EXERCISE: 09	**FILE HANDLING**
DATE	**12.10.2023**

1.AIM:

To create a class and handle the file operations for copying the contents from one file to another.

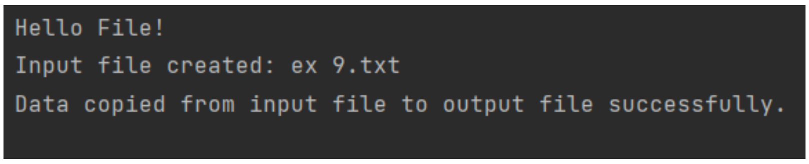
DESCRIPTION:

File Handling is an integral part of any programming language as file handling enables us to store the output of any particular program in a file and allows us to perform certain operations on it.In simple words, file handling means reading and writing data to a file.

PROGRAM:

```
import java.util.Scanner;
import java.io.*;
public class Main {
    public static void main(String[] args) {
        String input;
        System.out.println("Hello File!");
        File inputFile = new File("C:\\Users\\flora\\OneDrive\\Desktop\\ex 9.txt");
        File outputFile = new File("C:\\Users\\flora\\OneDrive\\Desktop\\output ex 9.txt");

        try {
            if (inputFile.createNewFile()) {
                System.out.println("Input file created: " + inputFile.getName());
            } else {
                System.out.println("Input file already exists.");
            }
        } catch (IOException e) {
            System.out.println("An error occurred while creating the input file: " +
e.getMessage());
        }

        try (FileInputStream inputStream = new FileInputStream(inputFile);
             FileOutputStream outputStream = new FileOutputStream(outputFile)) {

            byte[] buffer = new byte[1024]; // Buffer to read and write data in chunks
            int bytesRead;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }
            System.out.println("Data copied from input file to output file successfully.");

        } catch (IOException e) {
            System.out.println("An error occurred while copying data: " + e.getMessage());
        }
        try (Scanner sc1 = new Scanner(inputFile)) {
            while (sc1.hasNextLine()) {
                input = sc1.nextLine();
                System.out.println(input);
            }
        } catch (FileNotFoundException e) {
            System.out.println("Input file not found: " + e.getMessage());
        }
    }
}
```

OUTPUT:



```
Hello File!
Input file created: ex 9.txt
Data copied from input file to output file successfully.
```

RESULT:

The above program is successfully executed.

2. AIM:

To create a class and handle the file operations

DESCRIPTION:

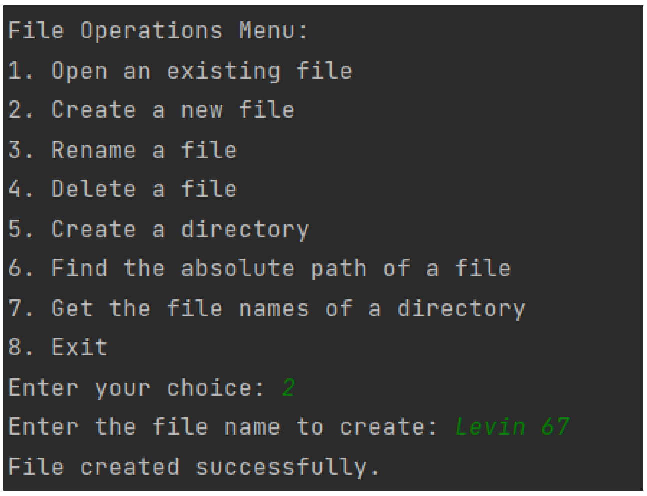
A class that extends the java.lang.Thread class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

PROGRAM:

```
import java.io.*;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("File Operations Menu:");
            System.out.println("1. Open an existing file");
            System.out.println("2. Create a new file");
            System.out.println("3. Rename a file");
            System.out.println("4. Delete a file");
            System.out.println("5. Create a directory");
            System.out.println("6. Find the absolute path of a file");
            System.out.println("7. Get the file names of a directory");
            System.out.println("8. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    openFile(scanner);
                    break;
                case 2:
                    createFile(scanner);
                    break;
                case 3:
                    renameFile(scanner);
                    break;
                case 4:
                    deleteFile(scanner);
                    break;
                case 5:
                    createDirectory(scanner);
                    break;
                case 6:
                    findAbsolutePath(scanner);
                    break;
                case 7:
                    getFileNamesInDirectory(scanner);
                    break;
                case 8:
                    System.out.println("Exiting program.");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 8);
    }

    private static void openFile(Scanner scanner) {
        System.out.print("Enter the file name to open: ");
        String fileName = scanner.nextLine();

        try {
            BufferedReader reader = new BufferedReader(new FileReader(fileName));
            String line;
            System.out.println("File Contents:");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }

    private static void createFile(Scanner scanner) {
        System.out.print("Enter the file name to create: ");
        String fileName = scanner.nextLine();

        try {
            File file = new File(fileName);
```

```
            if (file.createNewFile())  {
                System.out.println("File created successfully.");
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }

    private static void renameFile(Scanner scanner) {
        System.out.print("Enter the current file name: ");
        String currentFileName = scanner.nextLine();
        System.out.print("Enter the new file name: ");
        String newFileName = scanner.nextLine();

        File currentFile = new File(currentFileName);
        File newFile = new File(newFileName);

        if (currentFile.renameTo(newFile)) {
            System.out.println("File renamed successfully.");
        } else {
            System.out.println("Error renaming the file.");
        }
    }

    private static void deleteFile(Scanner scanner) {
        System.out.print("Enter the file name to delete: ");
        String fileName = scanner.nextLine();

        File file = new File(fileName);

        if (file.delete()) {
            System.out.println("File deleted successfully.");
        } else {
            System.out.println("Error deleting the file.");
        }
    }

    private static void createDirectory(Scanner scanner) {
        System.out.print("Enter the directory name to create: ");
        String directoryName = scanner.nextLine();

        File directory = new File(directoryName);
        if (directory.mkdirs()) {
            System.out.println("Directory created successfully.");
        } else {
            System.out.println("Error creating the directory.");
        }
    }

    private static void findAbsolutePath(Scanner scanner) {
        System.out.print("Enter the file name to find its absolute path: ");
        String fileName = scanner.nextLine();

        File file = new File(fileName);

        if (file.exists()) {
            System.out.println("Absolute path: " + file.getAbsolutePath());
        } else {
            System.out.println("File does not exist.");
        }
    }

    private static void getFileNamesInDirectory(Scanner scanner) {
        System.out.print("Enter the directory name to list its files: ");
        String directoryName = scanner.nextLine();

        File directory = new File(directoryName);

        if (directory.exists() && directory.isDirectory()) {
            String[] files = directory.list();
            if (files != null) {
                System.out.println("Files in the directory:");
                for (String file : files) {
                    System.out.println(file);
                }
            } else {
                System.out.println("No files in the directory.");
            }
        } else {
            System.out.println("Directory does not exist or is not a directory.");
        }
    }
}
```

OUTPUT:



```
File Operations Menu:
1. Open an existing file
2. Create a new file
3. Rename a file
4. Delete a file
5. Create a directory
6. Find the absolute path of a file
7. Get the file names of a directory
8. Exit
Enter your choice: 2
Enter the file name to create: Levin 67
File created successfully.
```

RESULT:
The above program is successfully executed.