بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Desktop cleaner

With java

Under the supervision of Dr. محمد النيل

Work done by:

محمد العشيوي

Course Title: programming 3

Major: Programming major

Level: 173

Finish Date: 2024/01/28

the source code will be provided at the end of this document

Desktop Cleaner app

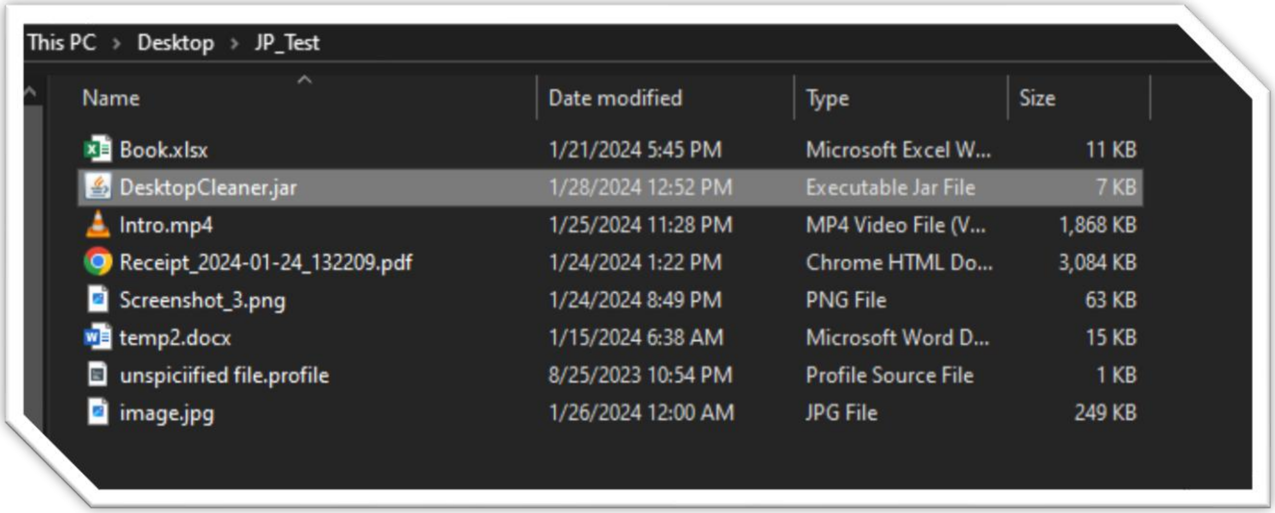
This program defines a "**DesktopCleaner**" class with methods to create folders for different file types, move files to appropriate folders, and execute the cleaning process. You can customize the file types and corresponding folders based on your needs.

***I personally use Eclips IDE ***

Its basically program to orgnize files based on your desired directory.

First of all, lets try this program:

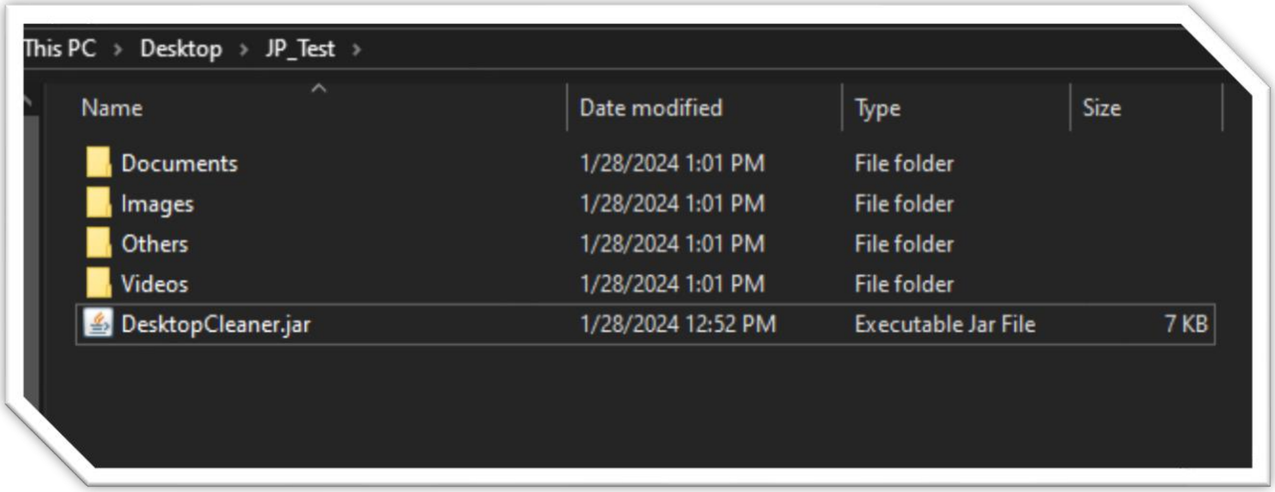
- I created subfolder in my desktop for testing purposes called "JP_Test"
- Inside it there is bunch of random files with different file types.
- There is also my program (called DesktopCleaner.jar)



This PC > Desktop > JP_Test

Name	Date modified	Type	Size
Book.xlsx	1/21/2024 5:45 PM	Microsoft Excel W...	11 KB
DesktopCleaner.jar	1/28/2024 12:52 PM	Executable Jar File	7 KB
Intro.mp4	1/25/2024 11:28 PM	MP4 Video File (V...	1,868 KB
Receipt_2024-01-24_132209.pdf	1/24/2024 1:22 PM	Chrome HTML Do...	3,084 KB
Screenshot_3.png	1/24/2024 8:49 PM	PNG File	63 KB
temp2.docx	1/15/2024 6:38 AM	Microsoft Word D...	15 KB
unspiciified file.profile	8/25/2023 10:54 PM	Profile Source File	1 KB
image.jpg	1/26/2024 12:00 AM	JPG File	249 KB

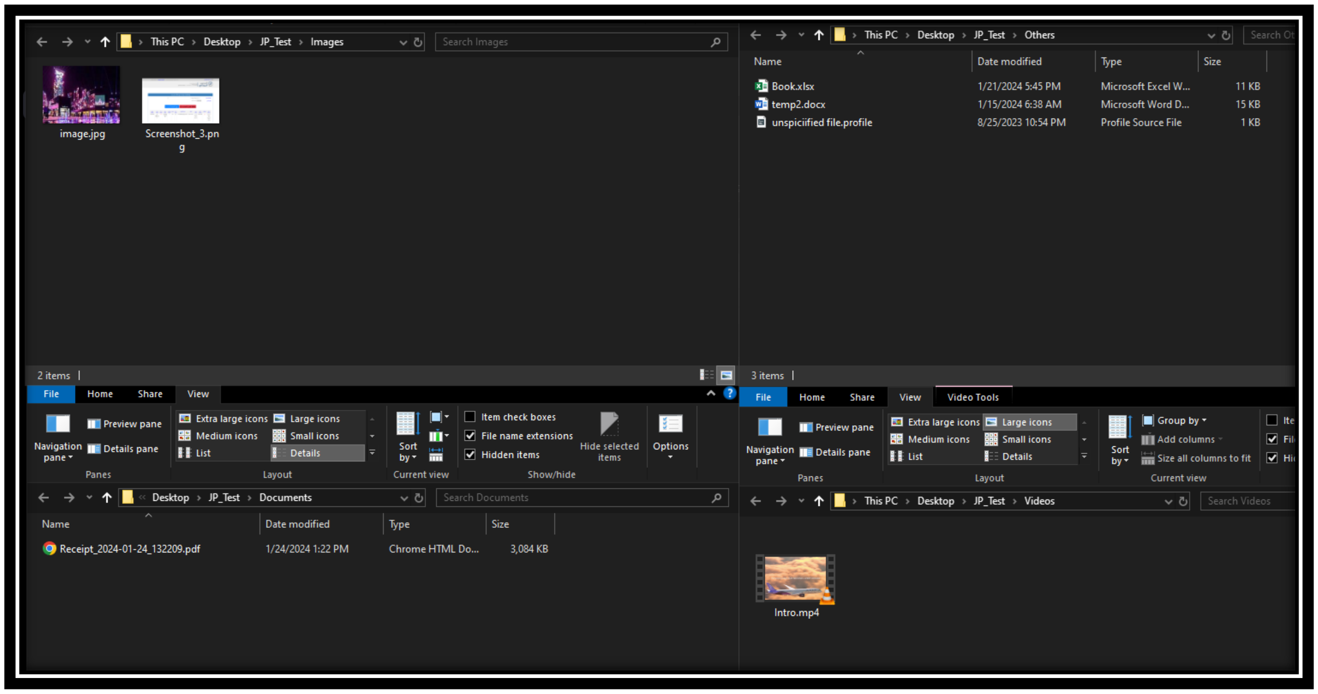
- Now I can click on this app or even lunch my code in the IDE, they both do the same thing.
- Notice if I click on the DesktopCleaner app (or run its code), there is deffrent files created (Documents, Images, Videos, Others), also all my files are gone !!



This PC › Desktop › JP_Test ›

Name	Date modified	Type	Size
Documents	1/28/2024 1:01 PM	File folder	
Images	1/28/2024 1:01 PM	File folder	
Others	1/28/2024 1:01 PM	File folder	
Videos	1/28/2024 1:01 PM	File folder	
DesktopCleaner.jar	1/28/2024 12:52 PM	Executable Jar File	7 KB

- They are now inside these folders and organized based on file type, for example the image.jpg file is now in the images folder along with Screenshot_3.png and so on.



This PC › Desktop › JP_Test › Images

image.jpg
Screenshot_3.png

2 items

This PC › Desktop › JP_Test › Others

Name	Date modified	Type	Size
Book.xlsx	1/21/2024 5:45 PM	Microsoft Excel W...	11 KB
temp2.docx	1/15/2024 6:38 AM	Microsoft Word D...	15 KB
unspiciified file.profile	8/25/2023 10:54 PM	Profile Source File	1 KB

3 items

Desktop › JP_Test › Documents

Name	Date modified	Type	Size
Receipt_2024-01-24_132209.pdf	1/24/2024 1:22 PM	Chrome HTML Do...	3,084 KB

This PC › Desktop › JP_Test › Videos

Intro.mp4

Here it breaks down to how this code works:

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardCopyOption;
```

- These lines import necessary classes for file manipulation, including working with paths, files, and copying options.

```
public class DesktopCleaner {
```

- This line defines the start of the DesktopCleaner class.

```
public static void main(String[] args) {
    cleanDesktop();
}
```

- The main method is the entry point of the program. It calls the cleanDesktop method to initiate the cleaning process.

```
public static void cleanDesktop() {
```

- This method is responsible for organizing and cleaning up files on the desktop.

```
String desktopPath =
System.getProperty("user.home") + "/Desktop";
```

- This line gets the path to the user's desktop.

```java
String[] folders = {"Documents", "Images",
"Videos", "Others"};

for (String folder : folders) {
    createFolder(desktopPath, folder);
}
```

- Here, it creates folders for different file types (Documents, Images, Videos, Others) using the createFolder method.

```java
File desktop = new File(desktopPath);
File[] files = desktop.listFiles();
```

- This section obtains a list of files on the desktop.

```java
if (files != null) {
    for (File file : files) {
        if (file.isFile()) {
            moveFile(file, desktopPath);
        }
    }
    System.out.println("Desktop cleaning
complete!");
} else {
    System.out.println("Error accessing desktop
files.");
}
```

- It checks if the list of files is not empty and then iterates through each file. If the item is a file (not a directory), it calls the moveFile method to move the file to its appropriate folder based on its type.

```
private static void createFolder(String
parentPath, String folderName) {
    File folder = new File(parentPath,
folderName);
    if (!folder.exists()) {
        folder.mkdir();
    }
}
```

- The createFolder method is used to create a folder with a given name in the specified parent path.

```
private static void moveFile(File file, String desktopPath)
{
    String fileType = getFileType(file.getName());
    String destinationFolder =
getDestinationFolder(fileType);
    Path sourcePath = file.toPath();
    Path destinationPath = new File(desktopPath + "/" +
destinationFolder + "/" + file.getName()).toPath();

    try {
        Files.move(sourcePath, destinationPath,
StandardCopyOption.REPLACE_EXISTING);
        System.out.println("Moved: " + file.getName() + " to
" + destinationFolder);
    } catch (IOException e) {
        System.out.println("Error moving file: " +
file.getName());
        e.printStackTrace();}}
```

- The moveFile method is responsible for moving a file to its appropriate folder. It uses the getFileType and getDestinationFolder methods to determine the file type and destination folder.

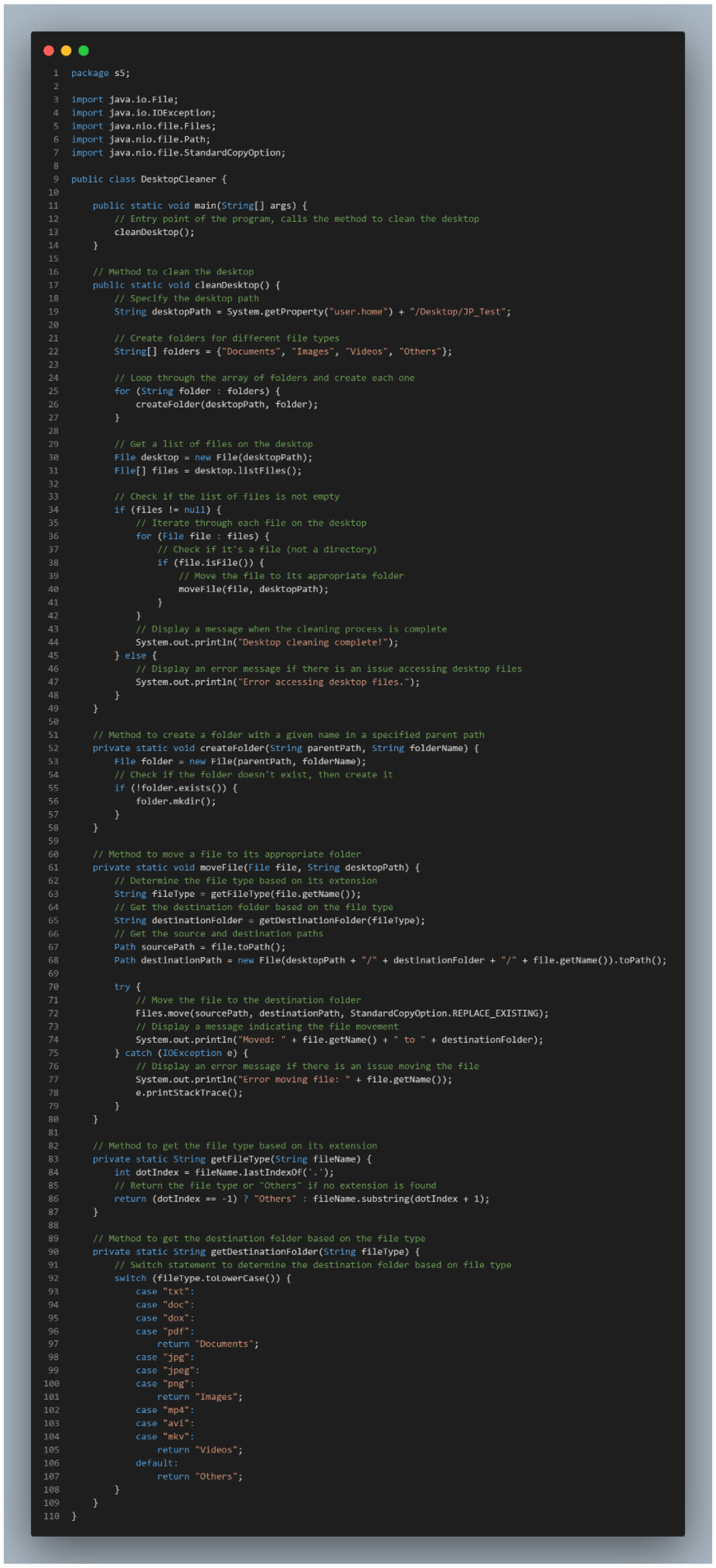
```
private static String getFileType(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    return (dotIndex == -1) ? "Others" :
fileName.substring(dotIndex + 1);
}
```

- The getFileType method extracts the file type based on the file name extension.

```
private static String
getDestinationFolder(String fileType) {
    switch (fileType.toLowerCase()) {
        case "txt":
        case "doc":
        case "pdf":
            return "Documents";
        case "jpg":
        case "jpeg":
        case "png":
            return "Images";
        case "mp4":
        case "avi":
        case "mkv":
            return "Videos";
        default:
            return "Others";
    }
}
```

- The getDestinationFolder method determines the destination folder based on the file type.

Here is in image of the source code:



```java
package s5;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardCopyOption;

public class DesktopCleaner {

    public static void main(String[] args) {
        // Entry point of the program, calls the method to clean the desktop
        cleanDesktop();
    }

    // Method to clean the desktop
    public static void cleanDesktop() {
        // Specify the desktop path
        String desktopPath = System.getProperty("user.home") + "/Desktop/JP_Test";

        // Create folders for different file types
        String[] folders = {"Documents", "Images", "Videos", "Others"};

        // Loop through the array of folders and create each one
        for (String folder : folders) {
            createFolder(desktopPath, folder);
        }

        // Get a list of files on the desktop
        File desktop = new File(desktopPath);
        File[] files = desktop.listFiles();

        // Check if the list of files is not empty
        if (files != null) {
            // Iterate through each file on the desktop
            for (File file : files) {
                // Check if it's a file (not a directory)
                if (file.isFile()) {
                    // Move the file to its appropriate folder
                    moveFile(file, desktopPath);
                }
            }
            // Display a message when the cleaning process is complete
            System.out.println("Desktop cleaning complete!");
        } else {
            // Display an error message if there is an issue accessing desktop files
            System.out.println("Error accessing desktop files.");
        }
    }

    // Method to create a folder with a given name in a specified parent path
    private static void createFolder(String parentPath, String folderName) {
        File folder = new File(parentPath, folderName);
        // Check if the folder doesn't exist, then create it
        if (!folder.exists()) {
            folder.mkdir();
        }
    }

    // Method to move a file to its appropriate folder
    private static void moveFile(File file, String desktopPath) {
        // Determine the file type based on its extension
        String fileType = getFileType(file.getName());
        // Get the destination folder based on the file type
        String destinationFolder = getDestinationFolder(fileType);
        // Get the source and destination paths
        Path sourcePath = file.toPath();
        Path destinationPath = new File(desktopPath + "/" + destinationFolder + "/" + file.getName()).toPath();

        try {
            // Move the file to the destination folder
            Files.move(sourcePath, destinationPath, StandardCopyOption.REPLACE_EXISTING);
            // Display a message indicating the file movement
            System.out.println("Moved: " + file.getName() + " to " + destinationFolder);
        } catch (IOException e) {
            // Display an error message if there is an issue moving the file
            System.out.println("Error moving file: " + file.getName());
            e.printStackTrace();
        }
    }

    // Method to get the file type based on its extension
    private static String getFileType(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        // Return the file type or "Others" if no extension is found
        return (dotIndex == -1) ? "Others" : fileName.substring(dotIndex + 1);
    }

    // Method to get the destination folder based on the file type
    private static String getDestinationFolder(String fileType) {
        // Switch statement to determine the destination folder based on file type
        switch (fileType.toLowerCase()) {
            case "txt":
            case "doc":
            case "dox":
            case "pdf":
                return "Documents";
            case "jpg":
            case "jpeg":
            case "png":
                return "Images";
            case "mp4":
            case "avi":
            case "mkv":
                return "Videos";
            default:
                return "Others";
        }
    }
}
```

Note That There Is Comments To All Code In The Image To Facilitate Project To User/Programmers To Understand, But The Comment Are Shown Only In The Image To Make The Code Shorter (The Source Code Not Its Image).

The Source Code:

```java
package s5;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.StandardCopyOption;

public class DesktopCleaner {

    public static void main(String[] args) {
        cleanDesktop();
    }

    public static void cleanDesktop() {
        String desktopPath = System.getProperty("user.home") + "/Desktop/JP_Test";

        String[] folders = {"Documents", "Images", "Videos", "Others"};

        for (String folder : folders) {
            createFolder(desktopPath, folder);
        }

        File desktop = new File(desktopPath);
        File[] files = desktop.listFiles();

        if (files != null) {
            for (File file : files) {
                if (file.isFile()) {
                    moveFile(file, desktopPath);
                }
            }
            System.out.println("Desktop cleaning complete!");
        } else {
            System.out.println("Error accessing desktop files.");
        }
    }

    private static void createFolder(String parentPath, String folderName) {
        File folder = new File(parentPath, folderName);
        if (!folder.exists()) {
            folder.mkdir();
        }
    }

    private static void moveFile(File file, String desktopPath) {
        String fileType = getFileType(file.getName());
        String destinationFolder = getDestinationFolder(fileType);
        Path sourcePath = file.toPath();
        Path destinationPath = new File(desktopPath + "/" + destinationFolder + "/" + file.getName()).toPath();

        try {
            Files.move(sourcePath, destinationPath, StandardCopyOption.REPLACE_EXISTING);
            System.out.println("Moved: " + file.getName() + " to " + destinationFolder);
        } catch (IOException e) {
            System.out.println("Error moving file: " + file.getName());
            e.printStackTrace();
        }
    }

    private static String getFileType(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        return (dotIndex == -1) ? "Others" : fileName.substring(dotIndex + 1);
    }

    private static String getDestinationFolder(String fileType) {
        switch (fileType.toLowerCase()) {
            case "txt":
            case "doc":
            case "dox":
            case "pdf":
                return "Documents";
            case "jpg":
            case "jpeg":
            case "png":
                return "Images";
            case "mp4":
            case "avi":
            case "mkv":
                return "Videos";
            default:
                return "Others";
        }
    }
}
```