Object Oriented Programming Interview Questions by Ammar Haider

Chapter · May 2022

CITATIONS
0

READS
7,162

1 author:

Ammar Haider Jafri
Forman Christian College
8 PUBLICATIONS **5** CITATIONS

SEE PROFILE

Object Oriented Programming Interview Questions

Author: Ammar Haider
Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

Basic OOP

***What is OOP?**

OOPs refers to Object-Oriented Programming. It is the programming technique that is defined using objects. Objects can be considered as real-world instances of entities like class, that have some characteristics and behaviours.

***Why we need OOP?**

- With OOPs, the readability, understandability, and maintainability of the code increase.
- Even very big software can be easily written and managed easily using OOP.
- Code can be reused through inheritance thereby reducing redundancy
- Data and code are bound together by encapsulation
- OOPs allows data hiding, therefore, private data is kept confidential
- Problems can be divided into different parts making it simple to solve
- The concept of polymorphism gives flexibility to the program by allowing the entities to have multiple forms.
- It allows us the reusability of code.
- Decompose a complex problem into smaller chunks.
- Programmers can reach their goals faster.

Disadvantages of OOP

- Proper planning is required.
- Program design is tricky.
- Programmer should be well skilled.
- Classes tend to be overly generalized.

***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.

***Write Basic Concept / Pillars / Features of OOP?**

There are four pillars of OOP

- Inheritance
- Encapsulation
- Abstraction
- Polymorphism

Classes and Objects

***What is Class?**

A class is a prototype that consists of objects in different states and with different behaviours. It has a number of methods that are common the objects present within that class.

***What is Object?**

An object is an instance of a class. It has its own state, behaviour, and identity.

***What is the difference between a class and a structure?**

Class: User-defined blueprint from which objects are created. It consists of methods or set of instructions that are to be performed on the objects.

Structure: A structure is basically a user-defined collection of variables which are of different data types.

***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

Inheritance

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

***What is Inheritance?**

Inheritance is a concept where one class shares the structure and behaviour defined in another class. If Inheritance applied to one class is called Single Inheritance, and if it depends on multiple classes, then it is called multiple Inheritance.

Inheritance is a feature of OOPs which allows classes inherit common properties from other classes. For example, if there is a class such as 'vehicle', other classes like 'car', 'bike', etc can inherit common properties from the vehicle class. This property helps you get rid of redundant code thereby reducing the overall size of the code.

***What are different types of Inheritance ?**

- Single inheritance
- Multiple inheritance
- Multilevel inheritance
- Hierarchical inheritance
- Hybrid inheritance

***What is Single Inheritance ?**

When a class inherited with one base class is called single inheritance.

Multiple inheritance	Multilevel inheritance
Multiple inheritance comes into picture when a class inherits more than one base class	Multilevel inheritance means a class inherits from another class which itself is a subclass of some other base class.
Example: A class defining a child inherits from two base classes Mother and Father	Example: A class describing a sports car will inherit from a base class Car which in turn inherits another class Vehicle

***What is hybrid inheritance?**

Hybrid inheritance is a combination of multiple and multi-level inheritance.

***What is hierarchical inheritance?**

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

Hierarchical inheritance refers to inheritance where one base class has more than one subclasses. For example, the vehicle class can have ‘car’, ‘bike’, etc as its subclasses.

***What are the limitations of inheritance?**

- Increases the time and effort required to execute a program as it requires jumping back and forth between different classes
- The parent class and the child class get tightly coupled
- Any modifications to the program would require changes both in the parent as well as the child class
- Needs careful implementation else would lead to incorrect results

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.

***What is a subclass / child class?**

The subclass is a part of Inheritance. The subclass is an entity, which inherits from another class. It is also known as the child class.

***Define a superclass / base class / parent class?**

Superclass is also a part of Inheritance. The superclass is an entity, which allows subclasses or child classes to inherit from itself.

***Is it possible for a class to inherit the constructor of its base class?**

No, a class cannot inherit the constructor of its base class.

***What is composition?**

Composition is one of the vital concepts in OOP. It describes a class that references one or more objects of other classes in instance variables. It allows us to model a has-a association between objects. We can find such relationships in the real world.

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

***What is the difference between Composition and Inheritance?**

Inheritance means an object inheriting reusable properties of the base class. Compositions mean that an object holds other objects. In Inheritance, there is only one object in memory (derived object) whereas, in Composition, the parent object holds references of all composed objects. From a design perspective, inheritance is "is a" relationship among objects whereas Composition is "has a" relationship among objects.

Polymorphism

***What is polymorphism?**

Polymorphism refers to the ability to exist in multiple forms. Multiple definitions can be given to a single interface. For example, if you have a class named Vehicle, it can have a method named speed, but you cannot define it because different vehicles have different speed. This method will be defined in the subclasses with different definitions for different vehicles.

***What is static polymorphism?**

Static polymorphism (static binding) is a kind of polymorphism that occurs at compile time. An example of compile-time polymorphism is method **overloading**.

***What is dynamic polymorphism?**

Runtime polymorphism or dynamic polymorphism (dynamic binding) is a type of polymorphism which is resolved during runtime. An example of runtime polymorphism is method **overriding**.

***What is method overloading?**

Method overloading is a feature of OOPs which makes it possible to give the same name to more than one methods within a class with different arguments/parameters (if the arguments passed differ).

***What is method overriding?**

Method overriding is a feature of OOPs by which the child class or the subclass can redefine methods present in the base class or parent class. Here, the method that is overridden has the same name as well as the signature meaning the arguments passed and the return type.

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

***What is operator overloading?**

Operator overloading refers to implementing operators using user-defined types based on the arguments passed along with it.

***Name the operators that cannot be overloaded?**

All the operators except the + operator cannot be overloaded.

Overloading	Overriding
Two or more methods having the same name but different parameters or signature in a same class.	Child class redefining methods present in the base class with the same parameters/ signature.
Resolved during compile-time.	Resolved during runtime.

***What is early and late Binding?**

Early binding (Overloading) refers to the assignment of values to variables during design time, whereas late Binding (overriding) refers to the assignment of values to variables during run time.

***What is a ternary operator?**

The ternary operator is said to be an operator which takes three arguments. Arguments and results are of different data types, and it depends on the function. The ternary operator is also called a conditional operator.

Encapsulation

***What is encapsulation?**

Encapsulation refers to binding the data and the code that works on that together in a single unit. For example, a class. Encapsulation also allows data-hiding as the data specified in one class is hidden from other classes.

Encapsulation is an attribute of an object, and it contains all data which is hidden. That hidden data can be restricted to the members of that class.

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

Levels are Public, Protected, Private, Internal, and Protected Internal.

***What are 'access specifiers'?**

Access specifiers or access modifiers are keywords that determine the accessibility of methods, classes, etc in OOPs. These access specifiers allow the implementation of encapsulation. The most common access specifiers are public, private, and protected. However, there are a few more which are specific to the programming languages.

Name	Accessibility from own class	Accessibility from derived class	Accessibility from world
Public	Yes	Yes	Yes
Private	Yes	No	No
Protected	Yes	Yes	No

Data Abstraction

***What is data abstraction?**

Data abstraction is a very important feature of OOPs that allows displaying only the important information and hiding the implementation details. For example, while riding a bike, you know that if you raise the accelerator, the speed will increase, but you don't know how it actually happens. This is data abstraction as the implementation details are hidden from the rider.

***How to achieve data abstraction?**

Data abstraction can be achieved through:

- Abstract class
- Abstract method

***What is an abstract class?**

An abstract class is a class that consists of abstract methods, and the name of the class also have abstract keyword. These methods are basically declared but not defined. If these methods are to be used in some subclass, they need to be exclusively defined in the subclass.

***Can you create an instance of an abstract class?**

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

No. 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.

***What is an interface?**

It is a concept of OOPs that allows you to declare methods without defining them. Interfaces, unlike classes, are not blueprints because they do not contain detailed instructions or actions to be performed. Any class that implements an interface defines the methods of the interfaces.

Data abstraction	Encapsulation
Solves the problem at the design level	Solves the problem at the implementation level
Allows showing important aspects while hiding implementation details	Binds code and data together into a single unit and hides it from the world

Methods and Functions

***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 (function overriding is achieve by implementing the virtual function).

***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 and this class is also called Abstract class (base class).

***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 and has no return type.

***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.

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

***Types of constructors ?**

Types of constructor is differed from language to language. However, all the possible constructors are:

- Default constructor
- Parameterized constructor
- Copy constructor
- Static constructor
- Private constructor

***What is a copy constructor?**

A copy constructor basically creates objects by copying variables from another object from the same class. The main focus of a copy constructor is to make a new object from an existing.

***Diff btw class and method / function?**

Class	Method
A class is basically a template that binds the code and data together into a single unit. Classes consist of methods, variables, etc	Callable set of instructions also called a procedure or function that are to be performed on the given data

***Diff btw abstract class and interface?**

Basis for comparison	Abstract Class	Interface
Methods	Can have abstract as well as other methods	Only abstract methods
Final Variables	May contain final and non-final variables	Variables declared are final by default
Accessibility of Data Members	Can be private, public, etc	Public by default
Implementation	Can provide the implementation of an interface	Cannot provide the implementation of an abstract class

***What is a final variable?**

A variable whose value does not change. It always refers to the same object by the property of non-transversely.

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

***What are manipulators?**

Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.

***What is an Inline function?**

An inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.

***What is a friend function?**

A friend function is a friend of a class that is allowed to access to Public, private, or protected data in that same class. If the function is defined outside the class cannot access such information.

A friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public, or protected.

Exception Handling

***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.

***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.

***What is the difference between an error and an exception?**

Error	Exception
Errors are problems that should not be encountered by applications	Conditions that an application might try to catch

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com

***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.

***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.

Limitations of OOPs

***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

Email: 213526914@formanite.fccollege.edu.pk
Shah34826@gmail.com