Java File Handling Study Notes - Interview Ready

Table of Contents

FileInputStream & FileOutputStream

When to use: For binary files (images, videos) or when you need byte-level control.

FileInputStream - Reading Binary Data

```java
import java.io.*;

public class FileInputStreamDemo {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("data.txt")) {
            int data;
            System.out.print("File content: ");
            while ((data = fis.read()) != -1) {
                System.out.print((char) data);
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
File content: Hello World from FileInputStream!
```

FileOutputStream - Writing Binary Data

```java
import java.io.*;

public class FileOutputStreamDemo {
    public static void main(String[] args) {
        String data = "Hello World from FileOutputStream!";

        try (FileOutputStream fos = new FileOutputStream("output.txt")) {
            fos.write(data.getBytes());
            System.out.println("Data written successfully!");
            System.out.println("Bytes written: " + data.getBytes().length);
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
Data written successfully!
Bytes written: 34
```

FileReader & FileWriter

When to use: For text files with character-by-character processing.

FileReader - Reading Text Characters

```java
import java.io.*;

public class FileReaderDemo {
    public static void main(String[] args) {
        try (FileReader reader = new FileReader("sample.txt")) {
            int ch;
            System.out.print("Content: ");
            while ((ch = reader.read()) != -1) {
                System.out.print((char) ch);
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
Content: This is sample text content.
```

FileWriter - Writing Text Characters

```java
import java.io.*;

public class FileWriterDemo {
    public static void main(String[] args) {
        try (FileWriter writer = new FileWriter("writer_output.txt")) {
            writer.write("This is written by FileWriter");
            System.out.println("Text written successfully!");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
Text written successfully!
```

BufferedReader & BufferedWriter

When to use: Most common for text files - provides line-by-line reading and better performance.

BufferedReader - Reading Line by Line

```java
import java.io.*;

public class BufferedReaderDemo {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new FileReader("sample.txt"))) {
            String line;
            int lineNum = 1;
            while ((line = reader.readLine()) != null) {
                System.out.println("Line " + lineNum++ + ": " + line);
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
Line 1: First line of text
Line 2: Second line of text
Line 3: Third line of text
```

BufferedWriter - Writing with Buffer

```java
import java.io.*;

public class BufferedWriterDemo {
    public static void main(String[] args) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("buffered.txt"))) {
            writer.write("Line 1");
            writer.newLine();
            writer.write("Line 2");
            writer.newLine();
            writer.write("Line 3");
            System.out.println("Buffered writing completed!");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
Buffered writing completed!
```

Text File Handling

Best Practice: Use BufferedReader/BufferedWriter or Files class for text files.

Reading Text File (Modern Way)

```java
import java.nio.file.*;
import java.util.List;

public class TextFileRead {
    public static void main(String[] args) {
        try {
            List<String> lines = Files.readAllLines(Paths.get("data.txt"));
            System.out.println("File contents:");
            for (String line : lines) {
                System.out.println(line);
            }
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
File contents:
Welcome to Java File Handling
This is line 2
This is line 3
```

Writing Text File (Modern Way)

```java
import java.nio.file.*;
import java.util.Arrays;

public class TextFileWrite {
    public static void main(String[] args) {
        try {
            Files.write(Paths.get("output.txt"),
                       Arrays.asList("Line 1", "Line 2", "Line 3"));
            System.out.println("File written successfully!");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
File written successfully!
```

JSON File Handling

Dependency: Add Jackson to pom.xml

```xml
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.13.0</version>
</dependency>
```

Complete JSON Example

java

```java
import com.fasterxml.jackson.databind.ObjectMapper;
import java.io.File;

public class JsonFileDemo {
    public static void main(String[] args) {
        // Create object
        Person person = new Person("John", 30, "john@email.com");

        // Write JSON
        writeJson(person);

        // Read JSON
        Person readPerson = readJson();
        System.out.println("Read: " + readPerson.getName() + ", Age: " + readPerson.getAge());
    }

    static void writeJson(Person person) {
        try {
            new ObjectMapper().writeValue(new File("person.json"), person);
            System.out.println("JSON written successfully!");
        } catch (Exception e) {
            System.out.println("Write error: " + e.getMessage());
        }
    }

    static Person readJson() {
        try {
            return new ObjectMapper().readValue(new File("person.json"), Person.class);
        } catch (Exception e) {
            System.out.println("Read error: " + e.getMessage());
            return null;
        }
    }

    static class Person {
        private String name;
        private int age;
        private String email;

        public Person() {} // Required for Jackson

        public Person(String name, int age, String email) {
            this.name = name;
            this.age = age;
            this.email = email;
        }

        // Getters and setters
        public String getName() { return name; }
        public void setName(String name) { this.name = name; }
        public int getAge() { return age; }
        public void setAge(int age) { this.age = age; }
        public String getEmail() { return email; }
        public void setEmail(String email) { this.email = email; }
    }
}
```

Output:

```
JSON written successfully!
Read: John, Age: 30
```

Generated JSON file:

```json
{"name":"John","age":30,"email":"john@email.com"}
```

CSV File Handling

Dependency: Add Commons CSV to pom.xml

```xml
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-csv</artifactId>
    <version>1.9.0</version>
</dependency>
```

Complete CSV Example

```java
import org.apache.commons.csv.*;
import java.io.*;

public class CsvFileDemo {
    public static void main(String[] args) {
        // Write CSV
        writeCsv();

        // Read CSV
        readCsv();
    }

    static void writeCsv() {
        try (CSVPrinter printer = new CSVPrinter(new FileWriter("data.csv"),
                CSVFormat.DEFAULT.withHeader("Name", "Age", "Email"))) {

            printer.printRecord("John", 30, "john@email.com");
            printer.printRecord("Jane", 25, "jane@email.com");
            System.out.println("CSV written successfully!");

        } catch (Exception e) {
            System.out.println("Write error: " + e.getMessage());
        }
    }

    static void readCsv() {
        try (CSVParser parser = new CSVParser(new FileReader("data.csv"),
                CSVFormat.DEFAULT.withFirstRecordAsHeader())) {

            System.out.println("CSV Contents:");
            for (CSVRecord record : parser) {
                System.out.println(record.get("Name") + " (" + record.get("Age") +
                                 ") - " + record.get("Email"));
            }

        } catch (Exception e) {
            System.out.println("Read error: " + e.getMessage());
        }
    }
}
```

Output:

```
CSV written successfully!
CSV Contents:
John (30) - john@email.com
Jane (25) - jane@email.com
```

Excel File Handling

Dependency: Add Apache POI to pom.xml

```xml
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.0</version>
</dependency>
```

Complete Excel Example

```
java
```

```java
import org.apache.poi.xssf.usermodel.*;
import java.io.*;

public class ExcelFileDemo {
    public static void main(String[] args) {
        // Write Excel
        writeExcel();

        // Read Excel
        readExcel();
    }

    static void writeExcel() {
        try (XSSFWorkbook workbook = new XSSFWorkbook()) {
            XSSFSheet sheet = workbook.createSheet("Data");

            // Header
            XSSFRow header = sheet.createRow(0);
            header.createCell(0).setCellValue("Name");
            header.createCell(1).setCellValue("Age");

            // Data
            XSSFRow row1 = sheet.createRow(1);
            row1.createCell(0).setCellValue("John");
            row1.createCell(1).setCellValue(30);

            // Write to file
            FileOutputStream out = new FileOutputStream("data.xlsx");
            workbook.write(out);
            out.close();

            System.out.println("Excel written successfully!");

        } catch (Exception e) {
            System.out.println("Write error: " + e.getMessage());
        }
    }

    static void readExcel() {
        try (FileInputStream file = new FileInputStream("data.xlsx");
             XSSFWorkbook workbook = new XSSFWorkbook(file)) {

            XSSFSheet sheet = workbook.getSheetAt(0);
            System.out.println("Excel Contents:");
            for (int i = 1; i <= sheet.getLastRowNum(); i++) {
                XSSFRow row = sheet.getRow(i);
                String name = row.getCell(0).getStringCellValue();
                double age = row.getCell(1).getNumericCellValue();
                System.out.println(name + " - " + (int)age);
            }

        } catch (Exception e) {
            System.out.println("Read error: " + e.getMessage());
        }
    }
}
```

Output:

```
Excel written successfully!
Excel Contents:
John - 30
```

Word Document Handling

Dependency: Add Apache POI to pom.xml

```xml
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-scratchpad</artifactId>
    <version>5.2.0</version>
</dependency>
```

Complete Word Example

java

```java
import org.apache.poi.xwpf.usermodel.*;
import java.io.*;

public class WordFileDemo {
    public static void main(String[] args) {
        // Write Word document
        writeWord();

        // Read Word document
        readWord();
    }

    static void writeWord() {
        try (XWPFDocument document = new XWPFDocument()) {
            // Create paragraph
            XWPFParagraph paragraph = document.createParagraph();
            XWPFRun run = paragraph.createRun();
            run.setText("This is a sample Word document created using Java!");

            // Write to file
            FileOutputStream out = new FileOutputStream("document.docx");
            document.write(out);
            out.close();

            System.out.println("Word document written successfully!");

        } catch (Exception e) {
            System.out.println("Write error: " + e.getMessage());
        }
    }

    static void readWord() {
        try (FileInputStream file = new FileInputStream("document.docx");
             XWPFDocument document = new XWPFDocument(file)) {

            System.out.println("Word Document Contents:");
            for (XWPFParagraph paragraph : document.getParagraphs()) {
                System.out.println(paragraph.getText());
            }

        } catch (Exception e) {
            System.out.println("Read error: " + e.getMessage());
        }
    }
}
```

Output:

```
Word document written successfully!
Word Document Contents:
This is a sample Word document created using Java!
```

Key Interview Points

1. **FileInputStream/FileOutputStream**: Use for binary data
2. **FileReader/FileWriter**: Use for character data
3. **BufferedReader/BufferedWriter**: Most common, use for line-by-line text processing
4. **Files class (NIO.2)**: Modern approach, simpler syntax
5. **Always use try-with-resources** for automatic resource management
6. **Jackson for JSON**, **Commons CSV for CSV**, **Apache POI for Excel/Word**

Common Interview Questions

Q: When to use BufferedReader vs FileReader? A: BufferedReader for better performance and line-by-line reading. FileReader for simple character-by-character reading.

Q: Difference between FileInputStream and FileReader? A: FileInputStream for binary data (bytes), FileReader for text data (characters).

Q: How to handle large files efficiently? A: Use buffered streams and process line by line instead of loading entire file in memory.