**Main.java**

package com.main;

import com.service.LockedMeApp;

public class Main {

public static void main(String[] args) {

LockedMeApp app = new LockedMeApp();

app.start();

}

}

**FileManager.java**

package com.model;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.\*;

import java.util.stream.Collectors;

public class FileManager {

private final String DEFAULT\_DIRECTORY = "lockedme\_files";

private final Path directoryPath;

private Set<String> virtualFileSystem;

public FileManager() {

this.directoryPath = Paths.get(DEFAULT\_DIRECTORY);

this.virtualFileSystem = new TreeSet<>(String.CASE\_INSENSITIVE\_ORDER);

loadExistingFiles();

}

private void loadExistingFiles() {

try {

if (Files.exists(directoryPath) && Files.isDirectory(directoryPath)) {

Files.list(directoryPath)

.filter(Files::isRegularFile)

.map(path -> path.getFileName().toString())

.forEach(virtualFileSystem::add);

}

} catch (IOException e) {

System.err.println("Error loading existing files: " + e.getMessage());

}

}

public boolean addFile(String fileName) throws IOException {

if (fileName == null || fileName.trim().isEmpty()) {

throw new IllegalArgumentException("File name cannot be null or empty");

}

fileName = fileName.trim();

if (virtualFileSystem.contains(fileName)) {

return false;

}

Path filePath = directoryPath.resolve(fileName);

try {

Files.createFile(filePath);

virtualFileSystem.add(fileName);

return true;

} catch (IOException e) {

throw new IOException("Failed to create file: " + fileName, e);

}

}

public boolean deleteFile(String fileName) throws IOException {

if (fileName == null || fileName.trim().isEmpty()) {

throw new IllegalArgumentException("File name cannot be null or empty");

}

final String trimmedFileName = fileName.trim();

boolean found = virtualFileSystem.stream()

.anyMatch(file -> file.equals(trimmedFileName));

if (!found) {

return false;

}

Path filePath = directoryPath.resolve(trimmedFileName);

try {

if (Files.exists(filePath)) {

Files.delete(filePath);

}

virtualFileSystem.removeIf(file -> file.equals(trimmedFileName));

return true;

} catch (IOException e) {

throw new IOException("Failed to delete file: " + trimmedFileName, e);

}

}

public boolean searchFile(String fileName) {

if (fileName == null || fileName.trim().isEmpty()) {

return false;

}

final String trimmedFileName = fileName.trim();

return virtualFileSystem.stream()

.anyMatch(file -> file.equals(trimmedFileName));

}

public List<String> getAllFilesSorted() {

List<String> sortedFiles = new ArrayList<>(virtualFileSystem);

Collections.sort(sortedFiles, String.CASE\_INSENSITIVE\_ORDER);

return sortedFiles;

}

public int getFileCount() {

return virtualFileSystem.size();

}

public boolean isEmpty() {

return virtualFileSystem.isEmpty();

}

public String getDirectoryPath() {

return directoryPath.toAbsolutePath().toString();

}

public void refresh() {

virtualFileSystem.clear();

loadExistingFiles();

}

public List<String> searchFilesByPattern(String pattern) {

if (pattern == null || pattern.trim().isEmpty()) {

return new ArrayList<>();

}

String regex = pattern.replace("\*", ".\*").replace("?", ".");

return virtualFileSystem.stream()

.filter(fileName -> fileName.matches(regex))

.sorted(String.CASE\_INSENSITIVE\_ORDER)

.collect(Collectors.toList());

}

public Map<String, Object> getFileStatistics() {

Map<String, Object> stats = new HashMap<>();

stats.put("totalFiles", virtualFileSystem.size());

stats.put("directoryPath", getDirectoryPath());

if (!virtualFileSystem.isEmpty()) {

stats.put("firstFile", Collections.min(virtualFileSystem, String.CASE\_INSENSITIVE\_ORDER));

stats.put("lastFile", Collections.max(virtualFileSystem, String.CASE\_INSENSITIVE\_ORDER));

}

return stats;

}

}

**LockedMeApp.java**

package com.service;

import com.model.FileManager;

import com.utils.InputValidator;

import java.util.Scanner;

public class LockedMeApp {

private FileManager fileManager;

private Scanner scanner;

private InputValidator validator;

private boolean isRunning;

// Application constants

private static final String APP\_NAME = "LockedMe.com";

private static final String VERSION = "v1.0";

private static final String COMPANY\_NAME = "Company Lockers Pvt. Ltd.";

private static final String DEVELOPER\_NAME = "Saksham Sengar";

public LockedMeApp() {

this.fileManager = new FileManager();

this.scanner = new Scanner(System.in);

this.validator = new InputValidator();

this.isRunning = true;

}

public void start() {

displayWelcomeScreen();

while (isRunning) {

try {

displayMainMenu();

int choice = getUserChoice();

handleMainMenuChoice(choice);

} catch (Exception e) {

System.out.println(" An error occurred: " + e.getMessage());

System.out.println("Please try again.\n");

}

}

closeApplication();

}

private void displayWelcomeScreen() {

System.out.println("\n" + "=".repeat(60));

System.out.println(" Welcome to " + APP\_NAME + " " + VERSION);

System.out.println("=".repeat(60));

System.out.println("File Management System");

System.out.println("Developed by:" + " " + DEVELOPER\_NAME);

System.out.println("Company: " + COMPANY\_NAME);

System.out.println("" + java.time.LocalDate.now());

System.out.println("=".repeat(60));

System.out.println("Application Features:");

System.out.println(" List files in ascending order");

System.out.println(" Add files to directory");

System.out.println(" Delete files from directory");

System.out.println(" Search for specific files");

System.out.println(" User-friendly navigation");

System.out.println("=".repeat(60));

System.out.println("Press Enter to continue...");

scanner.nextLine();

}

private void displayMainMenu() {

System.out.println("\n" + "─".repeat(40));

System.out.println(APP\_NAME + " - Main Menu");

System.out.println("─".repeat(40));

System.out.println("1. Display files in ascending order");

System.out.println("2. File operations menu");

System.out.println("3. Exit application");

System.out.println("─".repeat(40));

System.out.print("Please select an option (1-3): ");

}

private int getUserChoice() {

try {

String input = scanner.nextLine().trim();

return validator.validateMenuChoice(input, 1, 3);

} catch (NumberFormatException e) {

System.out.println(" Invalid input! Please enter a number between 1-3.");

return getUserChoice();

} catch (IllegalArgumentException e) {

System.out.println( e.getMessage());

return getUserChoice();

}

}

private void handleMainMenuChoice(int choice) {

switch (choice) {

case 1:

displayFilesInAscendingOrder();

break;

case 2:

handleFileOperationsMenu();

break;

case 3:

isRunning = false;

break;

default:

System.out.println("Invalid choice! Please select 1-3.");

}

}

private void displayFilesInAscendingOrder() {

System.out.println("\n" + "─".repeat(50));

System.out.println("Files in Directory (Ascending Order)");

System.out.println("─".repeat(50));

try {

var files = fileManager.getAllFilesSorted();

if (files.isEmpty()) {

System.out.println("Directory is empty - no files found.");

} else {

System.out.println("Found " + files.size() + " file(s):");

for (int i = 0; i < files.size(); i++) {

System.out.printf("%3d. %s%n", (i + 1), files.get(i));

}

}

} catch (Exception e) {

System.out.println("Error retrieving files: " + e.getMessage());

}

System.out.println("─".repeat(50));

System.out.println("Press Enter to return to main menu...");

scanner.nextLine();

}

private void handleFileOperationsMenu() {

boolean inSubMenu = true;

while (inSubMenu) {

try {

displayFileOperationsMenu();

int choice = getFileOperationChoice();

inSubMenu = handleFileOperationChoice(choice);

} catch (Exception e) {

System.out.println("An error occurred: " + e.getMessage());

System.out.println("Please try again.\n");

}

}

}

private void displayFileOperationsMenu() {

System.out.println("\n" + "─".repeat(40));

System.out.println("File Operations Menu");

System.out.println("─".repeat(40));

System.out.println("1. Add a file");

System.out.println("2. Delete a file");

System.out.println("3. Search for a file");

System.out.println("4. Return to main menu");

System.out.println("─".repeat(40));

System.out.print("Please select an option (1-4): ");

}

private int getFileOperationChoice() {

try {

String input = scanner.nextLine().trim();

return validator.validateMenuChoice(input, 1, 4);

} catch (NumberFormatException e) {

System.out.println("Invalid input! Please enter a number between 1-4.");

return getFileOperationChoice();

} catch (IllegalArgumentException e) {

System.out.println( e.getMessage());

return getFileOperationChoice();

}

}

private boolean handleFileOperationChoice(int choice) {

switch (choice) {

case 1:

addFile();

return true;

case 2:

deleteFile();

return true;

case 3:

searchFile();

return true;

case 4:

return false; // Return to main menu

default:

System.out.println("Invalid choice! Please select 1-4.");

return true;

}

}

private void addFile() {

System.out.println("\n" + "─".repeat(40));

System.out.println("Add New File");

System.out.println("─".repeat(40));

System.out.print("Enter file name to add: ");

String fileName = scanner.nextLine().trim();

if (validator.isValidFileName(fileName)) {

try {

boolean added = fileManager.addFile(fileName);

if (added) {

System.out.println("File '" + fileName + "' added successfully!");

} else {

System.out.println("File '" + fileName + "' already exists in the directory.");

}

} catch (Exception e) {

System.out.println("Error adding file: " + e.getMessage());

}

} else {

System.out.println("Invalid file name! Please enter a valid file name.");

}

System.out.println("Press Enter to continue...");

scanner.nextLine();

}

private void deleteFile() {

System.out.println("\n" + "─".repeat(40));

System.out.println("Delete File");

System.out.println("─".repeat(40));

System.out.print("Enter file name to delete (case-sensitive): ");

String fileName = scanner.nextLine().trim();

if (validator.isValidFileName(fileName)) {

try {

boolean deleted = fileManager.deleteFile(fileName);

if (deleted) {

System.out.println("File '" + fileName + "' deleted successfully!");

} else {

System.out.println("File Not Found (FNF): '" + fileName + "' does not exist in the directory.");

}

} catch (Exception e) {

System.out.println("Error deleting file: " + e.getMessage());

}

} else {

System.out.println("Invalid file name! Please enter a valid file name.");

}

System.out.println("Press Enter to continue...");

scanner.nextLine();

}

private void searchFile() {

System.out.println("\n" + "─".repeat(40));

System.out.println("Search File");

System.out.println("─".repeat(40));

System.out.print("Enter file name to search (case-sensitive): ");

String fileName = scanner.nextLine().trim();

if (validator.isValidFileName(fileName)) {

try {

boolean found = fileManager.searchFile(fileName);

if (found) {

System.out.println("File Found: '" + fileName + "' exists in the directory.");

} else {

System.out.println("File Not Found: '" + fileName + "' does not exist in the directory.");

}

} catch (Exception e) {

System.out.println("Error searching file: " + e.getMessage());

}

} else {

System.out.println("Invalid file name! Please enter a valid file name.");

}

System.out.println("Press Enter to continue...");

scanner.nextLine();

}

private void closeApplication() {

System.out.println("\n" + "=".repeat(50));

System.out.println("Closing " + APP\_NAME + "...");

System.out.println("Thank you for using our file management system!");

System.out.println("Goodbye!");

System.out.println("=".repeat(50));

scanner.close();

}

}

**InputValidator.java**

package com.utils;

import java.util.regex.Pattern;

public class InputValidator {

private static final Pattern INVALID\_CHARS\_PATTERN = Pattern.compile("[<>:\"/\\\\|?\*]");

// Constants for file name validation

private static final int MAX\_FILENAME\_LENGTH = 255;

private static final int MIN\_FILENAME\_LENGTH = 3;

public int validateMenuChoice(String input, int minChoice, int maxChoice)

throws NumberFormatException, IllegalArgumentException {

if (input == null || input.trim().isEmpty()) {

throw new IllegalArgumentException("Input cannot be empty");

}

int choice = Integer.parseInt(input.trim());

if (choice < minChoice || choice > maxChoice) {

throw new IllegalArgumentException(

String.format("Choice must be between %d and %d", minChoice, maxChoice));

}

return choice;

}

/\*\*

\* - Not null/empty

\* - Length between 3 and 255

\* - No invalid characters

\* - Must have an extension

\*/

public boolean isValidFileName(String fileName) {

if (fileName == null || fileName.trim().isEmpty()) {

return false;

}

fileName = fileName.trim();

if (fileName.length() < 3 || fileName.length() > 255) {

return false;

}

if (fileName.matches(".\*[<>:\"/\\\\|?\*].\*")) {

return false;

}

return fileName.contains(".");

}

public boolean isNotEmpty(String input) {

return input != null && !input.trim().isEmpty();

}

public String getFileNameValidationError(String fileName) {

if (fileName == null || fileName.trim().isEmpty()) {

return "File name cannot be empty";

}

fileName = fileName.trim();

if (fileName.length() < MIN\_FILENAME\_LENGTH) {

return "File name must be at least " + MIN\_FILENAME\_LENGTH + " characters long";

}

if (fileName.length() > MAX\_FILENAME\_LENGTH) {

return "File name cannot exceed " + MAX\_FILENAME\_LENGTH + " characters";

}

if (INVALID\_CHARS\_PATTERN.matcher(fileName).find()) {

return "File name contains invalid characters: < > : \" / \\ | ? \*";

}

if (!fileName.contains(".")) {

return "File name must have an extension (e.g., .txt, .doc)";

}

return "File name format is invalid";

}

}