25 IMPORTANT OOPS

Interview Questions

Omkar Srivastava

@omsrivastava

Omkar
@omsrivastava

Helps job aspirants to crack FAANG product companies

Q 1. What is object-oriented Programming ?

Ans –: Object-oriented programming (OOP) is a way of writing code where you create objects that hold data and code to work with that data.

- It's like using LEGO blocks - you build things by connecting different pieces (objects) together.
- OOP helps keep code organized, making it easier to understand and work with.

Q 2. What is Inheritence ?

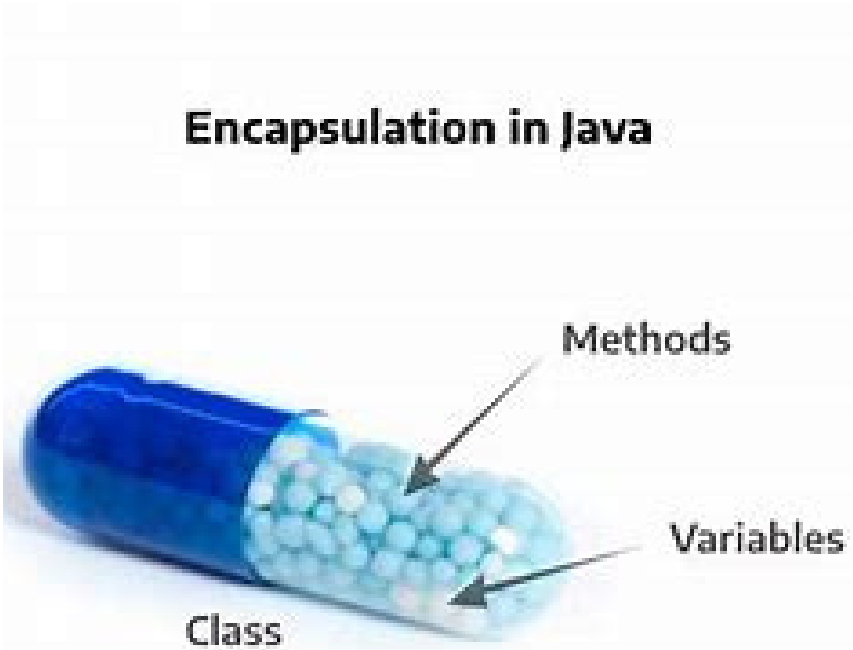
Ans –: Inheritance in object-oriented programming means a class can inherit attributes and behaviors from another class.

- This creates a hierarchy, with subclasses extending functionality from their parent class.
- It promotes code reuse, simplifies maintenance, and enables specialized classes while keeping common characteristics.

Omkar
@omsrivastava

Q 3. What is encapsulation ?

Ans -: Encapsulation in programming: Data and methods are bundled in a class, controlled via interfaces. Private elements are restricted to class methods, ensuring security and easy code maintenance.



Q 4. What is constructor ?

Ans -: A constructor in OOP is a special method that initializes an object when it's created.

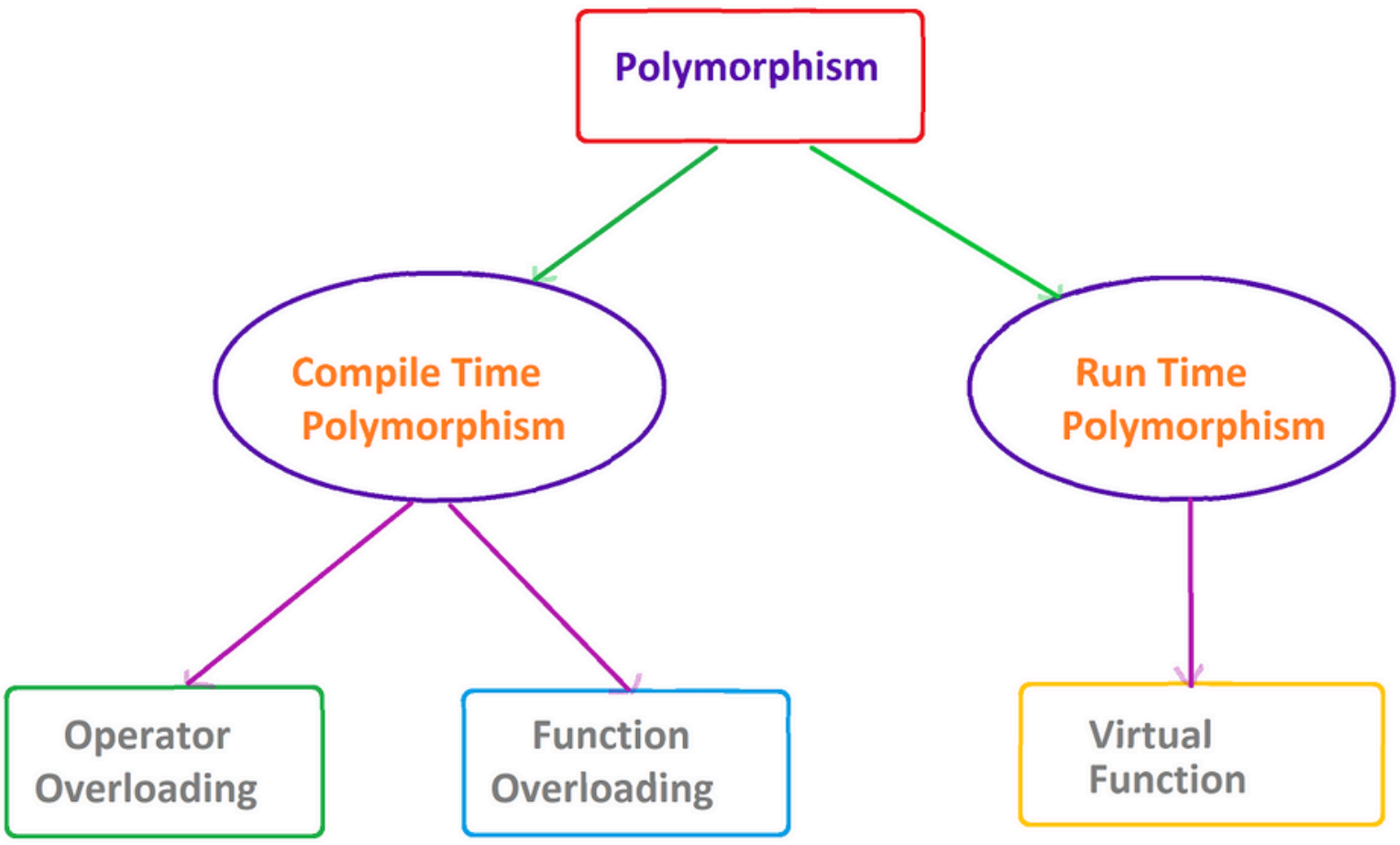
- It sets up the object's properties, often taking parameters for initial values. It's named the same as the class and handles tasks like memory allocation or variable initialization.


Omkar
@omsrivastava

Q 5. What is Polymorphism ?

Ans –: Polymorphism in OOP lets different class objects be treated as if they're from a common superclass.

- Methods can behave differently based on the object they're called on. It allows for generic code, achieved through method overriding and overloading.



Q 6. What is final class ?

Ans –: A final class in OOP cannot be subclassed or extended, ensuring its implementation remains unchanged. It's handy for utility classes or to prevent unintended modifications. They're declared using the `final` keyword in languages like Java.

Q 7. What is an abstract class ?

Ans –: An abstract class in OOP serves as a blueprint, unable to be directly instantiated. It contains abstract methods, declared but not implemented, which subclasses must define to be instantiated. Abstract classes promote code reuse and establish hierarchies by defining common behavior shared by subclasses.

Q 8. What is method hiding ?

Ans –: Method hiding happens when a subclass defines a static method with the same signature as a static method in its superclass, hiding the superclass method. It's determined at compile-time and applies to static methods, not dynamic polymorphism. It can cause confusion and is generally avoided.

Q 9. What is method overriding ?

Ans –: Method overriding in OOP lets a subclass provide its own implementation for a method inherited from its superclass. This allows for modifying or extending the behavior of the superclass method. It enables polymorphism, where objects of different types respond differently to the same method invocation.


Omkar
@omsrivastava

Q 10. What is method overloading ?

Ans –: Method overloading in OOP lets a class have multiple methods with the same name but different parameters.

- This simplifies code by allowing similar tasks to be performed with different inputs, chosen based on the parameters provided.

Q 11. What is a shallow copy and a deep copy ?

Ans –: Shallow copies duplicate only the top-level structure, while deep copies duplicate both top-level and nested objects, ensuring complete isolation.

- Shallow copies may share references, leading to potential side effects.

Omkar
@omsrivastava

Q 12. What is Composition ?

Ans –: Composition in OOP builds objects from other objects, unlike inheritance. It combines simpler components to create complex structures, promoting code organization, encapsulation, and flexibility.

Q 13. What is a Super Keyword ?

Ans –: The `super` keyword refers to the superclass of the current object. It accesses superclass members like methods and constructors. It's essential for method overriding and promotes code reusability and class hierarchy.

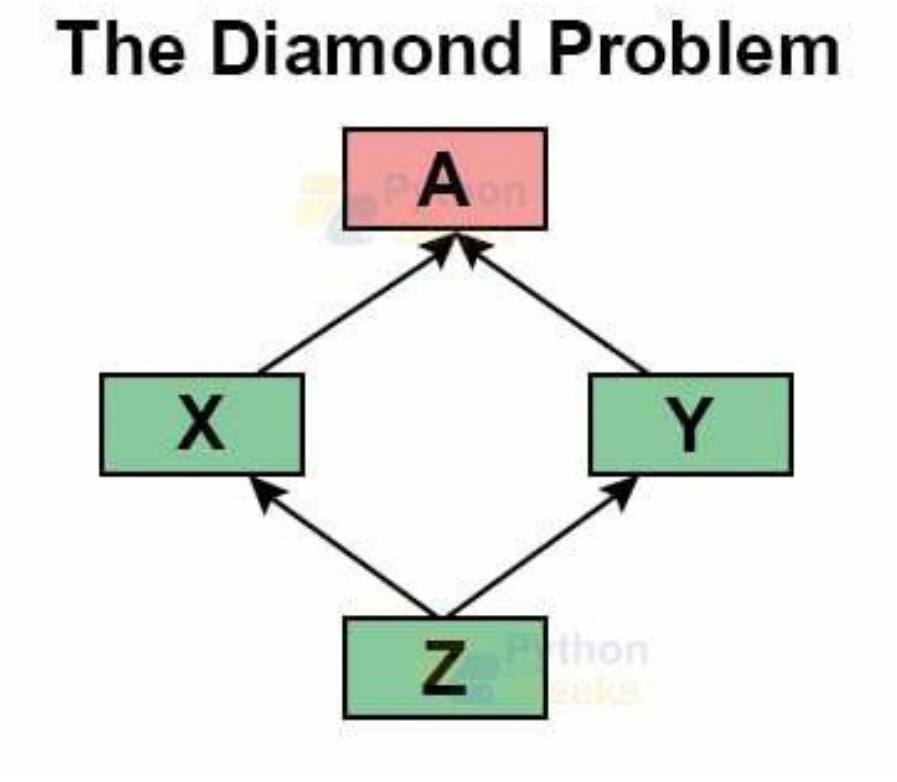
Q 14. What is a Virtual Method ?

Ans –: Virtual methods in OOP enable subclasses to override them for specialized implementations, allowing different objects to respond differently to the same method call based on their runtime types. This enhances code flexibility.

Omkar
@omsrivastava

Q 15. What is diamond problem in multiple inheritance ?

Ans -: The diamond problem arises in multiple inheritance when a class inherits from two classes with a common ancestor, causing ambiguity in method and property access. Techniques like virtual inheritance resolve this issue.



Q 16. What is static method?

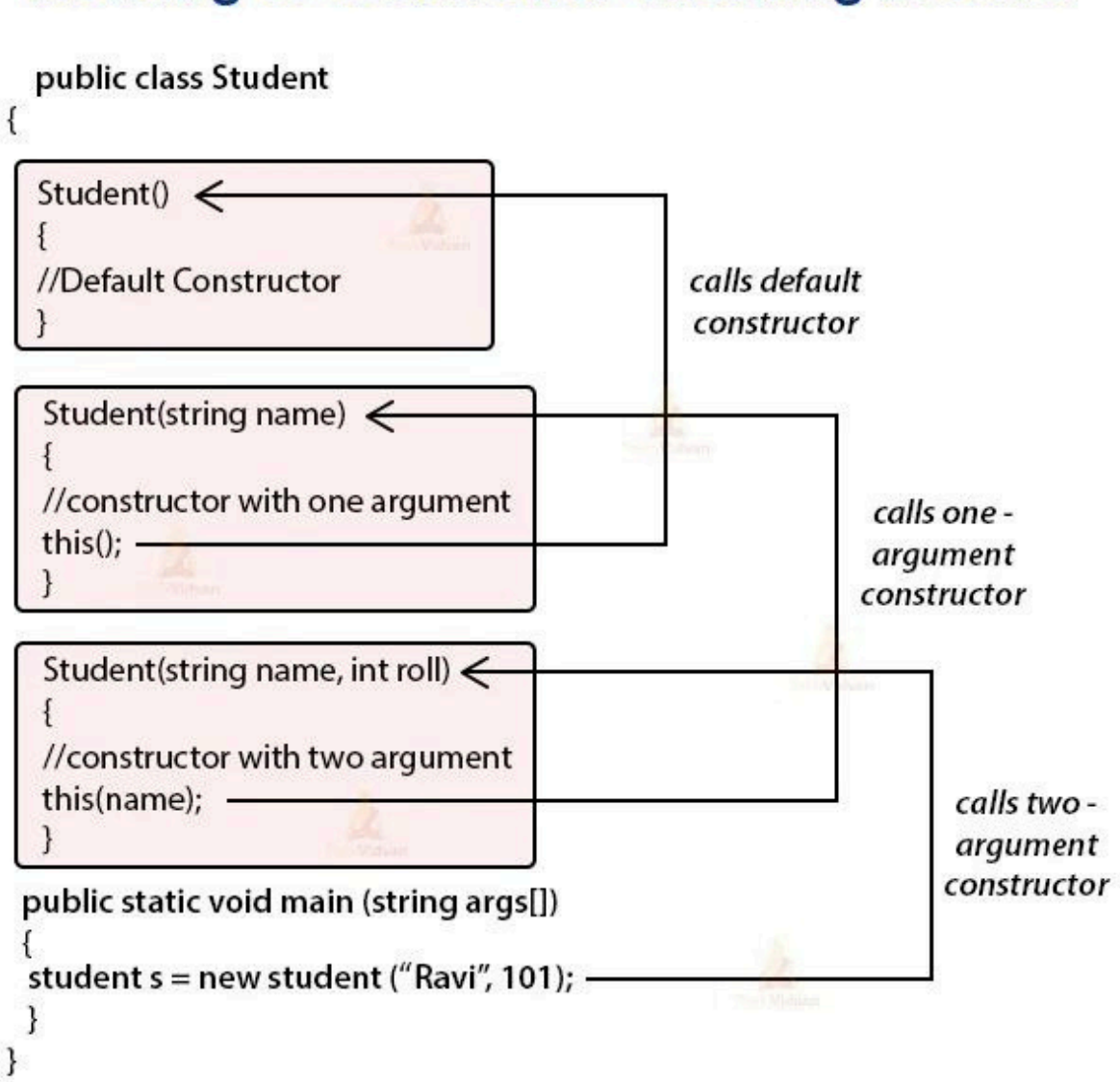
Ans -: A static method in OOP belongs to the class, not instances. It's called directly on the class without needing an instance. Static methods are used for utility functions or operations not tied to specific instances. They're declared with the `static` keyword and accessed using the class name, like `ClassName.staticMethod()`.


Omkar
@omsrivastava

Q 17. What is constructor chaining ?

Ans –: Constructor chaining in OOP involves a constructor calling another constructor within the same class or its superclass.

- It reuses initialization code and ensures consistent object initialization. In Java, it's done using `this()` for the same class or `super()` for the superclass. Constructor chaining reduces code duplication and maintains consistency in object initialization.



Q 18. How does Java achieve multiple inheritance ?

Ans -: In Java, multiple inheritance is achieved through interfaces. Classes can implement multiple interfaces, allowing them to inherit behavior from multiple sources without the complexities associated with traditional multiple inheritance.

Q 19. What is the purpose of final keyword in Java ?

Ans -: In Java, the `final` keyword indicates immutability for variables, method preservation, or class finality. It prevents modification of variables, overriding of methods, and subclassing of classes, ensuring code integrity and stability.

Q 20. How is encapsulation related to data hiding ?

Ans -: Encapsulation bundles data and methods in a class, including data hiding to restrict direct access. It enhances code organization, security, and maintainability by controlling access to implementation details.

Q 21. what is a SOLID Principle ?

Ans -: The SOLID principles in OOP promote clear design and modularity through five guidelines:
1. SRP: Single Responsibility
2. OCP: Open/Closed
3. LSP: Liskov Substitution
4. ISP: Interface Segregation
5. DIP: Dependency Inversion

Q 22. What is the difference between an abstract class and an interface ?

Ans -: Abstract classes can have both abstract and concrete methods, and instance variables, while interfaces can only have abstract methods. A class can extend only one abstract class but can implement multiple interfaces.

Q 23. what is the role of a destructor in C++ ?

Ans -: In C++, a destructor automatically releases resources when an object goes out of scope or is deleted. It prevents memory and resource leaks by deallocating memory, closing file handles, or terminating network connections. Destructors have the same name as the class preceded by a tilde (`~`).

Q 24. what is the difference between composition and inheritance?

Ans -: Composition involves combining simpler objects to create complex ones, while inheritance involves creating new classes by extending existing ones. Composition emphasizes a "has-a" relationship, while inheritance emphasizes an "is-a" relationship.

Q 25. What is an interface?

Ans -: An interface in OOP defines a set of methods that must be implemented by any class adopting it.

- It lacks implementation details, only declaring method signatures. Classes can implement multiple interfaces, enabling specific behavior.

- Interfaces enforce contracts between classes, enhancing code flexibility and modularity.