OOP – Object-Oriented Programming

1. What is Object-Oriented Programming?

Object-Oriented Programming (OOPs) is a type of programming that is based on objects rather than just functions and procedures. Individual objects are grouped into classes. OOPs implements real-world entities like inheritance, polymorphism, hiding, etc into programming. It also allows binding data and code together.

1. Basic OOP concepts:

* Abstraction – simplifying complexity;
* Encapsulation – Encapsulation refers to binding the data and the code that works on that together in a single unit;
* Inheritance – the possibility of a new class using attributes of an existing class;
* Polymorphism – refers to the ability to exist in multiple forms. An object takes many forms. A method takes a different quantity of parameters.

1. Why use OOPs?

* OOPs allow clarity in programming thereby allowing simplicity in solving complex problems;
* Code can be reused through inheritance thereby reducing redundancy;
* Data and code are bound together by encapsulation;
* OOPs allow data hiding, therefore, private data is kept confidential;
* Problems can be divided into different parts making them simple to solve;
* The concept of polymorphism gives flexibility to the program by allowing the entities to have multiple forms.

1. Why is OOPs so popular?

OOPs programming paradigm is considered as a better style of programming. Not only it helps in writing a complex piece of code easily, but it also allows users to handle and maintain them easily as well. Not only that, the main pillar of OOPs - Data Abstraction, Encapsulation, Inheritance, and Polymorphism, makes it easy for programmers to solve complex scenarios.

1. What is meant by Structured Programming?

Structured Programming refers to the method of programming which consists of a completely structured control flow. Here structure refers to a block, which contains a set of rules, and has a definitive control flow, such as (if/then/else), (while and for), block structures, and subroutines.

Nearly all programming paradigms include Structured programming, including the OOPs model.

1. What is an object?

* An object is an instance of a class;
* Different objects have different states or attributes, and behaviors.

1. What is a class?

User-defined blueprint from which objects are created. A class is a template that binds the code and data together into a single unit.

1. What is a method?

A callable set of instructions is also called a procedure or function that is to be performed on the given data.

1. What is a Structure?

A structure is a user-defined collection of variables that are of different data types.

1. Can you call the base class method without creating an instance?

Yes, you can call the base class without instantiating it if:

* It is a static method;
* The base class is inherited by some other subclass.

1. What is inheritance?

Inheritance is a feature of OOPs which allows classes to inherit common properties from other classes. This property helps you get rid of redundant code thereby reducing the overall size of the code.

1. Are there any limitations of Inheritance?

Yes, with more powers comes more complications. Inheritance is a very powerful feature in OOPs, but it has some limitations too. Inheritance needs more time to process, as it needs to navigate through multiple classes for its implementation. Also, the classes involved in Inheritance - the base class and the child class, are very tightly coupled together. So if one needs to make some changes, they might need to do nested changes in both classes. Inheritance might be complex for implementation, as well. So if not correctly implemented, this might lead to unexpected errors or incorrect outputs.

1. What are the different types of inheritance?

* Single inheritance
* Multiple inheritances
* Multilevel inheritance
* Hierarchical inheritance
* Hybrid inheritance

1. What is a superclass?

A superclass or base class is a class that acts as a parent to some other class or classes.

1. What is a subclass?

A class that inherits from another class is called the subclass.

1. What is method overloading?

Two or more methods have the same name but different parameters or signatures. Resolved during compile time.

1. What is method overriding?

Child class redefining methods present in the base class with the same parameters/ signature. Resolved during runtime.

1. Access modifiers:

* Public – gives the possibility to access from everywhere in the program;
* Private – gives the possibility to access only within the class;
* Protected – gives the possibility to access subclasses or within the same package;
* Default – gives the possibility of access only within the package.

1. What is an abstract class?

An abstract class is a class that consists of abstract methods. These methods are declared but not defined. If these methods are to be used in some subclass, they need to be exclusively defined in the subclass. Instances of an abstract class cannot be created because it does not have a complete implementation. However, instances of subclass inheriting the abstract class can be created.

1. What is an interface?

An interface refers to a special type of class, which contains methods, but not their definition. Only the declaration of methods is allowed inside an interface. To use an interface, you cannot create objects. Instead, you need to implement that interface and define the methods for their implementation.

1. What are virtual functions?

Virtual functions are functions that are present in the parent class and are overridden by the subclass. These functions are used to achieve runtime polymorphism.

1. What are pure virtual functions?

Pure virtual functions or abstract functions are functions that are only declared in the base class. This means that they do not contain any definition in the base class and need to be redefined in the subclass.

1. What is a constructor?

A constructor is a special type of method that has the same name as the class and is used to initialize objects of that class.

1. Types of constructors:

* Default constructor
* Parameterized constructor
* Copy constructor – creates objects by copying variables from another object of the same class;
* Static constructor
* Private constructor

1. What is a destructor?

A destructor is a method that is automatically invoked when an object is destroyed. The destructor also recovers the heap space that was allocated to the destroyed object, closes the files and database connections of the object, etc.

1. What is Garbage Collection (GC)?

GC is an implementation of automatic memory management. The Garbage collector frees up space occupied by objects that are no longer in existence.

1. What is the use of ‘finalize’?

Finalize as an object method used to free up unmanaged resources and cleanup before Garbage Collection (GC). It performs memory management tasks.

1. What is an error?

Error is an irrecoverable condition occurring in runtime.

1. What is an exception?

An exception is a kind of notification that interrupts the normal execution of a program. Exceptions provide a pattern to the error and transfer the error to the exception handler to resolve it. The state of the program is saved as soon as an exception is raised. Conditions that an application might try to catch.

1. What is exception handling?

Exception handling in Object-Oriented Programming is a very important concept that is used to manage errors. An exception handler allows errors to be thrown and caught and implements a centralized mechanism to resolve them.

1. What is a try/ catch block?

A try/ catch block is used to handle exceptions. The try block defines a set of statements that may lead to an error. The catch block basically catches the exception.

1. What is a finally block?

A finally block consists of code that is used to execute important code such as closing a connection, etc. This block executes when the try block exits. It also makes sure that finally block executes even in case some unexpected exception is encountered.

1. What are the limitations of OOPs?

* Usually not suitable for small problems
* Requires intensive testing
* Takes more time to solve the problem
* Requires proper planning
* The programmer should think of solving a problem in terms of objects

1. How much memory does a class occupy?

Classes do not consume any memory. They are just a blueprint based on which objects are created. Now when objects are created, they initialize the class members and methods and therefore consume memory.

1. Is it always necessary to create objects from class?

No. An object is necessary to be created if the base class has non-static methods. But if the class has static methods, then objects don’t need to be created. You can call the class method directly in this case, using the class name.

1. What is Overloading?

Overloading is a compile-time polymorphism feature in which an entity has multiple implementations with the same name.

1. What is Overriding?

Overriding is a runtime polymorphism feature in which an entity has the same name, but its implementation changes during execution.