**Project-5.3**

Code:

import java.io.IOException;

import java.nio.file.\*;

import java.util.\*;

import java.util.regex.Pattern;

// Item Interface

interface Item {

String MANUFACTURER = "OracleProduction";

void setProductionNumber(int number);

void setName(String name);

String getName();

Date getManufactureDate();

int getSerialNumber();

}

// ItemType Enum

enum ItemType {

AUDIO("AU"), VISUAL("VI"), AUDIO\_MOBILE("AM"), VISUAL\_MOBILE("VM");

private final String code;

ItemType(String code) {

this.code = code;

}

public String getCode() {

return code;

}

}

// Product Abstract Class

abstract class Product implements Item, Comparable<Product> {

private static int currentProductionNumber = 1;

private int serialNumber;

private String manufacturer;

private Date manufacturedOn;

private String name;

public Product(String name) {

this.name = name;

this.serialNumber = currentProductionNumber++;

this.manufacturedOn = new Date();

this.manufacturer = MANUFACTURER;

}

@Override

public void setProductionNumber(int number) {

this.serialNumber = number;

}

@Override

public void setName(String name) {

this.name = name;

}

@Override

public String getName() {

return name;

}

@Override

public Date getManufactureDate() {

return manufacturedOn;

}

@Override

public int getSerialNumber() {

return serialNumber;

}

@Override

public String toString() {

return "Manufacturer : " + manufacturer + "\n" +

"Serial Number : " + serialNumber + "\n" +

"Date : " + manufacturedOn + "\n" +

"Name : " + name;

}

@Override

public int compareTo(Product other) {

return this.getName().compareTo(other.getName());

}

}

// MultimediaControl Interface

interface MultimediaControl {

void play();

void stop();

void previous();

void next();

}

// AudioPlayer Class

class AudioPlayer extends Product implements MultimediaControl {

private String audioSpecification;

private ItemType mediaType;

public AudioPlayer(String name, String audioSpecification) {

super(name);

this.audioSpecification = audioSpecification;

this.mediaType = ItemType.AUDIO; // Default type

}

@Override

public void play() {

System.out.println("Playing");

}

@Override

public void stop() {

System.out.println("Stopped");

}

@Override

public void previous() {

System.out.println("Previous track");

}

@Override

public void next() {

System.out.println("Next track");

}

@Override

public String toString() {

return super.toString() + "\n" +

"Audio Spec : " + audioSpecification + "\n" +

"Type : " + mediaType.getCode();

}

}

// MonitorType Enum

enum MonitorType {

LCD, LED

}

// ScreenSpec Interface

interface ScreenSpec {

String getResolution();

int getRefreshRate();

int getResponseTime();

}

// Screen Class

class Screen implements ScreenSpec {

private String resolution;

private int refreshRate;

private int responseTime;

public Screen(String resolution, int refreshRate, int responseTime) {

this.resolution = resolution;

this.refreshRate = refreshRate;

this.responseTime = responseTime;

}

@Override

public String getResolution() {

return resolution;

}

@Override

public int getRefreshRate() {

return refreshRate;

}

@Override

public int getResponseTime() {

return responseTime;

}

@Override

public String toString() {

return "Resolution : " + resolution + "\n" +

"Refresh Rate : " + refreshRate + "\n" +

"Response Time : " + responseTime;

}

}

// MoviePlayer Class

class MoviePlayer extends Product implements MultimediaControl {

private Screen screen;

private MonitorType monitorType;

public MoviePlayer(String name, Screen screen, MonitorType monitorType) {

super(name);

this.screen = screen;

this.monitorType = monitorType;

}

@Override

public void play() {

System.out.println("Playing movie");

}

@Override

public void stop() {

System.out.println("Stopped movie");

}

@Override

public void previous() {

System.out.println("Previous scene");

}

@Override

public void next() {

System.out.println("Next scene");

}

@Override

public String toString() {

return super.toString() + "\n" +

"Monitor Type : " + monitorType + "\n" +

"Screen Details :\n" + screen;

}

}

// EmployeeInfo Class

class EmployeeInfo {

private StringBuilder name;

private String code;

private String deptId;

private Scanner in;

private Pattern p;

public EmployeeInfo() {

in = new Scanner(System.in);

p = Pattern.compile("[A-Z][a-z]{3}\\d{2}");

setName();

setDeptId();

}

public StringBuilder getName() {

return name;

}

public String getCode() {

return code;

}

public String getDeptId() {

return deptId;

}

private void setName() {

name = new StringBuilder(inputName());

if (checkName(name)) {

createEmployeeCode(name);

} else {

code = "guest";

}

}

private void createEmployeeCode(StringBuilder name) {

String[] parts = name.toString().split(" ");

code = (parts.length > 1) ? parts[0].charAt(0) + parts[1] : "guest";

}

private String inputName() {

System.out.print("Enter your full name (First Last): ");

return in.nextLine();

}

private boolean checkName(StringBuilder name) {

return name.toString().contains(" ");

}

private void setDeptId() {

deptId = getId();

if (!validId(deptId)) {

deptId = "None01";

} else {

deptId = reverseString(deptId);

}

}

private String getId() {

System.out.print("Enter your department ID (e.g., Abcd12): ");

return in.nextLine();

}

private boolean validId(String id) {

return p.matcher(id).matches();

}

@Override

public String toString() {

return "Employee Code: " + code + "\n" +

"Department ID: " + deptId;

}

public String reverseString(String id) {

return new StringBuilder(id).reverse().toString();

}

}

// ProcessFiles Class

class ProcessFiles {

private Path p;

private Path p2;

public ProcessFiles() {

p = Paths.get("C:", "LineTests");

p2 = p.resolve("TestResults.txt");

createDirectory();

}

private void createDirectory() {

if (!Files.exists(p)) {

try {

Files.createDirectory(p);

} catch (IOException e) {

e.printStackTrace();

}

}

}

public void writeFile(EmployeeInfo emp) throws IOException {

Files.write(p2, emp.toString().getBytes(), StandardOpenOption.APPEND, StandardOpenOption.CREATE);

}

public void writeFile(ArrayList<Product> products) throws IOException {

StringBuilder sb = new StringBuilder();

for (Product product : products) {

sb.append(product).append("\n\n");

}

Files.write(p2, sb.toString().getBytes(), StandardOpenOption.APPEND, StandardOpenOption.CREATE);

}

}

// ViewFileInfo Class

class ViewFileInfo {

public static void main(String[] args) {

Path path = Paths.get("C:", "LineTests", "TestResults.txt");

try {

List<String> lines = Files.readAllLines(path);

for (String line : lines) {

System.out.println(line);

}

} catch (IOException e) {

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

}

}

}

// ProductionLineApp Class

public class ProductionLineApp {

private static final Scanner scanner = new Scanner(System.in);

private static final ArrayList<Product> products = new ArrayList<>();

public static void main(String[] args) {

boolean running = true;

while (running) {

System.out.println("1. Add Product");

System.out.println("2. Show Products");

System.out.println("3. Show Statistics");

System.out.println("4. Exit");

int choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1:

addProduct();

break;

case 2:

showProducts();

break;

case 3:

showStatistics();

break;

case 4:

running = false;

break;

default:

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

}

}

}

private static void addProduct() {

System.out.println("Enter product type (1. AudioPlayer, 2. MoviePlayer): ");

int type = scanner.nextInt();

scanner.nextLine(); // consume newline

System.out.println("Enter product name: ");

String name = scanner.nextLine();

System.out.println("Enter audio specification (for AudioPlayer only): ");

String spec = scanner.nextLine();

if (type == 1) {

products.add(new AudioPlayer(name, spec));

} else if (type == 2) {

System.out.println("Enter screen resolution: ");

String resolution = scanner.nextLine();

System.out.println("Enter refresh rate: ");

int refreshRate = scanner.nextInt();

System.out.println("Enter response time: ");

int responseTime = scanner.nextInt();

scanner.nextLine(); // consume newline

Screen screen = new Screen(resolution, refreshRate, responseTime);

System.out.println("Enter monitor type (1. LCD, 2. LED): ");

MonitorType monitorType = (scanner.nextInt() == 1) ? MonitorType.LCD : MonitorType.LED;

scanner.nextLine(); // consume newline

products.add(new MoviePlayer(name, screen, monitorType));

} else {

System.out.println("Invalid product type");

}

}

private static void showProducts() {

for (Product product : products) {

System.out.println(product);

}

}

private static void showStatistics() {

System.out.println("Total products: " + products.size());

// Additional statistics can be added here

}

}

Output:

