# **Java File Handling - Complete Student Notes**

#### **Table of Contents**

- 1. Introduction to File Handling
- 2. File and FileWriter Classes
- 3. Reading Files
- 4. Writing Files
- 5. BufferedReader and BufferedWriter
- 6. Scanner for File Reading
- 7. File Operations
- 8. Exception Handling
- 9. Complete Examples

#### 1. Introduction to File Handling {#introduction}

File handling in Java allows you to read from and write to files stored on your computer. Java provides several classes in the java.io package for file operations.

## **Key Classes:**

- (File) Represents a file or directory
- (FileReader) Reads character files
- (FileWriter) Writes character files
- (BufferedReader) Efficient reading of text
- (BufferedWriter) Efficient writing of text
- (Scanner) Reading formatted input

## 2. File and FileWriter Classes {#file-filewriter}

### **Creating and Writing to Files**

java			

```
import java.io.*;
public class BasicFileWrite {
  public static void main(String[] args) {
     try {
       // Create a File object
        File myFile = new File("example.txt");
       // Create FileWriter object
        FileWriter writer = new FileWriter(myFile);
       // Write content to file
       writer.write("Hello, World!\n");
       writer.write("This is file handling in Java.\n");
       writer.write("Learning is fun!");
       // Close the writer
       writer.close();
        System.out.println("Successfully wrote to the file.");
     } catch (IOException e) {
        System.out.println("An error occurred: " + e.getMessage());
  }
}
```

#### **Appending to Files**

java

```
import java.io.*;

public class AppendToFile {
    public static void main(String[] args) {
        try {
            // Create FileWriter with append mode (true parameter)
            FileWriter writer = new FileWriter("example.txt", true);

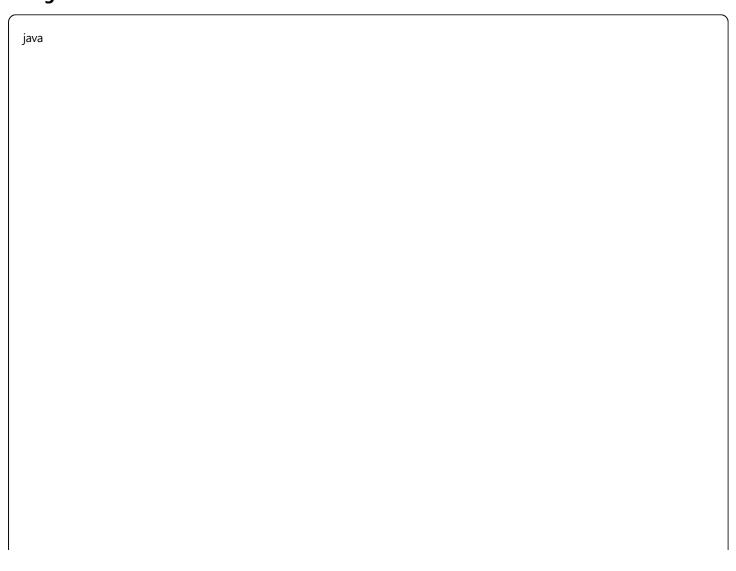
            writer.write("\nThis line is appended to the file.");
            writer.close();

            System.out.println("Content appended successfully.");

        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

## 3. Reading Files {#reading-files}

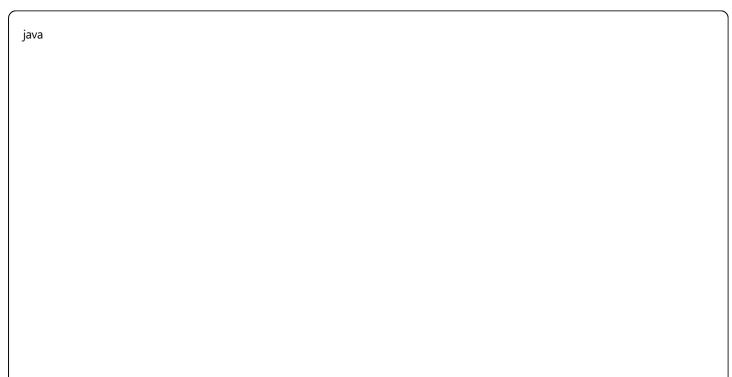
## **Using FileReader**



```
import java.io.*;
public class BasicFileRead {
  public static void main(String[] args) {
     try {
       File myFile = new File("example.txt");
       FileReader reader = new FileReader(myFile);
       int character;
       System.out.println("File contents:");
       // Read character by character
       while ((character = reader.read()) != -1) {
          System.out.print((char) character);
       }
       reader.close();
     } catch (FileNotFoundException e) {
       System.out.println("File not found: " + e.getMessage());
     } catch (IOException e) {
       System.out.println("Error reading file: " + e.getMessage());
     }
  }
}
```

## 4. Writing Files {#writing-files}

## **Complete File Writing Example**



```
import java.io.*;
import java.util.Scanner;
public class WriteStudentData {
  public static void main(String[] args) {
     Scanner input = new Scanner(System.in);
     try {
       FileWriter writer = new FileWriter("students.txt");
       System.out.println("Enter student details (type 'exit' to stop):");
       while (true) {
          System.out.print("Student Name: ");
          String name = input.nextLine();
          if (name.equalsIgnoreCase("exit")) {
            break;
          }
          System.out.print("Student ID: ");
          String id = input.nextLine();
          System.out.print("Grade: ");
          String grade = input.nextLine();
          // Write to file
          writer.write("Name: " + name + "\n");
          writer.write("ID: " + id + "n");
          writer.write("Grade: " + grade + "\n");
          writer.write("-----\n");
       }
       writer.close();
       System.out.println("Student data saved successfully!");
     } catch (IOException e) {
       System.out.println("Error: " + e.getMessage());
     input.close();
  }
}
```

## 5. BufferedReader and BufferedWriter {#buffered-classes}

#### **BufferedWriter Example**

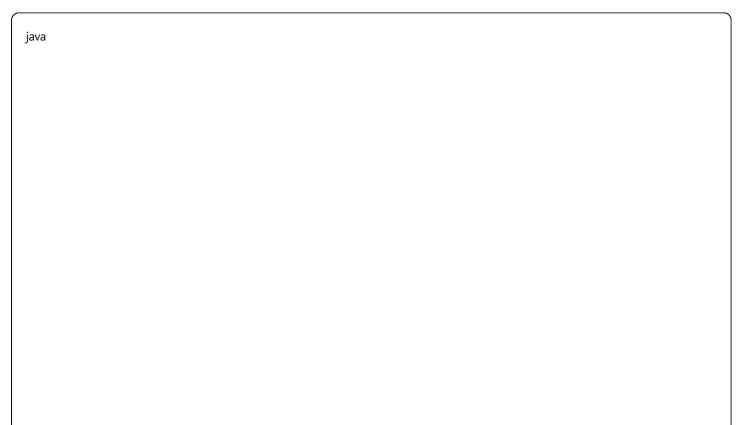
```
java
import java.io.*;
public class BufferedWriteExample {
  public static void main(String[] args) {
     try {
       FileWriter fileWriter = new FileWriter("buffered_example.txt");
       BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
       // Write multiple lines efficiently
       bufferedWriter.write("Line 1: Introduction to Java");
       bufferedWriter.newLine();
       bufferedWriter.write("Line 2: Object-Oriented Programming");
       bufferedWriter.newLine();
       bufferedWriter.write("Line 3: File Handling");
       bufferedWriter.newLine();
       // Always close BufferedWriter
       bufferedWriter.close();
       System.out.println("File written using BufferedWriter.");
     } catch (IOException e) {
       System.out.println("Error: " + e.getMessage());
     }
  }
}
```

## **BufferedReader Example**

java

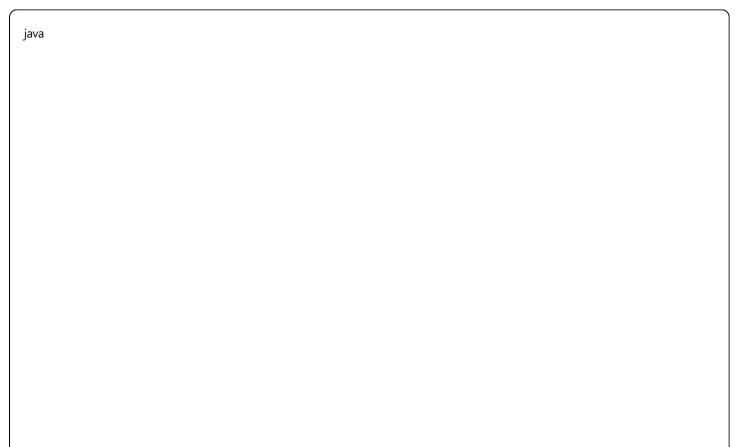
```
import java.io.*;
public class BufferedReadExample {
  public static void main(String[] args) {
       FileReader fileReader = new FileReader("buffered_example.txt");
       BufferedReader bufferedReader = new BufferedReader(fileReader);
       String line;
       System.out.println("Reading file line by line:");
       // Read line by line
       while ((line = bufferedReader.readLine()) != null) {
          System.out.println(line);
       }
       bufferedReader.close();
     } catch (FileNotFoundException e) {
       System.out.println("File not found: " + e.getMessage());
     } catch (IOException e) {
       System.out.println("Error reading file: " + e.getMessage());
  }
}
```

## 6. Scanner for File Reading {#scanner-file}



```
import java.io.*;
import java.util.Scanner;
public class ScannerFileRead {
  public static void main(String[] args) {
     try {
       File file = new File("example.txt");
       Scanner scanner = new Scanner(file);
       System.out.println("Reading file using Scanner:");
       // Read 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());
     }
  }
}
```

## **Reading Different Data Types with Scanner**



```
import java.io.*;
import java.util.Scanner;
public class ScannerDataTypes {
  public static void createDataFile() {
     try {
       FileWriter writer = new FileWriter("data.txt");
       writer.write("John 25 85.5\n");
       writer.write("Alice 22 92.0\n");
       writer.write("Bob 24 78.5\n");
       writer.close();
     } catch (IOException e) {
       System.out.println("Error creating file: " + e.getMessage());
     }
  }
  public static void main(String[] args) {
     // First create the data file
     createDataFile();
     try {
       File file = new File("data.txt");
       Scanner scanner = new Scanner(file);
       System.out.println("Student Information:");
       System.out.println("Name\tAge\tScore");
        System.out.println("----");
       while (scanner.hasNext()) {
          String name = scanner.next();
          int age = scanner.nextInt();
          double score = scanner.nextDouble();
          System.out.println(name + "\t" + age + "\t" + score);
       }
       scanner.close();
     } catch (FileNotFoundException e) {
       System.out.println("File not found: " + e.getMessage());
  }
}
```

## 7. File Operations {#file-operations}

#### **File Information and Operations**

```
java
import java.io.*;
public class FileOperations {
  public static void main(String[] args) {
     File file = new File("example.txt");
     // Check if file exists
     if (file.exists()) {
        System.out.println("File exists!");
       // File information
        System.out.println("File name: " + file.getName());
        System.out.println("Absolute path: " + file.getAbsolutePath());
        System.out.println("File size: " + file.length() + " bytes");
        System.out.println("Can read: " + file.canRead());
        System.out.println("Can write: " + file.canWrite());
        System.out.println("Is directory: " + file.isDirectory());
        System.out.println("Is file: " + file.isFile());
     } else {
        System.out.println("File does not exist.");
       // Create new file
        try {
          if (file.createNewFile()) {
             System.out.println("File created successfully.");
          }
       } catch (IOException e) {
          System.out.println("Error creating file: " + e.getMessage());
       }
     }
     // Delete file (uncomment to test)
     // if (file.delete()) {
     // System.out.println("File deleted successfully.");
     //}
  }
```

### **Directory Operations**

```
import java.io.*;
public class DirectoryOperations {
  public static void main(String[] args) {
     File directory = new File("test_directory");
     // Create directory
     if (directory.mkdir()) {
        System.out.println("Directory created successfully.");
     } else {
        System.out.println("Directory already exists or couldn't be created.");
     // List files in current directory
     File currentDir = new File(".");
     String[] files = currentDir.list();
     System.out.println("\nFiles in current directory:");
     if (files != null) {
        for (String fileName: files) {
          System.out.println(fileName);
       }
     }
     // List files with File objects
     File[] fileObjects = currentDir.listFiles();
     System.out.println("\nDetailed file information:");
     if (fileObjects != null) {
        for (File f : fileObjects) {
          if (f.isFile()) {
             System.out.println("File: " + f.getName() + " (" + f.length() + " bytes)");
          } else if (f.isDirectory()) {
             System.out.println("Directory: " + f.getName());
       }
  }
```

## 8. Exception Handling {#exception-handling}

## **Try-with-Resources (Recommended)**

java

```
import java.io.*;
public class TryWithResources {
  public static void writeFile() {
     // Try-with-resources automatically closes the file
     try (FileWriter writer = new FileWriter("auto_close.txt")) {
       writer.write("This file will be automatically closed.");
       writer.write("\nEven if an exception occurs!");
       System.out.println("File written successfully.");
     } catch (IOException e) {
       System.out.println("Error writing file: " + e.getMessage());
     }
  }
  public static void readFile() {
     try (BufferedReader reader = new BufferedReader(new FileReader("auto_close.txt"))) {
       while ((line = reader.readLine()) != null) {
          System.out.println(line);
       }
     } catch (IOException e) {
       System.out.println("Error reading file: " + e.getMessage());
     }
  }
  public static void main(String[] args) {
     writeFile();
     readFile();
  }
```

## 9. Complete Examples {#complete-examples}

### **Student Grade Management System**

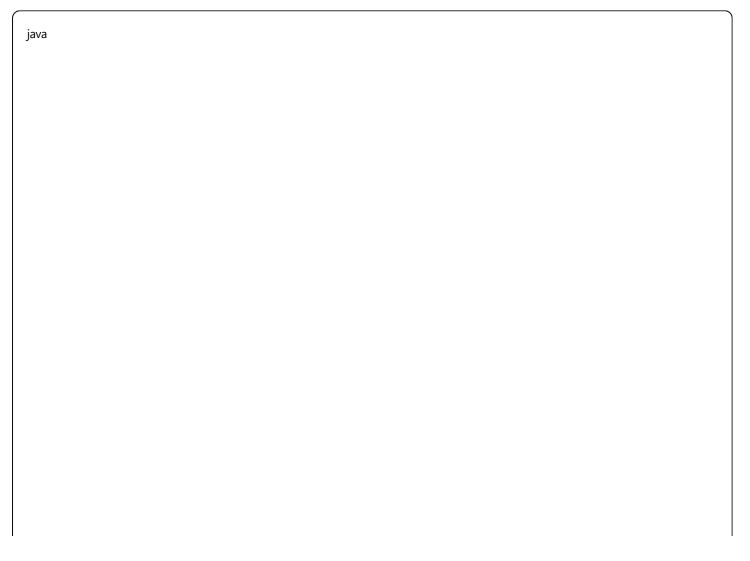


```
import java.io.*;
import java.util.Scanner;
public class StudentGradeManager {
  private static final String FILENAME = "student_grades.txt";
  public static void addStudent() {
    Scanner input = new Scanner(System.in);
    try (FileWriter writer = new FileWriter(FILENAME, true)) {
       System.out.print("Enter student name: ");
       String name = input.nextLine();
       System.out.print("Enter student ID: ");
       String id = input.nextLine();
       System.out.print("Enter grade: ");
       double grade = input.nextDouble();
       writer.write(name + "," + id + "," + grade + "\n");
       System.out.println("Student added successfully!");
    } catch (IOException e) {
       System.out.println("Error adding student: " + e.getMessage());
    }
  }
  public static void displayAllStudents() {
    try (BufferedReader reader = new BufferedReader(new FileReader(FILENAME))) {
       String line;
       System.out.println("\n--- Student Records ---");
       System.out.println("Name\t\tID\t\tGrade");
       System.out.println("-----");
       while ((line = reader.readLine()) != null) {
         String[] parts = line.split(",");
         if (parts.length == 3) {
            System.out.printf("%-12s\t%-8s\t%.2f\n",
              parts[0], parts[1], Double.parseDouble(parts[2]));
         }
       }
    } catch (FileNotFoundException e) {
       System.out.println("No student records found.");
    } catch (IOException e) {
       System.out.println("Error reading file: " + e.getMessage());
```

```
public static void searchStudent() {
  Scanner input = new Scanner(System.in);
  System.out.print("Enter student ID to search: ");
  String searchId = input.nextLine();
  try (BufferedReader reader = new BufferedReader(new FileReader(FILENAME))) {
     String line;
     boolean found = false;
     while ((line = reader.readLine()) != null) {
       String[] parts = line.split(",");
       if (parts.length == 3 && parts[1].equals(searchId)) {
          System.out.println("\nStudent Found:");
          System.out.println("Name: " + parts[0]);
          System.out.println("ID: " + parts[1]);
          System.out.println("Grade: " + parts[2]);
         found = true;
         break:
       }
     }
     if (!found) {
       System.out.println("Student with ID " + searchId + " not found.");
    }
  } catch (FileNotFoundException e) {
     System.out.println("No student records found.");
  } catch (IOException e) {
     System.out.println("Error reading file: " + e.getMessage());
  }
}
public static void main(String[] args) {
  Scanner input = new Scanner(System.in);
  while (true) {
     System.out.println("\n=== Student Grade Manager ===");
     System.out.println("1. Add Student");
     System.out.println("2. Display All Students");
     System.out.println("3. Search Student");
     System.out.println("4. Exit");
     System.out.print("Choose an option: ");
     int choice = input.nextInt();
```

```
input.nextLine(); // Consume newline
       switch (choice) {
          case 1:
            addStudent();
            break;
          case 2:
            displayAllStudents();
            break;
          case 3:
            searchStudent();
            break;
          case 4:
            System.out.println("Goodbye!");
          default:
            System.out.println("Invalid choice. Please try again.");
       }
}
```

# **File Copy Utility**



```
import java.io.*;
public class FileCopyUtility {
  public static void copyFile(String sourcePath, String destinationPath) {
     try (BufferedReader reader = new BufferedReader(new FileReader(sourcePath));
        BufferedWriter writer = new BufferedWriter(new FileWriter(destinationPath))) {
       String line;
       while ((line = reader.readLine()) != null) {
          writer.write(line);
          writer.newLine();
       }
       System.out.println("File copied successfully from " + sourcePath + " to " + destinationPath);
    } catch (FileNotFoundException e) {
       System.out.println("Source file not found: " + e.getMessage());
    } catch (IOException e) {
       System.out.println("Error copying file: " + e.getMessage());
    }
  }
  public static void main(String[] args) {
    // Create a sample file first
    try (FileWriter writer = new FileWriter("original.txt")) {
       writer.write("This is the original file.\n");
       writer.write("It contains some sample text.\n");
       writer.write("We will copy this to another file.");
    } catch (IOException e) {
       System.out.println("Error creating sample file: " + e.getMessage());
       return;
    // Copy the file
    copyFile("original.txt", "copy.txt");
    // Verify the copy
    System.out.println("\nContents of copied file:");
     try (BufferedReader reader = new BufferedReader(new FileReader("copy.txt"))) {
       String line;
       while ((line = reader.readLine()) != null) {
          System.out.println(line);
    } catch (IOException e) {
       System.out.println("Error reading copied file: " + e.getMessage());
```

### **Key Points to Remember**

- 1. Always close files after use to free system resources
- 2. Use try-with-resources for automatic resource management
- 3. **Handle exceptions** properly with try-catch blocks
- 4. **BufferedReader/BufferedWriter** are more efficient for large files
- 5. **Scanner** is convenient for reading formatted data
- 6. FileWriter with append mode adds content to existing files
- 7. **Check file existence** before attempting operations
- 8. **Use absolute paths** when files are in different directories

### **Common File Operations Summary**

- **Create file**: File.createNewFile()
- **Delete file**: (File.delete())
- Check existence: (File.exists())
- Get file size: (File.length())
- Create directory: (File.mkdir())
- List directory contents: (File.list()) or (File.listFiles())

#### **Practice Exercises**

- 1. Create a program that reads a text file and counts the number of words
- 2. Write a program that merges two text files into a third file
- 3. Create a simple log file writer that appends timestamped messages
- 4. Build a program that reads student data from a file and calculates average grades
- 5. Implement a file backup utility that copies files with timestamps