🌟 1️⃣ What is File Handling?

👉 Simple Meaning:

File Handling means:

Storing data permanently in a file
Reading data from a file
Updating data in a file

Because:

🔹 Variables store data temporarily (RAM)
🔹 Files store data permanently (Disk)

🌟 2️⃣ Why Do We Need File Handling?

Without files:

- Data lost after program ends ❌
- No persistence ❌

With files:

- Save student data
- Save logs
- Save configuration
- Save reports

🌟 3️⃣ Where is File Stored?

📍 On Disk (Hard Drive / SSD)

Memory Difference:

RAM	Disk
Temporary	Permanent
Fast	Slower
Volatile	Non-Volatile

🌟 4️⃣ Java File Handling Package

All file classes are in:

java.io

IO = Input Output

🌟 5️⃣ Types of File Streams

Java works using **Streams**

A stream is:

A flow of data from source to destination

Two types:

1️⃣ Byte Stream

- Works with binary data
- Example: images, pdf
- Classes:
  - FileInputStream
  - FileOutputStream

2️⃣ Character Stream

- Works with text data
- Uses Unicode
- Classes:
  - FileReader
  - FileWriter

You are mostly using this in exams.

🌟 6️⃣ Core Classes in File Handling

Let’s understand one by one deeply.

🔹 1️⃣ File Class

Used to:

- Create file
- Delete file
- Rename file
- Get file info

Example:

File file = new File("data.txt");

It does NOT read/write data.
It only manages file metadata.

🔹 2️⃣ FileWriter

Used to write characters into file.

FileWriter fw = new FileWriter("data.txt");

fw.write("Hello");

fw.close();

Important Points:

- Overwrites file by default
- Use true for append mode
- Must close file

🔹 3️⃣ FileReader

Used to read characters from file.

FileReader fr = new FileReader("data.txt");

int ch = fr.read();

Returns:

- Character as int
- -1 when file ends

🔹 4️⃣ BufferedWriter

Wraps FileWriter

Why needed?

Because:

Without buffer:

- Every write → directly goes to disk (slow)

With buffer:

- Data stored in memory buffer

- Written in bulk
- Faster

BufferedWriter bw = new BufferedWriter(new FileWriter("data.txt"));

🔹 5️⃣ BufferedReader

Wraps FileReader

Advantage:

- Reads line by line
- Faster than FileReader

BufferedReader br = new BufferedReader(new FileReader("data.txt"));

🔹 6️⃣ PrintWriter

Advanced writer

Provides:

- println()
- printf()

Easy formatting.

🌟 7️⃣ Internal Working (Memory Flow)

Let’s understand what happens internally when writing:

Without Buffer:

Program → FileWriter → Disk

Each write → Disk access → Slow

With Buffer:

Program → BufferedWriter → Memory Buffer → Disk

Data stored in memory first
Then written to disk in bulk
Much faster

🌟 8️⃣ Why Close is Important?

When writing:

Data is first stored in:

👉 Memory Buffer

If you don’t close:

❌ Data may not go to disk
❌ File may remain incomplete

close() does:

1️⃣ Flush buffer
2️⃣ Release resource
3️⃣ Free memory

🌟 9️⃣ Append Mode

Default:

new FileWriter("data.txt");

It deletes old content.

Append:

new FileWriter("data.txt", true);

Adds content at end.

🌟 🔟 Reading Methods Comparison

Class	Method	Reads
FileReader	read()	character
BufferedReader	readLine()	line
Scanner	nextLine()	line
Scanner	next()	word

🌟 1️⃣1️⃣ Important File Operations

Using File class:

file.exists();

file.length();

file.delete();

```
file.renameTo();

file.getAbsolutePath();
```

🌟 1️⃣ 2️⃣ File Handling Workflow (Full Flow)

Writing:

1️⃣ Create FileWriter
2️⃣ Wrap with BufferedWriter (optional but recommended)
3️⃣ Write data
4️⃣ Close file

Reading:

1️⃣ Create FileReader
2️⃣ Wrap with BufferedReader
3️⃣ Read line by line
4️⃣ Close file

🌟 1️⃣ 3️⃣ Common Exceptions in File Handling

Exception	When Occurs
IOException	General IO error
FileNotFoundException	File not found
EOFException	End of file
SecurityException	No permission

🌟 1️⃣ 4️⃣ try-with-resources (Modern Way)

Best way to handle file closing:

```
try (BufferedReader br = new BufferedReader(new FileReader("data.txt"))) {

   System.out.println(br.readLine());

}
```

Auto closes file.

🌟 1️⃣ 5️⃣ File Handling vs Database

File	Database
Simple storage	Structured storage
No relations	Supports relations
Hard to query	Easy querying

🌟 1️⃣6️⃣ Common Mistakes Students Make

❌ Forgetting close()
❌ Not handling exceptions
❌ Using FileWriter without append when needed
❌ Using FileReader for large files instead of BufferedReader

🌟 1️⃣7️⃣ Performance Understanding

FileReader → Slow
BufferedReader → Fast

FileWriter → Slow
BufferedWriter → Fast

Because disk access is costly.

🌟 1️⃣8️⃣ Real World Usage

- Logging systems
- Saving user preferences
- Exporting reports
- Reading config files
- Importing CSV data

🌟 1️⃣9️⃣ Important Differences

FileWriter vs BufferedWriter

FileWriter	BufferedWriter
Direct write	Uses buffer
Slower	Faster
No newLine()	Has newLine()

FileReader vs BufferedReader

FileReader	**BufferedReader**
Char by char	Line by line
Slower	Faster

🌟 2️⃣0️⃣ Big Picture Understanding

File Handling =

👉 Managing files
👉 Reading data
👉 Writing data
👉 Appending data
👉 Deleting/renaming
👉 Getting file info

All through:

Streams + Buffer + IO Classes

🎯 Final Clear Concept

Java File Handling is built on:

1️⃣ Streams
2️⃣ Buffering
3️⃣ Character / Byte handling
4️⃣ Resource management