Object Oriented Programming (JAVA)



## Semester: Fall 2024

**Software Engineering**

**Faculty of Information Technology UCP Lahore, Pakistan**

**Instructor:**

**Danish Ali Khan**

Table of Contents

[Semester: Fall 2024 1](#_Toc183179127)

[Output: 4](#_Toc183179128)

[Importing Packages: We import the necessary classes from the java.io package. 5](#_Toc183179129)

[File Creation: 5](#_Toc183179130)

[We create a File object with the specified filename (example.txt). 5](#_Toc183179131)

[We use the createNewFile() method to create the file. This method returns true if the file was created successfully, and false if it already exists. 5](#_Toc183179132)

[Writing to the File: 5](#_Toc183179133)

[We create a FileWriter object to write to the file. 5](#_Toc183179134)

[The write() method is used to write the specified string to the file. 5](#_Toc183179135)

[Finally, we close the FileWriter using close() to release system resources. 5](#_Toc183179136)

[Error Handling: We handle potential IOExceptions that may occur during file operations. 5](#_Toc183179137)

[Reading from a file: 5](#_Toc183179138)

[Output: 5](#_Toc183179139)

[**Lab Tasks** 7](#_Toc183179140)

[**Task 1: Basic File Operations - Create, Write, and Read a File** 7](#_Toc183179141)

[**Task 2: Write your data on file** 7](#_Toc183179142)

[**Task 3: Read from File from data.txt file and print all the students data having 70 above marks.** 7](#_Toc183179143)

[**Task 4: Extending File Handling to Store Multiple Homes’ Configurations** 7](#_Toc183179144)

[**Scenario Based** 8](#_Toc183179145)

[Task 1: Creating Files 8](#_Toc183179146)

[Task 2: Writing Data to Files 8](#_Toc183179147)

[Task 3: Reading Data from Files 8](#_Toc183179148)

[Task 4: Extending File Handling to Multiple Homes 9](#_Toc183179149)

|  |  |
| --- | --- |
| **Week 7** | |
| **Topic** | **File Handling** |
| **Objective** | * Understand the importance of file handling in Java. * Learn the classes used for reading from and writing to files (File, FileReader, FileWriter, BufferedReader, BufferedWriter). * Learn how to create, write and read data from a file in Java. * Extend file handling to store multiple homes’ configurations. * Save and load the configurations for multiple homes from a file. |

**Why File Handling is Required?**

* File Handling is an integral part of any programming language as file handling enables us to store the output of any particular program in a file and allows us to perform certain operations on it.
* In simple words, file handling means reading and writing data to a file.

**Streams in Java**

* In Java, a sequence of data is known as a stream.
* This concept is used to perform I/O operations on a file.

There are two types of streams :

**1. Input Stream:**

The Java InputStream class is the superclass of all input streams.

**2. Output Stream:**

The output stream is used to write data to numerous output devices like the monitor, file, etc.

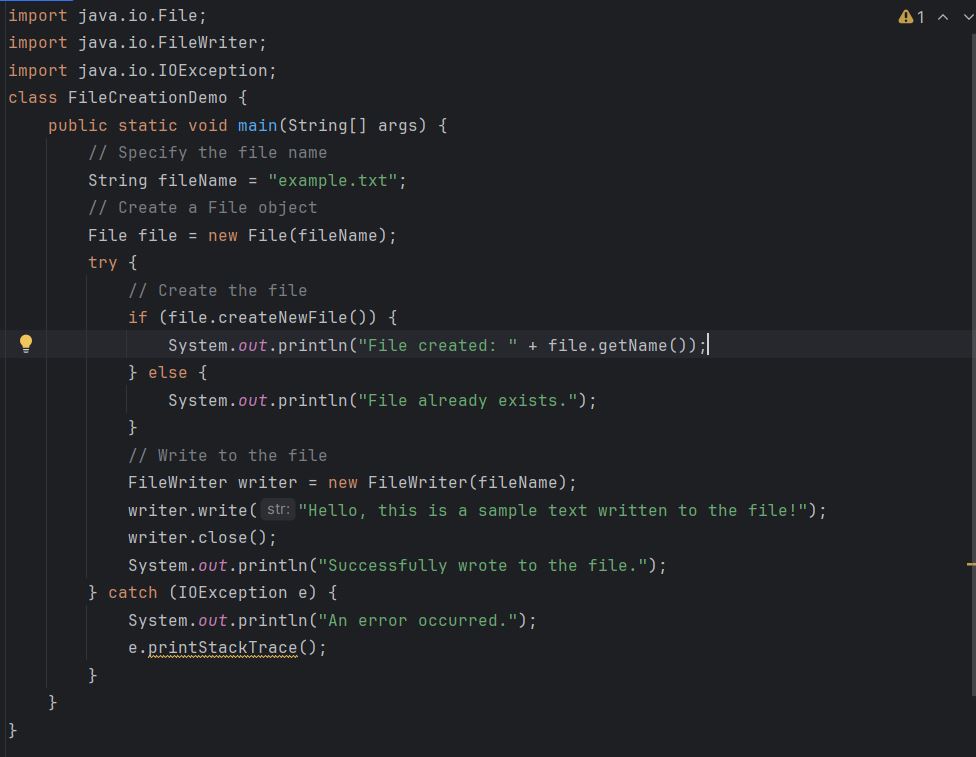
**File operations in Java**

The following are the several operations that can be performed on a file in Java :

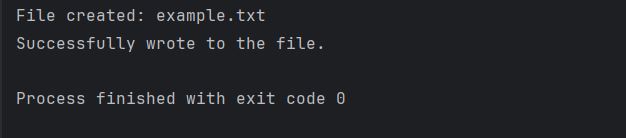
1. **Create a File**
2. **Read from a File**
3. **Write to a File**

***Creating and Writing to a file:***

Creating a file in Java can be done using the java.io package, specifically using the File and FileWriter classes. Below is a simple example that demonstrates how to create a file and write some text into it.

****

Output:

****

*Explanation:*

* + Packages:

We import the necessary classes from the java.io package.

* + File Creation:

We create a File object with the specified filename (example.txt).

We use the createNewFile() method to create the file. This method returns true if the file was created successfully, and false if it already exists.

* + Writing to the File:

We create a FileWriter object to write to the file.

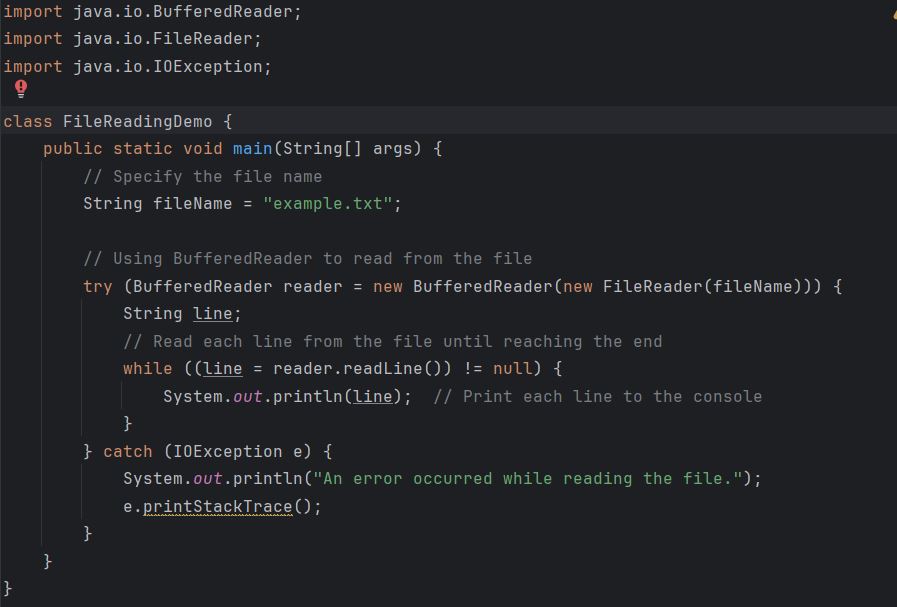
The write() method is used to write the specified string to the file.

Finally, we close the FileWriter using close() to release system resources.

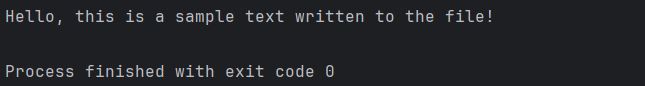
* + Error Handling:

We handle potential IOExceptions that may occur during file operations.

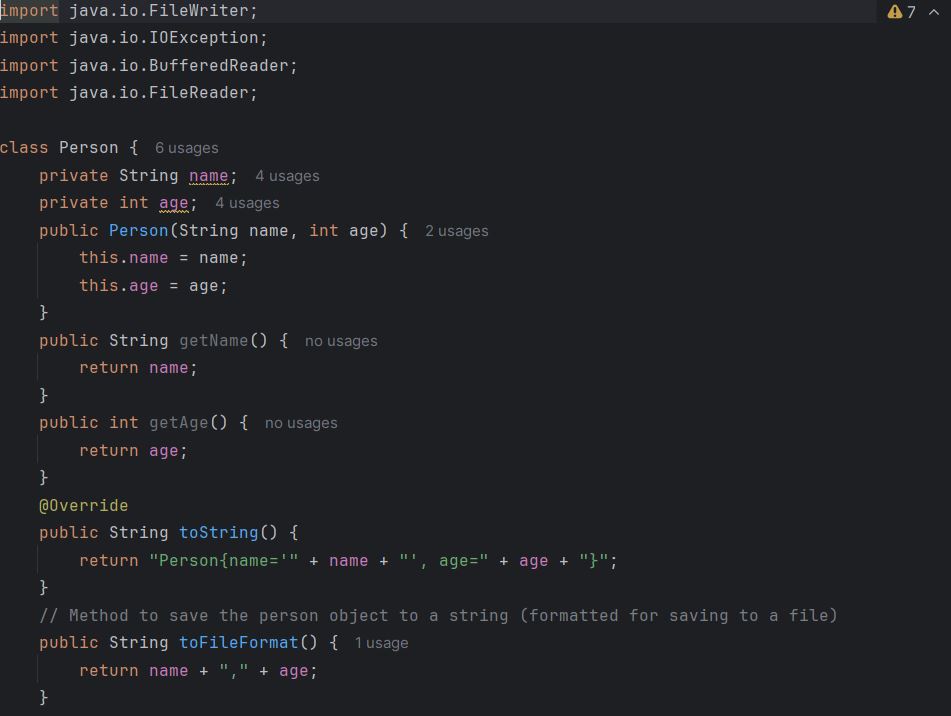
*Reading from a file:*

****

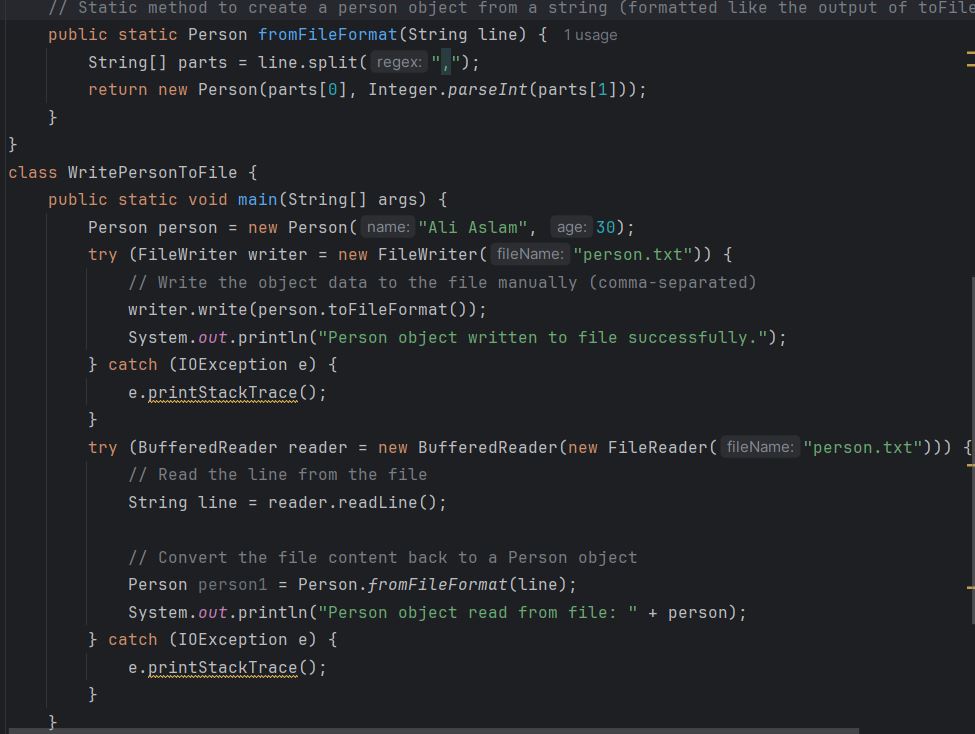
Output:

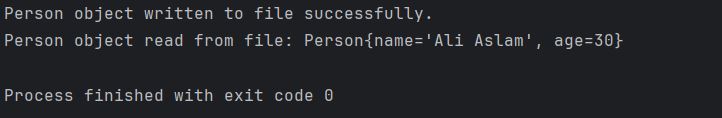
****

**An example to demonstrate how to populate an object on the file and read an object from the file:**

****

Output:

****

****Output:

**Lab Tasks**

**Task 1: Basic File Operations - Create, Write, and Read a File**

**Steps**

1. **Create a File**:
   * Use the File class to create a file in a specific directory.
2. **Write to a File**:
   * Use FileWriter and BufferedWriter to write text data to the file.
3. **Read from a File**:
   * Use FileReader and BufferedReader to read text data from the file.

**Task 2: Write your data on file**

Write your name your age and your marks on file Also write 4 to 5 students info on file name data.txt.

**Task 3: Read from File from data.txt file and print all the students data having 70 above marks.**

**Task 4: Extending File Handling to Store Multiple Homes’ Configurations**

**Scenario**

Design a **Home** class to represent a home with properties like number of bedrooms, number of bathrooms, and additional features. Store multiple home configurations in a file, then load and display them.

**Steps**

1. **Define the Home Class**:
   * Attributes: int bedrooms, int bathrooms, String additionalFeatures.
   * Constructor: Initialize the attributes.
   * Method toString(): Returns a formatted string of the home configuration.
2. **Write Multiple Homes to a File**:
   * Create multiple Home objects and write each configuration to the file.
3. **Read Multiple Homes from a File**:
   * Read each line from the file and create a Home object, then display the configurations.

**Scenario Based**

### Task 1: Creating Files

**Objective:** Create a text file to store data about the SmartHome system.

**Instructions:**

1. Use the File class to create a text file named smart\_home.txt.
2. Write a simple method in Java to check if the file exists. If it doesn’t, create the file.

**Expected Output:**

* The program checks whether the smart\_home.txt file exists, creating it if necessary.

### Task 2: Writing Data to Files

**Scenario:** You need to store the current configuration of SmartHome and its associated SmartLight objects in a file for later retrieval.

**Instructions:**

1. Extend the SmartHome and SmartLight classes.
2. Write a method saveToFile() in the SmartHome class that writes the details of the smart home and its lights to the file smart\_home.txt.
3. Use FileWriter and BufferedWriter to write the details.

**Expected Output:**

* The details of SmartHome and its lights are written to smart\_home.txt. The file should contain something like:

Smart Home: My Home

Lights:

Living Room Light, On, Brightness: 75

Kitchen Light, Off, Brightness: 30

### Task 3: Reading Data from Files

**Scenario:** After saving the smart home configuration, you now want to retrieve it by reading the smart\_home.txt file.

**Instructions:**

1. Implement a loadFromFile() method in the SmartHome class that reads the data from smart\_home.txt.
2. Use FileReader and BufferedReader to read the file content line by line and display the information.

**Expected Output:**

Smart Home: My Home

Lights:

Living Room Light, On, Brightness: 75

Kitchen Light, Off, Brightness: 30

### 

### Task 4: Extending File Handling to Multiple Homes

**Instructions:**

1. Modify the saveToFile() and loadFromFile() methods to handle multiple smart homes.
2. Write a method saveMultipleHomes() to save multipl
3. e SmartHome objects into smart\_home.txt.
4. Write a method loadMultipleHomes() to read and display the data for multiple homes.

.