Lecture 2

2.1 Principles of Object-Oriented Programming

2.1.1 Software Evolution

The s/w evolution has had distinct phases or layers of growth. These layers were built up one by one. With each layer representing an improvement over the previous one. Fig. (2.1) had shown layers of s/w.

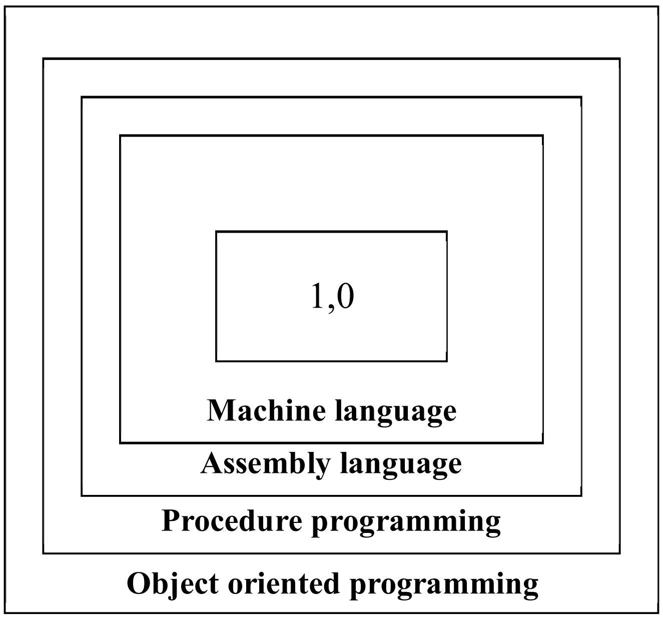


Fig. (2.1) layers of s/w technology

Since the invention of the computer, many programming approaches have been tried. These included techniques such as **modular programming**, **top-down programming, bottom-up programming and structured programming**. The primary motivation in each case has been the concern to handle the increasing complexity of programs that are reliable & maintainable. These techniques became popular among programmers over the last two decades. With the advent of languages such as C, structured programming become very popular and was the main technique of the 1980s. Structured programming was a powerful tool that enabled programmers to write moderately complex programs fairly easily.

Object-Oriented Programming (OOP)

Is an approach to programs 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.

2.1.2 Procedure–Oriented Programming

Conventional programming using high level languages (COBOL, FORTRAN and C) is commonly known as procedure oriented programming. In the procedure-oriented approach, the problem is viewed as a sequence of things to be done, such as reading, calculating and printing. A number of functions are written to accomplish these tasks. A typical program structure for procedure programming is show in fig. (2.2).

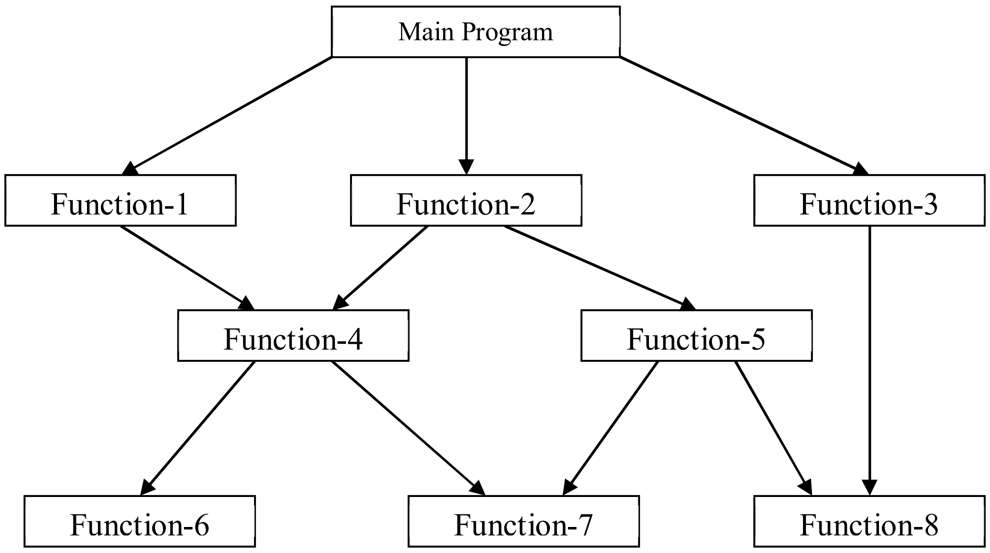


Fig. (2.2) Typical structure of procedure oriented programs

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 a flowchart to organize these actions and represent the flow of control from one action to another.

In 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. Fig. (2.3) shows the relationship of data and functions in a procedure-oriented program.

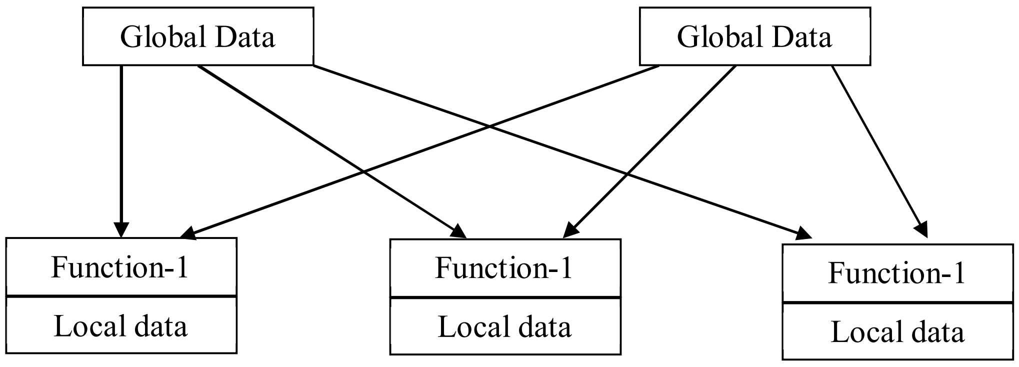


Fig. (2.3) Relationship of data and functions in procedural programming.

2.2 Disadvantage of Global data

- ❖ Global data are more vulnerable to an inadvertent change by a function.
- ❖ In a large- program it is very difficult to identify what data is used by which function.
- ❖ When we need to revise an external data structure, we should also revise all function that accesses the data.

Drawback of procedure approach is that it does not model real world problems very well. Because the functions are action-oriented and do not really correspond to the elements of the problem.

2.3 Characteristics exhibited by procedure-oriented programming

- ❖ 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.
- ❖ Function transforms data from one form to another.
- ❖ Employs top-down approach in program design.

2.4 Object-Oriented Programming Paradigm (OOP)

The major motivating factor in the invention of OOP is to salvage some of the flaws encountered the procedural approach.
OOP treats data 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 functions that operate on it and protects it from accidental modification from outside functions.

OOP allow us to decompose a problem in to a number of entities call objects and then built data and functions around these entities. The organization of data and functions in OOP is shown in Fig. (2.4).
OOP: - is approach that provides away 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.
The object-oriented approach to programming is an easy way to master the management and complexity in developing software systems that take advantage of the strengths of data abstraction.

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

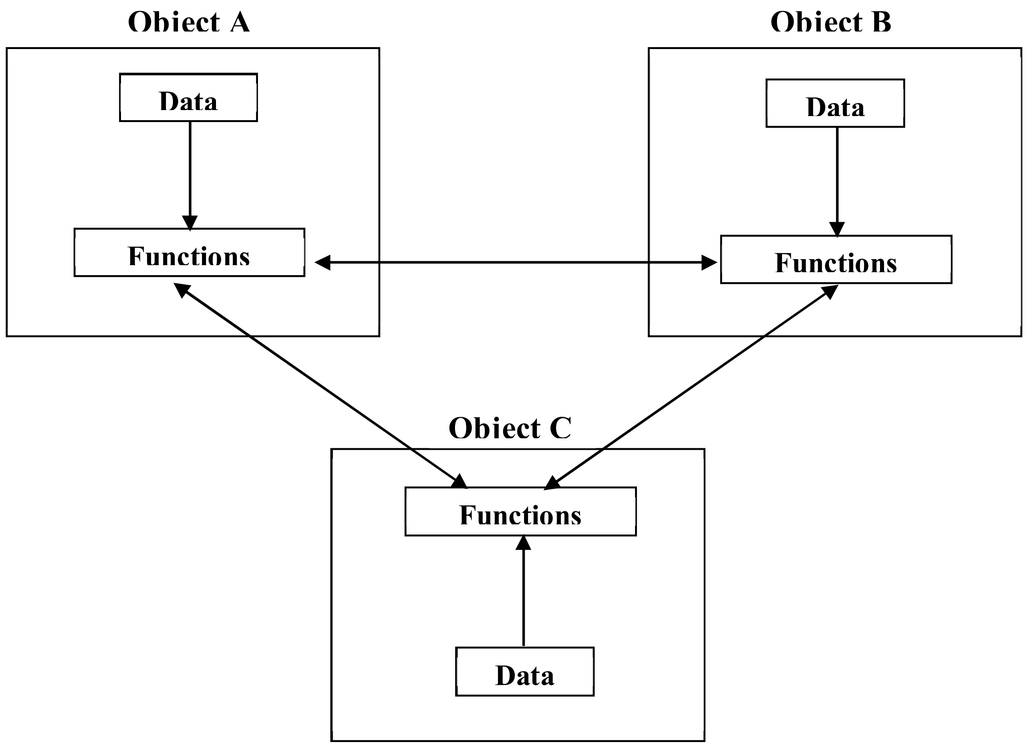


Fig. (2.4) organization of data and function in OOP

2.5Features of OOP: -

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

2.6 Basic Concepts of OOP: -

The concepts used extensively in OOP are:

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

Classes and Objects: -

The concepts of object-oriented technology must be represented in object-oriented programming languages. Only then, complex problems can be solved in the same manner as they are solved in real-world situations. OOP languages use classes and objects for representing the concepts of abstraction and encapsulation.

The software structure that supports data abstraction is known as class. A class is a data type capturing the essence of an abstraction. It is characterized by a number of features. The class is a prototype or blue print or model that defines different features. A feature may be a data or an operation. Data are represented by instance variables or data variables in a class. The operations are also known as behaviors, or methods, or functions. They are represented by member functions of a class in C++ and methods in Java and C#.

A class is a data type and hence it cannot be directly manipulated. It describes a set of objects. For example, apple is a fruit implies that apple is an example of fruit. The term “fruit” is a type of food and apple is an instance of fruit. Likewise, a class is a type of data (data type) and object is

an instance of class. Similarly car represents a class (a model of vehicle) and there are a number of instances of car. Each instance of car
is an object and the class car does not physically mean a car. An object is also known as class variable because it is created by the class data type. Actually, each object in an object-oriented system corresponds to a real-world thing, which may be a person, or a product, or an entity.

. Fig. (2.5) shows two notations that are popularly used in object-oriented analysis and design.

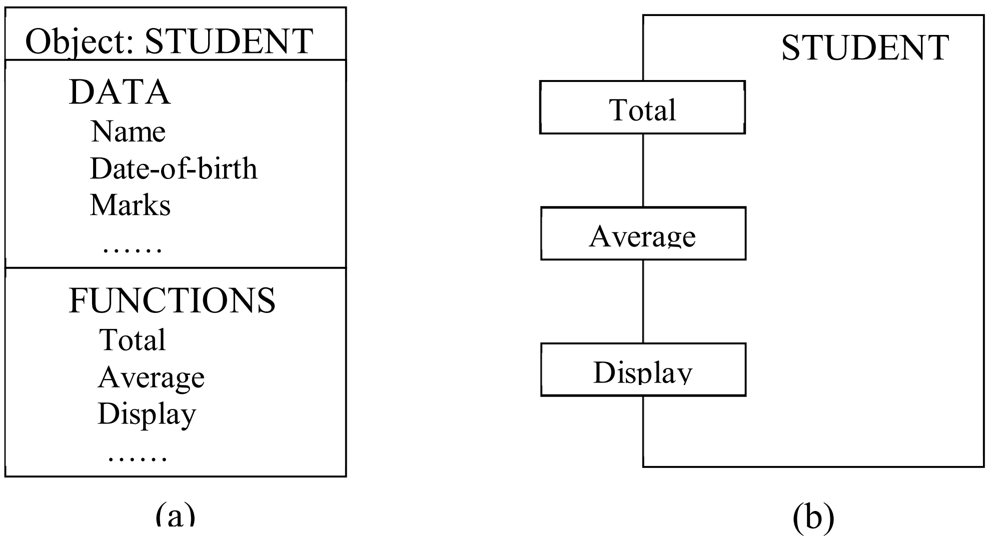


Fig. (2.5) Two ways of representing an object

Table (2.1) Comparisons of Class and Object

Class	**Object**
Class is a data type.	Object is an instance of class data type
It generates object.	It gives life to a class.
It is the prototype or model.	It is a container for storing its features.
Does not occupy memory location.	It occupies memory location.
It cannot be manipulated because it is not available in the memory.	It can be manipulated.

Instantiation of an object is defined as the process of creating an object of a particular class.

An object has:

1. States or properties.
2. Operations.
3. Identity

Properties maintain the internal state of an object. Operations provide the appropriate functionality to the object. Identity differentiates one object from the other. Object name is used to identify the object. Hence, object name itself is an identity. Sometimes, the object name is mixed with a property to differentiate two objects. For example, differentiation of two similar types of cars, say MARUTI 800 may be differentiated by colors. If colors are also same, the registration number is used. Unique identity is important and hence the property reflecting unique identity must be used in an object.

Data Abstraction and Encapsulation

The wrapping up of data and functions into a single unit (called class) is known as encapsulation. Data 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 .Encapsulation provides a layer of security around manipulated data, protecting it from external interference and misuse. This insulation of the data from direct access by the program is called data hiding.

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes. Since the classes use the concept of data abstraction they are known as abstract data types (ADT).

Comparison of Abstraction and Encapsulation Abstraction

1- Abstraction separates interface and implementation.

2- User knows only the interfaces of the object and how to use them according to abstraction. Thus, it provides access to a specific part of data.

3- Abstraction gives the coherent picture of what the user wants to know. The degree of relatedness of an encapsulated unit is defined as cohesion. High cohesion is achieved by means of good abstraction.

4- Abstraction is defined as a data type called class which separates interface from implementation.

5- The ability to encapsulate and isolate design from execution information is known as abstraction.

Encapsulation

1. Encapsulation groups related concepts into one item.
2. Encapsulation hides data and the user cannot access the same directly (data hiding).
3. Coupling means dependency. Good systems have low coupling. Encapsulation results in lesser dependencies of one object on other objects in a system that has low coupling. Low coupling may be achieved by designing a good encapsulation.
4. Encapsulation packages data and functionality and hides the implementation details (information hiding).
5. Encapsulation is a concept embedded in abstraction.

Classes and objects represent abstractions in OOP languages. Class is a common representation with definite attributes and operations having a unique name. Class can be viewed as a user-defined data type.

Is a declaration of variables in C. This statement conveys to the compiler that year and mark are instances of integer data type. Likewise, in OOP, a class is a data type. A variable of a class data type is known as an object. An object is defined as an instance of a class. For example,

if Book is a defined class,

Book c Book, java Book;

Declares the variables c Book and java Book of the Book class type. Thus, classes are software prototypes for objects. Creation of a class variable or an object is known as instantiation (creation of an instance of a class). The objects must be allocated in memory. Classes cannot be allocated in memory.

Inheritance

Inheritance is the process by which objects of one class acquire the properties of objects of another class. For example, the bird “robin” is a part of the class “flying bird” which is again apart of the class “bird”. As illustrated in fig. (2-6), the principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived.

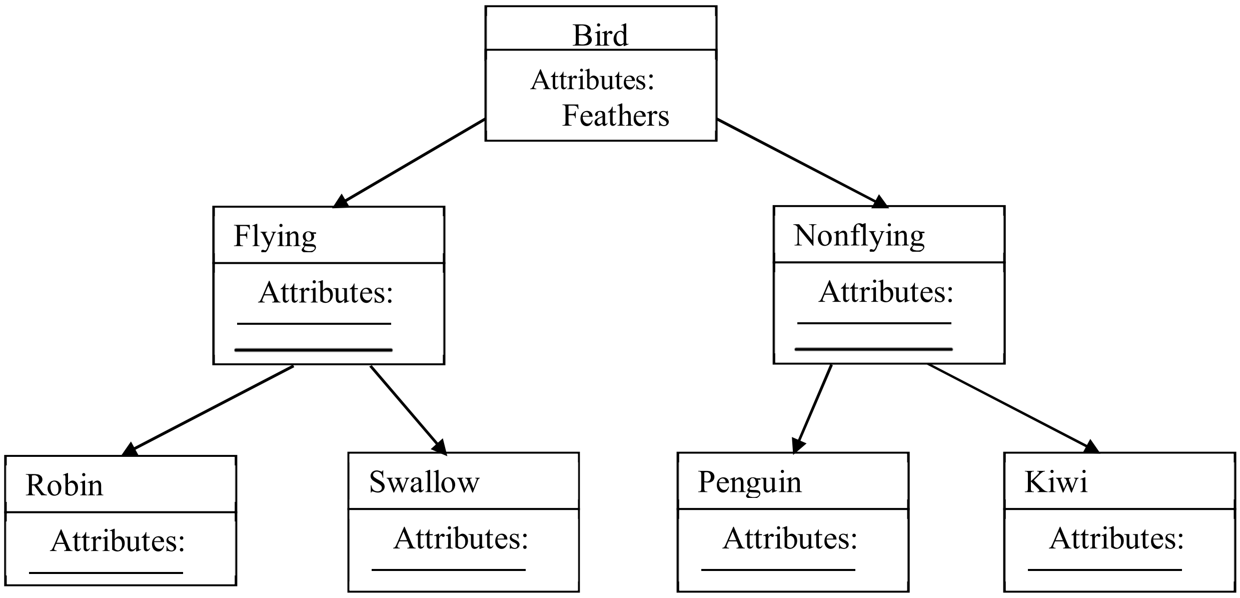


Fig. (2.6) Property inheritance

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 will have the combined features of both the classes.

The new derived class inherits the members of the base class and also adds its own.

For example, a banking system would expect to have customers, of which we keep information such as name, address, etc.

A subclass of customer could be customers who are students, where not only we keep their name and address, but we also track the educational institution they are enrolled in. Inheritance is mostly useful for two programming strategies: extension and specialization. Extension uses inheritance to develop new classes from existing ones by adding new features. Specialization makes use of inheritance to refine the behavior of a general class.

Polymorphism

Polymorphism is another important OOP concept. Polymorphism means the ability to take more than one form. For example 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". If the operands are strings, then the operation would produce a third stringy concatenation. Figure (3.7) illustrates that a single function name can be used to handle different number and different types of arguments.

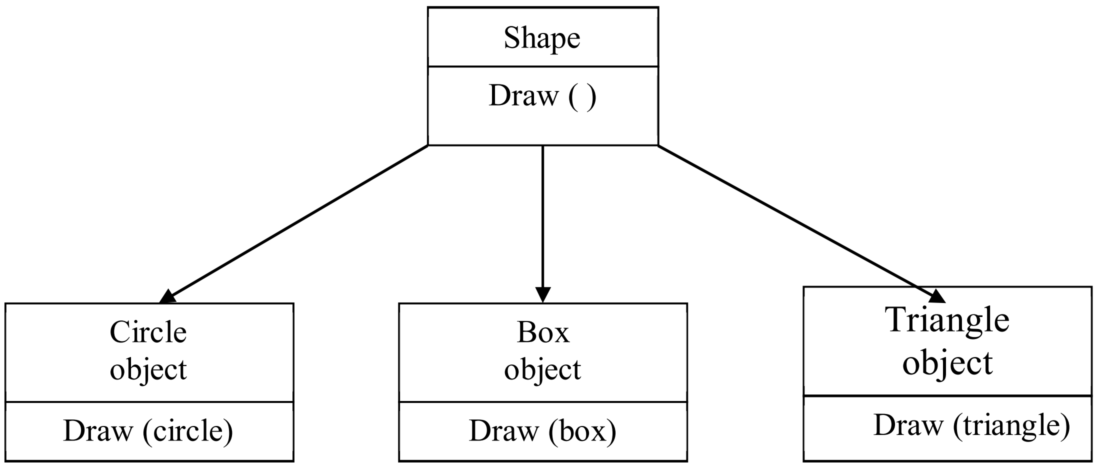


Fig. (2.7) Polymorphism.

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 actions 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 polymorphism reference depends on the dynamic type of that reference.

Consider the procedure “draw” in fig. (2-7). 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 communication

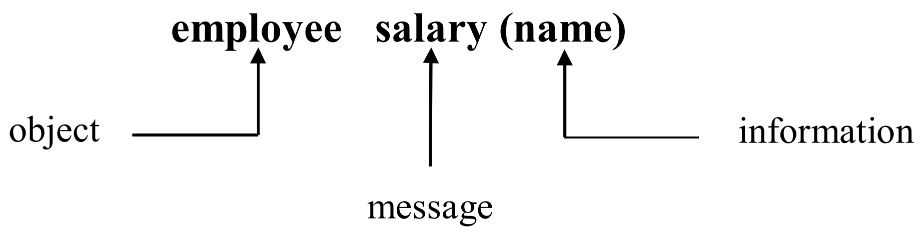
An object-oriented program consists of a set of objects that communicate with each other. The process of programming in an object-oriented language therefore involves the following basic steps:

- ❖ Creating classes that define objects and their behavior.
- ❖ Creating objects from class definitions.
- ❖ 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.

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 result.

Message passing involves specifying the name of the object the name of the function message and the information to be sent. Example:



Benefits of OOP

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 s/w products. The principal advantages are:

1. Through inheritance, we can eliminate redundant code and extend the use of existing classes.

2. We can build programs from the standard working modules that communicate with one another. This leads to saving of development time and higher productivity.
3. The principle of data hiding helps the programmer to build secure programs that cannot be by code in other parts of program.
4. It is possible to have multiple instances of an object to co-exist without any interference.
5. It is possible to map object in the problem domain to those objects in the program.
6. It is easy to partition the work in a project base on objects.
7. Object-oriented systems can be easily upgraded from small to large systems.
8. Massage passing techniques for communication between objects makes the interface descriptions with external systems much simpler.
9. S/w complexity can be easily managed.

Object oriented languages

The object-oriented approach to programming is an easy way to master the management and complexity in developing software systems that take advantage of the strengths of data abstraction. Data-driven methods of programming provide a disciplined approach to the problems of data abstraction, resulting in the development of object-based languages that support only data abstraction. These object-based languages do not support the features of the object-oriented paradigm, such as inheritance or polymorphism. Depending on the object features supported, there are two categories of object languages:

1. Object-Based Programming Languages.
2. Object-Oriented Programming Languages.

Object-based programming languages support encapsulation and object identity (unique property to differentiate it from other objects) without supporting important features of OOP languages such as polymorphism, inheritance, and message based communication, although these features may be emulated to some extent. Ada, C, and Haskell are three examples of typical object-based programming languages.

Object-based language = Encapsulation + Object Identity

Object-oriented languages incorporate all the features of object-based programming languages, along with inheritance and polymorphism. Therefore, an object-oriented programming language is defined by the following statement:

Object-oriented language = Object-based features + Inheritance + Polymorphism

Object-oriented programming languages for projects of any size use modules to represent the physical building blocks of these languages. A module is a logical grouping of related declarations, such as objects
Or procedures, and replaces the traditional concept of subprograms that existed in earlier languages.

Object-oriented technology is built upon object models. An Object is anything having crisply defined conceptual boundaries. Book, pen, train, employee, student, machine, etc., are examples of objects. But the Entities that do not have crisply defined boundaries are not objects. Beauty, river, sky, etc., are not objects.

Model is the description of a specific view of a real-world problem domain showing those aspects, which are considered to be important to the observer (user) of the problem domain. Object-oriented programming language directly influences the way in which we view the world. It uses the programming paradigm to address the problems in everyday life. It addresses the solution closer to the problem domain.

Object model is defined by means of classes and objects. The development of programs using object model is known as object-oriented development. To learn object-oriented programming concepts, it is very important to view the problem from the user's perspective and model the solution using object model.

Procedure oriented programming	**Object oriented programming**
Emphasis is on doing things (algorithms). **Logical structure**	Emphasis is on data rather than procedure.
Large programs are divided into smaller programs known as "functions". **Execution as functions**	Programs are divided into 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.
Most of the functions share global data. **Focus on code**	Data is hidden and cannot be accessed by external functions.
Data move openly around the system from function to function. **Less data security**	Object may communicate with each other through functions.
Function transforms data from one form to another.	New data and function can be easily added whenever necessary.
Employs top-down approach in program design.	Follows bottom-up approach in program design.