Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Практическое занятие №9.2**»

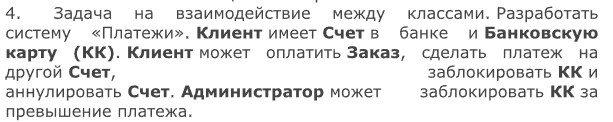
Выполнил: Валиев Айнур Ильдусович

Группа: ПР-22

Преподаватель: Мирошниченко Г. В.

2024

**Задание:**

****

**Входные и выходные данные**

**clientName – string/строка**

**bankAccountNumber – int / число**

**bankAccountBalance – Double/Число с плавающей запятой**

**bankAccount1**

**creditCardLimit – Double / Число с плавающей запятой**

**creditCard1**

**client1**

**administrator**

**order – string / строка**

**orderTotalAmount – Double/ Число с плавающей запятой**

**order1**

**Блок-схема**

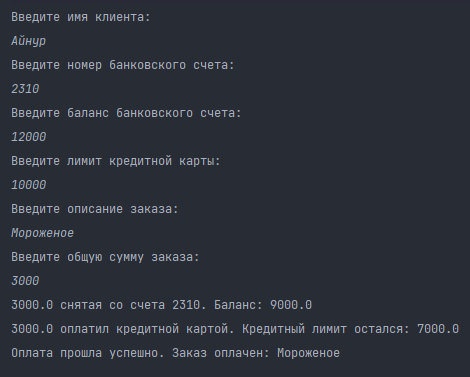
**Листинг программы (если есть)**

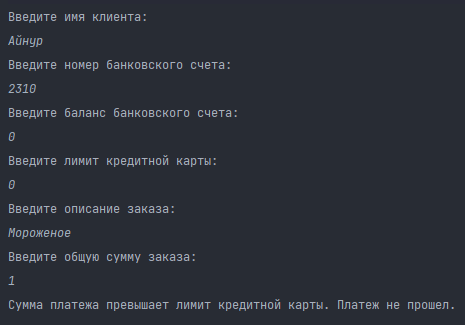
*fun* main() {  
 *try* {  
 *println*("Введите имя клиента: ")  
 *val* clientName = *readLine*()!!  
 *println*("Введите номер банковского счета: ")  
 *val* bankAccountNumber = *readLine*()!!  
 *println*("Введите баланс банковского счета: ")  
 *val* bankAccountBalance = *readLine*()!!.*toDouble*()  
 *val* bankAccount1 = BankAccount(bankAccountNumber, bankAccountBalance)  
 *println*("Введите лимит кредитной карты: ")  
 *val* creditCardLimit = *readLine*()!!.*toDouble*()  
 *val* creditCard1 = CreditCard(creditCardLimit)  
 *val* client1 = Client(clientName, bankAccount1, creditCard1)  
 *val* administrator = Administrator()  
 *println*("Введите описание заказа: ")  
 *val* order = *readLine*()!!.toString()  
 *println*("Введите общую сумму заказа: ")  
 *val* orderTotalAmount = *readLine*()!!.*toDouble*()  
 *val* order1 = Orders(order, orderTotalAmount)  
 client1.makePayment(order1)  
 } *catch* (e:Exception) { *println*("Неверно веденно значение(-я)")}  
}

=============================================================================

*class* Client(*val* name: String, *val* bankAccount: BankAccount, *var* creditCard: CreditCard) {  
 *fun* makePayment(order: Orders){  
 *val* paymentAmount = order.totalAmount  
 *if*(creditCard.isBlocked){  
 *println*("Кредитная карта заблокирована. Платеж не прошел.")  
 } *else if*(paymentAmount > creditCard.limit){  
 *println*("Сумма платежа превышает лимит кредитной карты. Платеж не прошел.")  
 } *else* {  
 bankAccount.withdraw(paymentAmount)  
 creditCard.makePayment(paymentAmount)  
 *println*("Оплата прошла успешно. Заказ оплачен: ${order.description}")  
 }  
 }

**Тестовые ситуации**

****

****

**Задание 2:**

**Входные и выходные данные**

**productName – string/строка**

**productPrice – double/число с плавающей запятой**

**product**

**customerName – string/строка**

**salesperson**

**quantity – int/число**

**order**

**blspi – int/число**

**Блок-схема**

**Листинг программы (если есть)**

*fun* main() {  
 *try* {  
 *print*("Введите название продукта: ")  
 *val* productName = *readLine*()!!  
 *print*("Введите цену товара: ")  
 *val* productPrice = *readLine*()!!.*toDouble*()  
 *val* product = Product(productName, productPrice)  
 *print*("Введите имя клиента: ")  
 *val* customerName = *readLine*()!!  
 *val* customer = Customer(customerName)  
 *val* salesperson = Salesperson()  
 product.getInfo()  
 *print*("Введите количество для заказа: ")  
 *val* quantity = *readLine*()!!.*toInt*()  
 customer.makeOrder(product, quantity)  
 *val* order = Order(product, quantity)  
 salesperson.registerSale(order)  
 *print*("Оплатил ли покупатель товар? 1-Да, 2-Нет : ")  
 *val* blspi = *readLine*()!!.*toInt*()  
 *if* (blspi==1)  
 {  
 *print*("Спасибо за покупку!")  
 }  
 *if* (blspi==2)  
 {  
 salesperson.addToBlacklist(customer)  
 }  
 } *catch* (e:Exception) { *println*("Неверно веденно значение(-я)")}  
}

=============================================================================

*class* Product(*val* name: String, *var* price: Double){  
 *fun* getInfo(): String {  
 *return* "Product: $name, Price: $price"  
 }  
 }  
 *class* Customer(*val* name: String){  
 *fun* makeOrder(product: Product, quantity: Int){  
 *val* order = Order(product, quantity)  
 order.checkout()  
 }  
 }  
 *class* Order(*val* product: Product, *val* quantity: Int){  
 *fun* checkout(){  
 *val* totalPrice = product.price \* quantity  
 *println*("Заказ подтвержден: ${product.name} x $quantity. Итоговая цена: $totalPrice")  
 }  
 }  
 *class* Salesperson{  
 *fun* registerSale(order: Order){  
 *println*("Продажа зарегистрирована на продукт: ${order.product.name} x ${order.quantity}")  
 }  
 *fun* addToBlacklist(customer: Customer){  
 *println*("Клиент ${customer.name} добавлен в черный список.")  
 }  
 }