**Name:** Mukesh Kanna

**Project:** Virtual Key Repository (Java prototype)

**Virtual Keys Repository:**

Source Code

/Main/Java

**Package**: org.example.virtualkey

**File Name:** VirtualKeyApplication.java

**package** org.example.virtualkey;

**import** org.example.virtualkey.screens.WelcomeScreen;

**public** **class** VirtualKeyApplication {

**public** **static** **void** main(String[] args) {

WelcomeScreen welcome = **new** WelcomeScreen();

welcome.introWS();

welcome.GetUserInput();

}

}

**Package:** org.example.virtualkey.entities

**File Name:** Directory.java

package org.example.virtualkey.entities;

import java.util.ArrayList;

import java.util.Collections;

import java.io.File;

import java.nio.file.FileSystems;

import java.nio.file.Path;

public class Directory {

public static final String name = "src/main/directory/";

private ArrayList<File> files = new ArrayList<File>();

Path path = FileSystems.getDefault().getPath(name).toAbsolutePath();

File Dfiles = path.toFile();

public String getName() {

return name;

}

public void print() {

System.out.println("Existing Files: ");

files.forEach(f -> System.out.println(f));

}

public ArrayList<File> fillFiles() {

File[] directoryFiles = Dfiles.listFiles();

files.clear();

for (int i = 0; i < directoryFiles.length; i++) {

if (directoryFiles[i].isFile()) {

files.add(directoryFiles[i]);

}

}

Collections.sort(files);

return files;

}

public ArrayList<File> getFiles() {

fillFiles();

return files;

}

public void mergeSort() {

}

}

**Package:** org.example.virtualkey.screens

**File Name:** FileOptionsScreen.java

package org.example.virtualkey.screens;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.file.FileSystems;

import java.nio.file.Path;

import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.Scanner;

import org.example.virtualkey.entities.Directory;

import org.example.virtualkey.services.ScreenService;

public class FileOptionsScreen implements Screen {

private Directory dir = new Directory();

private ArrayList<String> options = new ArrayList<>();

public FileOptionsScreen() {

options.add("1. To Add a New File");

options.add("2. To Delete a File");

options.add("3. Search for a File");

options.add("4. Return to Main Menu");

}

@Override

public void Show() {

System.out.println("File Option-Menu");

for (String s : options) {

System.out.println(s);

}

}

public void GetUserInput() {

int selectedOption;

while ((selectedOption = this.getOption()) != 4) {

this.NavigateOption(selectedOption);

}

}

@Override

public void NavigateOption(int option) {

switch(option) {

case 1:

this.AddFile();

this.Show();

break;

case 2:

this.DeleteFile();

this.Show();

break;

case 3:

this.SearchFile();

this.Show();

break;

default:

System.out.println("Invalid Option");

break;

}

}

public void AddFile() {

System.out.println("Please enter the Filename: ");

String fileName = this.getInputString();

System.out.println("Enterted Filename: " + fileName);

System.out.println("Enter the text: ");

String inputtext = this.getInputString();

System.out.println("Entered Text: "+ inputtext);

try {

Path path = FileSystems.getDefault().getPath(Directory.name + fileName).toAbsolutePath();

File file = new File(dir.getName() + fileName);

if (file.createNewFile()) {

System.out.println("Created File: " + file.getName());

dir.getFiles().add(file);

BufferedWriter out = new BufferedWriter(

new FileWriter(file));

out.write(inputtext);

out.close();

} else {

System.out.println("File Already Exits");

}

}catch (IOException e){

System.out.println(e);

}

}

public void DeleteFile() {

System.out.println("Please Enter the Filename: ");

String fileName = this.getInputString();

System.out.println("Entered Filename: " + fileName);

Path path = FileSystems.getDefault().getPath(Directory.name + fileName).toAbsolutePath();

File file = path.toFile();

if (file.delete()) {

System.out.println("Deleted File: " + file.getName());

dir.getFiles().remove(file);

} else {

System.out.println(fileName + ": File Not Found");

}

}

public void SearchFile() {

Boolean found = false;

System.out.println("Please Enter the Filename: ");

String fileName = this.getInputString();

System.out.println("Entered Filename: " + fileName);

ArrayList<File> files = dir.getFiles();

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

if(files.get(i).getName().equals(fileName)) {

System.out.println("Found " + fileName);

found = true;

}

}

if (found == false) {

System.out.println("File not found");

}

}

private String getInputString() {

Scanner in = new Scanner(System.in);

return(in.nextLine());

}

private int getOption() {

Scanner in = new Scanner(System.in);

int returnOption = 0;

try {

returnOption = in.nextInt();

}

catch (InputMismatchException ex) {

System.out.println("Invalid input");

}

return returnOption;

}

}

**Package:** org.example.virtualkey.screens

**File Name:** Screen.java

**package** org.example.virtualkey.screens;

**public** **interface** Screen {

**public** **void** Show();

**public** **void** NavigateOption(**int** option);

**public** **void** GetUserInput();

}

**Package:** org.example.virtualkey.screens

**File Name:** WelcomeScreen.java

package org.example.virtualkey.screens;

import org.example.virtualkey.services.DirectoryService;

import org.example.virtualkey.services.ScreenService;

import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.Scanner;

public class WelcomeScreen implements Screen {

private String welcome = "Welcome to Virtual Keys Repository";

private String devName = "Developer Name: Mukesh Kanna";

private ArrayList<String> options = new ArrayList<>();

public WelcomeScreen() {

options.add("1. Show All Files");

options.add("2. Show File Option-Menu");

options.add("3. Exit");

}

public void introWS() {

System.out.println(welcome);

System.out.println(devName);

System.out.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Show();

}

@Override

public void Show() {

System.out.println("\n");

System.out.println("Main Menu");

for (String s : options) {

System.out.println(s);

}

}

public void GetUserInput() {

int selectedOption = 0;

while ((selectedOption = this.getOption()) != 3) {

this.NavigateOption(selectedOption);

}

}

@Override

public void NavigateOption(int option) {

switch(option) {

case 1:

this.ShowFiles();

this.Show();

break;

case 2:

ScreenService.setCurrentScreen(ScreenService.FileOptionsScreen);

ScreenService.getCurrentScreen().Show();

ScreenService.getCurrentScreen().GetUserInput();

this.Show();

break;

default:

System.out.println("Invalid Option");

break;

}

}

public void ShowFiles() {

DirectoryService.PrintFiles();

}

private int getOption() {

Scanner in = new Scanner(System.in);

int returnOption = 0;

try {

returnOption = in.nextInt();

}

catch (InputMismatchException ex) {

}

return returnOption;

}

}

**Package:** org.example.virtualkey.services

**File Name:** DirectoryService.java

package org.example.virtualkey.services;

import java.io.File;

import org.example.virtualkey.entities.Directory;

public class DirectoryService {

private static Directory fileDirectory = new Directory();

public static void PrintFiles() {

fileDirectory.fillFiles();

if(fileDirectory.getFiles().size()==0) {

System.out.println("Found Not Found");

}else {

for (File file : DirectoryService.getFileDirectory().getFiles())

{

System.out.println(file.getName());

}

}

}

public static Directory getFileDirectory() {

return fileDirectory;

}

public static void setFileDirectory(Directory fileDirectory) {

DirectoryService.fileDirectory = fileDirectory;

}

}

**Package:** org.example.virtualkey.services

**File Name:** ScreenService.java

**package** org.example.virtualkey.services;

**import** org.example.virtualkey.screens.FileOptionsScreen;

**import** org.example.virtualkey.screens.Screen;

**import** org.example.virtualkey.screens.WelcomeScreen;

**public** **class** ScreenService {

**public** **static** WelcomeScreen *WelcomeScreen* = **new** WelcomeScreen();

**public** **static** FileOptionsScreen *FileOptionsScreen* = **new** FileOptionsScreen();

**public** **static** Screen *CurrentScreen* = *WelcomeScreen*;

**public** **static** Screen getCurrentScreen() {

**return** *CurrentScreen*;

}

**public** **static** **void** setCurrentScreen(Screen currentScreen) {

*CurrentScreen* = currentScreen;

}

}