Interview Q&A

1. Difference between FileReader and BufferedReader?

FileReader reads data character by character from a file source. It is an unbuffered reader that directly accesses the file system, making it slower for large files. **BufferedReader** wraps FileReader and provides a buffer, reading data in chunks. It has additional methods like readLine() for reading entire lines at once, making it more efficient for processing text files line by line.

2. What is try-with-resources?

Try-with-resources is a statement introduced in Java 7 that automatically closes any resource implementing AutoCloseable interface. The syntax is try (ResourceType resource = new Resource()) { }. It ensures resources are closed even if an exception occurs, eliminating the need for explicit finally blocks to close resources. This prevents resource leaks and makes code cleaner.

3. How to handle IOException?

IOException is handled using try-catch blocks. The syntax is try { fileOperations(); } catch (IOException e) { e.printStackTrace(); }. You can also use try-with-resources for automatic closure, propagate the exception using throws keyword, or use multiple catch blocks for different IO exceptions. Always log the exception details for debugging.

4. What are checked and unchecked exceptions?

Checked exceptions are verified at compile-time and must be caught or declared using the throws keyword. Examples include IOException, SQLException. **Unchecked exceptions** (RuntimeException) are not checked at compile-time and don't require explicit handling. Examples include NullPointerException, ArrayIndexOutOfBoundsException. Checked exceptions are used for recoverable errors.

5. How does file writing work in Java?

File writing in Java uses FileWriter or FileOutputStream. FileWriter creates a character stream to write text data. Steps: 1) Create FileWriter with file path 2) Call write() method to write data 3) Call flush() to ensure data is written 4) Close the writer to release resources. Append mode adds to existing content, overwrite mode replaces content completely.

6. What is the difference between append and overwrite mode?

Append mode (new FileWriter(file, true)) adds new data to the end of existing file content. **Overwrite mode** (new FileWriter(file, false) or default) replaces the entire file content with new data, deleting previous content. Append is useful for logging, overwrite for data replacement. Choose based on whether you want to preserve existing data.

7. What is exception propagation?

Exception propagation occurs when a method throws an exception instead of catching it, passing it to the calling method. The exception propagates up the call stack until caught by a catch block or handled at the top level. Syntax: methodA() throws IOException { ... }. Useful when a method shouldn't handle a specific exception, allowing caller to decide how to handle it.

8. How to log exceptions?

Exceptions can be logged using: 1) e.printStackTrace() for console output 2) Try-catch blocks to capture and process 3) Logging frameworks like Log4j or SLF4J 4) Custom logging methods 5) System.err.println() for error streams. Include timestamp, error message, stack trace, and context information. Avoid silent failures; always log exceptions for debugging.

9. What is a stack trace?

A stack trace is a report showing the sequence of method calls that led to an exception. It displays the method name, file name, and line number for each call in the execution stack. Format: at ClassName.methodName(FileName.java:lineNumber). Stack traces help identify where exceptions occur and understand the execution flow. Printed using e.printStackTrace().

10. When to use finally block?

Use the finally block to execute code that must run regardless of whether an exception occurred. Common uses: 1) Closing resources (now largely replaced by try-with-resources) 2) Cleanup operations 3) Releasing locks 4) Logging completion. Finally always executes except when JVM exits or threads are interrupted. Prefer try-with-resources for resource management.