1

import java.util.Scanner;

class Logger {

private static Logger instance;

private Logger() {}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("[LOG] " + message);

}

}

public class SingletonTest {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("First message");

logger2.log("Second message");

System.out.println("Same instance? " + (logger1 == logger2));

}

}

OUTPUT: [LOG] First message

[LOG] Second message

Same instance? true

2

import java.util.Scanner;

interface Document {

String getType();

}

class WordDocument implements Document {

public String getType() {

return "Word Document";

}

}

class PdfDocument implements Document {

public String getType() {

return "PDF Document";

}

}

class ExcelDocument implements Document {

public String getType() {

return "Excel Document";

}

}

abstract class DocumentFactory {

abstract Document createDocument();

}

class WordFactory extends DocumentFactory {

Document createDocument() {

return new WordDocument();

}

}

class PdfFactory extends DocumentFactory {

Document createDocument() {

return new PdfDocument();

}

}

class ExcelFactory extends DocumentFactory {

Document createDocument() {

return new ExcelDocument();

}

}

public class FactoryTest {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter document type (word/pdf/excel): ");

String input = sc.nextLine().toLowerCase();

DocumentFactory factory;

switch (input) {

case "word":

factory = new WordFactory(); break;

case "pdf":

factory = new PdfFactory(); break;

case "excel":

factory = new ExcelFactory(); break;

default:

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

}

Document doc = factory.createDocument();

System.out.println("Created: " + doc.getType());

sc.close();

}

}

OUTPUT: Enter document type (word/pdf/excel): pdf

Created: PDF Document

Enter document type (word/pdf/excel): word

Created: Word Document

Enter document type (word/pdf/excel): excel

Created: Excel Document

3

class Computer {

private String cpu;

private String ram;

private String storage;

private Computer(Builder builder) {

this.cpu = builder.cpu;

this.ram = builder.ram;

this.storage = builder.storage;

}

public String toString() {

return "Computer [CPU=" + cpu + ", RAM=" + ram + ", Storage=" + storage + "]";

}

static class Builder {

private String cpu;

private String ram;

private String storage;

public Builder setCPU(String cpu) {

this.cpu = cpu;

return this;

}

public Builder setRAM(String ram) {

this.ram = ram;

return this;

}

public Builder setStorage(String storage) {

this.storage = storage;

return this;

}

public Computer build() {

return new Computer(this);

}

}

}

public class BuilderTest {

public static void main(String[] args) {

Computer gamingPC = new Computer.Builder()

.setCPU("Intel i9")

.setRAM("32GB")

.setStorage("1TB SSD")

.build();

Computer officePC = new Computer.Builder()

.setCPU("Intel i5")

.setRAM("16GB")

.setStorage("512GB SSD")

.build();

System.out.println("Gaming PC: " + gamingPC);

System.out.println("Office PC: " + officePC);

}

}

OUTPUT: Gaming PC: Computer [CPU=Intel i9, RAM=32GB, Storage=1TB SSD]

Office PC: Computer [CPU=Intel i5, RAM=16GB, Storage=512GB SSD]

4

interface PaymentProcessor {

void processPayment(double amount);

}

class PayPalGateway {

void send(double amt) {

System.out.println("Paid Rs. " + amt + " using PayPal");

}

}

class StripeGateway {

void charge(double amt) {

System.out.println("Charged Rs. " + amt + " using Stripe");

}

}

class PayPalAdapter implements PaymentProcessor {

PayPalGateway gateway = new PayPalGateway();

public void processPayment(double amount) {

gateway.send(amount);

}

}

class StripeAdapter implements PaymentProcessor {

StripeGateway gateway = new StripeGateway();

public void processPayment(double amount) {

gateway.charge(amount);

}

}

public class AdapterTest {

public static void main(String[] args) {

PaymentProcessor paypal = new PayPalAdapter();

PaymentProcessor stripe = new StripeAdapter();

paypal.processPayment(2000);

stripe.processPayment(1500);

}

}

OUTPUT: Paid Rs. 2000.0 using PayPal

Charged Rs. 1500.0 using Stripe

5

interface Notifier {

void send(String message);

}

class EmailNotifier implements Notifier {

public void send(String message) {

System.out.println("Email sent: " + message);

}

}

abstract class NotifierDecorator implements Notifier {

protected Notifier wrappee;

NotifierDecorator(Notifier notifier) {

this.wrappee = notifier;

}

public void send(String message) {

wrappee.send(message);

}

}

class SMSNotifierDecorator extends NotifierDecorator {

SMSNotifierDecorator(Notifier notifier) {

super(notifier);

}

public void send(String message) {

super.send(message);

System.out.println("SMS sent: " + message);

}

}

class SlackNotifierDecorator extends NotifierDecorator {

SlackNotifierDecorator(Notifier notifier) {

super(notifier);

}

public void send(String message) {

super.send(message);

System.out.println("Slack sent: " + message);

}

}

public class DecoratorTest {

public static void main(String[] args) {

Notifier notifier = new SlackNotifierDecorator(new SMSNotifierDecorator(new EmailNotifier()));

notifier.send("System maintenance at 10PM");

}

}

OUTPUT: Email sent: System maintenance at 10PM

SMS sent: System maintenance at 10PM

Slack sent: System maintenance at 10PM

6

interface Image {

void display();

}

class RealImage implements Image {

String fileName;

RealImage(String fileName) {

this.fileName = fileName;

loadFromServer();

}

void loadFromServer() {

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

}

public void display() {

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

}

}

class ProxyImage implements Image {

RealImage realImage;

String fileName;

ProxyImage(String fileName) {

this.fileName = fileName;

}

public void display() {

if (realImage == null) {

realImage = new RealImage(fileName);

}

realImage.display();

}

}

public class ProxyTest {

public static void main(String[] args) {

Image img = new ProxyImage("photo.jpg");

img.display();

img.display();

}

}

OUTPUT: Loading photo.jpg

Displaying photo.jpg

Displaying photo.jpg

7

import java.util.\*;

interface Observer {

void update(String stock, double price);

}

interface Stock {

void register(Observer o);

void deregister(Observer o);

void notifyObservers();

}

class StockMarket implements Stock {

List<Observer> observers = new ArrayList<>();

String stockName;

double stockPrice;

public void register(Observer o) {

observers.add(o);

}

public void deregister(Observer o) {

observers.remove(o);

}

public void setStock(String name, double price) {

this.stockName = name;

this.stockPrice = price;

notifyObservers();

}

public void notifyObservers() {

for (Observer o : observers) {

o.update(stockName, stockPrice);

}

}

}

class MobileApp implements Observer {

public void update(String stock, double price) {

System.out.println("MobileApp - " + stock + " updated to Rs. " + price);

}

}

class WebApp implements Observer {

public void update(String stock, double price) {

System.out.println("WebApp - " + stock + " updated to Rs. " + price);

}

}

public class ObserverTest {

public static void main(String[] args) {

StockMarket market = new StockMarket();

Observer mobile = new MobileApp();

Observer web = new WebApp();

market.register(mobile);

market.register(web);

market.setStock("TCS", 3500);

market.setStock("Infosys", 1480);

}

}

OUTPUT: MobileApp - TCS updated to Rs. 3500.0

WebApp - TCS updated to Rs. 3500.0

MobileApp - Infosys updated to Rs. 1480.0

WebApp - Infosys updated to Rs. 1480.0

8

import java.util.Scanner;

interface PaymentStrategy {

void pay(double amount);

}

class CreditCardPayment implements PaymentStrategy {

public void pay(double amount) {

System.out.println("Paid Rs. " + amount + " via Credit Card");

}

}

class PayPalPayment implements PaymentStrategy {

public void pay(double amount) {

System.out.println("Paid Rs. " + amount + " via PayPal");

}

}

class PaymentContext {

private PaymentStrategy strategy;

public void setPaymentStrategy(PaymentStrategy strategy) {

this.strategy = strategy;

}

public void executePayment(double amount) {

strategy.pay(amount);

}

}

public class StrategyExample {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

PaymentContext context = new PaymentContext();

System.out.println("Enter payment amount:");

double amount = sc.nextDouble();

System.out.println("Select payment method (1=CreditCard, 2=PayPal):");

int choice = sc.nextInt();

if (choice == 1) context.setPaymentStrategy(new CreditCardPayment());

else context.setPaymentStrategy(new PayPalPayment());

context.executePayment(amount);

sc.close();

}

}

OUTPUT: Enter payment amount:

5000

Select payment method (1=CreditCard, 2=PayPal):

2

Paid Rs. 5000.0 via PayPal

9

interface Command {

void execute();

}

class Light {

void on() {

System.out.println("Light turned ON");

}

void off() {

System.out.println("Light turned OFF");

}

}

class LightOnCommand implements Command {

Light light;

LightOnCommand(Light light) {

this.light = light;

}

public void execute() {

light.on();

}

}

class LightOffCommand implements Command {

Light light;

LightOffCommand(Light light) {

this.light = light;

}

public void execute() {

light.off();

}

}

class RemoteControl {

Command command;

void setCommand(Command command) {

this.command = command;

}

void pressButton() {

command.execute();

}

}

public class CommandExample {

public static void main(String[] args) {

Light light = new Light();

RemoteControl remote = new RemoteControl();

remote.setCommand(new LightOnCommand(light));

remote.pressButton();

remote.setCommand(new LightOffCommand(light));

remote.pressButton();

}

}

OUTPUT: Light turned ON

Light turned OFF

10

class Student {

private String name;

private int id;

private String grade;

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getGrade() { return grade; }

public void setGrade(String grade) { this.grade = grade; }

}

class StudentView {

public void displayStudentDetails(String name, int id, String grade) {

System.out.println("Name: " + name + ", ID: " + id + ", Grade: " + grade);

}

}

class StudentController {

private Student student;

private StudentView view;

public StudentController(Student student, StudentView view) {

this.student = student;

this.view = view;

}

public void updateView() {

view.displayStudentDetails(student.getName(), student.getId(), student.getGrade());

}

public void setStudentName(String name) { student.setName(name); }

public void setStudentId(int id) { student.setId(id); }

public void setStudentGrade(String grade) { student.setGrade(grade); }

}

public class MVCExample {

public static void main(String[] args) {

Student s = new Student();

StudentView v = new StudentView();

StudentController c = new StudentController(s, v);

c.setStudentName("Ananya");

c.setStudentId(101);

c.setStudentGrade("A");

c.updateView();

}

}

OUTPUT: Name: Ananya, ID: 101, Grade: A

11

interface CustomerRepository {

String findCustomerById(int id);

}

class CustomerRepositoryImpl implements CustomerRepository {

public String findCustomerById(int id) {

return "Customer #" + id + ": Raj Sharma";

}

}

class CustomerService {

private CustomerRepository repository;

public CustomerService(CustomerRepository repository) {

this.repository = repository;

}

public void showCustomer(int id) {

System.out.println(repository.findCustomerById(id));

}

}

public class DIExample {

public static void main(String[] args) {

CustomerRepository repo = new CustomerRepositoryImpl();

CustomerService service = new CustomerService(repo);

service.showCustomer(7);

}

}

OUTPUT: Customer #7: Raj Sharma