Day 24 aug 6

Name Aravind Kasanagottu

Id mvsnarav

// Task 1: Factory Method Pattern

// Interface

public interface Login {

void login(String username, String password);

}

// Concrete Classes

public class WebLogin implements Login {

@Override

public void login(String username, String password) {

System.out.println("Logging in through Web with username: " + username);

}

}

public class MobileLogin implements Login {

@Override

public void login(String username, String password) {

System.out.println("Logging in through Mobile with username: " + username);

}

}

// Factory Class

public class LoginFactory {

public static Login getLogin(String type) {

if (type.equalsIgnoreCase("web")) {

return new WebLogin();

} else if (type.equalsIgnoreCase("mobile")) {

return new MobileLogin();

}

throw new IllegalArgumentException("Unknown login type: " + type);

}

}

// Main Class

public class Main {

public static void main(String[] args) {

Login webLogin = LoginFactory.getLogin("web");

webLogin.login("webUser", "webPass");

Login mobileLogin = LoginFactory.getLogin("mobile");

mobileLogin.login("mobileUser", "mobilePass");

}

}

// Task 2: Singleton Pattern

public class SingletonDPConfigManager {

private static SingletonDPConfigManager configManagerInstance;

private Properties properties;

private SingletonDPConfigManager() {

properties = new Properties();

try (InputStream input = getClass().getClassLoader().getResourceAsStream("config.prop")) {

if (input == null) {

System.out.println("Cannot find config properties");

return;

}

properties.load(input);

} catch (IOException e) {

e.printStackTrace();

}

}

public static synchronized SingletonDPConfigManager getConfigManagerInstance() {

if (configManagerInstance == null) {

configManagerInstance = new SingletonDPConfigManager();

}

return configManagerInstance;

}

public String getProperty(String key) {

return properties.getProperty(key);

}

}

// Task 3: Abstract Factory Pattern

// See original code as it's quite long and complete

// Task 4: Prototype Pattern

abstract class HumanBeing implements Cloneable {

protected String name;

protected String type;

public abstract void walk();

public abstract void see();

public abstract void talk();

@Override

public Object clone() {

try {

return super.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

return null;

}

}

}

class SpiderMan extends HumanBeing {

public SpiderMan() {

type = "SpiderMan";

}

@Override

public void walk() {

System.out.println("SpiderMan walks silently");

}

@Override

public void see() {

System.out.println("SpiderMan sees with spider-sense");

}

@Override

public void talk() {

System.out.println("SpiderMan talks humorously");

}

}

// Add other classes (BatMan, IronMan) similarly  
  
  
  
  
// Task 3: Abstract Factory Pattern

package Aug6;

// Interfaces

interface NoteBook {

void writing();

}

interface TextBook {

void reading();

}

// Concrete Products for NoteBook

class LongBookNote implements NoteBook {

public void writing() {

System.out.println("Writing in Long NoteBook");

}

}

class ShortBookNote implements NoteBook {

public void writing() {

System.out.println("Writing in Short NoteBook");

}

}

// Concrete Products for TextBook

class LongBookText implements TextBook {

public void reading() {

System.out.println("Reading from Long TextBook");

}

}

class ShortBookText implements TextBook {

public void reading() {

System.out.println("Reading from Short TextBook");

}

}

// Abstract Factory Interface

interface BookFactory {

NoteBook createLongBookNote();

NoteBook createShortBookNote();

TextBook createLongBookText();

TextBook createShortBookText();

}

// Concrete Factory for Notebooks

class NoteBookFactory implements BookFactory {

public NoteBook createLongBookNote() {

return new LongBookNote();

}

public NoteBook createShortBookNote() {

return new ShortBookNote();

}

public TextBook createLongBookText() {

return null;

}

public TextBook createShortBookText() {

return null;

}

}

// Concrete Factory for Textbooks

class TextBookFactory implements BookFactory {

public TextBook createLongBookText() {

return new LongBookText();

}

public TextBook createShortBookText() {

return new ShortBookText();

}

public NoteBook createLongBookNote() {

return null;

}

public NoteBook createShortBookNote() {

return null;

}

}

// Driver Class

public class Driver {

public static void main(String[] args) {

System.out.println("=== Using NoteBookFactory ===");

BookFactory notebookFactory = new NoteBookFactory();

NoteBook longNote = notebookFactory.createLongBookNote();

NoteBook shortNote = notebookFactory.createShortBookNote();

if (longNote != null) longNote.writing();

if (shortNote != null) shortNote.writing();

System.out.println("\n=== Using TextBookFactory ===");

BookFactory textbookFactory = new TextBookFactory();

TextBook longText = textbookFactory.createLongBookText();

TextBook shortText = textbookFactory.createShortBookText();

if (longText != null) longText.reading();

if (shortText != null) shortText.reading();

}

}

// Task 4: Prototype Pattern

import java.util.HashMap;

// Prototype Abstract Class

abstract class HumanBeing implements Cloneable {

protected String name;

protected String type;

public abstract void walk();

public abstract void see();

public abstract void talk();

@Override

public Object clone() {

Object clone = null;

try {

clone = super.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

return clone;

}

}

// Concrete Prototype: SpiderMan

class SpiderMan extends HumanBeing {

public SpiderMan() {

type = "SpiderMan";

}

@Override

public void walk() {

System.out.println("SpiderMan walks silently");

}

@Override

public void see() {

System.out.println("SpiderMan sees with spider-sense");

}

@Override

public void talk() {

System.out.println("SpiderMan talks humorously");

}

}

// Concrete Prototype: BatMan

class BatMan extends HumanBeing {

public BatMan() {

type = "BatMan";

}

@Override

public void walk() {

System.out.println("BatMan walks in the shadows");

}

@Override

public void see() {

System.out.println("BatMan sees with night vision");

}

@Override

public void talk() {

System.out.println("BatMan talks deeply");

}

}

// Concrete Prototype: IronMan

class IronMan extends HumanBeing {

public IronMan() {

type = "IronMan";

}

@Override

public void walk() {

System.out.println("IronMan walks with style");

}

@Override

public void see() {

System.out.println("IronMan sees with AI");

}

@Override

public void talk() {

System.out.println("IronMan talks sarcastically");

}

}

// Cache/Registry for prototypes

class CloneOfHuman {

private static HashMap<String, HumanBeing> sizeMap = new HashMap<>();

public static void loadCache() {

SpiderMan sp = new SpiderMan();

BatMan bm = new BatMan();

IronMan im = new IronMan();

sizeMap.put("SpiderMan", sp);

sizeMap.put("BatMan", bm);

sizeMap.put("IronMan", im);

}

public static HumanBeing getSize(String type) {

HumanBeing cachedHuman = sizeMap.get(type);

return (HumanBeing) cachedHuman.clone();

}

}

// Driver class

public class PrototypeMethodDP {

public static void main(String[] args) {

CloneOfHuman.loadCache();

HumanBeing clone1 = CloneOfHuman.getSize("SpiderMan");

System.out.println("Cloned: " + clone1.type);

clone1.walk();

clone1.see();

clone1.talk();

System.out.println();

HumanBeing clone2 = CloneOfHuman.getSize("BatMan");

System.out.println("Cloned: " + clone2.type);

clone2.walk();

clone2.see();

clone2.talk();

System.out.println();

HumanBeing clone3 = CloneOfHuman.getSize("IronMan");

System.out.println("Cloned: " + clone3.type);

clone3.walk();

clone3.see();

clone3.talk();

}

}s