Basic Concepts of Object-Oriented Programming

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

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 terms 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.

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 contains data, and code to manipulate the data. Objects can interact without having to know details of each other's data or code. It is sufficient to know the type of message accepted, and the type of response returned by the objects.

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 a 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 collection of objects of similar type. For example, mango, apple and orange are members of the class fruit. Classes are user-defined data types and behave like the built-in types of a programming language.

Data Abstraction and Encapsulation

The wrapping up of data and functions, into a single unit (called class) is known as encapsulation. Date 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 art 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. They encapsulate all the essential properties of the objects 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 functions.

Inheritance
Inheritance is the process- by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. For example, the bird 'robin' is a part of the class 'flying bird" which in again a part of the class "bird". The principle behind this Sort of division is that each derived class shares common characteristics with the class from which it is derived.

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 features of both the classes. The real appeal and power of the inheritance mechanism is that it allows the programmer to reuse a class.

Polymorphism
Polymorphism is another important OOP concept. Polymorphism means the ability to take more than one form. An operation may exhibit different behaviors 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 string by concatenation, The process of making an operator to exhibit different behaviors in different instances is known as operator overloading a single Function name to perform different types of tasks known as function overloading.

Dynamic Binding
Binding refers to the linking of a procedure call to the code to be executed in response to the call, Dynamic binding (also known as late 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.

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, therefore, involves the following basic steps:

1. Creating classes that objects 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 result. Message passing involves specifying the name of the object, the name of the function (message} and the information to be sent.

Benefits of OOP
Through inheritance, we can eliminate redundant code and 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 Baring of development time and higher productivity.
- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program
- It is possible to have multiple instances of an object to co-exist without, any interference.
- Tt is possible to map objects, 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 details of a model in implementable form.
- Object-oriented systems can be easily upgraded from small to large systems.
- Message passing techniques, for communication between objects makes the interface descriptions with external systems much simpler.
- Software complexity can be easily managed.

Object-Oriented Languages

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, and
2. Object-oriented programming languages.

Object base programming is the style- of programming that primarily supports encapsulation and object identity. Major features 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 be object-based programming languages. They do not support inheritance and dynamic binding. Ada is a typical object based programming language.

Object-oriented programming 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 statement

Object-based features + Inheritance + dynamic binding

Languages that support these features include C++, Smalltalk, Object Pascal and Java,

Features of the Object Oriented Programming

The fundamental features of the OOPs are the following:

- Encapsulation
- Data Abstraction
- Inheritance
- Polymorphism
- Message Passing
- Extensibility
- Persistence

- Delegation
- Genericity
- Multiple Inheritance

Encapsulation: It is a mechanism that associates the code and the data it manipulates into a single unit and keeps them safe from external interference and misuse. In C++, this is supported by a construct called class. An **instance** of a class is known as an **object,** which represents a real-world entity.

Data Abstraction: The technique of creating new data types that are well suited ***to*** an application to be programmed is known as data abstraction. It provides the ability ***to*** create user-defined data types, for modeling a real world object, having the properties of built-in data types and a set of permitted operators. The class is a constructed in C++ for creating user-defined data types called abstract data types (ADTs).

Inheritance: It allows the extension and reuse of existing code without having to rewrite the code from scratch. Inheritance involves the creation of new classes (derived classes) from the existing ones (base classes), thus enabling the creation o f a hierarchy of classes that simulate the class and subclass concept of the real world. The new derived class inherits the members of the base class and also adds its own.

Polymorphism: It allows a single name/operator to be associated with different operations depending on the type of data passed to it. In C++, it is achieved by function overloading, operator overloading, and dynamic binding (virtual functions).

Message Passing: It is the process of invoking an operation on an object. In response to a message, the corresponding method (function) is executed in the object. It is supported in C++.

Extensibility: It is a feature, which allows the extension of the functionality of the existing software components. In C++, this is achieved through abstract classes and inheritance.
Persistence: The phenomenon where the object (data) outlives the program execution time and exists between executions o f a program is known as persistence. All database system support persistence. In C++, this is not supported. However, the user can build it explicitly using file ***streams*** in a program.

Delegation: It is an alternative to class inheritance. Delegation is a way of making object composition as powerful as inheritance. In delegation, two objects are involved in handling a request, a receiving object delegates operations to its delegate. This is analogous to the child classes sending requests to the parent classes. In C++, delegation is realized by using object composition. Here, new functionality is obtained by assembling or composing objects. This approach takes a view that an object can be a collection of many objects and the relationship is called the has-a relationship or containership.

Genericity: It is a technique for defining software components that have more than one interpretation depending on the data type of parameters. Thus, it allows the declaration of data items without specifying their exact data type. Such unknown data types (generic data type) are resolved at the time of their usage (function call) based on the data type of parameters. For example, a sort function can be parameterized by the type of elements it sorts. To invoke the parameterized sort ( ) , just supply the required data type parameters to it

and the compiler will take care of issues such as creation of actual function and invoking that transparently. In C++, genericity is realized through function templates and class templates.

Difference between C and C++

C	C++
1. C is Procedural Language.	**1.** C++ is non Procedural i.e., Object oriented Language.
2. No virtual Functions are present in C	**2.** The concept of virtual Functions are used in C++.
3. In C, Polymorphism is not possible.	**3.** The concept of polymorphism is used in C++. Polymorphism is the most Important Feature of OOPS.
4. Operator overloading is not possible in C.	**4.** Operator overloading is one of the greatest Feature of C++.
5. Top down approach is used in Program Design.	**5.** Bottom up approach adopted in Program Design.
6. No namespace Feature is present in C Language.	**6.** Namespace Feature is present in C++ for avoiding Name collision.
7. Multiple Declaration of global variables are allowed.	**7.** Multiple Declaration of global varioables are not allowed.
8. In C • scanf() Function used for Input. • printf() Function used for output.	**8.** In C++ • cin>> Function used for Input. • cout<< Function used for output.
9. Mapping between Data and Function is difficult and complicated.	**9.** Mapping between Data and Function can be used using "Objects"
10. In C, we can call main() Function through other Functions	**10.** In C++, we cannot call main() Function through other functions.
11. C requires all the variables to be defined at the starting of a scope.	**11.** C++ allows the declaration of variable anywhere in the scope i.e at time of its First use.
12. No inheritance is possible in C.	**12.** Inheritance is possible in C++
13. In C, malloc() and calloc() Functions are used for Memory Allocation and free() function	**13.** In C++, new and delete operators are used for Memory Allocating and

for memory Deallocating.	Deallocating.
14. It supports built-in and primitive data types.	**14.** It supports both built-in and user defined data types.
15. In C, Exception Handling is not present.	**15.** In C++, Exception Handling is done with Try and Catch block.