Java File Handling

Here's a **beginner-friendly guide** to **File Handling in Java**, starting from scratch with step-by-step **code examples** and **explanations**.

🔰 What is File Handling?

File Handling in Java means **reading**, **writing**, **creating**, and **deleting** files. Java provides classes in the **`java.io`** and **`java.nio.file`** packages for handling files.

📦 Basic Classes Used in File Handling

Class	Purpose
`File`	Create, check, delete a file
`FileReader`	Read data from a file
`BufferedReader`	Read text efficiently
`FileWriter`	Write data to a file
`BufferedWriter`	Write text efficiently
`Scanner`	Read file using Scanner
`Files` (Java 7+)	File operations (modern way)

1️⃣ Creating a New File

```
java


import java.io.File;
import java.io.IOException;

public class CreateFileExample {
    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            if (file.createNewFile()) {
                System.out.println("File created: " + file.getName());
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

✅ Explanation:

- **`File file = new File("example.txt");`** → Creates a File object.
- **`createNewFile()`** → Creates the actual file if it doesn't exist.

- Always wrap in `try-catch` to handle `IOException`.

2️⃣ Writing to a File

```java
import java.io.FileWriter;
import java.io.IOException;

public class WriteToFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("example.txt");
            writer.write("Hello, Java File Handling!\nWelcome to file writing.");
            writer.close();  // Always close the writer
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

✅ Explanation:

- `FileWriter` writes characters to the file.
- `write()` method writes content.
- `close()` must be called to flush and close the writer.

3️⃣ Reading from a File (Using `Scanner`)

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class ReadFromFile {
    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            Scanner scanner = new Scanner(file);
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();  // Reads one line at a time
                System.out.println(line);
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
            e.printStackTrace();
        }
    }
}
```

4️⃣ Appending to a File (Preserve existing content)

```java
import java.io.FileWriter;
import java.io.IOException;

public class AppendToFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("example.txt", true); // true = append mode
            writer.write("\nThis line is appended.");
            writer.close();
            System.out.println("Data appended to the file.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

5️⃣ Deleting a File

```java
import java.io.File;

public class DeleteFile {
    public static void main(String[] args) {
        File file = new File("example.txt");
        if (file.delete()) {
            System.out.println("Deleted the file: " + file.getName());
        } else {
            System.out.println("Failed to delete the file.");
        }
    }
}
```

🧠 Summary

Task	Class Used
Create file	**`File`**
Write file	**`FileWriter`**
Read file	**`Scanner`**, **`BufferedReader`**
Append file	**`FileWriter(true)`**
Delete file	**`File.delete()`**

📝 Practice Assignment

1. Create a file **`notes.txt`**.
2. Write multiple lines of text to it.
3. Read and display the content.
4. Append a timestamp to the file.
5. Delete the file after confirmation.

Would you like me to bundle all of these in a ready-to-run Java project or provide more advanced file handling (like using **`BufferedReader`**, **`BufferedWriter`**, or **`NIO`**)?