Sabahat siddique

Fa20-bse-054

import java.util.ArrayList;

import java.util.List;

// Singleton for Settings

class Settings {

private static Settings instance;

private Settings() {

// private constructor to prevent instantiation

}

public static synchronized Settings getInstance() {

if (instance == null) {

instance = new Settings();

}

return instance;

}

// Add settings specific to your application

}

// Mediator to control coupling

class Mediator {

// Mediator logic for communication between different components

}

// Facade for easy interface

class ExamSystemFacade {

private Mediator mediator;

public ExamSystemFacade() {

this.mediator = new Mediator();

}

// Facade methods for simplified interactions

// e.g., scheduleExam, monitorProgress, etc.

}

// Strategy for different exam conduction

interface ExamConductionStrategy {

void conductExam();

}

class TimedExamStrategy implements ExamConductionStrategy {

@Override

public void conductExam() {

// Implementation for a timed exam

}

}

class OpenBookExamStrategy implements ExamConductionStrategy {

@Override

public void conductExam() {

// Implementation for an open-book exam

}

}

// Builder for Exam

class ExamBuilder {

private List<Question> questions = new ArrayList<>();

public ExamBuilder addQuestion(Question question) {

questions.add(question);

return this;

}

public Exam build() {

return new Exam(questions);

}

}

class Exam {

private List<Question> questions;

public Exam(List<Question> questions) {

this.questions = questions;

}

// Methods to get/set questions, conduct exam, etc.

}

// Iterator for Stakeholder or Contents Iteration

interface Iterator<T> {

boolean hasNext();

T next();

}

class StakeholderIterator implements Iterator<Stakeholder> {

// Implementation for iterating over stakeholders

}

// Template Method for Taking Exams

abstract class ExamTemplate {

public void takeExam() {

// Common steps for taking an exam

conductInstructions();

conductExam();

submitExam();

}

protected abstract void conductInstructions();

protected abstract void conductExam();

protected abstract void submitExam();

}

// Observer for Real-Time Communication

interface Observer {

void update();

}

class ParentObserver implements Observer {

// Implementation for updating parent on child's progress

}

// Example Question classes

abstract class Question {

// Common properties/methods for all types of questions

}

class MCQQuestion extends Question {

// Implementation for multiple-choice questions

}

class TrueFalseQuestion extends Question {

// Implementation for true/false questions

}

// Other question types...

// Example Stakeholder classes

class Stakeholder {

// Common properties/methods for all stakeholders

}

public class Main {

public static void main(String[] args) {

// Singleton for Settings

Settings settings = Settings.getInstance();

// Facade for easy interface

ExamSystemFacade examSystemFacade = new ExamSystemFacade();

// Strategy for different exam conduction

ExamConductionStrategy timedExamStrategy = new TimedExamStrategy();

ExamConductionStrategy openBookExamStrategy = new OpenBookExamStrategy();

// Builder for Exam

ExamBuilder examBuilder = new ExamBuilder();

Exam exam = examBuilder

.addQuestion(new MCQQuestion())

.addQuestion(new TrueFalseQuestion())

// Add other types of questions as needed

.build();

// Iterator for Stakeholder or Contents Iteration

Iterator<Stakeholder> stakeholderIterator = new StakeholderIterator();

// Template Method for Taking Exams

ExamTemplate examTemplate = new ExamTemplate() {

@Override

protected void conductInstructions() {

// Implementation for conducting instructions

}

@Override

protected void conductExam() {

// Implementation for conducting the exam

}

@Override

protected void submitExam() {

// Implementation for submitting the exam

}

};

// Observer for Real-Time Communication

Observer parentObserver = new ParentObserver();

// Example: Using the classes

examSystemFacade.scheduleExam(exam, timedExamStrategy);

examSystemFacade.monitorProgress(parentObserver);

// Example of using the Template Method

examTemplate.takeExam();

// Other interactions and demonstrations...

}

}