120 JAVA INTERVIEW QUESTIONS

1. What defines Java?

   - Java is a popular high-level, object-oriented programming language used for various applications like web, desktop, and mobile development.

2. Differentiate Java and JavaScript.

   - Java is for application development, while JavaScript adds interactivity to web pages.

3. Main principle of Java?

   - Java follows "write once, run anywhere" principle enabling execution on any platform.

4. Java's main features?

   - Java features include platform independence, OOP, garbage collection, and strong typing.

5. Define class in Java.

   - A class in Java serves as a blueprint for creating objects.

6. Explain object in Java.

   - An object in Java represents an instance of a class.

7. Purpose of method in Java?

- A method performs a specific task in Java.

8. Class vs. object difference?

- A class is a blueprint, while an object is an instance of a class.

9. What's inheritance in Java?

- Inheritance allows classes to inherit properties from others.

10. Java's inheritance types?

- Single and multiple inheritance through interfaces are supported in Java.

11. Define polymorphism in Java.

- Polymorphism enables objects to take multiple forms.

12. What are access modifiers?

- Access modifiers control visibility of classes, methods, and variables.

13. Explain encapsulation in Java.

- Encapsulation hides internal details, offering a public interface.

14. Purpose of constructor in Java?

- Constructors initialize objects in Java.

15. Difference between constructor and method?

- Constructors initialize objects automatically; methods perform tasks explicitly.

16. What's Java Virtual Machine (JVM)?

- JVM executes Java bytecode across platforms.

17. What's Java Development Kit (JDK)?

- JDK facilitates Java program development, compilation, and execution.

18. JDK vs. JRE difference?

- JDK is for development, JRE for running Java applications.

19. Define package in Java.

- Packages organize related Java classes and interfaces.

20. Abstract class vs. interface difference?

- Abstract classes can have both abstract and concrete methods; interfaces only declare abstract methods.

21. Purpose of static method in Java?

- Static methods belong to classes, callable without object instantiation.

22. Usage of "final" keyword?

- "final" makes variables constant, methods un-overridable, and classes unextendable.

23. What's method overloading?

- Method overloading defines multiple methods with the same name but different parameters.

24. What's method overriding?

- Method overriding provides a subclass's implementation of a superclass method.

25. Difference between overloading and overriding?

- Overloading changes method parameters; overriding changes method implementation.

26. Explain "this" keyword.

- "this" refers to the current class instance in Java.

27. What's a static variable?

   - Static variables are shared among class instances.

28. Purpose of "final" keyword in method parameters?

   - "final" in parameters ensures their immutability within methods.

29. Role of "static" keyword in Java?

   - "static" defines class-level variables and methods.

30. Difference between "==" and ".equals()"?

   - "==" compares object references; ".equals()" compares object values.

31. Role of "super" keyword?

   - "super" refers to the superclass in Java.

32. What's a thread in Java?

   - A thread enables concurrent execution within a program.

33. Creating and starting a thread?

   - Extend "Thread" class or implement "Runnable" interface, then call "start()".

34. Define synchronization.

- Synchronization controls access to shared resources in Java.

35. Difference between synchronized block and method?

- Block synchronizes specific code; method synchronizes entire method.

36. Purpose of "volatile" keyword?

- "volatile" ensures visibility of variable changes across threads.

37. Define exception.

- An exception is an event disrupting normal program flow.

38. Difference between checked and unchecked exceptions?

- Checked exceptions are verified at compile-time; unchecked exceptions are not.

39. Handling exceptions in Java?

- Use try-catch blocks to handle exceptions.

40. Purpose of "finally" block?

- "finally" block executes cleanup code irrespective of exception occurrence.

41. Difference between "throw" and "throws"?

- "throw" manually throws exceptions; "throws" declares exceptions in method signature.

42. Difference between checked and unchecked exceptions?

- Checked exceptions must be handled or declared; unchecked exceptions need not be.

43. Java API purpose?

- Java API provides classes and methods for application development.

44. Difference between ArrayList and LinkedList?

- ArrayList uses resizable array; LinkedList uses doubly-linked list.

45. Difference between HashSet and TreeSet?

- HashSet stores elements in no order; TreeSet maintains sorted order.

46. Difference between "equals()" and "hashCode()"?

- "equals()" compares object values; "hashCode()" calculates hash code.

47. Difference between shallow and deep copy?

- Shallow copy shares references; deep copy duplicates objects.

48. Define lambda expression.

- Lambda expression is an anonymous function for simplifying code.

49. Define functional programming.

- Functional programming emphasizes pure functions and immutable data.

50. Java 8 features for functional programming?

- Java 8 introduced lambda expressions, Stream API, and default methods.

51. Difference between interface and abstract class?

- Interface declares methods; abstract class provides method implementations.

52. Purpose of "default" keyword in interface?

- "default" defines default method implementations in interfaces.

53. Difference between BufferedReader and Scanner?

- BufferedReader efficiently reads text; Scanner parses various data types.

54. Purpose of "StringBuilder" class?

- "StringBuilder" manipulates mutable character sequences efficiently.

55. Difference between "Comparable" and "Comparator"?

- "Comparable" defines natural ordering; "Comparator" customizes sorting.

56. Purpose of "assert" keyword?

- "assert" verifies conditions during development and testing.

57. Difference between local and instance variables?

- Local variables have limited scope; instance variables are accessible to all methods.

58. Purpose of "transient" keyword?

- "transient" prevents serialization of variables.

59. Purpose of "static" block?

- "static" block initializes static variables or performs one-time tasks.

60. Purpose of "strictfp" keyword?

- "strictfp" ensures consistent floating-point calculations across platforms.

61. Difference between public and default class?

- Public class accessible from any package; default class restricted to its package.

62. Purpose of "enum" keyword?

- "enum" defines fixed set of constants.

63. Purpose of "break" and "continue"?

- "break" exits loop or switch; "continue" skips iteration.

64. Purpose of "try-with-resources"?

- "try-with-resources" automatically closes resources.

65. Purpose of "instanceof" operator?

- "instanceof" checks object type.

66. Difference between pre-increment and post-increment?

- Pre-increment returns incremented value; post-increment returns original value.

67. Difference between pre-decrement and post-decrement?

- Pre-decrement returns decremented value; post-decrement returns original value.

68. Purpose of "Math" class?

- "Math" class performs mathematical operations.

69. Purpose of "StringBuffer" class?

- "StringBuffer" manipulates mutable character sequences in a thread-safe manner.

70. Purpose of "Math.random()"?

- "Math.random()" generates random double between 0.0 and 1.0.

71. Purpose of "Character" class?

- "Character" class manipulates individual characters.

72. Purpose of "Integer" class?

- "Integer" class works with integer values.

73. Purpose of "Double" class?

- "Double" class works with double-precision floating-point values.

74. Purpose of "System" class?

- "System" class provides system resources and interaction.

75. Purpose of "File" class?

- "File" class represents and manipulates file paths.

76. Purpose of "FileNotFoundException"?

- "FileNotFoundException" handles missing files.

77. Purpose of "NullPointerException"?

- "NullPointerException" handles null references.

78. Purpose of "ArrayIndexOutOfBoundsException"?

- "ArrayIndexOutOfBoundsException" handles invalid array indices.

79. Purpose of "ArithmeticException"?

- "ArithmeticException" handles arithmetic errors.

80. Purpose of "NumberFormatException"?
- "NumberFormatException" handles invalid number formats.

Of course! Let's continue:

81. What's JavaFX?

- JavaFX is a platform for creating rich internet applications (RIAs) using Java.

82. Purpose of "FXML" in JavaFX?

- FXML is an XML-based markup language used to define JavaFX GUIs.

83. Difference between JavaFX and Swing?

- JavaFX offers modern UI controls and effects; Swing provides older UI components.

84. Purpose of "EventHandler" in JavaFX?

- "EventHandler" handles events in JavaFX applications.

85. Difference between "ActionEvent" and "MouseEvent"?

- "ActionEvent" handles user actions like button clicks; "MouseEvent" handles mouse events.

86. Purpose of "FXMLLoader" in JavaFX?

- "FXMLLoader" loads FXML files and instantiates corresponding JavaFX objects.

87. Purpose of "ObservableList" in JavaFX?

- "ObservableList" provides a dynamic list that notifies listeners of changes.

88. Purpose of "FXMLController" in JavaFX?

- "FXMLController" controls the behavior of JavaFX GUIs defined in FXML files.

89. What's JPA?

- Java Persistence API (JPA) is a Java specification for object-relational mapping (ORM).

90. Purpose of annotations in JPA?

- Annotations configure JPA entities, relationships, and queries.

91. Difference between EntityManager and EntityManagerFactory?

- EntityManager manages entity instances; EntityManagerFactory creates EntityManager instances.

92. What's Hibernate?

- Hibernate is a popular Java ORM framework implementing JPA specifications.

93. Purpose of Hibernate SessionFactory?

- SessionFactory creates Hibernate Session instances.

94. Difference between Hibernate and JPA?

- Hibernate is an ORM framework; JPA is a specification implemented by Hibernate.

95. What's JDBC?

- Java Database Connectivity (JDBC) enables Java programs to interact with databases.

96. Steps to connect to a database using JDBC?

- Load driver, establish connection, create statement, execute queries, handle results.

97. What's PreparedStatement in JDBC?

- PreparedStatement precompiles SQL statements for efficient execution.

98. Difference between Statement and PreparedStatement?

- Statement executes static SQL queries; PreparedStatement executes precompiled queries.

99. What's ResultSet in JDBC?

- ResultSet represents the result set of a database query.

100. Purpose of "SQLException" in JDBC?

- "SQLException" handles errors occurring during database operations.

101. What's ORM?

- Object-Relational Mapping (ORM) maps objects to relational database tables.

102. Purpose of "persistence.xml" in JPA?

- "persistence.xml" configures JPA entities, database connection, and properties.

103. What's the "EntityManager" interface in JPA?

- "EntityManager" manages entity instances and performs CRUD operations.

104. Purpose of "Criteria API" in JPA?

- "Criteria API" provides a type-safe way to build queries dynamically.

105. What's a JAR file?

- Java Archive (JAR) file packages Java classes, metadata, and resources.

106. Purpose of "manifest.mf" in a JAR file?

- "manifest.mf" specifies metadata about the JAR file, including main class.

107. Difference between WAR and JAR files?

- WAR (Web ARchive) packages web application resources; JAR contains Java classes.

108. What's bytecode?

- Bytecode is intermediate code compiled from Java source for JVM execution.

109. What's reflection in Java?

- Reflection allows runtime examination and manipulation of class metadata.

110. Purpose of "Class" class in reflection?

- "Class" class represents classes and interfaces during runtime.

111. Purpose of "newInstance()" method in reflection?

- "newInstance()" creates new instances of classes during runtime.

112. What's the "ClassLoader" in Java?

- "ClassLoader" loads classes into memory dynamically at runtime.

113. Purpose of "getResource()" method in ClassLoader?

- "getResource()" retrieves resources from classpath during runtime.

114. What's the "java.lang.reflect" package for?

- "java.lang.reflect" provides classes and interfaces for reflection.

115. What's serialization in Java?

- Serialization converts object state to a byte stream for storage or transmission.

116. Purpose of "Serializable" interface?

- "Serializable" marks classes as serializable for object serialization.

117. Difference between serialization and deserialization?

- Serialization converts objects to byte stream; deserialization reconstructs objects from byte stream.

118. What's the purpose of "ObjectInputStream" and "ObjectOutputStream"?

- "ObjectInputStream" reads serialized objects; "ObjectOutputStream" writes serialized objects.

119. What's the "transient" keyword for in serialization?

- "transient" excludes variables from object serialization.

120. What's the "Externalizable" interface for?

- "Externalizable" provides custom serialization and deserialization implementations.

These questions cover a wide range of topics in Java, including core language features, object-oriented programming concepts, JavaFX, database connectivity, ORM frameworks, file handling, and advanced topics like reflection and serialization.