----------OOPS Interview Questions----------

1. What is Object Oriented Programming (OOPs)?

Object-oriented programming (OOPs) is a programming paradigm that is based on the concept of objects. An object is a collection of data and the methods which operate on that data.

2. Advantages of using oops ?

- OOPs is very helpful in solving very complex level of problems.
- Highly complex programs can be created, handled, and maintained easily using object-oriented programming.
- OOPs, promote code reuse, thereby reducing redundancy.
- OOPs also helps to hide the unnecessary details with the help of Data Abstraction.
- OOPs, are based on a bottom-up approach, unlike the Structural programming paradigm, which uses a top-down approach.
- Polymorphism offers a lot of flexibility in OOPs.

3. What is a Class?

It is a user-defined data type that contains the data members and member functions that operate on the data members. It is like a blueprint or template of objects having common properties and methods.

4. **What is an Object?**

An **object** is an instance of a class. Data members and methods of a class cannot be used directly. We need to create an object (or instance) of the class to use them. In simple terms, they are the actual world entities that have a state and behavior.

5. **Explain 4 Pillars of oops ?**

1. Encapsulation
2. Data Abstraction
3. Polymorphism
4. Inheritance

6. **What is encapsulation?**

Encapsulation can also be defined in two different ways:

1) **Data hiding:** Encapsulation is the process of hiding unwanted information, such as restricting access to any member of an object.

2) **Data binding:** Encapsulation is the process of binding the data members and the methods together as a whole, as a class

7. What is abstraction ?

It means showing only the necessary information and hiding the other irrelevant information from the user. Abstraction is implemented using classes and interfaces.

For example, consider a car. You only need to know how to run a car, and not how the wires are connected inside it. This is obtained using Abstraction.

8. **Explain Polymorphism ?**

The word "**Polymorphism**" means having many forms. It is the property of some code to behave differently for different contexts.

- Compile Time Polymorphism
- Runtime Polymorphism

A) Compile-Time Polymorphism

Compile time polymorphism, also known as static polymorphism or early binding is the type of polymorphism where the binding of the call to its code is done at the compile time. Method overloading or operator overloading are examples of compile-time polymorphism.

B) Runtime Polymorphism

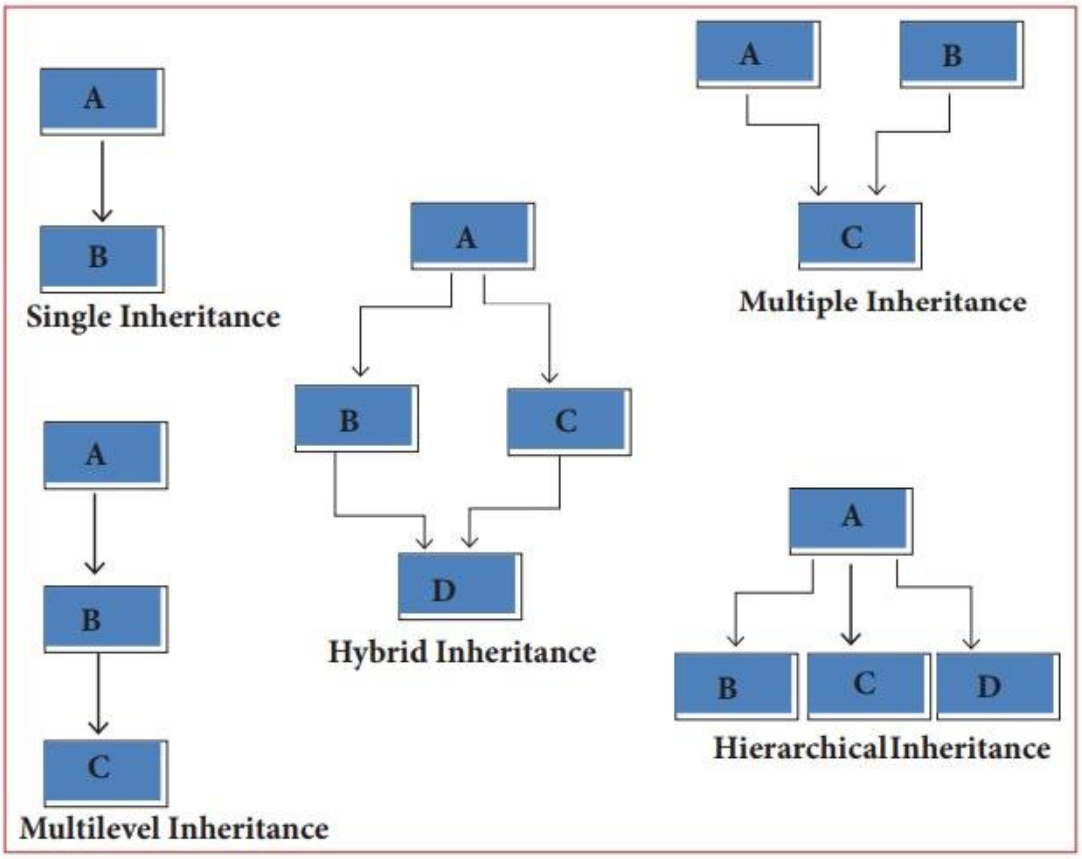
Also known as dynamic polymorphism or late binding, runtime polymorphism is the type of polymorphism where the actual implementation of the function is determined during the runtime or execution. Method overriding is an example of this method.

9. Explain Inheritance ?

In object-oriented programming, inheritance is the mechanism by which an object or class (referred to as a child) is created using the definition of another object or class (referred to as a parent). Inheritance not only helps to keep the implementation simpler but also helps to facilitate code reuse

10. **What are the various types of inheritance?**

- Single inheritance
- Multiple inheritances
- Multi-level inheritance
- Hierarchical inheritance
- Hybrid inheritance



12. What are access specifiers?

Private, Public, and Protected are examples of access specifiers or access modifiers.

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

13. **What is the difference between overloading and overriding?**

A compile-time polymorphism feature called **overloading** allows an entity to have numerous implementations of the same name. Method overloading and operator overloading are two examples.

Overriding is a form of runtime polymorphism where an entity with the same name but a different implementation is executed. It is implemented with the help of virtual functions.

14. **What is subclass ?**

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**.**

15. Define a superclass?

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

16. **What is Constructor?**

Constructors are special methods whose name is the same as the class name. The constructors serve the special purpose of initializing the objects.

17. What are the various types of constructors in C++?

The most common classification of constructors includes:

1. Default Constructor
2. Non-Parameterized Constructor
3. Parameterized Constructor
4. Copy Constructor

A. Default Constructor

The default constructor is a constructor that doesn't take any arguments. It is a non-parameterized constructor that is automatically defined by the compiler when no explicit constructor definition is provided.

It initializes the data members to their default values.

B. Non-Parameterized Constructor

It is a user-defined constructor having no arguments or parameters.

C. Parameterized Constructor

The constructors that take some arguments are known as parameterized constructors.

D. Copy Constructor

A copy constructor is a member function that initializes an object using another object of the same class.

18. What is a destructor?

A destructor is a method that is automatically called when the object is made of scope or destroyed.

In C++, the destructor name is also the same as the class name but with the (**~**) **tilde symbol** as the prefix.

19. Can we overload the constructor in a class?

Yes , We can overload the constructor in a class in Java. Constructor Overloading is done when we want constructor with different constructor with different parameter(Number and Type).

20. Can we overload the destructor in a class?

No, a destructor cannot be overloaded in a class. There can only be one destructor present in a class.

21. What is meant by Garbage Collection in OOPs world?

Object-oriented programming revolves around entities like objects. Each object consumes memory and there can be multiple objects of a class. So if these objects and their memories are not handled

properly, then it might lead to certain memory-related errors and the system might fail.

Garbage collection refers to this mechanism of handling the memory in the program. Through garbage collection, the unwanted memory is freed up by removing the objects that are no longer needed.

22. What are the characteristics of an abstract class?

An abstract class is a class that is declared as abstract. It cannot be instantiated and is always used as a base class. The characteristics of an abstract class are as follows:

- Instantiation of an abstract class is not allowed. It must be inherited.
- An abstract class can have both abstract and non-abstract methods.
- An abstract class must have at least one abstract method.
- You must declare at least one abstract method in the abstract class.
- It is always public.

23. What is constructor chaining?

In OOPs, constructor chaining is a sequence of invoking constructors (of the same class) upon initializing an object. It is used when we want to invoke a number of constructors, one after another by using only an instance.

24. Name the operators that cannot be overload.

1. Scope Resolution Operator (::)
2. Ternary Operator (? :)
3. Member Access or Dot Operator (.)
4. Pointer to Member Operator (.*)
5. sizeof operator

25. What are the types of variables in OOP?

Instance Variable: It is an object-level variable. It should be declared inside a class but must be outside a method, block, and constructor. It is created when an object is created by using the new keyword. It can be accessed directly by calling the variable name inside the class.

Static Variable: It is a class-level variable. It is declared with keyword **static** inside a class but must be outside of the method, block, and constructor. It stores in static memory. Its visibility is the same as the instance variable. The default value of a static variable is the same as the instance variable. It can be accessed by calling the **class_name.variable_name**.

Local Variable: It is a method-level variable. It can be declared in method, constructor, or block. Note that the use of an access modifier is not allowed with local variables. It is visible only to the method, block, and constructor in which it is declared. Internally, it is implemented at the stack level. It must be declared and initialized before use.

Another type of variable is used in object-oriented programming is the **reference** variable.

Reference Variable: It is a variable that points to an object of the class. It points to the location of the object that is stored in the memory.

26. **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

27. What is the virtual function?

A virtual function is a function that is used to override a method of the parent class in the derived class. It is used to provide abstraction in a class.

In C++, a virtual function is declared using the virtual keyword,

In Java, every public, non-static, and non-final method is a virtual function.

28. What is pure virtual function?

A pure virtual function, also known as an abstract function is a member function that doesn't contain any statements. This function is defined in the derived class if needed.

29. **Explain full static keyword (function / variable)?**

Refer this : [Static Keyword in C++ - GeeksforGeeks]

30. Difference between class & structure?

Class: A class is like a blueprint for creating objects in object-oriented programming (OOP). It combines data and behavior into a single unit. Objects made from a class are instances of that class, each with its own data set. Classes are essential for modeling real-world concepts and implementing OOP features like inheritance and polymorphism.

Structure: A structure is a way to group variables of different types under a single name. It's commonly used in languages like C to define custom data types holding multiple elements. Unlike classes, structures don't support methods or inheritance. They're mainly used to organize related data for easier management and manipulation within a program.

Tricky Questions

1. What is the difference between a class and an object?

- **Answer:** A class is a blueprint or template for creating objects, while an object is an instance of a class. For example, Car is a class, and Tesla Model S is an object of the Car class.

---

2. Can you create an object of an abstract class?

- **Answer:** No, you cannot instantiate an abstract class directly. It must be subclassed, and the subclass must provide implementations for all abstract methods.

3. What is the difference between method overloading and method overriding?

- **Answer:**
  - **Method Overloading:** Same method name but different parameters (compile-time polymorphism).
  - **Method Overriding:** Same method name and parameters in a subclass (runtime polymorphism).

---

4. What is the super keyword used for?

- **Answer:** The super keyword is used to refer to the immediate parent class object. It is often used to call the parent class constructor or methods.

---

5. What is the difference between == and .equals() in Java?

- **Answer:**
  - == checks for reference equality (whether two objects point to the same memory location).
  - .equals() checks for content equality (whether two objects have the same value).

---

6. What is a constructor? Can a constructor be private?

- **Answer:** A constructor is a special method used to initialize objects. Yes, a constructor can be private, often used in Singleton design patterns.

---

7. What is the difference between static and instance methods?

- **Answer:**
    - **Static methods:** Belong to the class and can be called without creating an object.
    - **Instance methods:** Belong to the object and require an instance to be called.

---

8. What is the final keyword used for?

- **Answer:**
    - For variables: Makes the value constant (cannot be changed).
    - For methods: Prevents overriding in subclasses.
    - For classes: Prevents inheritance.

---

9. What is the difference between composition and inheritance?

- **Answer:**
    - **Inheritance:** "is-a" relationship (e.g., Car is a Vehicle).
    - **Composition:** "has-a" relationship (e.g., Car has an Engine).

---

10. What is a Singleton class?

- **Answer:** A Singleton class ensures that only one instance of the class is created and provides a global point of access to it.

---

11. What is the this keyword used for?

- **Answer:** The this keyword refers to the current instance of the class. It is often used to differentiate between instance variables and parameters.

---

12. What is the difference between an interface and an abstract class?

- **Answer:**
  - **Interface:** Contains only abstract methods (no implementation) and supports multiple inheritance.
  - **Abstract Class:** Can contain both abstract and concrete methods and does not support multiple inheritance.

---

13. What is polymorphism?

- **Answer:** Polymorphism allows objects of different classes to be treated as objects of a common superclass. It can be achieved through method overriding (runtime) or method overloading (compile-time).

---

14. What is encapsulation?

- **Answer:** Encapsulation is the bundling of data (attributes) and methods (functions) that operate on the data into a single unit (class). It also restricts direct access to some of the object's components.

---

15. What is the difference between shallow copy and deep copy?

- **Answer:**

- **Shallow Copy:** Copies only the references of objects, not the objects themselves.
- **Deep Copy:** Creates a new copy of the objects and their references.

---

16. What is a static block?

- **Answer:** A static block is used to initialize static variables and is executed when the class is loaded into memory.

---

17. What is the diamond problem in inheritance?

- **Answer:** The diamond problem occurs in multiple inheritance when a class inherits from two classes that have a common ancestor. This can lead to ambiguity in method resolution.

---

18. What is the purpose of the finalize() method?

- **Answer:** The finalize() method is called by the garbage collector before an object is reclaimed. It can be used to perform cleanup operations.

---

19. What is method hiding?

- **Answer:** Method hiding occurs when a subclass defines a static method with the same signature as a static method in the superclass. The subclass method hides the superclass method.

---

20. What is the difference between private, protected, and public access modifiers?

- **Answer:**

    - private: Accessible only within the class.
    - protected: Accessible within the class, subclasses, and the same package.
    - public: Accessible from any class.

---

21. What is a marker interface?

- **Answer:** A marker interface is an empty interface used to mark a class for special behavior (e.g., Serializable in Java).

---

22. What is the difference between instanceof and getClass()?

- **Answer:**
    - instanceof checks if an object is an instance of a class or its subclass.
    - getClass() returns the exact runtime class of the object.

---

23. What is the purpose of the volatile keyword?

- **Answer:** The volatile keyword ensures that a variable's value is always read from and written to main memory, not cached in a thread's local memory.

---

24. What is the difference between ArrayList and LinkedList?

- **Answer:**
    - ArrayList: Dynamic array, fast for random access.
    - LinkedList: Doubly linked list, fast for insertions and deletions.

25. What is the transient keyword used for?

- **Answer:** The transient keyword is used to indicate that a variable should not be serialized.

---

26. What is the difference between String, StringBuilder, and StringBuffer?

- **Answer:**
  - String: Immutable.
  - StringBuilder: Mutable, not thread-safe.
  - StringBuffer: Mutable, thread-safe.

---

27. What is the purpose of the default method in interfaces?

- **Answer:** The default method allows adding new methods to interfaces without breaking existing implementations.

---

28. What is a lambda expression?

- **Answer:** A lambda expression is a concise way to represent an anonymous function, often used in functional programming.

---

29. What is the difference between Comparable and Comparator?

- **Answer:**
  - Comparable: Defines natural ordering within the class.
  - Comparator: Defines external ordering outside the class.

---

30. What is the purpose of the try-with-resources statement?

- **Answer:** The try-with-resources statement ensures that resources (e.g., files, sockets) are closed automatically after the try block is executed.

---

31. What is the difference between HashSet and TreeSet?

- **Answer:**
  - HashSet: Unordered, uses hashing.
  - TreeSet: Ordered, uses a Red-Black tree.

---

32. What is the purpose of the synchronized keyword?

- **Answer:** The synchronized keyword ensures that only one thread can access a method or block at a time, preventing race conditions.

---

33. What is the difference between HashMap and Hashtable?

- **Answer:**
  - HashMap: Not thread-safe, allows one null key.
  - Hashtable: Thread-safe, does not allow null keys.

---

34. What is the purpose of the enum type?

- **Answer:** The enum type is used to define a fixed set of constants, making the code more readable and type-safe.

---

35. What is the difference between fail-fast and fail-safe iterators?

- **Answer:**
    - fail-fast: Throws ConcurrentModificationException if the collection is modified during iteration.
    - fail-safe: Does not throw exceptions and works on a clone of the collection.

---

36. What is the purpose of the assert keyword?

- **Answer:** The assert keyword is used for debugging and testing to ensure that a condition is true during program execution.

---

37. What is the difference between checked and unchecked exceptions?

- **Answer:**
    - checked: Must be handled at compile time (e.g., IOException).
    - unchecked: Occurs at runtime (e.g., NullPointerException).

---

38. What is the purpose of the var keyword in Java?

- **Answer:** The var keyword allows type inference, letting the compiler determine the type of a variable based on its value.

---

39. What is the difference between Stream and Collection?

- **Answer:**
    - Collection: A data structure to store and manipulate objects.

- Stream: A sequence of elements supporting sequential and parallel operations.

---

40. What is the purpose of the Optional class?

- **Answer:** The Optional class is used to represent an object that may or may not contain a value, reducing the risk of NullPointerException.

---

41. What is the difference between flatMap and map in Java Streams?

- **Answer:**
  - map: Transforms each element of the stream.
  - flatMap: Transforms and flattens nested structures into a single stream.

---

42. What is the purpose of the @FunctionalInterface annotation?

- **Answer:** The @FunctionalInterface annotation indicates that an interface is intended to be a functional interface, meaning it has exactly one abstract method.

---

43. What is the difference between Predicate and Function in Java?

- **Answer:**
  - Predicate: Takes an input and returns a boolean.
  - Function: Takes an input and returns an output of any type.

---

44. What is the purpose of the BiFunction interface?

- **Answer:** The BiFunction interface represents a function that takes two arguments and produces a result.

---

45. What is the difference between parallelStream and stream?

- **Answer:**
  - stream: Processes elements sequentially.
  - parallelStream: Processes elements in parallel.

---

46. What is the purpose of the Collectors class?

- **Answer:** The Collectors class provides utility methods for collecting stream elements into collections or other data structures.

---

47. What is the difference between peek and forEach in Java Streams?

- **Answer:**
  - peek: Used for debugging and intermediate operations.
  - forEach: Used for terminal operations.

---

48. What is the purpose of the Supplier interface?

- **Answer:** The Supplier interface represents a supplier of results, often used for lazy initialization.

---

49. What is the difference between reduce and collect in Java Streams?

- **Answer:**
    - reduce: Combines elements into a single result.
    - collect: Accumulates elements into a collection.

---

50. What is the purpose of the CompletableFuture class?

- **Answer:** The CompletableFuture class is used for asynchronous programming, allowing chaining and combining of asynchronous tasks.

Reference :

GeeksforGeeks : 30 OOPs Interview Questions and Answers (2024) Updated (geeksforgeeks.org)

Simplilearn : https://www.simplilearn.com/tutorials/java-tutorial/oops-interview-questions

Interviewbit : 40+ OOPs Interview Questions and Answers (2024) - InterviewBit

Javatpoint : OOPs Interview Questions (2024) - javatpoint

KN ACADEMY
E-LEARNING