ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

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

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



|  |
| --- |
|  |

ЗЕРТХАНАЛЫҚ ЖҰМЫС

Тақырыбы **Проект принциптері**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **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

*//SINGLE-RESPONSIBILITY-PRINCIPLE каждый класс имеет одну  
// ответственность.  
/\*  
import java.util.List;  
  
class Item {  
 public double price;  
  
 public Item(double price) {  
 this.price = price;  
 }  
}  
  
public class Invoice {  
 private int id;  
 private List<Item> items;  
 private double taxRate;  
  
 public Invoice(int id, List<Item> items, double taxRate) {  
 this.id = id;  
 this.items = items;  
 this.taxRate = taxRate;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public List<Item> getItems() {  
 return items;  
 }  
  
 public double getTaxRate() {  
 return taxRate;  
 }  
}  
  
class InvoiceCalculator {  
 public double calculateTotal(Invoice invoice) {  
 double subTotal = invoice.getItems().stream().mapToDouble(item -> item.price).sum();  
 return subTotal + (subTotal \* invoice.getTaxRate());  
 }  
}  
  
class InvoiceRepository {  
 public void saveToDatabase(Invoice invoice) {  
 System.out.println("Invoice with ID: " + invoice.getId() + " saved to the database.");  
 }  
}  
\*/  
  
//OPEN-CLOSED-PRINCIPLE классы должны быть открыты для расширения,  
// но закрыты для модификации. То есть- классы можно расширять, не  
// изменяя существующий код.  
/\*  
interface DiscountStrategy {  
 double applyDiscount(double amount);  
}  
  
class RegularDiscount implements DiscountStrategy {  
 public double applyDiscount(double amount) {  
 return amount;  
 }  
}  
  
class SilverDiscount implements DiscountStrategy {  
 public double applyDiscount(double amount) {  
 return amount \* 0.9;  
 }  
}  
  
class GoldDiscount implements DiscountStrategy {  
 public double applyDiscount(double amount) {  
 return amount \* 0.8;  
 }  
}  
  
class PlatinumDiscount implements DiscountStrategy {  
 public double applyDiscount(double amount) {  
 return amount \* 0.99;  
 }  
}  
  
class DiscountCalculator {  
 public double calculateDiscount(DiscountStrategy strategy, double amount) {  
 return strategy.applyDiscount(amount);  
 }  
}  
\*/  
  
//INTERFACE-SEGREGATION-PRINCIPLE интерфейсы разделены по конкретным  
// обязанностям, чтобы классы реализовывали только нужные методы.  
/\*  
interface Worker {  
 void work();  
}  
  
interface Eater {  
 void eat();  
}  
  
interface Sleeper {  
 void sleep();  
}  
  
class HumanWorker implements Worker, Eater, Sleeper {  
 public void work() {  
 System.out.println("Human is working");  
 }  
  
 public void eat() {  
 System.out.println("Human is eating");  
 }  
  
 public void sleep() {  
 System.out.println("Human is sleeping");  
 }  
}  
  
class RobotWorker implements Worker {  
 public void work() {  
 System.out.println("Robot is working");  
 }  
}  
 \*/  
  
//DEPENDECY-INVERSION-PRINCIPLE высокоуровневые модули не  
// должны зависеть от низкоуровневых, оба должны зависеть от абстракций.  
/\*  
interface NotificationService {  
 void send(String message);  
}  
  
class EmailService implements NotificationService {  
 public void send(String message) {  
 System.out.println("Sending Email: " + message);  
 }  
}  
  
class Notification {  
 private NotificationService service;  
  
 public Notification(NotificationService service) {  
 this.service = service;  
 }  
  
 public void send(String message) {  
 service.send(message);  
 }  
}  
 \*/*