**🔷 BASICS OF OOPS (Q1–Q10)**

**1. What is OOPS?**

**Answer:**  
Object-Oriented Programming System is a programming paradigm based on **objects and classes**.

**2. What are the 4 main pillars of OOPS?**

**Answer:**

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

**3. What is a class in Java?**

**Answer:**  
A **class** is a blueprint or template for creating objects.

java

CopyEdit

class Car {

String color;

void drive() {

System.out.println("Driving...");

}

}

**4. What is an object in Java?**

**Answer:**  
An **object** is an instance of a class.

java

CopyEdit

Car myCar = new Car();

**5. Difference between class and object?**

| **Class** | **Object** |
| --- | --- |
| Blueprint/template | Real-world entity |
| Doesn’t occupy memory | Occupies memory |

**6. What is a constructor?**

**Answer:**  
A **constructor** initializes an object. It has the same name as the class.

java

CopyEdit

class Car {

Car() {

System.out.println("Constructor called");

}

}

**7. Types of constructors in Java?**

**Answer:**

* Default Constructor
* Parameterized Constructor
* Copy Constructor (manually created in Java)

**8. What is the this keyword?**

**Answer:**  
Refers to the **current object** inside a class.

java

CopyEdit

this.name = name;

**9. Can we overload constructors?**

**Answer:**  
Yes, by changing the number or type of parameters.

**10. Can we overload the main method?**

**Answer:**  
Yes, but JVM calls only the standard public static void main(String[] args).

**🔶 ENCAPSULATION (Q11–Q15)**

**11. What is Encapsulation?**

**Answer:**  
Wrapping data (variables) and methods together and restricting access using **getters/setters**.

java

CopyEdit

class Student {

private int age;

public void setAge(int a) { age = a; }

public int getAge() { return age; }

}

**12. Why use Encapsulation?**

* Data hiding
* Code maintainability
* Security

**13. What are access modifiers in Java?**

**Answer:**

* private – within class
* default – within package
* protected – package + subclass
* public – everywhere

**14. Can private data be accessed outside a class?**

**Answer:**  
Yes, using **getters and setters**.

**15. Is Java fully encapsulated?**

**Answer:**  
Yes, Java classes can be made fully encapsulated using private fields and public methods.

**🔷 ABSTRACTION (Q16–Q22)**

**16. What is Abstraction?**

**Answer:**  
Hiding internal implementation and showing only essential details.

**17. How is abstraction achieved in Java?**

* **Abstract classes**
* **Interfaces**

**18. What is an abstract class?**

**Answer:**  
A class that **cannot be instantiated** and may contain **abstract and non-abstract methods**.

java

CopyEdit

abstract class Animal {

abstract void sound();

}

**19. What is an interface?**

**Answer:**  
An interface contains **only abstract methods** (till Java 7), used to achieve **100% abstraction**.

java

CopyEdit

interface Vehicle {

void start();

}

**20. Difference between abstract class and interface?**

| **Abstract Class** | **Interface** |
| --- | --- |
| Can have method body | Only method declarations (Java 7) |
| Constructors allowed | No constructors |
| Supports variables | Only constants |

**21. Can we create an object of an abstract class?**

**Answer:**  
No, we cannot instantiate abstract classes directly.

**22. Can we have a constructor in an interface?**

**Answer:**  
No, interfaces cannot have constructors.

**🔶 INHERITANCE (Q23–Q31)**

**23. What is Inheritance?**

**Answer:**  
A mechanism where a class inherits properties and methods from another class using extends.

**24. Types of Inheritance in Java?**

1. Single
2. Multilevel
3. Hierarchical  
   (✖ Multiple via classes – not allowed)

**25. What is the super keyword?**

**Answer:**  
Used to refer to the **parent class** and access its methods or constructors.

**26. Example of single inheritance:**

java

CopyEdit

class Animal {

void eat() { System.out.println("Eating"); }

}

class Dog extends Animal {

void bark() { System.out.println("Barking"); }

}

**27. Is multiple inheritance supported in Java?**

**Answer:**  
Not via classes (due to ambiguity). Supported via **interfaces**.

**28. What is constructor chaining?**

**Answer:**  
Calling one constructor from another using this() or super().

**29. Can a subclass override constructor?**

**Answer:**  
No, constructors are **not inherited**, so they can't be overridden.

**30. What is IS-A and HAS-A relationship?**

* **IS-A** → Inheritance
* **HAS-A** → Composition (object inside another class)

**31. Can a class extend multiple classes in Java?**

**Answer:**  
No. Java does **not support multiple inheritance** using classes.

**🔷 POLYMORPHISM (Q32–Q40)**

**32. What is Polymorphism?**

**Answer:**  
Polymorphism means "many forms". One method behaves differently based on context.

**33. Types of Polymorphism?**

1. Compile-time (Method Overloading)
2. Runtime (Method Overriding)

**34. What is Method Overloading?**

**Answer:**  
Same method name, different parameters.

java

CopyEdit

void add(int a, int b)

void add(double a, double b)

**35. What is Method Overriding?**

**Answer:**  
Child class provides a new implementation of parent method.

**36. Can we override a static method?**

**Answer:**  
No, static methods are not polymorphic.

**37. Can we overload main method?**

**Answer:**  
Yes, but JVM will always call the standard signature.

**38. What is dynamic method dispatch?**

**Answer:**  
Runtime decision of which method to call in case of method overriding.

**39. Example of overriding:**

java

CopyEdit

class Animal {

void sound() { System.out.println("Animal sound"); }

}

class Dog extends Animal {

void sound() { System.out.println("Bark"); }

}

**40. Can constructors be overloaded?**

**Answer:**  
Yes, by changing number/types of parameters.

**🔶 MISC. OOPS CONCEPTS (Q41–Q50)**

**41. What is the difference between == and .equals()?**

**Answer:**

* == → reference comparison
* .equals() → content comparison

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

| **Overloading** | **Overriding** |
| --- | --- |
| Same class | Different class |
| Compile-time | Runtime |

**43. What is an anonymous object?**

**Answer:**  
Object created without reference.

java

CopyEdit

new Car().drive();

**44. Can abstract class have main method?**

**Answer:**  
Yes, abstract class can have a main method.

**45. What is cohesion and coupling?**

* **Cohesion**: How related the methods in a class are
* **Coupling**: How dependent classes are on each other

**46. What is an inner class?**

**Answer:**  
A class defined inside another class.

**47. What is object slicing?**

**Answer:**  
When subclass object is assigned to superclass reference, subclass-specific fields/methods are not accessible.

**48. What is downcasting and upcasting?**

* **Upcasting:** Child → Parent (safe)
* **Downcasting:** Parent → Child (unsafe, needs explicit cast)

**49. What is the difference between shallow copy and deep copy?**

* **Shallow Copy:** References copied
* **Deep Copy:** New object created with copied data

**50. What is the difference between instance and static?**

| **Instance** | **Static** |
| --- | --- |
| Belongs to object | Belongs to class |
| Memory allocated per object | Only one copy shared |