1

Introduction to Object Oriented Programming Concepts

UNIT I
PRINCIPLES OF OBJECT ORIENTED PROGRAMMING

Learning Scope

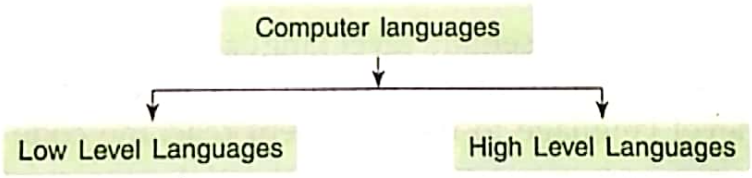
Introduction, Low level languages, High level languages (advantages and disadvantages), Compiler and Interpreter, Types of High level languages, Structure Oriented Programming, Procedural Oriented programming and Object Oriented programming, Features of Object Oriented programming, Difference between Procedural Oriented programming and Object Oriented programming, All the four principles of OOP viz. Data abstraction, Inheritance, Polymorphism, Encapsulation with real life examples.

INTRODUCTION

Computer is an electronic device that accepts instructions in a specific language. Hence, it is very difficult to establish communication between the user and the computer if he/she is not aware of computer language. This is the reason why learning computer language is very essential for working on the computer.

TYPES OF COMPUTER LANGUAGES

The computer languages are basically categorised into two levels. They are:



1. Low Level Languages

The low level languages are the type of computer languages in which the computer recognizes the instructions, without conversion into any other form. They are sort of cryptic languages which are not directly understood by the users. These type of languages were used in the early computers when only the experts were able to code the instructions. The low level languages are further categorised into two types:

- *Machine Level Language*

It is the form of computer language in which instructions are coded in terms of binary digits (bits), i.e., in the form of zeroes and ones (0's and 1's). This form of binary instructions is also called the machine code or the object code.

The computer understands the machine codes easily (*i.e.* it does not need to be transformed or converted into any other form). The only advantage of this language is that the program executes faster because the machine code is directly operated by the CPU.

Limitations of Machine Level language:

- The user needs to remember all the instruction codes.
- The error detection and correction is difficult.
- It is a machine dependent language.

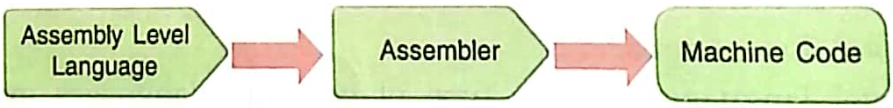
The user must be aware of the internal hardware structure of the computer for coding instructions; and so it is known as a machine dependent language.

• ***Assembly Level Language***

A language, in which the instructions are coded in terms of mnemonics and op-codes (or operation codes), is known as an assembly level language. The mnemonics are the abbreviated form of the instructions, used for writing a program. Whereas, op-codes are the numeric codes of the instructions that are actually fed to the computer for execution. The instructions in assembly level language are readable to some extent as compared to machine level language. Below are some mnemonics and op-codes, illustrated for your reference:

MNEMONICS	DESCRIPTION	OP – CODES
LD A	Enter the value in A	3B
LD B	Enter the value in B	73
ADD A, B	Add the values of A and B	C2
ST C	Store the sum in C	3F
HLT	Stop	EF

The instructions given in assembly level language are not directly understood by the computer. Hence, a translator is required to convert the instructions coded in the assembly level language to its equivalent machine code which is known as the *Assembler*.



Advantages of Assembly Level language:

- It is easier to write the instructions as compared to machine level language.
- The error detection and correction is also comparatively easier.
- The code can easily be modified.

Limitations of Assembly Level language:

- It is also a machine dependent language.
- The users need to remember all the mnemonics.
- A translator is required to convert the instructions into machine code.

2. High Level Language

To overcome the disadvantages of low level languages, the experts (System Developers) developed another category of languages, which are referred to as the High Level Languages (HLL). These languages allow the user to write the instructions in simple English phrases or sentences. It also uses common English words and mathematical symbols. Thus, it makes the instructions easier to understand in the user's native language. It allows a user to write the instructions, even if he is not aware of the hardware architecture of the computer. The programming style and context is comparatively easier to learn. The entire code focuses on the specific program to be created.

BASIC, C/C++, Java and Python are some popular examples of high level languages.

Advantages of High Level language:

- It is a machine independent language.
- The instructions can be written using English words or phrases.
- It is easier to understand and develop the program logic.
- The error detection and correction is easier.

Disadvantages of High Level language:

- It requires a translator to convert the source code (program) into machine code. cause it takes longer... etc. 2
- The machine code of high level instructions might be less efficient than the machine code generated from assembly level instructions.

One of the major disadvantages of the High level language is that the machine cannot understand this language directly (a machine understands only machine level language). Hence, we need to convert the instructions written in high level language into their equivalent instructions, in machine level language, so that they can be operated upon by the CPU. The translators/language processors used to perform above mentioned task, are referred to as *Compilers* and *Interpreters* Let us understand the functions of these translators.

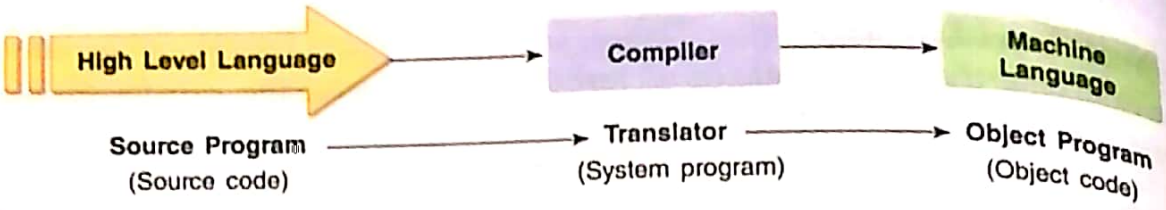
COMPILER AND INTERPRETER

A program in high level language needs to be converted into machine code (binar code) so that the computer can understand the instructions for processing. If th program contains any error then it must be corrected for successful execution

The conversion of high-level language (source code) to machine level languag i.e., MLL (binary code) can be done in two ways; either by using a Compiler o an Interpreter.

A software that accepts the whole program written in high level language ar converts it into its equivalent program in machine level language, is known the *compiler*.

The program, which the compiler uses for conversion, is known as the *sour program* or *source code*. The program converted into the machine level langua is known as the *object program* or *object code*.



A compiler is designed exclusively to convert a program written in a specific high level language. This means that each high level programming language would require a separate compiler for conversion. For example, a FORTRAN compiler is capable of translating only a FORTRAN program. A computer system may have more than one compiler for more than one high level language.

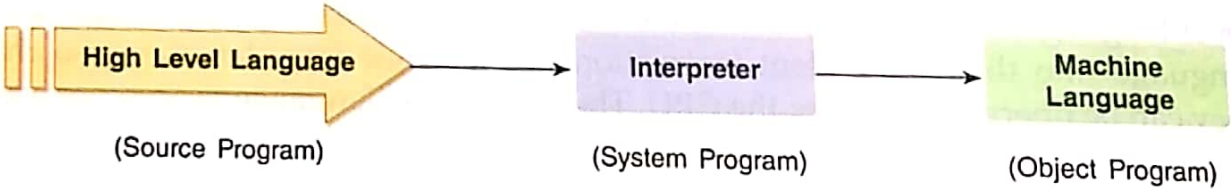
The software which converts the instructions written in high level languages into their equivalent instructions in machine level language, line by line or statement by statement, is known as the *Interpreter*. If an error is found on any line, the execution stops there till it is corrected. This process of correcting errors is easier but the program takes more time to execute successfully.

Note

A compiler is not capable of diagnosing logical errors. It can only identify the syntax errors in the program.

An Interpreter is generally used in micro computers. It helps the programmer find out the errors and correct them before the control moves to the next statement. Compilers and interpreters are basically system softwares, which are also known as the *language processors*.

The compiler is comparatively faster than the interpreter but sometimes, it becomes difficult to correct errors since it displays all the errors together.

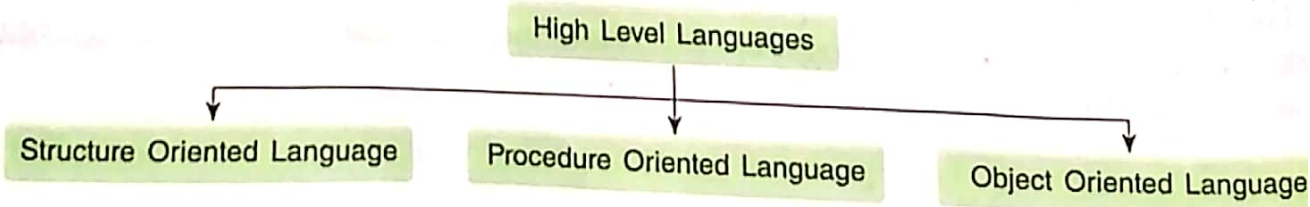


Differences between a Compiler and an Interpreter

Compiler	Interpreter
1. It converts the whole source program into the object program at once.	1. It converts the source program into the object program one line at a time.
2. It displays the errors for the whole program together, after the compilation.	2. It displays the errors, one line at a time and only after debugging that error the control goes to the next line.

TYPES OF HIGH LEVEL LANGUAGES

High level languages are broadly classified into three major categories. They ar



1. Structure Oriented Programming Language

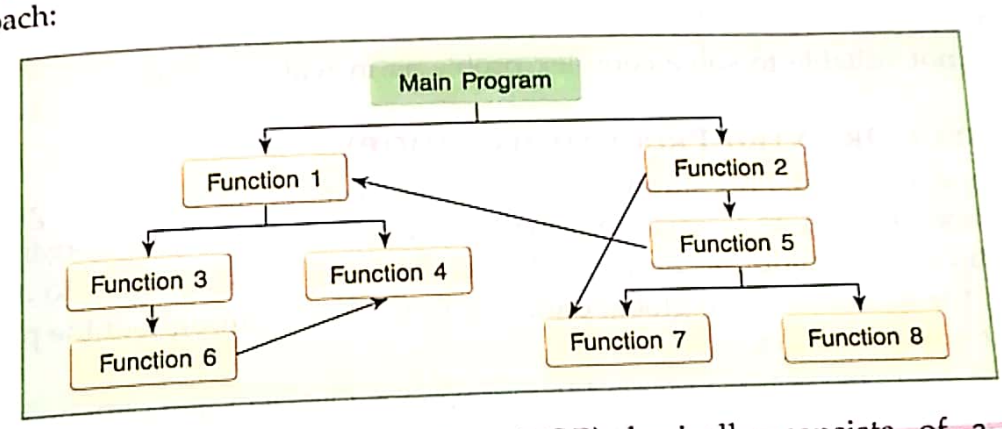
Structure Oriented language uses a modular approach to improve the clarity, quality and the development time of the programming steps. It uses various logical structures like the structure of selective control flow, structure of looping block, structure of subroutines or functions etc. ALGOL and PASCAL are examples of structure oriented programming languages.

2. Procedure Oriented Programming Language

The procedure oriented approach allows the users to develop their logic by using a number of functions that would enhance the program's productivity.

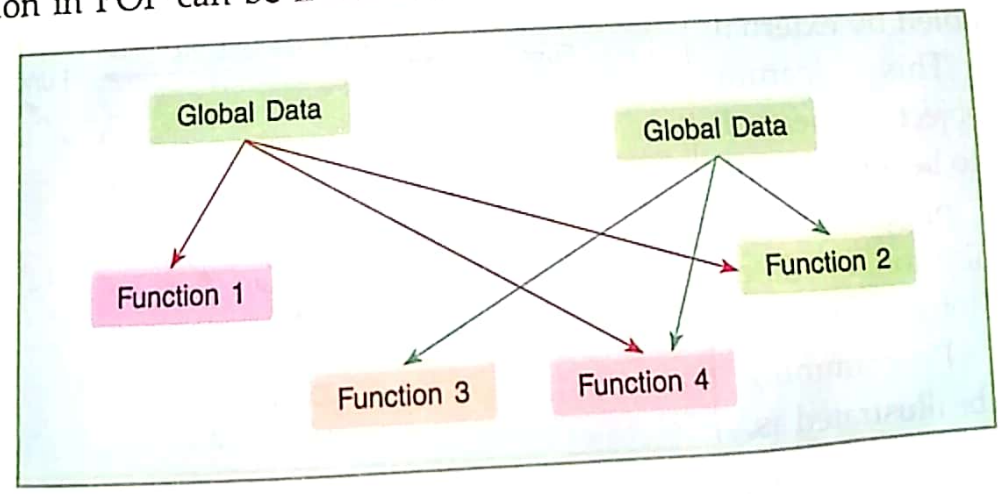
Conventional programming using high level languages such as BASIC, COBOL, FORTRAN and C are commonly known as Procedure Oriented Programming (POP) languages.

Here, a sample flow diagram is illustrated to depict the procedural programming approach:



Procedure Oriented Programming (POP) basically consists of a list of instructions for the computer to follow and these are organized into groups, known as *functions*. We normally use a flow chart to organize these actions and represent the flow of control from one function to another. In Procedure Oriented Programming, most of the functions share global data and this data moves more openly around the system from one function to the other.

When we deal with a program containing many functions, important dat items are globally used by all the functions, however, a function may contai its own local data to deal with logical situations. The organisation of data an its function in POP can be illustrated as:



In this system, the global data are loosely attached to the functions. They keep floating throughout the program. In order to make any change in the function, you may need to reschedule the associated data values. This may affect the normal sequencing of the program logic.

In Procedure Oriented Programming system, the emphasis is on Functions rather than Data Items.

Characteristics of Procedure Oriented Programming

- Emphasis is on functions (logical steps).
- Functions share global data.
- Data values can keep floating from one function to another.
- It uses top down approach of programming.

Limitations of Procedure Oriented Programming (POP)

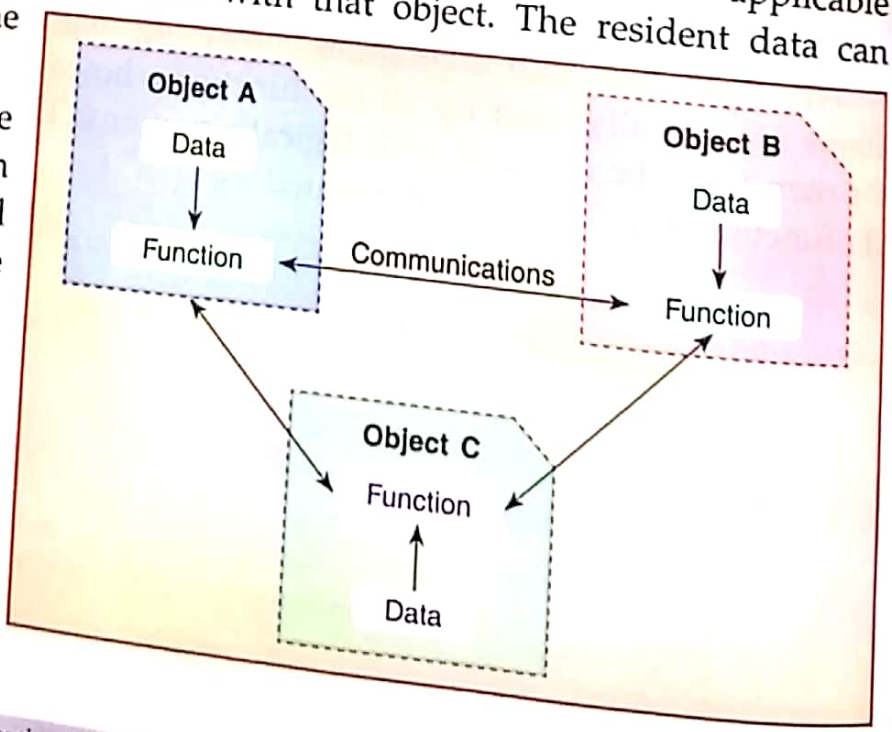
- As data values are global to all the functions, you may require to make necessary changes in all the functions, in case of any change in the data values.
- It is not suitable to solve complex problems in real situations.

3. OBJECT ORIENTED PROGRAMMING (OOP)

Procedure Oriented Programming basically consists of an organised group of instructions known as *function*. Normally, flow charts are used to organise the instructions and guide the movement of control from one function to another. Most of the functions share global data that flows freely throughout the program without any restriction.

In contrast to Procedure Oriented Programming, Object Oriented Programming (OOP) is an approach to standardize the programs by creating partitioned memory area for both the data and the functions. It does not allow data to low freely from one function to another. In this system, the complete problem is lecomposed into a number of entities called the *objects*. Each object includes a set f data items and related functions. The data values of an object are applicable nly within the functions associated with that object. The resident data can ever be handled by the ternal functions.

In this way, the data are otected and secure from ing troubled by external irces. This feature ows object oriented roach to be a powerful in programming. organisation of data functions in Object nted Programming) can be illustrated as:



The different Object Oriented Programming languages commonly being used are C++, Java, Python, Smalltalk, Ruby, Eiffel, etc.

An Object Oriented Programming (OOP) is a modular approach, which allows the data to be applied within a stipulated program area. It also provides the reusability feature to develop productive logic, which means to give more emphasis on data.

Features of Object Oriented Programming (OOP)

Some of the features of Object Oriented Programming are listed below:

- It gives importance to the data items rather than the functions.
- It makes the complete program/problem simpler by dividing it into a number of objects.
- The objects can be used as a bridge to have data flow from one function to another.
- The concept of data hiding enhances the security in programs.
- It is highly beneficial to solve complex problems.

Differences between Procedural Oriented Programming (POP) and Object Oriented Programming (OOP)

Now, you have learnt that both are programming processes where OOP stands for 'Object Oriented Programming' and POP stands for 'Procedure Oriented Programming'. They are different types of high-level programming languages with different approaches. Some of the differences between them are mentioned as under:

Procedural Oriented Programming	Object Oriented Programming
1. The emphasis is put on the function rather than the data.	1. The emphasis is put on the data rather than the functions.
2. It allows data to flow freely throughout the program.	2. The data is restricted, to be used in a specific program area.
3. It follows top-down programming approach.	3. It follows bottom-top programming approach.

KEYWORDS RELATED TO OBJECT ORIENTED PROGRAMMING

- **Object:** It is a unique entity, which contains data and functions (characteristics and behaviour) together in an Object Oriented Programming (OOP) Language.
- **Real World Object:** The objects that we experience or use in our day to day life. Each real world object contains characteristics and behaviour. The characteristics basically comprises the parts of its body or specifications whereas behaviour refers to the purpose of its use or its function.

- **Software Objects:** A software object may be defined as an object that is created while writing a Java program. When we compare a software object with the real world object then the characteristics and behaviours of real world objects are referred to as the data members and member methods (functions) of the software objects, respectively.
- **Class:** The class is a template or blue print for similar type of objects. Each object of a class possesses some attributes and common behaviour as defined within the class. Thus, a class is also referred to as a blue print or prototype of an object.

BASIC PRINCIPLES OF OBJECT ORIENTED PROGRAMMING (OOP)

The Object Oriented Programming (OOP) has the following basic principles:

- Data Abstraction
- Polymorphism
- Inheritance
- Encapsulation

Data Abstraction

In real life situations, you might have noticed that we do not require to know the details of the technologies to operate any system.

For example, a digital camera is used to capture a photograph. It simply provides a number of buttons and switches to control the operations needed to take the photograph. In fact, these buttons make the working so eay that anybody could operate the camera, even though he/she may not be aware of the technology.

Have you ever thought of how does the camera manage to take photograph or what mechanism is followed in the internal part of the camera?

You may not be able to answer such questions because you are unaware of technology used inside the camera.

So, you may say that we use only the essential features of the camera to take a photograph without knowing the internal mechanism.

'Data Abstraction' is an act of representing the essential features without knowing the background details. It is always relative to the purpose or the user.

It may be noted that the abstraction of an object depends upon the area of applications.

Inheritance

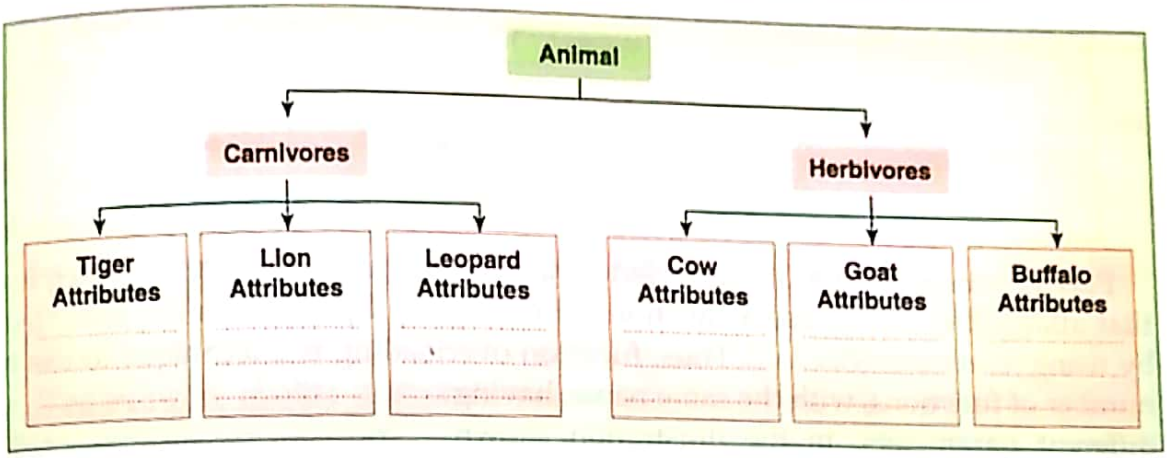
You might have studied the term 'Heredity' in biology, which means the transmission of genetically based characteristics from parents to the offsprings. We inherit thousands of characteristics (inheritable features) from our parents who in turn, had inherited those from their parents, and so on.

Similarly, a class acquires some properties from another class. This is possible by deriving a new class from the existing class. The new class will have the

combined features of both the classes. The class that gets inherited to another class is known as *the Base class* or *the Super class*. The class that inherits from a Base class is known as *the Derived class* or *the Sub-class* or *Target*.

Let us take an example of a class 'Animal' which can be broadly classified into 'Carnivores' (flesh eating animals) and 'Herbivores' (plants eating animals).

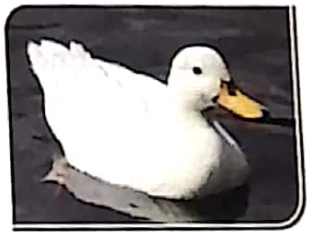
So, you will find that some of the characteristics or properties of the class 'Animal' will be inherited by the classes Carnivores and Herbivores.



During inheritance, the elements of the base class are shared by the derived class. As a result, it may happen that the elements performing a specific task in the base class can be used to perform another task in derived class. This feature is called *Reusability*.

The term Inheritance means to link and share some common properties of one class with the other class. This can be done by extending a class into another class and using thus using both it.

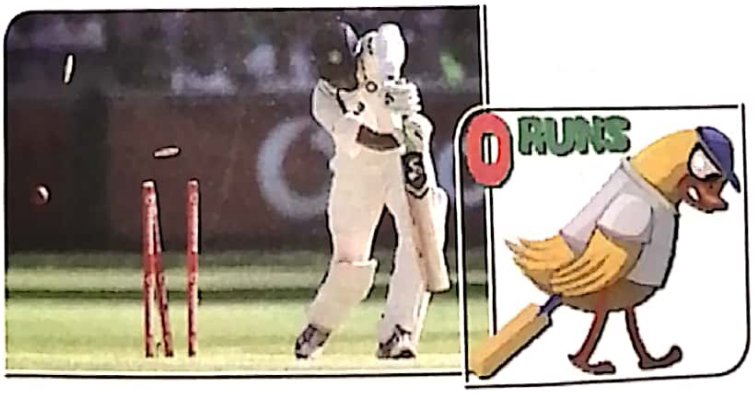
Polymorphism



You know that a single word can have many meanings. Similarly, an operation may show an entirely different behaviour for a different set of data and environment.

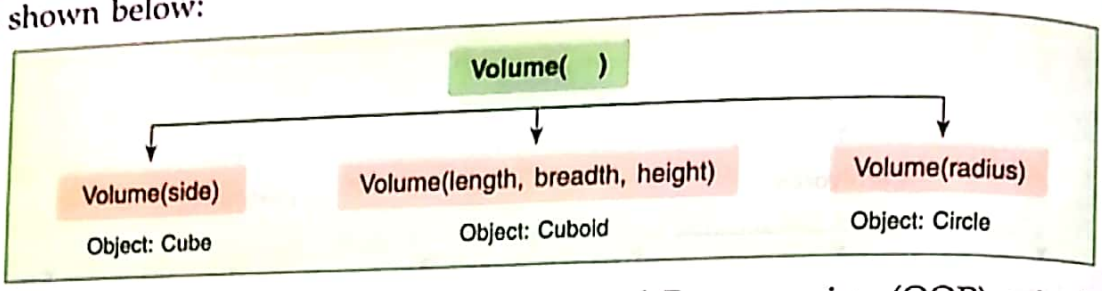
Let us take an example of the English word 'duck'.

The word 'duck' defines a water bird with a broad bill, short legs and webbed feet whereas, the same word 'duck' in a cricket match means a batsman who got out with no score.



With the reference to the above examples, you can notice that the same word 'duck' is used for two different purposes. Similarly, you can find many such examples in real life situations also.

With reference to Object Oriented Programming, if the function name is Volume( ), then you can calculate the volume of different geometrical solid figures viz. a cube, a cuboid, a sphere, a cylinder by using different parameters, as shown below:



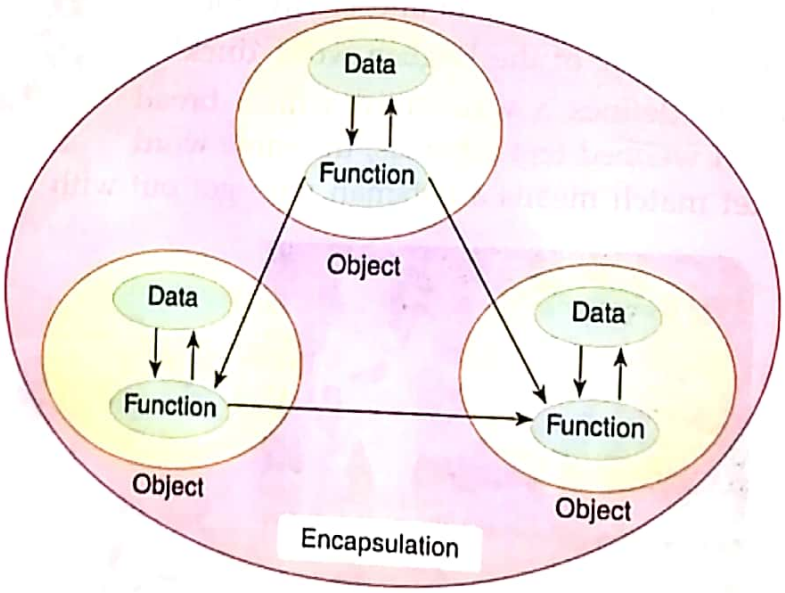
Polymorphism is one of the Object Oriented Programming (OOP) principles that allows the user to use a function for multiple purposes. It is implemented by using *function overloading*. Thus, function overloading is a technique to use a number of functions, with the same name, having different parameters. In the illustration shown above, three functions are defined with the same name 'volume'. Hence, they are overloaded. The execution of these functions will depend upon the data types and number of parameters.

The term 'Polymorphism' is defined as the process of using a function/method for more than one purposes.

Encapsulation

Encapsulation is another feature of the Object Oriented Programming which restricts the free flow of data from one object to another. The data and functions are wrapped together in an object such that the data can only be accessed in the associated functions.

Thus, we can say that the data is kept hidden and cannot be accessed directly outside the object, although it is available in the same program. The following illustration of Encapsulation depicts how the data and functiions are inter-linked:



With reference to the above context, it could be inferred that the state and behaviour are encapsulated in an Object Oriented Programming to prevent their isolation from each other.

The system of wrapping data and functions into a single unit is known as Encapsulation.

Thus, encapsulation is helpful in Object Oriented Programming in the following ways:

(a) The source code of an object could be maintained independently.

(b) The object maintains the privacy of the data members. However, the changes that take place in the methods do not affect the other object.

Benefits of Object Oriented Programming

There are many reasons of preferring Object Oriented Programming over Procedure Oriented Programming. Some of them are mentioned below:

- The reusability of the program code is enhanced.
- The software quality and performance are improved.
- Modularity is achieved.
- Data abstraction makes the software easier to handle.
- Software for complex tasks can easily be developed.

Limitations of Object Oriented Programming

Even though, OOP languages are preferred in solving problems ranging from simple to complex, there are some limitations of Object Oriented Programming, which are mentioned as under:

- Object Oriented Programming languages require intensive testing processes.
- Solving problems is more time consuming as compared to Procedure Oriented Programming.

Student's Notes

REVIEW INSIGHT

(a) Name any two Procedure Oriented Programming languages.

Ans. (*i*) BASIC (*ii*) COBOL

(b) What is meant by Procedure Oriented Programming languages?

Ans. Procedure Oriented Programming Language focuses on the functions rather than the data. The data values may keep floating throughout the program in an unrestricted and insecure manner.

(c) Name any two Object Oriented Programming languages other than Java.

Ans. (*i*) C++ (*ii*) Python

(d) Define source code and object code.

Ans. A program written in high level language that needs to be converted into machine code with the help of a translator (Compiler/Interpreter) is known as the source code. Whereas, the machine code that is accepted by the computer for execution is known as the object code.

(e) Name any two basic principles of Object Oriented Programming.

(i) Data Abstraction (ii) Encapsulation

(f) What do you understand by the term data abstraction? Explain with an example. [ICSE-2010]

Ans. It is an act of representing essential features of a class without including the background details.

For example, while driving a car, you are only aware of its important parts (using essential features) viz. clutch, brake, gears and accelerator. By using brake, you can stop the car and thus you do not require to know the internal mechanism to be followed when brake is pressed.

(g) What does reusability mean?

Ans. During inheritance, the components used to perform a task in the base class may be used for an other task in the derived class. This feature is known as reusability.

(h) Name an Object Oriented principle that allows a function to be used for multiple purposes.

Ans. An Object Oriented principle that allows a function to be used for multiple purposes is called Polymorphism.

(i) State the Java concept that is implemented through:

(*i*) A superclass and a subclass

(*ii*) The act of representing essential features of a class without including the background details. [ICSE-2013]

Ans. (*i*) Inheritance (*ii*) Data Abstraction

(j) What is Inheritance? [ICSE-2017]

Ans. Inheritance is an OOP principle according to which a class acquires some features from another class. It promotes a characteristic called reusability.

(k) Define Encapsulation. [ICSE-2016]

Ans. The wrapping of the data and functions of a class so that they can be used as a unit is termed as Encapsulation.

(l) In what ways are Encapsulation and Data Abstraction inter-related?

Ans. Encapsulation is a mechanism of wrapping data and functions into a single unit. Moreover, the data is kept hidden which cannot be accessed directly outside the class although it is available in the same program.
Data abstraction is an act of representing essential features without including background details.