**Project-1**

**Virtual Key for Repositories**

**(Core Java)**



**\*Github Repository link:**

**SUBMITTED BY:**

**KOUSAR JAMADAR**

**DATE: 16/08/2022**

**SUBMITTED TO:**

**SIMLILEARN**

**COMPANY: LOCKERS Pvt Ltd**

**Project objective:**

As a Full Stack Developer, I have completed the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

**Background of the problem statement:**

As a Full Stack Developer at company Lockers Pvt. Ltd. They aim to digitize their products and chose LockedMe.com as their first project to start with. I have asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. The goal of the company is to deliver a high-end quality product as early as possible with some requirements:

* Specification document - Product’s capabilities, appearance, and user interactions
* Number and duration of sprints required
* Setting up Git and GitHub account to store and track your enhancements of the prototype
* Java concepts being used in the project
* Generic features and three operations:
  + Retrieving the file names in an ascending order
  + Business-level operations:
    1. Option to add a user specified file to the application
    2. Option to delete a user specified file from the application
    3. Option to search a user specified file from the application
    4. Navigation option to close the current execution context and return to the main context
  + Option to close the application

**The flow and features of the application:**

* Plan more than two sprints to complete the application
* Document the flow of the application and prepare a flow chart
* List the core concepts and algorithms being used to complete this application
* Code to display the welcome screen. It should display:
  1. Application name and the developer details
  2. The details of the user interface such as options displaying the user interaction information
  3. Features to accept the user input to select one of the options listed
* The first option should return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it
* The second option should return the details of the user interface such as options displaying the following:
  + - 1. Add a file to the existing directory list
      2. You can ignore the case sensitivity of the file names
      3. Delete a user specified file from the existing directory list
    - We can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list
    - Return a message if FNF (File not found)
  + Search a user specified file from the main directory
    - We can add the case sensitivity on the file name to retrieve the correct file
    - Display the result upon successful operation
    - Display the result upon unsuccessful operation
  + Option to navigate back to the main context
* There should be a third option to close the application
* Implement the appropriate concepts such as exceptions, collections, and sorting techniques for source code optimization and increased performance

 **Used the following:**

* Eclipse: An IDE to code for the application
* Java: A programming language to develop the prototype
* Git: To connect and push files from the local system to GitHub
* GitHub: To store the application code and track its versions
* Scrum: An efficient agile framework to deliver the product incrementally
* Specification document: Any open-source document

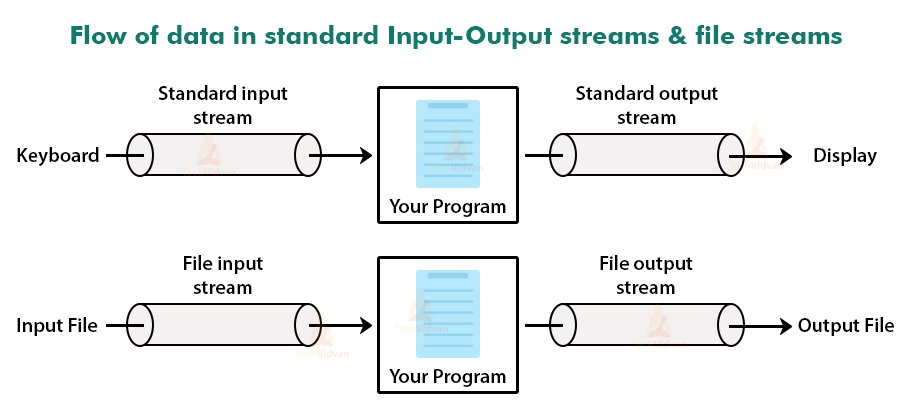
**Detailed Description**:

**Java File Handling**:

In common terms, **File Handling** is the task of maintaining and managing the contents of a file. File Handling is a term used for reading data from the File and writing data to the File.

In Java Programming Language, a package called **java.io** allows us to do all such types of File Handling tasks. This package provides us with the functionality of various formats of File.

In this package, we have to provide an **Input Source** and an **Output Destination.** While working with File, first we have to **create a File** for this you need to construct an object of the class and define the destination location.



**Fig: Flow of data in standard I/O stream and file stream**

Java uses a concept of **Stream** for such tasks related to file. So, before falling on the concept of file, we have to understand what a stream is**.**

What is a Stream in Java?

**Stream** in Java Programming Language is the sequence of the objects that are pipelined to get desired results.

The Stream is not a data structure instead it just takes input from collections of I/O. Stream doesn’t change the original methods or data structure it just pipelines it and provides our result.

There are two brands of Stream in Java Programming Language:

**1. Input Stream:**

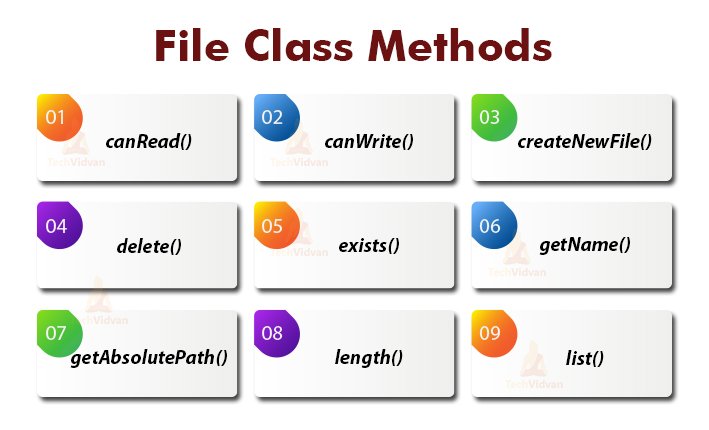
The Input Stream provides the functionality of taking the Input from a Source. The Input Stream provides the set of Objects and pipelines together to provide us the functionality of Input the collection in I/O File.

**2. Output Stream:**

The Output Stream provides the functionality of writing the data to the Destination. The Output Stream provides the set of Objects and pipelines together to provide us the functionality of Output to the collection in I/O File.

**File Class Methods:**

The File class is widely used to provide the functionality of File Handling in the Java programming language.



**FIG: File class methods**

**The method used to check the permission of the File:**

#### **1. canRead()**

Type: **Boolean**

**Description:** The **canRead()** method used to check whether the file is readable or not.

The method returns a Boolean value i.e. **True or False**. If the method returns True then File is readable and if False then unreadable.

#### **2. canWrite()**

Type: **Boolean**

**Description:** The **canWrite()** method used to check whether the files are writable or not. The method also returns a Boolean value.

If the method returns True then File is writable i.e. we can edit the File and write our Content also and if False then it is not writable.

**The method used to create, delete and check the existence of the File:**

#### **3. createNewFile()**

Type: **Boolean**

**Description:** The **createNewFile()** method used to create a New File with the Name and the Location provided in the File object.

It also returns Boolean Value which denotes whether the file is successfully created or not.

**Note:** There are very few chances that the method returns False value. The method returns False when there is already a file that exists or memory is not sufficient to create the File.

#### **4. delete()**

Type:**Boolean**

**Description:** The **delete()** method used to delete the existing Files with the Name and the location provided in the File Object. It returns a false value if there is no such File exists.

#### **5. exists()**

Type:**Boolean**

**Description:** The **exists()** method is used to check whether the particular file exists or not.

**The method used to get the information of the File:**

#### **6. getName()**

Type: **String**

**Description:** The **getName()** method used to extract the Name of the file. In this method, we just provide the object of the File class, the method follows the object and extracts the Name of the File.

#### **7. getAbsolutePath()**

Type: **String**

**Description:** The getAbsolutePath() method used to get the Absolute path of the File i.e. the location of the FIle in our System.

The getAbsolutePath() method is called by the File Object and the method follows the object and extracts the location of the File.

#### **8. length()**

Type: **Long**

**Description:** The **length()** method used to get the length of the file in **bytes**. The method simply traverses the File and calculates the number of bytes present in the file and returns the size of the file.

**The method used to get the information of the directory:**

#### **9. list()**

Type: **String [ ] (String Array)**

**Description:** The**list()** method used to get details about the directory in which our files are stored. The method returns a string array with the name of the file that exists in the directory.

### Operations on File in Java

In the above section of this article, we discussed the methods of the file class in Java. Now it’s time to take a deep dive into the operations in Java.

There are 4 types of operations in Java that a beginner must know when working with files in Java.

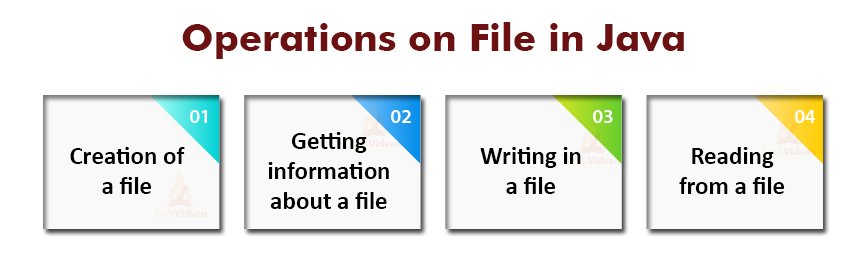
**These 4 operations are**:

Let’s discuss these operations on file in detail with the implementation examples:

#### **1. Creation of File in Java**

Before applying all the other operations on the file, first we need to create a file on a specified location.

Java provides a method createNewFile() that provides a functionality to create a new file on a specified directory. The createNewFile() method returns a boolean value based on the creation of the file.

[](https://techvidvan.com/tutorials/wp-content/uploads/sites/2/2020/06/Operations-on-File-in-Java.jpg)

**FIG: Operations on File in java**

We inherit this method from the File class that we previously discussed in this article.

For creating a new file, first we have to create an object of a File class then assign the location and the file name to this object.

Now we call the createNewFile() method with the help of this object. To check if the file is created or not, we simply execute an if statement.

#### **2. Getting Information about File in Java**

There are mainly four types of methods used to get information about the file in Java.

To get the information about the file in Java, we extract the length of the file, the path to the file, and the permissions granted to the file(read and write) with the help of different methods.

#### **3. Writing on the File in Java**

A user usually uses the file to store and save the data for future preference. Java provides us the functionality so that we can add any content in the file through the Java programs.

In Java, there is a class called FileWriter that is responsible for writing the content into the file. If we want to write the content in the file, we need to create an object of the FileWriter class and then provide the absolute path of the file to that object.

Inside the FileWriter method, there is a method called write() that can be called with the FileWriter object. This method takes the content of the file as its parameter.

The parameter can be of 2 types- Either directly write the content in the parameter, or create a list and pass the reference of the list as the parameter.

**Note:** It is compulsory to close the object of the FileWriter object in order to save the data, otherwise there can be some inconsistencies.

#### **4. Reading from the File in Java**

To read the content or data from the file, we need two classes File and Scanner. The Scanner class reads the contents from the file.

The File object acts as a bridge between the Scanner and the content of the file. We pass the object of the file class to the Scanner class.

**Flow Chart:**

**Fulfilled requirements are:**

* The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.
* The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.
* Document the step-by-step process starting from sprint planning to the product release.
* Application should not close, exit, or throw an exception if the user specifies an invalid input.

**Submitted the final specification document which includes:**

* + Project and developer details
  + Sprints planned and the tasks achieved in them
  + Algorithms and flowcharts of the application
  + Core concepts used in the project
  + Links to the GitHub repository to verify the project completion