/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* The Java Input/Output Processing Program. \*

\* @author: Vladimir Efros \*

\* This program demonstrates the Java Input/Output \*

\* Processing Input/Output of the files \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.io.File;

import java.io.IOException;

import java.util.Scanner;

/\*\*

\* The Test Driver of the program.

\*/

**public class JavaIOApp**

{

**public static void main(String[] args) throws IOException**

{

// Get Directory where the files are stored.

String fileDirectory = getDirectory();

// Create the new Directory Object.

File folder = new File(fileDirectory);

// Create an array of the files that is to be processed.

File[] listOfFiles = folder.listFiles();

// Find the number of actual files excluding directories.

int filesNum = getNumFiles(listOfFiles.length, listOfFiles);

// Create an array of file objects, the size is the actual number of the files.

FileObj[] file = new FileObj[filesNum];

// Determine whether the element is File or Folder.

// If the file exists, the Object File will created.

createObjFile(listOfFiles, file);

// Get the names of the created files.

getFileName(filesNum, file);

// Get the number of Objects.

getNumOfObj();

// Display the contents of each file.

readFile(filesNum, file);

// Write data to a file. SOS: wipes all the data in the file.

writeFile(filesNum, file);

// Write data to a file.

appendFile(filesNum, file);

// Display the contents of each file.

readFile(filesNum, file);

}

        /\*\*

\* The appendFile method writes data to the end of the file.

\* @param num The amount of files that is to be modified.

\* @param f The array containing Object Files that are to be modified.

\* @throws IOException if the the file is not found.

\*/

**private static void appendFile(int num, FileObj[] f) throws IOException**

{

for(int i = 0; i < num; i++)

{

System.out.println("FILE # " + (i + 1));

f[i].appendToFile();

System.out.println();

}

}

        /\*\*

\* The writeFile method writes info to the file.

\* @param num The amount of files that is to be modified.

\* @param f The array containing Object Files that are to be modified.

\* @throws IOException if the the file is not found.

\*/

**private static void writeFile(int num, FileObj[] f) throws IOException**

{

for(int i = 0; i < num; i++)

{

System.out.println("FILE # " + (i + 1));

f[i].writeToFile();

System.out.println();

}

}

        /\*\*

\* The readFile method reads the contents of the file.

\* @param num The amount of files that is to be read.

\* @param f The array containing Object Files that are to be read.

\* @throws IOException if the the file is not found.

\*/

**private static void readFile(int num, FileObj[] f) throws IOException**

{

for(int i = 0; i < num; i++)

{

System.out.println("FILE # " + (i + 1));

f[i].readFile();

System.out.println();

}

}

        /\*\*

\* The createObjFile method creates the File object if the element processed is file.

\* If the element is directory, object file is not created and message is displayed.

\* @param list An array of the files that is to be processed.

\* @param f The array containing Object Files that are to be created.

\*/

**private static void createObjFile(File[] list, FileObj[] f)**

{

int i = 0; // Total file/Directory counter

int j = 0; // Object file counter

while(i < list.length)

{

// Is it a folder?

if(list[i].isDirectory())

{

System.out.println("Directory! Object is not created! " + list[i].getName());

}

// Is it a file?

else if(list[i].isFile())

{

f[j] = new FileObj(list[i]);

j++;

}

i++;

}

}

        /\*\*

\* The getNumOfObj method displays the number of object files created.

\*/

private static void getNumOfObj()

{

System.out.println("The number of object files created: " + FileObj.getObjNum());

}

        /\*\*

\* The getFileName names displays the name of the object.

\* @param num The amount of files that is to be processed.

\* @param f The array containing Object Files that are to be processed.

\*/

**private static void getFileName(int num, FileObj[] f)**

{

for(int i = 0; i < num; i++)

System.out.println(f[i]);

System.out.println();

}

**private static int getNumFiles(int len, File[] list)**

{

int i = 0;

int filesNum = 0; // Cout the number of valid files

while(i < len)

{

if(list[i].getName().endsWith(".txt") || list[i].isFile())

filesNum++;

i++;

}

return filesNum;

}

        /\*\*

\* The getDirectory method obtains a String containing the path of

\* the specific directory that is to be analyzed.

\* @return The directory path of the file.

\*/

**private static String getDirectory()**

{

// Create a Scanner object for keyboard input.

Scanner keyboard = new Scanner(System.in);

// Get the file name.

String fileDirectory;

// Check the String output.

do

{

// Store the directory into a String.

System.out.print("Enter the Directory: ");

                        // "/home/vladimir/Desktop/JavaIO/TextFiles";

fileDirectory = keyboard.nextLine();

}

while(fileDirectory == null || fileDirectory.length() <= 0);

return fileDirectory;

}

}

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Scanner;

/\*\*

\* The representation of the File Object that is obtained

\* from the specified directory.

\*/

**public class FileObj**

{

// Declaration of the private fields of the Field Object.

private File file;

private static int fileNum = 0; // Counter of the total number of objects created.

/\*\*

\* The constructor method instantiates the File Object.

\* @param f The file that is passed to the the Object.

\*/

**public FileObj(File f)**

{

file = f;

fileNum++;

}

        /\*\*

\* The getObjectNum method returns the current number of the object.

\*/

**public static int getObjNum()**

{

return fileNum;

}

/\*\*

\* The readFile method reads the contents of the File Object.

\* @throws IOException if file is not found.

\*/

**public void readFile() throws IOException**

{

// Open the file.

Scanner inputFile = new Scanner(file);

// Read lines from the file until the end of the file.

while(inputFile.hasNext())

{

// Read the next name.

String line = inputFile.nextLine();

// Display the line.

System.out.println(line);

}

// Close the file.

inputFile.close();

System.out.println("Data read from the file!");

}

/\*\*

\* The writeFile method writes info to the file.

\* @throws IOException

\*/

**public void writeToFile() throws IOException**

{

Scanner keyboard = new Scanner(System.in);

String str;

// Enter the information to the file.

do

{

System.out.print("Enter information to be entered in a file: ");

str = keyboard.nextLine();

}

while(str == null || str.length() <= 0);

// Open the file.

PrintWriter outputFile = new PrintWriter(file);

// Write the data to the file.

outputFile.println(str);

// Close the file.

outputFile.close();

System.out.println("Data written to the file!");

}

/\*\*

\* The appendFile method writes data to the end of the file.

\* @throws IOException If the file is not found.

\*/

**public void appendToFile() throws IOException**

{

Scanner keyboard = new Scanner(System.in);

String str;

// Enter the information to the file.

do

{

System.out.print("Enter information to be appended to a file: ");

str = keyboard.nextLine();

}

while(str == null || str.length() <= 0);

// Open the file.

FileWriter fileWriter = new FileWriter(file, true);

PrintWriter outputFile = new PrintWriter(fileWriter);

// Write the data to the file.

outputFile.println(str);

// Close the file.

outputFile.close();

System.out.println("Data appended to the file!");

}

        /\*\*

\* The toString method overrides the toString method and displays the state of the object.

\*/

**public String toString()**

{

StringBuilder builder = new StringBuilder();

builder.append("File " + "Name is " + file.getName() + "!");

return builder.toString();

}

}