**JAVA INTERVIEW QUESTIONS**

**1. What are the main features of Java?**

**Answer:** Java has several key features:

* **Platform Independence** – Java code runs on any platform via the Java Virtual Machine (JVM).
* **Object-Oriented** – Java follows the object-oriented programming paradigm.
* **Automatic Memory Management** – Garbage Collection handles memory management.
* **Multithreading** – Java supports concurrent execution of multiple threads.
* **Secure** – Java has built-in security features like bytecode verification.
* **Robust** – Java has strong memory management, exception handling, and type checking.
* **Portable** – Java programs can be easily moved across systems.

**2. What is the difference between JDK, JRE, and JVM?**

**Answer:**

* **JDK (Java Development Kit):** It includes JRE and development tools for writing and compiling Java applications.
* **JRE (Java Runtime Environment):** It contains libraries and JVM needed to run Java applications.
* **JVM (Java Virtual Machine):** It is an abstract machine that executes Java bytecode.

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

**Answer:**

* == compares **references** (memory addresses).
* .equals() compares **actual content** (for objects, if overridden).

Example:

java

CopyEdit

String s1 = new String("Hello");

String s2 = new String("Hello");

System.out.println(s1 == s2); // false (different objects)

System.out.println(s1.equals(s2)); // true (same content)

**4. What are the different types of memory in Java?**

**Answer:** Java memory is divided into:

* **Stack** – Stores method calls and local variables.
* **Heap** – Stores objects and class instances.
* **Method Area** – Stores class structures like metadata and static variables.
* **PC Register** – Stores address of the currently executing instruction.
* **Native Method Stack** – Manages native (non-Java) method calls.

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

| **Feature** | **Abstract Class** | **Interface** |
| --- | --- | --- |
| Methods | Can have both abstract and concrete methods | Only abstract methods (Java 7), default & static methods (Java 8) |
| Variables | Can have instance variables | Only public static final constants |
| Multiple Inheritance | Not supported | Supported (A class can implement multiple interfaces) |
| Constructor | Yes | No |

Example:

java

CopyEdit

abstract class Animal {

abstract void makeSound();

}

interface Pet {

void beFriendly();

}

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

| **Feature** | **String** | **StringBuffer** | **StringBuilder** |
| --- | --- | --- | --- |
| Mutability | Immutable | Mutable | Mutable |
| Thread-Safe | Yes | Yes (synchronized) | No |
| Performance | Slow (new object on modification) | Faster than String | Fastest |

Example:

java

CopyEdit

StringBuffer sb = new StringBuffer("Hello");

sb.append(" World"); // Modifies the same object

**7. What are wrapper classes in Java?**

**Answer:** Wrapper classes convert primitive types into objects.

* int → Integer
* double → Double
* boolean → Boolean
* char → Character

Example:

java

CopyEdit

int num = 10;

Integer obj = Integer.valueOf(num); // Boxing

int num2 = obj.intValue(); // Unboxing

**8. What is autoboxing and unboxing?**

**Answer:**

* **Autoboxing:** Automatic conversion of a primitive type to its corresponding wrapper class.
* **Unboxing:** Automatic conversion of a wrapper class to its primitive type.

Example:

java

CopyEdit

Integer obj = 5; // Autoboxing

int num = obj; // Unboxing

**9. What is the difference between final, finally, and finalize()?**

| **Keyword** | **Usage** |
| --- | --- |
| final | Used with variables (constant), methods (cannot override), and classes (cannot inherit) |
| finally | A block that always executes after try-catch, used for cleanup |
| finalize() | A method called by the Garbage Collector before object deletion |

Example:

java

CopyEdit

class Example {

final int value = 10;

@Override

protected void finalize() {

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

}

}

**10. What is method overloading and overriding?**

| **Feature** | **Method Overloading** | **Method Overriding** |
| --- | --- | --- |
| Definition | Same method name with different parameters | Same method name and parameters in a subclass |
| Class Relationship | Same class | Parent-child (inheritance) |
| Return Type | Can be different | Must be same or covariant |
| Access Modifier | Can be changed | Cannot reduce visibility |

Example:

java

CopyEdit

class Parent {

void show() { System.out.println("Parent"); }

}

class Child extends Parent {

@Override

void show() { System.out.println("Child"); }

}

**11. What is the difference between static and instance methods?**

* **Static Methods:** Belong to the class, called using the class name.
* **Instance Methods:** Belong to an object, called using an object reference.

Example:

java

CopyEdit

class Example {

static void staticMethod() { System.out.println("Static"); }

void instanceMethod() { System.out.println("Instance"); }

}

**12. What are the types of exceptions in Java?**

**Checked Exceptions:** Must be handled using try-catch or throws.  
Examples: IOException, SQLException.

**Unchecked Exceptions:** Occur at runtime, don’t require handling.  
Examples: NullPointerException, ArrayIndexOutOfBoundsException.

Example:

java

CopyEdit

try {

int num = 10 / 0; // ArithmeticException

} catch (ArithmeticException e) {

System.out.println("Cannot divide by zero.");

}

**13. What is the difference between ArrayList and LinkedList?**

| **Feature** | **ArrayList** | **LinkedList** |
| --- | --- | --- |
| Implementation | Resizable array | Doubly linked list |
| Access Speed | Fast (O(1) for get) | Slow (O(n) for get) |
| Insertion/Deletion | Slow (shifting required) | Fast (O(1) for add/remove) |

Example:

java

CopyEdit

ArrayList<Integer> list = new ArrayList<>();

LinkedList<Integer> linkedList = new LinkedList<>();

**14. What is the difference between HashMap and HashSet?**

| **Feature** | **HashMap** | **HashSet** |
| --- | --- | --- |
| Stores | Key-value pairs | Unique elements |
| Duplicates | Keys cannot be duplicated | No duplicates allowed |
| Performance | O(1) for get/put | O(1) for add/remove |

Example:

java

CopyEdit

HashMap<Integer, String> map = new HashMap<>();

HashSet<Integer> set = new HashSet<>();

**15. What are functional interfaces in Java?**

**Answer:**  
Functional interfaces have only **one abstract method** and can be used with **lambda expressions**.  
Example: Runnable, Callable, Predicate, Consumer.

Example:

java

CopyEdit

@FunctionalInterface

interface MyFunctionalInterface {

void sayHello();

}

public class Test {

public static void main(String[] args) {

MyFunctionalInterface obj = () -> System.out.println("Hello!");

obj.sayHello();

}

}