ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ институты

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_кафедрасы



|  |
| --- |
|  |

ЗЕРТХАНАЛЫҚ ЖҰМЫС

Тақырыбы **Паттерны поведения. Стратегия. Наблюдатель**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Орындалған жұмыстың сапасы** | **Баға диапазоны** | **Алған %** |
| 1 | Орындалмаған  Себепсіз сабақтан қалуы | 0% |  |
| 2 | Орындалуы және білім алушының белсенділігі | 0-50% |  |
| 3 | Жұмыстың рәсімделуі | 0-20% |  |
| 4 | Анықтамаларды, техникалық әдебиеттерді, дәріс конспектілерін, пәннің оқу-әдістемелік кешенін пайдалана білуі | 0-5% |  |
| 5 | Техникалық құралдарды пайдалана білуі | 0-5% |  |
| 6 | Жұмысты қорғауы | 0-20% |  |
|  | Қорытынды | 0-100% |  |

Білім алушының аты-жөні

Шахмуханбетов Ханкелді

Мамандық шифрі

6B06102 Computer Science

Оқытушының аты-жөні

Иманбекова У.Н.

Алматы 2024 ж

Ф ҚазҰТЗУ 706-04. Зертханалық жұмыс

**Strategy**

public interface IShippingStrategy {  
 double calculateShippingCost(double weight, double distance);  
}

public class StandardShippingStrategy implements IShippingStrategy {  
 @Override  
 public double calculateShippingCost(double weight, double distance) {  
 return weight \* 0.5 + distance \* 0.1;  
 }  
}

public class ExpressShippingStrategy implements IShippingStrategy {  
 @Override  
 public double calculateShippingCost(double weight, double distance) {  
 return (weight \* 0.75 + distance \* 0.2) + 10; *// Жылдамдық үшін қосымша ақы* }  
}

public class InternationalShippingStrategy implements IShippingStrategy {  
 @Override  
 public double calculateShippingCost(double weight, double distance) {  
 return weight \* 1.0 + distance \* 0.5 + 15; *//Таможнялық сборлар* }  
}

public class DeliveryContext {  
 private IShippingStrategy shippingStrategy;  
  
 *//Жеткізу стратегиясын орындау* public void setShippingStrategy(IShippingStrategy strategy) {  
 this.shippingStrategy = strategy;  
 }  
  
 *//Жеткізу бағасы* public double calculateCost(double weight, double distance) {  
 if (shippingStrategy == null) {  
 throw new IllegalStateException("Стратегия доставки не установлена.");  
 }  
 if (weight <= 0 || distance <= 0) {  
 throw new IllegalArgumentException("Вес и расстояние должны быть положительными числами.");  
 }  
 return shippingStrategy.calculateShippingCost(weight, distance);  
 }  
}

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 DeliveryContext deliveryContext = new DeliveryContext();  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Выберите тип доставки: 1 - Стандартная, 2 - Экспресс, 3 - Международная");  
 String choice = scanner.nextLine();  
 switch (choice) {  
 case "1":  
 deliveryContext.setShippingStrategy(new StandardShippingStrategy());  
 break;  
 case "2":  
 deliveryContext.setShippingStrategy(new ExpressShippingStrategy());  
 break;  
 case "3":  
 deliveryContext.setShippingStrategy(new InternationalShippingStrategy());  
 break;  
 default:  
 System.*out*.println("Неверный выбор.");  
 return;  
 }  
 System.*out*.println("Введите вес посылки (кг):");  
 double weight = scanner.nextDouble();  
 System.*out*.println("Введите расстояние доставки (км):");  
 double distance = scanner.nextDouble();  
 try {  
 double cost = deliveryContext.calculateCost(weight, distance);  
 System.*out*.printf("Стоимость доставки: %.2f%n", cost);  
 } catch (Exception e) {  
 System.*out*.println("Ошибка: " + e.getMessage());  
 }  
 }  
}

Өзін-өзі тексеру сұрақтары:

1. **Осы жобада Стратегия үлгісін пайдаланудың қандай пайдасы бар?**

«Стратегия» үлгісінің артықшылықтары:

* Клиент кодын өзгертпестен жеткізу құнын есептеу алгоритмін оңай өзгертуге мүмкіндік береді.
* Жаңа есептеу әдістерін (стратегияларды) бұрыннан барларын бұзбай қосуды жеңілдетеді.

1. **Қолданыстағы кодты өзгертпей, бағдарламаның әрекетін қалай өзгертуге болады?**

Сіз жаңа стратегияны (мысалы, түнде жеткізу) қосып, оны бар кодты өзгертпестен жай ғана контекстке жібере аласыз.

1. **Әрбір есептеу әдісінің тәуелсіз болуы және жеке сыныпта жүзеге асырылуы неліктен маңызды?**

Әрбір стратегия кодты икемді және қолдауға болатын етіп нақты есептеу логикасын қамтиды.

**Observer**

public interface IObserver {  
 void update(float temperature);  
}

public interface ISubject {  
 void registerObserver(IObserver observer);  
 void removeObserver(IObserver observer);  
 void notifyObservers();  
}

import java.util.ArrayList;  
import java.util.List;  
*//температураны сақтайды және бақылаушыларға хабарлайды*public class WeatherStation implements ISubject {  
 private List<IObserver> observers;  
 private float temperature;  
 public WeatherStation() {  
 observers = new ArrayList<>();  
 }  
 @Override  
 public void registerObserver(IObserver observer) {  
 observers.add(observer);  
 }  
 @Override  
 public void removeObserver(IObserver observer) {*//Қатені өңдеу (мысалы, жоқ бақылаушыны жою әрекеті):* if (observers.contains(observer)) {  
 observers.remove(observer);  
 } else {  
 System.*out*.println("Наблюдатель не найден.");  
 }  
 }  
 @Override  
 public void notifyObservers() {  
 for (IObserver observer : observers) {  
 observer.update(temperature);  
 }  
 }  
 public void setTemperature(float newTemperature) {  
 System.*out*.println("Изменение температуры: " + newTemperature + "°C");  
 temperature = newTemperature;  
 notifyObservers();  
 }  
}

public class WeatherDisplay implements IObserver {*//жаңартылған температура деректерін көрсетеді* private String name;  
 public WeatherDisplay(String name) {  
 this.name = name;  
 }  
 @Override  
 public void update(float temperature) {  
 System.*out*.println(name + " показывает новую температуру: " + temperature + "°C");  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 WeatherStation weatherStation = new WeatherStation();  
 WeatherDisplay mobileApp = new WeatherDisplay("Мобильное приложение");  
 WeatherDisplay digitalBillboard = new WeatherDisplay("Электронное табло");  
  
 weatherStation.registerObserver(mobileApp);  
 weatherStation.registerObserver(digitalBillboard);  
 *//Бақылаушыларды хабардар ету үшін температураны өзгерту* weatherStation.setTemperature(25.0f);  
 weatherStation.setTemperature(30.0f);  
 *//Дисплейлердің бірін алып тастап, температураны қайтадан өзгертеміз* weatherStation.removeObserver(digitalBillboard);  
 weatherStation.setTemperature(28.0f);  
 }  
}

public class EmailNotification implements IObserver {*//Функционалдылықты кеңейту - бақылаушының жаңа түрін қосу* private String email;  
 public EmailNotification(String email) {  
 this.email = email;  
 }  
 @Override  
 public void update(float temperature) {  
 System.*out*.println("Отправка email на " + email + ": Температура изменилась на " + temperature + "°C");  
 }  
}

Өзін-өзі тексеру сұрақтары:

1. **Осы жобада Observer үлгісін пайдаланудың қандай пайдасы бар?**

Observer үлгісінің артықшылықтары:

* Актер кодын өзгертпестен бақылаушыларды (мысалы, әртүрлі деректерді көрсету интерфейстері) оңай қосуға немесе жоюға мүмкіндік береді.
* Субъектінің күйі өзгерген кезде барлық жазылған бақылаушыларға автоматты хабарландыру береді.

1. **Тақырып кодын өзгертпей бақылаушылар тізімін қалай өзгертуге болады?**

Тақырып кодын өзгертпестен, registerObserver() және removeObserver() әдістерін пайдаланып бақылаушыны оңай қосуға немесе жоюға болады.

1. **Жүйені асинхронды ету үшін іске асыруға қандай өзгерістер енгізуге болады?**

Асинхронды хабарландыруды жүзеге асыру үшін әр бақылаушыға хабарландыруларды бөлек ағында жіберу арқылы көп ағынды пайдалануға болады.