File Permissions in Java

Difficulty Level : Easy Last Updated : 27 Jan, 2022

Java provides a number of method calls to check and change the permission of a file, such as a read-only file can be changed to have permissions to write. File permissions are required to be changed when the user wants to restrict the operations permissible on a file. For example, file permission can be changed from write to read-only because the user no longer wants to edit the file.

Checking the Current File Permissions

A file can be in any combination of the following permissible permissions depicted by methods below in tabular format/

Method	Action Performed
canExecutable()	Returns true if and only if the abstract pathname exists and the application is allowed to execute the file
canRead()	Tests whether the application can read the file denoted by this abstract pathname
canWrite()	Returns true if and only if the file system actually contains a file denoted by this abstract pathname and the application is allowed to write to the file; false otherwise

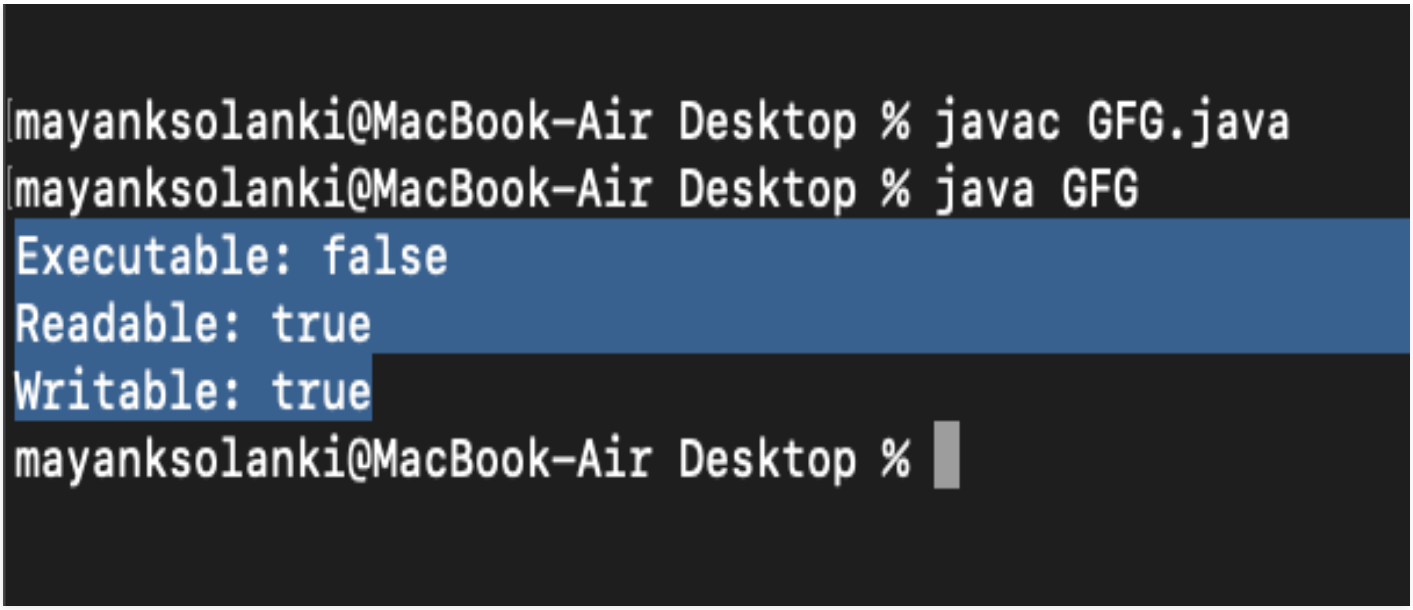
Implementation: A file can be readable and writable but not executable. Here's a Java program to get the current permissions associated with a file.

Example:

```
// Java Program to Check the Current File Permissions

// Importing required classes
import java.io.*;
```

```
// Main class
public class Test {

    // Main driver method
    public static void main(String[] args)
    {

        // Creating a file by
        // creating object of Fiel class
        File file
            = new File("C:\\Users\\Mayank\\Desktop\\1.txt");

        // Checking if the file exists
        // using exists() method of File class
        boolean exists = file.exists();
        if (exists == true) {

            // Printing the permissions associated
            // with the file
            System.out.println("Executable: "
                               + file.canExecute());
            System.out.println("Readable: "
                               + file.canRead());
            System.out.println("Writable: "
                               + file.canWrite());
        }

        // If we enter else it means
        // file does not exists
        else {
            System.out.println("File not found.");
        }
    }
}
```

Output:



Changing File Permissions

A file in Java can have any combination of the following permissions:

- Executable
- Readable
- Writable

Here are methods to change the permissions associated with a file as depicted in a tabular format below as follows:

Method	Action Performed
setExecutable()	Set the owner's execute permission for this abstract pathname
setReadable()	Set the owner's read permission for this abstract pathname
setWritable()	Set the owner's write permission for this abstract pathname

Note:

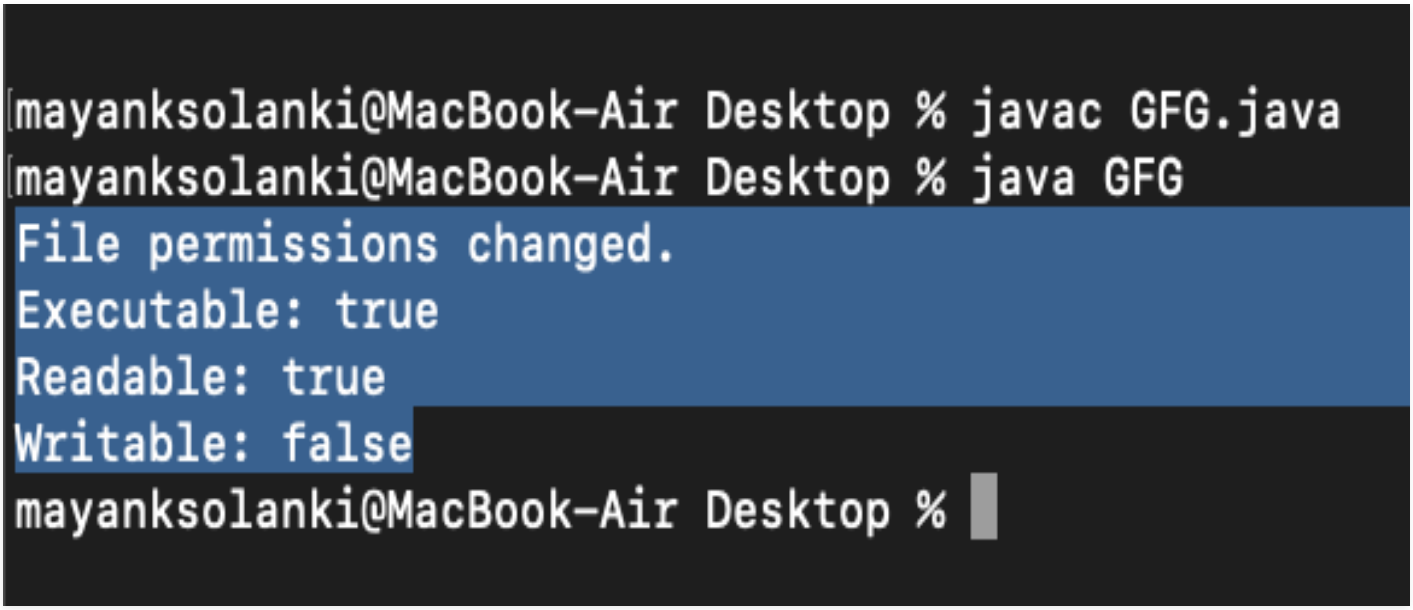
- *setReadable() Operation will fail if the user does not have permission to change t he access permissions of this abstract path name. If readable is false and the un derlying file system does not implement a read permission, then the operation wil l fail.*
- *setWritable() Operation will fail if the user does not have permission to change th*

e access permissions of this abstract pathname.

Example:

```
// Java Program to Change File Permissions

// Importing required classes
import java.io.*;

// Main class
public class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        // Creating a new file by
        // creating object of File class where
        // local directory is passed as in argumenyt
        File file
            = new File("C:\\Users\\Mayank\\Desktop\\1.txt");

        // Checking if file exists
        boolean exists = file.exists();
        if (exists == true) {

            // Changing the file permissions
            file.setExecutable(true);
            file.setReadable(true);
            file.setWritable(false);
            System.out.println("File permissions changed.");

            // Printing the permissions associated with the
            // file currently
            System.out.println("Executable: "
                               + file.canExecute());
            System.out.println("Readable: "
                               + file.canRead());
            System.out.println("Writable: "
                               + file.canWrite());
        }

        // If we reach here, file is not found
        else {
            System.out.println("File not found");
        }
    }
}
```

Output:



This article is contributed by **Mayank Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.