Lab3: Implementation of façade Structural Pattern

**Name: Faizan**

**Registration No.: FA21-BSE-011**

**Date: 22-Oct-2024**

Contents

[Task 1: Home Automation System 3](#_Toc180488831)

[Amplifier Class: 3](#_Toc180488832)

[DVDPlayer Class: 3](#_Toc180488833)

[Lights Class: 4](#_Toc180488834)

[Projector Class: 4](#_Toc180488835)

[HomeTheaterFaçade Class: 4](#_Toc180488836)

[Main Class: 5](#_Toc180488837)

[Car Ignition System: 6](#_Toc180488838)

[Engine Class: 6](#_Toc180488839)

[Battery Class: 6](#_Toc180488840)

[FuelPump Class: 6](#_Toc180488841)

[Radiator Class: 7](#_Toc180488842)

[Car Façade Class: 7](#_Toc180488843)

[Main Class: 8](#_Toc180488844)

[Computer Startup System: 8](#_Toc180488845)

[CPU Class: 8](#_Toc180488846)

[Memory Class: 9](#_Toc180488847)

[HardDrive Class: 9](#_Toc180488848)

[Computer Façade Class: 9](#_Toc180488849)

[Main Class: 10](#_Toc180488850)

[Travel System: 10](#_Toc180488851)

[Flight Class: 10](#_Toc180488852)

[Hotel Class: 11](#_Toc180488853)

[Car Rental Class: 11](#_Toc180488854)

[Travel façade Class: 12](#_Toc180488855)

[Main Class: 12](#_Toc180488856)

[Bank Account Management System: 13](#_Toc180488857)

[Account Class: 13](#_Toc180488858)

[Notification Class: 14](#_Toc180488859)

[Transaction Class: 14](#_Toc180488860)

[Bank Façade Class: 15](#_Toc180488861)

[Main Class: 17](#_Toc180488862)

# ****Github Link:****

<https://github.com/Faizan-Kh/Solid-Design-Principles/tree/main/src/facade_structural_pattern>

# ****Task 1: Home Automation System****

## Amplifier Class:

/\*\*

\*

\* @author fa21-bse-011

\*/

class Amplifier {

public void on() {

System.out.println("Amplifier is ON.");

}

public void setVolume(int volume) {

System.out.println("Setting volume to " + volume);

}

public void off() {

System.out.println("Amplifier is OFF.");

}

}

## DVDPlayer Class:

class DVDPlayer {

public void on() {

System.out.println("DVD Player is ON.");

}

public void play(String movie) {

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

}

public void off() {

System.out.println("DVD Player is OFF.");

}

}

## Lights Class:

class Lights {

public void dim(int level) {

System.out.println("Dimming lights to " + level + "%.");

}

}

## Projector Class:

class Projector {

public void on() {

System.out.println("Projector is ON.");

}

public void off() {

System.out.println("Projector is OFF.");

}

}

## HomeTheaterFaçade Class:

class HomeTheaterFacade {

private Projector projector;

private Amplifier amplifier;

private DVDPlayer dvdPlayer;

private Lights lights;

public HomeTheaterFacade(Projector projector, Amplifier amplifier, DVDPlayer dvdPlayer, Lights lights) {

this.projector = projector;

this.amplifier = amplifier;

this.dvdPlayer = dvdPlayer;

this.lights = lights;

}

public void watchMovie(String movie) {

System.out.println("Get ready to watch a movie...");

lights.dim(10);

projector.on();

amplifier.on();

amplifier.setVolume(5);

dvdPlayer.on();

dvdPlayer.play(movie);

}

public void endMovie() {

System.out.println("Shutting down the movie theater...");

projector.off();

amplifier.off();

dvdPlayer.off();

}

}

## Main Class:

public class FacadePatternTest {

public static void main(String[] args) {

Projector projector = new Projector();

Amplifier amplifier = new Amplifier();

DVDPlayer dvdPlayer = new DVDPlayer();

Lights lights = new Lights();

HomeTheaterFacade homeTheater = new HomeTheaterFacade(projector, amplifier, dvdPlayer, lights);

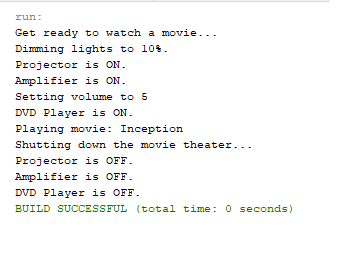
homeTheater.watchMovie("Inception");

homeTheater.endMovie();

}

}

## Output:



# Car Ignition System:

## Engine Class:

class Engine {

public void start() {

System.out.println("Engine started.");

}

public void stop() {

System.out.println("Engine stopped.");

}

}

## Battery Class:

class Battery {

Public void supplyPower () {

System.out.println ("Battery supplying power.");

}

}

## FuelPump Class:

class FuelPump {

public void pump() {

System.out.println("Fuel pump pumping fuel.");

}

}

## Radiator Class:

class Radiator {

public void on() {

System.out.println("Radiator is ON.");

}

public void off() {

System.out.println("Radiator is OFF.");

}

}

## Car Façade Class:

class CarFacade {

private Engine engine;

private Battery battery;

private FuelPump fuelPump;

private Radiator radiator;

public CarFacade(Engine engine, Battery battery, FuelPump fuelPump, Radiator radiator) {

this.engine = engine;

this.battery = battery;

this.fuelPump = fuelPump;

this.radiator = radiator;

}

public void startCar() {

System.out.println("Starting the car...");

battery.supplyPower();

fuelPump.pump();

radiator.on();

engine.start();

}

public void stopCar() {

System.out.println("Stopping the car...");

engine.stop();

radiator.off();

}

}

## Main Class:

public class FacadePatternCarTest {

public static void main(String[] args) {

Engine engine = new Engine();

Battery battery = new Battery();

FuelPump fuelPump = new FuelPump();

Radiator radiator = new Radiator();

CarFacade carFacade = new CarFacade(engine, battery, fuelPump, radiator);

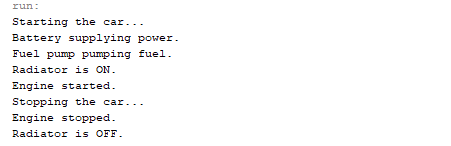
carFacade.startCar();

carFacade.stopCar();

}

}

## Output:



# Computer Startup System:

## CPU Class:

class CPU {

public void freeze() {

System.out.println("CPU is freezing...");

}

public void jump(long position) {

System.out.println("CPU is jumping to position " + position);

}

public void execute() {

System.out.println("CPU is executing instructions.");

}

}

## Memory Class:

class Memory {

public void load(long position, byte[] data) {

System.out.println("Loading data into memory from position " + position);

}

}

## HardDrive Class:

class HardDrive {

public byte[] read(long lba, int size) {

System.out.println("Reading data from hard drive at position " + lba);

return new byte[size]; // Simulating read data

}

}

## Computer Façade Class:

class ComputerFacade {

private CPU cpu;

private Memory memory;

private HardDrive hardDrive;

public ComputerFacade() {

cpu = new CPU();

memory = new Memory();

hardDrive = new HardDrive();

}

public void start() {

System.out.println("Starting the computer...");

cpu.freeze();

memory.load(0x001, hardDrive.read(0x001, 64));

cpu.jump(0x001);

cpu.execute();

}

public void shutdown() {

System.out.println("Shutting down the computer...");

}

}

## Main Class:

public class ComputerFacadeTest {

public static void main(String[] args) {

ComputerFacade computer = new ComputerFacade();

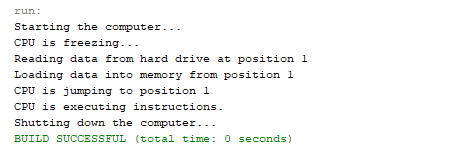
computer.start();

computer.shutdown();

}

}

## Output:



# Travel System:

## Flight Class:

class Flight {

public void bookFlight(String origin, String destination) {

System.out.println("Booking flight from " + origin + " to " + destination + ".");

}

public void cancelFlight() {

System.out.println("Flight canceled.");

}

}

## Hotel Class:

class Hotel {

public void bookHotel(String location, int nights) {

System.out.println("Booking hotel in " + location + " for " + nights + " nights.");

}

public void cancelHotel() {

System.out.println("Hotel booking canceled.");

}

}

## Car Rental Class:

class CarRental {

public void bookCar(String location) {

System.out.println("Booking car rental in " + location + ".");

}

public void cancelCarRental() {

System.out.println("Car rental canceled.");

}

}

## Travel façade Class:

class TravelFacade {

private Flight flight;

private Hotel hotel;

private CarRental carRental;

public TravelFacade() {

flight = new Flight();

hotel = new Hotel();

carRental = new CarRental();

}

public void bookTrip(String origin, String destination, String hotelLocation, int nights) {

System.out.println("Booking your trip...");

flight.bookFlight(origin, destination);

hotel.bookHotel(hotelLocation, nights);

carRental.bookCar(destination);

System.out.println("Trip booked successfully!");

}

public void cancelTrip() {

System.out.println("Canceling your trip...");

flight.cancelFlight();

hotel.cancelHotel();

carRental.cancelCarRental();

System.out.println("Trip canceled successfully!");

}

}

## Main Class:

public class TravelBookingTest {

public static void main(String[] args) {

TravelFacade travelFacade = new TravelFacade();

// Booking a trip

travelFacade.bookTrip("New York", "Los Angeles", "Los Angeles", 3);

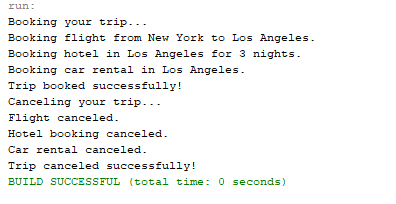
// Canceling the trip

travelFacade.cancelTrip();

}

}

## Output:



# Bank Account Management System:

## Account Class:

import java.util.HashMap;

import java.util.Map;

class Account {

private String accountNumber;

private double balance;

public Account(String accountNumber) {

this.accountNumber = accountNumber;

this.balance = 0.0;

}

public void deposit(double amount) {

balance += amount;

System.out.println("Deposited: $" + amount + ". New balance: $" + balance);

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrew: $" + amount + ". New balance: $" + balance);

} else {

System.out.println("Insufficient funds for withdrawal of $" + amount);

}

}

public double getBalance() {

return balance;

}

public String getAccountNumber() {

return accountNumber;

}

}

## Notification Class:

class Notification {

public void sendNotification(String message) {

System.out.println("Notification: " + message);

}

}

## Transaction Class:

class Transaction {

public void recordTransaction(String accountNumber, String type, double amount) {

System.out.println("Transaction recorded for account " + accountNumber + ": " + type + " of $" + amount);

}

}

## Bank Façade Class:

import java.util.HashMap;

import java.util.Map;

class BankFacade {

private Map<String, Account> accounts;

private Transaction transaction;

private Notification notification;

public BankFacade() {

accounts = new HashMap<>();

transaction = new Transaction();

notification = new Notification();

}

public void createAccount(String accountNumber) {

accounts.put(accountNumber, new Account(accountNumber));

System.out.println("Account created: " + accountNumber);

}

public void deposit(String accountNumber, double amount) {

Account account = accounts.get(accountNumber);

if (account != null) {

account.deposit(amount);

transaction.recordTransaction(accountNumber, "Deposit", amount);

notification.sendNotification("Deposited: $" + amount + " to account " + accountNumber);

} else {

System.out.println("Account does not exist: " + accountNumber);

}

}

public void withdraw(String accountNumber, double amount) {

Account account = accounts.get(accountNumber);

if (account != null) {

account.withdraw(amount);

transaction.recordTransaction(accountNumber, "Withdrawal", amount);

notification.sendNotification("Withdrew: $" + amount + " from account " + accountNumber);

} else {

System.out.println("Account does not exist: " + accountNumber);

}

}

public void checkBalance(String accountNumber) {

Account account = accounts.get(accountNumber);

if (account != null) {

System.out.println("Current balance for account " + accountNumber + ": $" + account.getBalance());

} else {

System.out.println("Account does not exist: " + accountNumber);

}

}

}

## Main Class:

import java.util.Scanner;

public class BankAccountManagementTest {

public static void main(String[] args) {

BankFacade bankFacade = new BankFacade();

Scanner scanner = new Scanner(System.in);

String choice;

// Creating accounts

System.out.println("Creating accounts...");

bankFacade.createAccount("123456789");

bankFacade.createAccount("987654321");

do {

System.out.println("\nChoose an option: ");

System.out.println("1. Deposit");

System.out.println("2. Withdraw");

System.out.println("3. Check Balance");

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

choice = scanner.nextLine();

switch (choice) {

case "1":

System.out.print("Enter account number: ");

String depositAccount = scanner.nextLine();

System.out.print("Enter amount to deposit: ");

double depositAmount = Double.parseDouble(scanner.nextLine());

bankFacade.deposit(depositAccount, depositAmount);

break;

case "2":

System.out.print("Enter account number: ");

String withdrawAccount = scanner.nextLine();

System.out.print("Enter amount to withdraw: ");

double withdrawAmount = Double.parseDouble(scanner.nextLine());

bankFacade.withdraw(withdrawAccount, withdrawAmount);

break;

case "3":

System.out.print("Enter account number: ");

String balanceAccount = scanner.nextLine();

bankFacade.checkBalance(balanceAccount);

break;

case "4":

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice. Please try again.");

}

} while (!choice.equals("4"));

scanner.close();

}

}

## Output:

