**File Handling**

DESCRIPTION:

**Objective:**

write a program to read, write, and append to a file.

**Background of the problem statement:**

As a developer, write a Java code to read, write, and append to a file.

**You must use the following:**

* Eclipse/IntelliJ: An IDE to code for the application.
* Java: A programming language.
* Git: To connect and push files from the local system to GitHub.
* GitHub: To store the application code and track its versions.

**Steps needed to write a program to read , write and append to a file:**

1. Creating a new project in Eclipse

2. Writing a program in Java code to read, write, and append to a file.

3. Executing the program and verifying the result according to the particular operation.

4. Print the result.

5. Pushing the code to GitHub repositories.

**Step1:**

**Creating a new project in Eclipse:**

* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* Type in project name as **Third-Project** and click on “Finish.”
* Select your project and go to File -> New -> Package.
* Enter **com.simplilearn.test** as Package name.
* Select your project and go to File -> New -> Class.
* Enter **FileHandling** as class name, check the checkbox “public static void main(String[] args)”., and click on “Finish.”
* Generate try-catch block for performing the methods for creating and reading a file

try {

createFileUsingFileClass( );

readFile();

}

catch(IOException e){

e.printStackTrace();

}

static void createFileUsingFileClass() throws IOException

{

File myfile= new File("D:\\Files\\File1.txt");

if(myfile.createNewFile())

System.out.println("File Created Successfully");

else

System.out.println("File Already Exists");

The try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

createFileUsingFileClass( );

readFile();

The above method is used to create a new file and the next one is used to read a file

File myfile= new File("D:\\Files\\File1.txt");

It shows the path of the file is created or exists

if(myfile.createNewFile())

System.out.println("File Created Successfully");

else

System.out.println("File Already Exists");

If-else is used to show whether the new file is created or the file already exists

**//Write to a File**

FileWriter writer = new FileWriter(myfile);

writer.write("\n Hello Everyone Welcome To File Handling");

System.out.println("Data Written Successfully");

writer.close();

It is used to write data into the file. FileWriter is used for that purpose.

**//Append to a File**

FileWriter writer1 = new FileWriter(myfile,true);

writer1.write("\n Hello Everyone Welcome To File Handling,Testing Testing");

System.out.println("Data updated Successfully");

writer1.close();

}

Here the new data is added to the file without removing old data by using Append

**//To Read a File**

static void readFile() throws IOException {

FileReader reader = new FileReader("D:\\Files\\File1.txt");

char data[ ] = new char[200];

reader.read(data);

System.out.println(data);

System.out.println("Data read Successfully");

reader.close();

}

FileReader is used to read data in the file. It reads data by the count of characters. So, we have to give the number of characters we want to read.

**Step2:**

**Writing a program in Java to create read, write, and append to a file:**

package com.simplilearn.test;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileHandling {

public static void main(String[ ] args) {

try {

createFileUsingFileClass( );

readFile();

}

catch(IOException e){

e.printStackTrace();

}

}

static void createFileUsingFileClass() throws IOException

{

File myfile= new File("D:\\Files\\File1.txt");

if(myfile.createNewFile())

System.out.println("File Created Successfully");

else

System.out.println("File Already Exsists");

**//Write to a File**

FileWriter writer = new FileWriter(myfile);

writer.write("\n Hello Everyone Welcome To File Handling");

System.out.println("Data Written Successfully");

writer.close();

**//Append to a File**

FileWriter writer1 = new FileWriter(myfile,true);

writer1.write("\n Hello Everyone Welcome To File Handling,Testing Testing");

System.out.println("Data updated Successfully");

writer1.close();

}

**//To Read a File**

static void readFile() throws IOException {

FileReader reader = new FileReader("D:\\Files\\File1.txt");

char data[ ] = new char[200];

reader.read(data);

System.out.println(data);

System.out.println("Data read Successfully");

reader.close();

}

}

**Step 3:**

**Executing the program and verifying the result according to the particular operation:**

Before executing the program, check for syntactical corrections. If no errors are found, follow the steps mentioned below:

* [*Right click*] in the program space
* Select *Run As Java Application*

**Step4:**

**Print the result:**

File Created Successfully

Data Written Successfully

Data updated Successfully

Hello Everyone Welcome To File Handling

Hello Everyone Welcome To File Handling,Testing Testing

Data read Successfully

* New file is created Successfully and data is written in it.
* Data is added using append.
* The data in the file is read.

In conclusion, the program in Java code to read, write, and append to a file is running successfully.

**Step5:**

**Pushing the code to your GitHub repositories:**

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**