object oriented programming.
- 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.
- Encapsulation means hiding the important features of a class which is not been needed to be exposed outside of a class and exposing only necessary things of a class.

- For example : Medical Store

Medicines Member variables

Chemist Member Methods

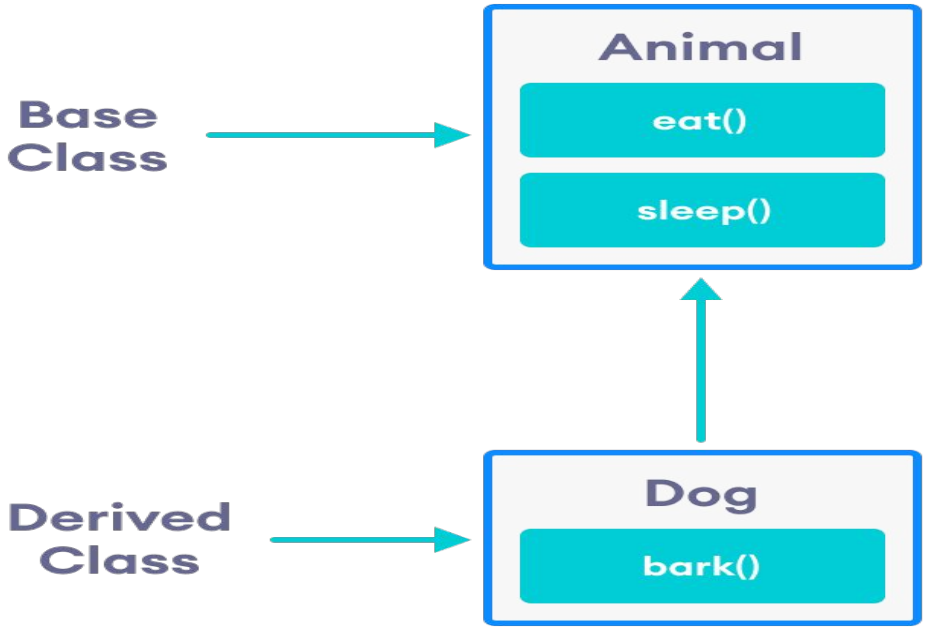
You External Application or piece of code

Data Abstraction

- 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.
- The exposed part of a class acts like Abstraction.

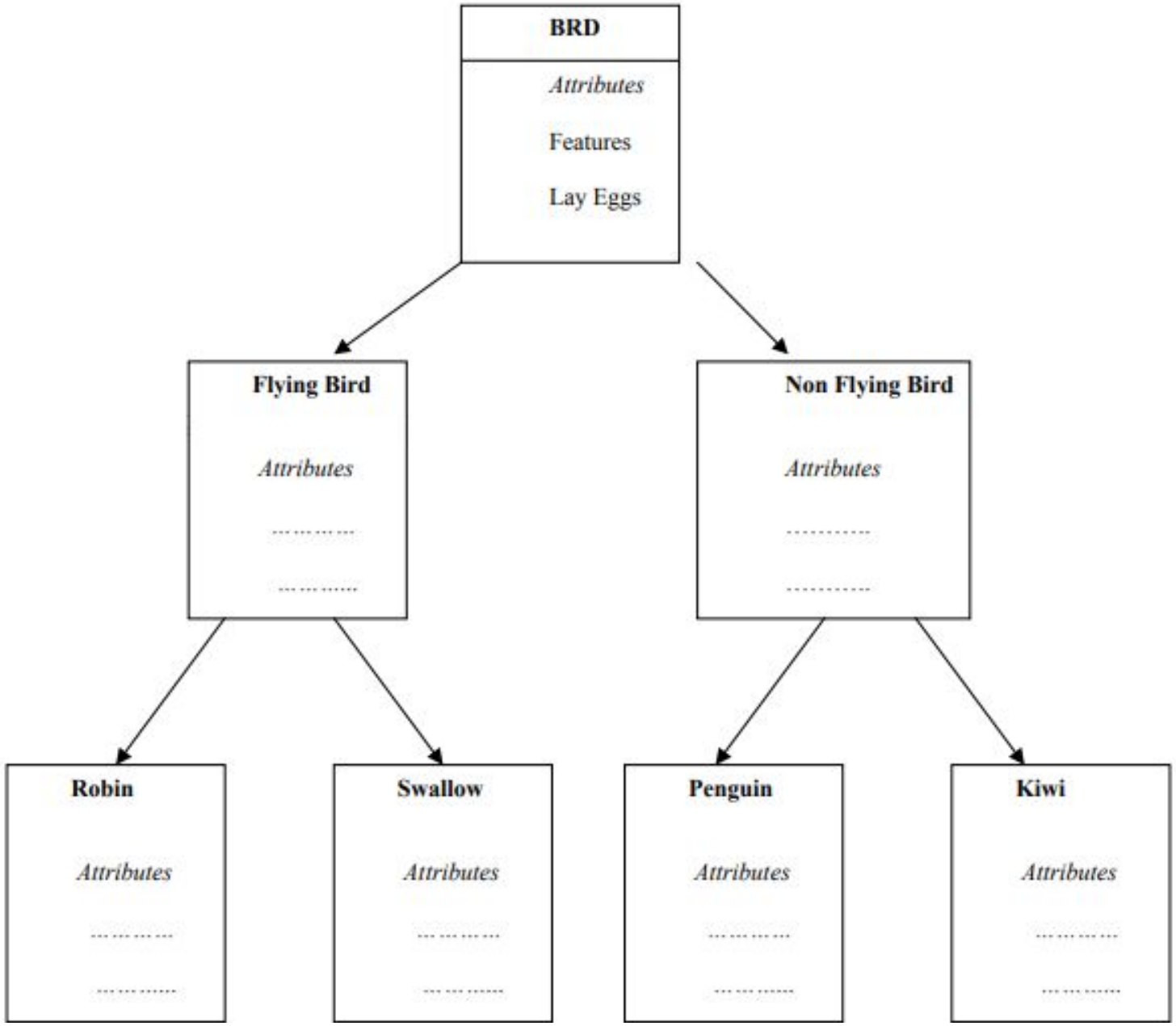
Inheritance

- Inheritance is the process by which objects of one class acquired the properties of objects of another classes.



- The process of deriving a new class from the existing one is called inheritance. The new class will have the combined feature of both the classes.
- The old class is known as base class or the parent class while the new class is known as derived class or sub class.

Fig. 1.6 Property inheritances



- 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 1.6.

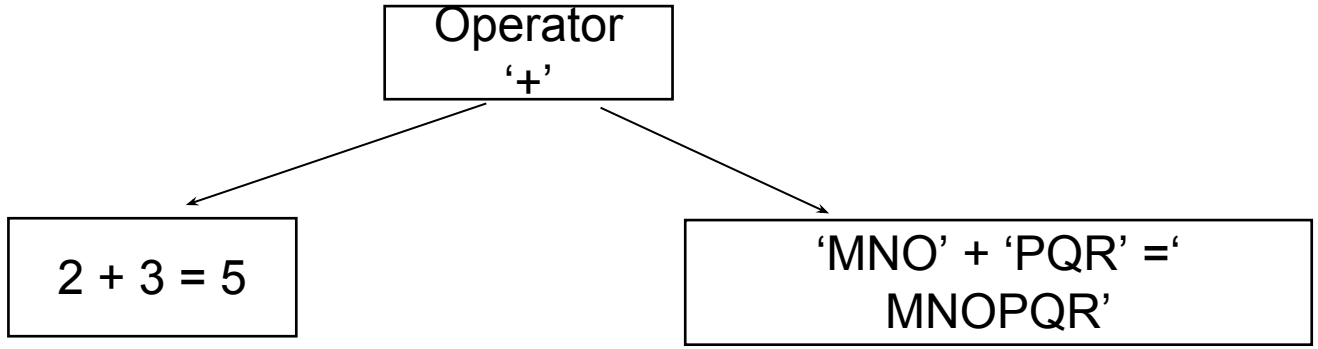
- Inheritance is the most powerful feature of oop and provides the idea of reusability, which in turn will increase the quality of work and productivity.

- Different types of Inheritance :

1. Single Inheritance
2. Hierarchical Inheritance
3. Multiple Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance

Polymorphism

- Polymorphism, a Greek term which means the ability to take more than one form.
- An operation may exhibit different behavior in 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 of two numbers and 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. 1.7 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*.

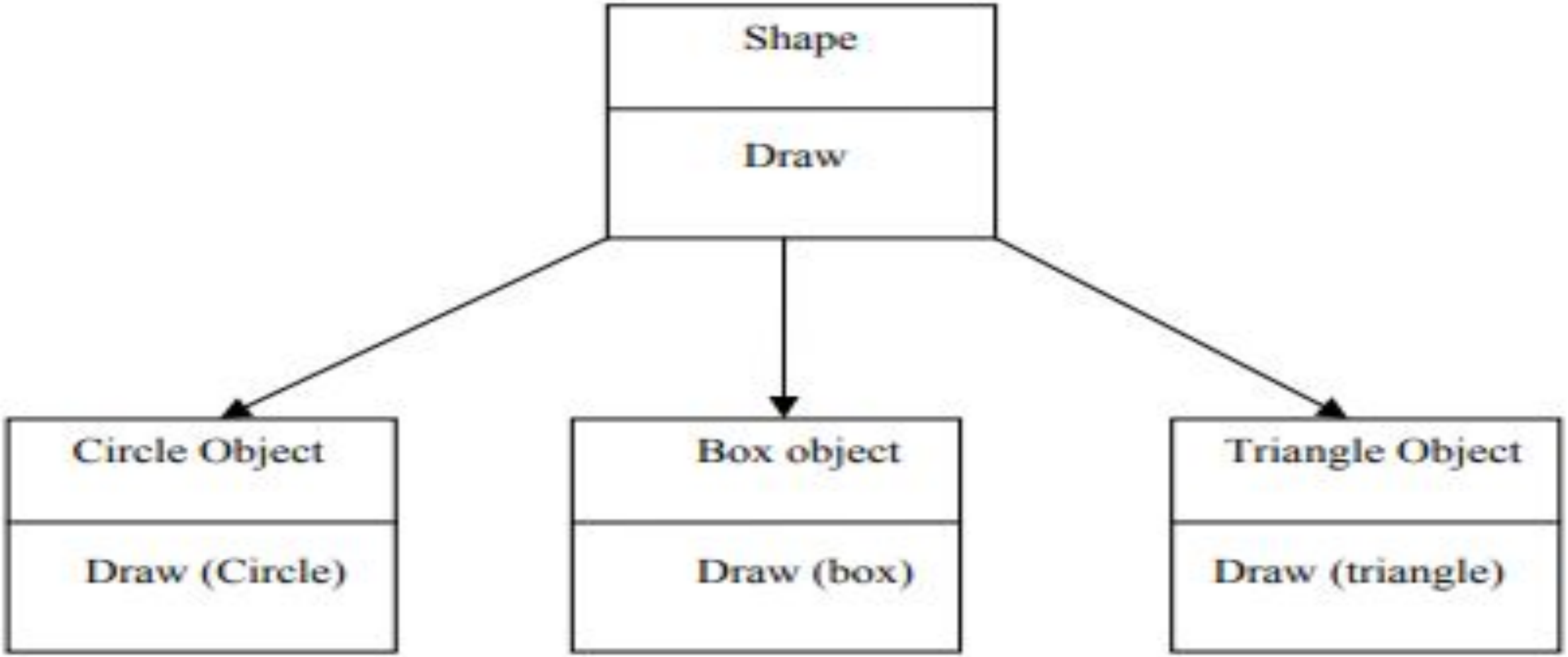


Fig. 1.7 Polymorphism

Dynamic Binding (Late Binding)

- **Binding** refers to the link between function call and code to be execute (function definition).
- It is the process of linking of a function call to the actual code of the function.
- Dynamic Binding: When binding takes place at the run time.
- In Dynamic binding, the actual code to be executed is not known to the compiler until run-time.
- For example:

Function call:

func();

func(a);

Function definition:

Void func() {cout<<“hello”;}

Void func(int a) {cout<<a;}

Message Passing

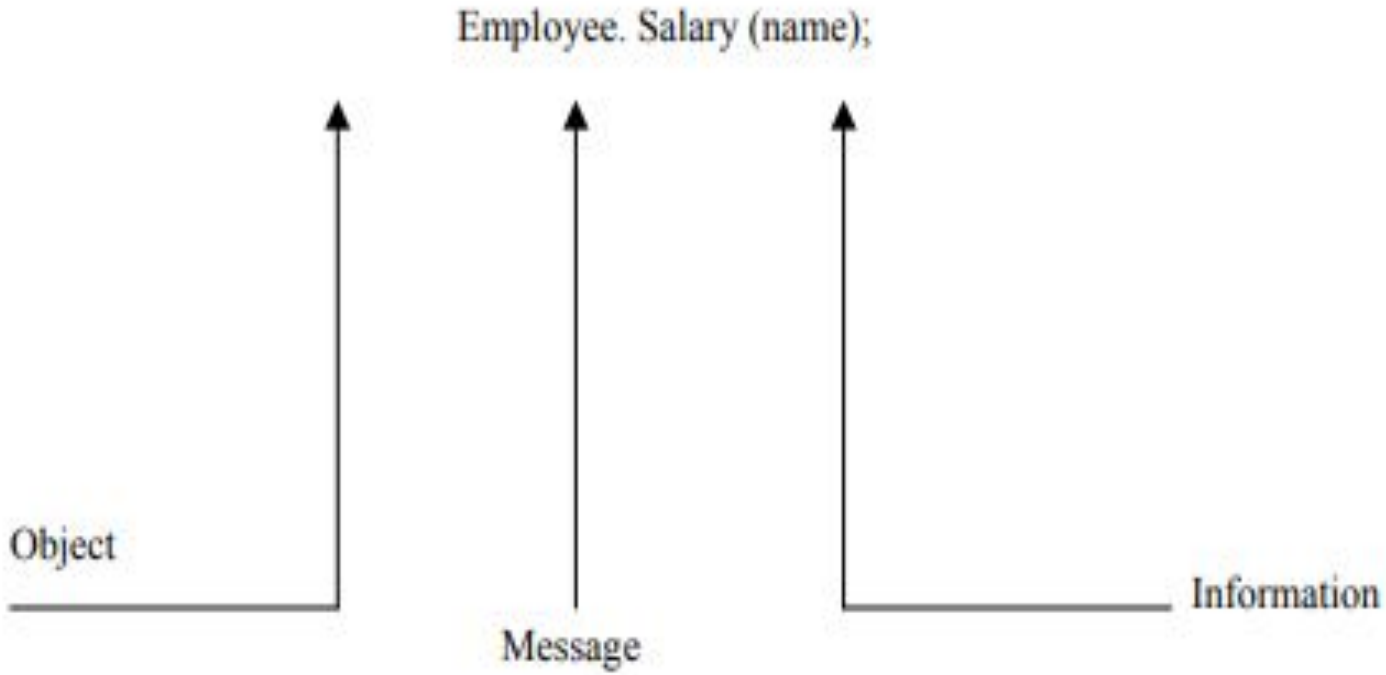
- Objects are made to communicate or interact with each other with the help of a mechanism called message passing.
- Objects can send or receive message or information.

- Two synchronized methods are used:

  putMessage(): Add a message in the message queue.

  getMessage(): Extracts the message from the message queue.

- 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.
- For Example:



Object-based vs Object-oriented programming languages

Object based Languages	Object Oriented Languages
Object based languages supports the usage of object and encapsulation. They does not support inheritance or, polymorphism or, both.	Object Oriented Languages supports all the features of Oops including inheritance and polymorphism.
Object based languages does not supports built-in objects.	They support built-in objects.
JavaScript, VB are the examples of object based languages.	C++, C#, Java, VB. Net are the examples of object oriented languages.