150 Important Viva Questions for Object-Oriented Programming (CSC-122)

These questions are based on the CSC-122 Object-Oriented Programming course outline, covering topics such as Moving from C to Java, OOP principles, Java fundamentals, classes, inheritance, polymorphism, exception handling, GUI programming, and Java Database Connectivity. They are designed to be logical and time-consuming to prepare you for your viva.

Week 1: Moving from C to Java

1. Explain the key differences between procedural programming in C and object-oriented programming in Java.
2. Why is Java considered a platform-independent language? Discuss the role of JVM in achieving this.
3. Describe the history of Java and its evolution as a dominant programming language.
4. What are the Java buzzwords? Explain how each contributes to Java's design philosophy.
5. Differentiate between JDK, JRE, and JVM with a detailed example of their interaction during program execution.
6. How does Java's memory management differ from C's manual memory management?
7. Why is Java preferred over C for large-scale applications? Provide specific examples.
8. Explain how Java's "write once, run anywhere" principle is implemented in practice.
9. Compare the compilation and execution process of a C program with a Java program.
10. Discuss the limitations of Java compared to C in terms of performance and low-level operations.

Week 2: Introduction to OOP

11. Define programming paradigms and explain how OOP differs from procedural and functional programming.
12. What is abstraction in OOP? Provide a real-world example implemented in Java.
13. Explain the three core OOP principles (Encapsulation, Inheritance, Polymorphism) with Java code snippets.
14. How do encapsulation and inheritance work together to improve code modularity?
15. Discuss the advantages of OOP over procedural programming with respect to software maintenance.
16. Explain how polymorphism enables flexible and reusable code in Java.
17. Provide an example where abstraction and encapsulation are used together in a Java program.
18. How does OOP facilitate teamwork in large software projects compared to procedural programming?
19. Write a Java program demonstrating the concept of abstraction using abstract classes.

20. Discuss the challenges of transitioning from procedural to object-oriented thinking for a C programmer.

Week 3: Introduction to Java

21. Write and explain the structure of your first Java application, including the `main` method.
22. Demonstrate the use of `if` and `for` control statements in a Java program with a practical example.
23. Explain the significance of block code in Java and how it affects variable scope.
24. What are lexical issues in Java? Provide examples of naming conventions and their importance.
25. Discuss the importance of Java documentation (Javadoc) and demonstrate how to generate it for a class.
26. Compare the programming style in Java with C, focusing on readability and maintainability.
27. Write a Java program that uses nested `if` statements and explain its control flow.
28. How does Java handle whitespace compared to C? Explain with examples.
29. Explain the role of the `public static void main(String[] args)` method in Java programs.
30. Write a Java program that demonstrates proper commenting and documentation practices.

Week 4: Fundamental Elements of Language

31. List and explain all primitive data types in Java with their memory sizes and use cases.
32. What are literals in Java? Provide examples of integer, floating-point, and string literals.
33. Explain escape sequences in Java with a program that uses at least three different sequences.
34. Discuss the scope and lifetime of variables in Java with examples of local and instance variables.
35. Demonstrate type conversion and casting in Java with a program that handles both widening and narrowing conversions.
36. What is automatic type promotion in expressions? Provide a Java code example.
37. Write a Java program to create and manipulate a two-dimensional array.
38. Explain the concept of uneven multidimensional arrays in Java with a code example.
39. Discuss the differences between arrays in Java and C, focusing on memory allocation.
40. Write a Java program that demonstrates array initialization and iteration using enhanced `for` loops.

Week 5: Operators and Control Statements

41. Explain the difference between arithmetic, bitwise, relational, and logical operators in Java with examples.

42. Write a Java program that uses the ternary operator to determine the maximum of three numbers.
43. Discuss the precedence and associativity of operators in Java with a complex expression example.
44. Explain the role of selection, iteration, and jump statements in Java with code snippets.
45. Write a Java program that uses `switch` statements to handle multiple user inputs.
46. Demonstrate the use of `break` and `continue` statements in a loop with a practical example.
47. Explain how Java's bitwise operators can be used for low-level data manipulation.
48. Write a Java program that combines logical and relational operators to validate user input.
49. Discuss the advantages of using `do-while` loops over `while` loops in specific scenarios.
50. Write a Java program that demonstrates nested loops to print a pattern (e.g., a triangle).

Week 7: Classes & Methods

51. Explain the role of constructors in Java and demonstrate a parameterized constructor.
52. Write a Java program that uses the `this` keyword to resolve instance variable hiding.
53. Discuss the purpose of the `new` operator in Java with a code example.
54. Explain garbage collection in Java and the role of the `finalize()` method.
55. Write a Java program that demonstrates method overloading with different parameter types.
56. Discuss the differences between instance and static methods in Java with examples.
57. Explain how Java handles object creation and destruction compared to C++.
58. Write a Java program that uses a constructor to initialize an object's state.
59. Demonstrate the use of access specifiers (`public`, `private`, `protected`) in a Java class.
60. Explain the concept of recursion in Java with a program to calculate factorial.

Week 8: Encapsulation

61. Define encapsulation and explain how it is achieved in Java with access specifiers.
62. Write a Java program that demonstrates encapsulation using getter and setter methods.
63. Discuss the benefits of encapsulation in terms of data security and code maintenance.
64. Explain the difference between `public`, `private`, and `protected` access specifiers with examples.
65. Write a Java program that uses encapsulation to model a bank account with private fields.
66. Discuss how encapsulation prevents unauthorized access to an object's state.
67. Explain the role of references in Java and how they differ from pointers in C.
68. Write a Java program that demonstrates passing objects as parameters to methods.
69. Discuss the concept of nested classes in Java with a practical example.
70. Explain the difference between static and non-static nested classes in Java.

Week 9: Inheritance

71. Define inheritance and explain its advantages in Java with a real-world example.
72. Write a Java program that demonstrates single inheritance using the `extends` keyword.
73. Explain the role of the `super` keyword in Java with a program that uses it.
74. Discuss multilevel inheritance in Java with a code example involving three classes.
75. Write a Java program that demonstrates method overriding in an inheritance hierarchy.
76. Explain how constructors are called in an inheritance hierarchy with a code example.
77. Discuss the limitations of inheritance in Java, such as single inheritance.
78. Write a Java program that uses inheritance to model a vehicle class hierarchy.
79. Explain the difference between method overriding and method overloading with examples.
80. Discuss how inheritance promotes code reusability and extensibility in Java.

Week 10: Polymorphism

81. Define polymorphism and explain its types (compile-time and runtime) in Java.
82. Write a Java program that demonstrates method overloading (compile-time polymorphism).
83. Explain runtime polymorphism in Java with a program using method overriding.
84. Discuss the role of abstract classes in achieving polymorphism in Java.
85. Write a Java program that uses an abstract class to model a shape hierarchy.
86. Explain the use of the `final` keyword in preventing method overriding and inheritance.
87. Discuss how polymorphism improves code flexibility in large-scale applications.
88. Write a Java program that demonstrates polymorphic behavior using interfaces.
89. Explain the difference between abstract classes and interfaces in achieving polymorphism.
90. Discuss the advantages and disadvantages of polymorphism in Java programming.

Week 11: Packages & Interfaces

91. Explain the purpose of packages in Java and how they organize code.
92. Write a Java program that creates and uses a custom package.
93. Discuss the process of importing packages in Java with examples of `import` statements.
94. Explain the concept of interfaces in Java and their role in achieving abstraction.
95. Write a Java program that defines and implements an interface with multiple methods.
96. Discuss how interfaces support multiple inheritance in Java with a code example.
97. Explain the difference between partial and complete implementation of an interface.
98. Write a Java program that demonstrates extending an interface.
99. Discuss the benefits of using packages for code modularity and reusability.
100. Explain how interfaces enable polymorphic behavior in Java applications.

Week 12: Exception Handling

101. Define exception handling in Java and explain its importance in robust programming.
102. Write a Java program that demonstrates the use of `try` and `catch` blocks.

103. Explain the difference between checked and unchecked exceptions in Java with examples.
104. Write a Java program that uses the `throw` keyword to create a custom exception.
105. Discuss the role of the `throws` keyword in method declarations with a code example.
106. Explain the purpose of the `finally` block in exception handling with a practical example.
107. Write a Java program that handles multiple exceptions in a single `try` block.
108. Discuss the hierarchy of exception classes in Java, including `Throwable`, `Error`, and `Exception`.
109. Explain how exception handling in Java differs from error handling in C.
110. Write a Java program that demonstrates chaining exceptions using `initCause()`.

Week 13: File I/O

111. Explain the concept of streams in Java and differentiate between byte and character streams.
112. Write a Java program that writes data to a file using `FileOutputStream`.
113. Write a Java program that reads data from a file using `FileInputStream`.
114. Discuss the advantages of using `BufferedReader` over `FileReader` for text files.
115. Write a Java program that appends data to an existing file using `FileWriter`.
116. Explain the role of the `Serializable` interface in Java file I/O with a code example.
117. Write a Java program that copies the contents of one file to another.
118. Discuss the differences between file I/O in Java and C with respect to error handling.
119. Explain how Java handles file permissions and security during I/O operations.
120. Write a Java program that reads and displays the contents of a CSV file.

Week 14: Graphical User Interface

121. Explain the difference between AWT and Swing in Java GUI programming.
122. Write a Java program that creates a simple GUI using `JFrame` and `JButton`.
123. Discuss the role of layout managers in Java GUI design with examples of `FlowLayout` and `GridLayout`.
124. Write a Java program that demonstrates event handling with a button click.
125. Explain the steps to create a Java GUI application using Swing components.
126. Write a Java program that uses `BorderLayout` to arrange multiple components.
127. Discuss the advantages of using Swing over AWT for modern GUI applications.
128. Explain the concept of composition vs. inheritance in Java GUI design with examples.
129. Write a Java program that creates a form with text fields and labels using `GridLayout`.

130. Discuss the challenges of creating responsive GUIs in Java and how to address them.

Week 15: Java Database Connectivity

131. Explain the role of JDBC in connecting Java applications to databases.
132. Write a Java program that connects to a database using the JDBC-ODBC bridge.
133. Discuss the four types of JDBC drivers and their use cases.
134. Write a Java program that executes an SQL query using a `PreparedStatement`.
135. Explain the difference between `Statement` and `PreparedStatement` in JDBC.
136. Write a Java program that retrieves data from a database and displays it in a GUI.
137. Discuss the steps to perform database updates using JDBC with a code example.
138. Explain how JDBC handles database transactions with commit and rollback operations.
139. Write a Java program that inserts multiple records into a database table.
140. Discuss the security considerations when using JDBC in a production environment.

General and Project-Related Questions

141. How would you break down a complex problem into objects for an OOP-based solution?
142. Design a Java project that implements all OOP principles (encapsulation, inheritance, polymorphism).
143. Explain how you would choose appropriate Java libraries for a specific project.
144. Discuss the process of designing and implementing a Java project from scratch.
145. Write a Java program that integrates GUI, database connectivity, and exception handling.
146. Explain how you would test and debug a Java application to ensure it meets requirements.
147. Discuss the role of tools like Eclipse or IntelliJ IDEA in Java development.
148. How would you optimize a Java application for performance and scalability?
149. Explain the importance of code documentation and version control in Java projects.
150. Design a Java application that demonstrates socket programming for client-server communication.