Introduction to File Handling in Java

Marcin Kurzyna

December 10, 2024

Contents

1 Introduction

File handling is a key skill that allows programmers to store, retrieve, and manage data persistently. This document describes:

- How to read data from a file.
- How to write data to a file.
- Best practices for file handling in Java.

2 Reading from a File

To read data from a file in Java, the `Scanner` class or the `BufferedReader` class can be used. Below is an example demonstrating how to use `Scanner` to read data from a text file.

2.1 Example Code

```java
// Import required classes
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
```

```java

public class ReadFileExample {
    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            Scanner scanner = new Scanner(file);

            // Read data line by line
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                System.out.println(line);
            }

            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage()
               );
        }
    }
}
```

2.2 Explanation

- `File` class is used to represent the file.
- `Scanner` reads the file line by line.
- The `FileNotFoundException` is handled to manage errors when the file is missing.

3 Writing to a File

Writing data to a file is essential for storing output or logs. Java provides the `FileWriter` and `BufferedWriter` classes for this purpose.

3.1 Example Code

```java
// Import required classes
import java.io.FileWriter;
import java.io.IOException;

public class WriteFileExample {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("output.txt");

            // Write data to the file
            writer.write("Hello, World!\n");
            writer.write("This is a test file.\n");

            writer.close();
            System.out.println("Data successfully written to file.");
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

3.2 Explanation

- `FileWriter` creates or overwrites a file.
- The `write()` method is used to write data to the file.
- The `IOException` is handled to manage errors during file operations.

4 Best Practices for File Handling

- Always close file streams to avoid memory leaks.
- Use `try-with-resources` for automatic resource management.
- Handle exceptions gracefully to prevent application crashes.

For further reading, visit the official Java documentation: Java File I/O Tutorial.