МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет

имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

**Отчет**

**по лабораторной работе №4**

**по дисциплине «Объектно-ориентированное программирование»**

Выполнил

студент группы Б22-191-2: Балабанов В. А.

Принял: старший преподаватель, Чернышев К. С.

Ижевск, 2023

**Постановка задачи**

Нужно выделить классы, необходимые для описания предметной области; разработать структуру каждого класса (необходимые методы и поля). После этого реализовать приложение с использованием разработанных типов данных. Всего необходимо реализовать не менее 3–4 классов в рамках автоматизации задачи.

Описание предметной области: Учет взаиморасчетов с клиентами.

При проектировании приложения следует придерживаться следующих

принципов; их использование позволяет сделать код более

профессиональным и качественным и обеспечить простое сопровождение,

поддержку и модификацию:

Имя класса должно отображать его роль, назначение.

Разрабатывать класс нужно таким образом, чтобы внесение в него

изменений происходило относительно просто.

Назначение класса должно быть простым.

Код конструктора должен только инициализировать состояние объекта,

в нем не должно быть длительных операций. Необходимо избегать вызовов

других методов в классе (за исключение других конструкторов и методов

final).

Необходимо использовать инкапсуляцию нестатических и

неконстантных полей.

Необходимо избегать слишком длинных методов.

Необходимо избегать длинных списков аргументов (возможно следует

объединять группы аргументов в новый тип).

Если метод используется только другими методами данного класса, то

следует объявлять его как private.

**Текст программы**

import java.util.ArrayList;

import java.util.List;

class Transaction

{

private String description;

private double amount;

public Transaction(String description, double amount)

{

this.description = description;

this.amount = amount;

}

public String getDescription()

{

return description;

}

public double getAmount()

{

return amount;

}

}

class Client {

private String name;

private double balance;

private List<Transaction> transactions;

public Client(String name)

{

this.name = name;

this.balance = 0.0;

this.transactions = new ArrayList<>();

}

public String getName()

{

return name;

}

public double getBalance()

{

return balance;

}

public void addTransaction(Transaction transaction)

{

transactions.add(transaction);

balance += transaction.getAmount();

}

public void removeTransaction(Transaction transaction)

{

transactions.remove(transaction);

balance -= transaction.getAmount();

}

public List<Transaction> getTransactions()

{

return transactions;

}

public double checkBalance()

{

return balance;

}

public void printTransactionHistory()

{

System.out.println("История транзакций для клиента " + name + ":");

for (Transaction transaction : transactions)

{

System.out.println(transaction.getDescription() + ": " + transaction.getAmount());

}

}

}

class AccountingSystem

{

private List<Client> clients;

public AccountingSystem()

{

this.clients = new ArrayList<>();

}

public void addClient(Client client)

{

clients.add(client);

}

public void processTransaction(Client sender, Client receiver, Transaction transaction)

{

sender.addTransaction(new Transaction("Дебет: " + transaction.getDescription(), -transaction.getAmount()));

receiver.addTransaction(new Transaction("Кредит: " + transaction.getDescription(), transaction.getAmount()));

}

public List<Client> getClients()

{

return clients;

}

public void printAllBalances()

{

System.out.println("Балансы всех клиентов:");

for (Client client : clients)

{

System.out.println(client.getName() + ": " + client.getBalance());

}

}

public void removeTransaction(Client client, Transaction transaction)

{

if (client.getTransactions().contains(transaction))

{

client.removeTransaction(transaction);

System.out.println("Транзакция удалена у клиента: " + client.getName() + ".");

} else

{

System.out.println("Транзакция не найдена для клиента: " + client.getName() + ".");

}

}

}

public class Main

{

public static void main(String[] args)

{

Client client1 = new Client("Кудряшов Е.П.");

Client client2 = new Client("Теплова О.С.");

AccountingSystem accountingSystem = new AccountingSystem();

accountingSystem.addClient(client1);

accountingSystem.addClient(client2);

Transaction payment = new Transaction("Оплата за услуги", 100.0);

accountingSystem.processTransaction(client1, client2, payment);

for (Client client : accountingSystem.getClients())

{

System.out.println("Клиент: " + client.getName());

System.out.println("Баланс: " + client.getBalance());

System.out.println("Транзакция:");

for (Transaction transaction : client.getTransactions())

{

System.out.println(transaction.getDescription() + ": " + transaction.getAmount());

}

System.out.println("--------------------------");

}

accountingSystem.printAllBalances();

Transaction transactionToRemove = client1.getTransactions().get(0);

accountingSystem.removeTransaction(client1, transactionToRemove);

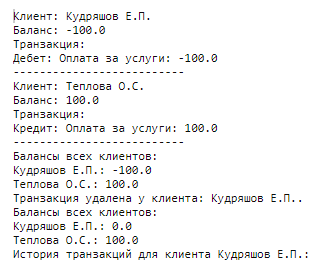
accountingSystem.printAllBalances();

client1.printTransactionHistory();

}

}

**Пример работы программы**



**Вывод**

В процессе выполнения лабораторной работы я освоил навыки проектирования и разработки приложений, сфокусированных на использовании объектов. Я приобрел опыт создания объектов с использованием конструкторов различной сигнатуры и освоил этап объектного проектирования. Также я научился решать задачи автоматизации процессов в предметной области с помощью приложений, функционирующих в рамках объектно-ориентированной парадигмы.

Код лабораторной работы направлен на создание приложения для учета взаиморасчетов с клиентами, следуя принципам объектно-ориентированного программирования (ООП). В программе реализованы несколько классов, обеспечивающих структурированное хранение и управление данными по товарам и их группам.