ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ПЕТРА ВЕЛИКОГО»**

**Институт компьютерных наук и кибербезопасности**

Курсовая работа

по курсу «Основы программирования и алгоритмизации»

**Упрощенная система блокчейн-транзакций**

Авторы работы:

Студенты группы 5130203/40001

Очной формы обучения

Головаха Ольга Александровна

Дворниченко Тимур Иванович

Киричек Мария Романовна

Чеботарев Сергей Александрович

Шутылева Яна Олеговна

Руководитель работы:

Ассистент ин. — “Высшая школа

технологий искусственного интеллекта”

Эспинола Ривера Хольгер Элиас

САНКТ-ПЕТЕРБУРГ 2025

**Постановка задачи**

Цель работы — разработать упрощенную модель системы блокчейн-транзакций, включающую:

* Добавление новых клиентов;
* Возможность создать несколько кошельков у клиентов;
* Иерархия клиентов;
* Создание транзакций;
* Удаление транзакций;

**Описание исходных данных задачи**

* Параметры клиента: ID, имя, тип клиента (Gold, Standard, Platinum), суммарный баланс;
* Параметры кошелька: ID, баланс;
* Параметры транзакции: ID, отправитель, получатель, сумма.

**Термины предметной области**

Блокчейн- система, состоящая из цепочки блоков с данными.

Транзакция- запись о каких-либо изменениях данных в блоках.

Децентрализованная система- система, в которой управление распределено по всей системе и нет иерархии.

Блок- структурная единица данных, в которой содержится информация о транзакциях, своем хеше и хеше предыдущего блока.

Хеш- данные, которые были преобразованы с помощью криптографии и имеют уникальное шифрование.

**Описание методов**

**Пример метода: Wallet::withdraw**

* Название: **withdraw**
* Параметры: **double amoun**t
* Возвращаемое значение: **bool** (успех или неудача)
* Назначение: снять указанную сумму с кошелька
* Описание реализации: если на счете достаточно средств, вычесть сумму и вернуть **true**, иначе — **false**

bool Wallet::withdraw(double amount) {

if (amount > 0 && balance >= amount) {

balance -= amount;

return true;

} else {

cerr << "Error: Cannot withdraw " << amount << ". Current balance: " << balance << endl;

}

return false;

}

**Пример метода: Transaction::getDetailes()**

* Название: getDetailes
* Параметры: нет
* Возвращаемое значение: string (строка с деталями транзакции)
* Назначение: получить форматированную строку с полной информацией о транзакции
* Описание реализации: формирует строку с **ID**, кошельками отправителя/получателя, суммой (2 знака после запятой), типом **TRANSFER** и комиссией (2 знака)

\*/ string Transaction::getDetailes() const {

stringstream ss;

ss << "Transaction ID: " << id << ", From Wallet: " << senderWalletId << ", To Wallet: " << receiveWalletId << ", Amount: " << fixed << setprecision(2) << amount << ", Type: TRANSFER, Commission: " << fixed << setprecision(2) << commission;

return ss.str();}

**Пример метода:TransactionList::getTransaction**

* Название: **getTransaction**
* Параметры: **const string& id** - идентификатор транзакции
* Возвращаемое значение: **Transaction\*** (указатель на найденную транзакцию или **nullptr**)
* Назначение: найти транзакцию по идентификатору в списке
* Описание реализации: последовательно проверяет каждый узел списка, пока не найдет транзакцию с указанным **ID** или не достигнет конца списка

Transaction\* TransactionList::getTransaction(const string& id) const {

TransactionNode\* current = head;

while (current != nullptr) {

if (current->data && current->data->getId() == id) {

return current->data;

}

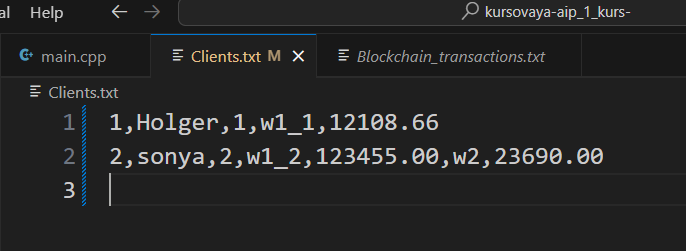
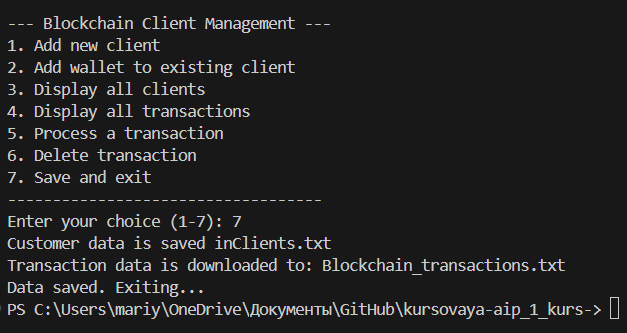
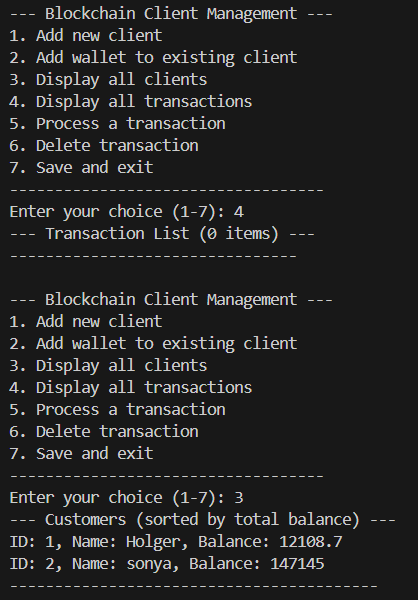
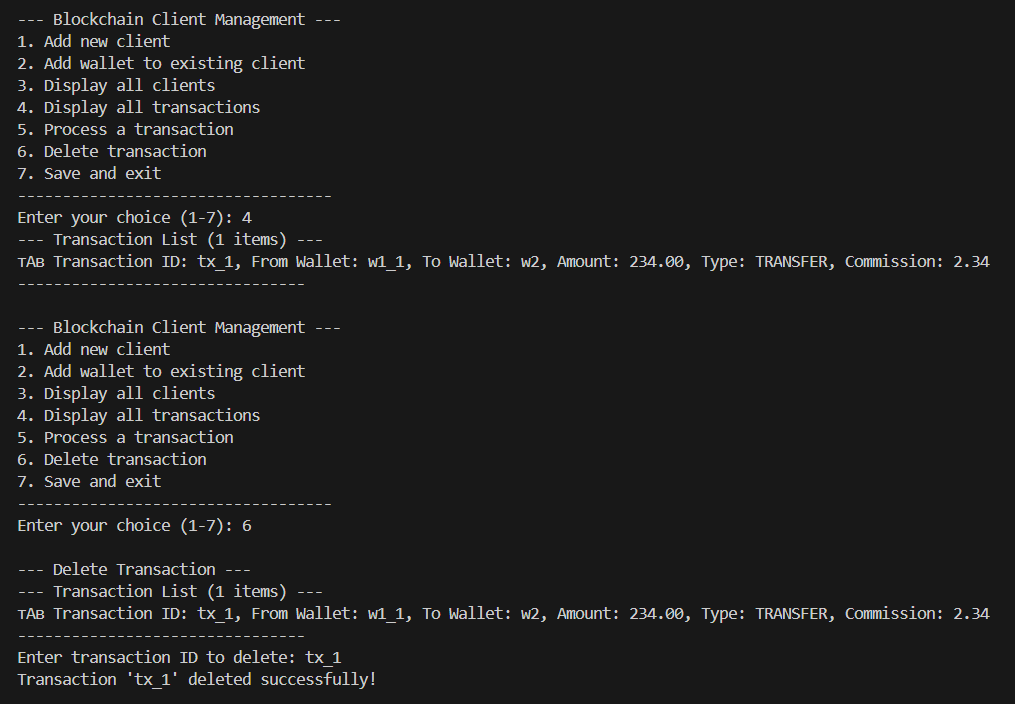
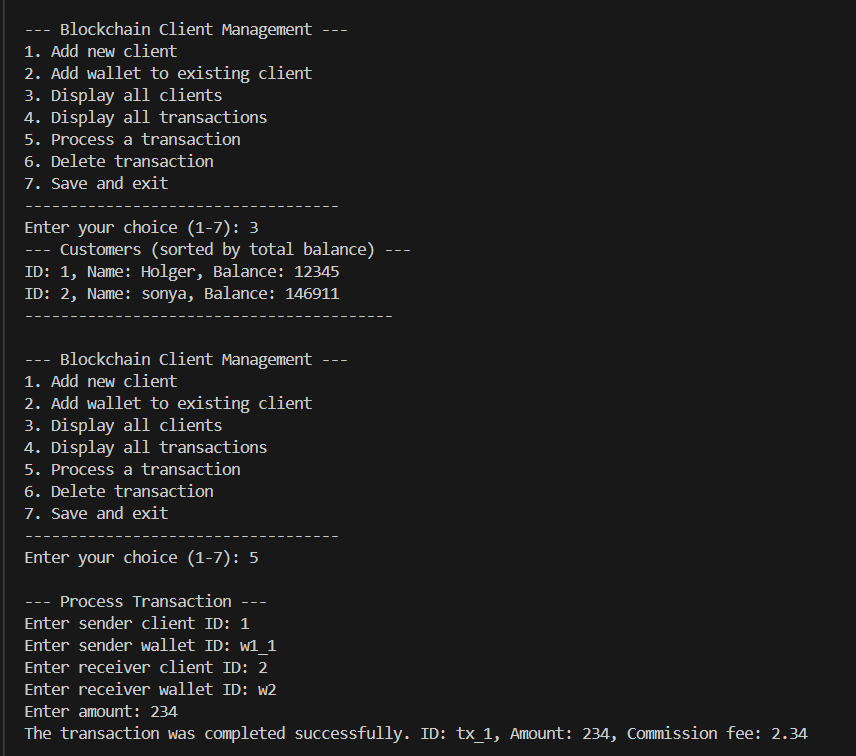
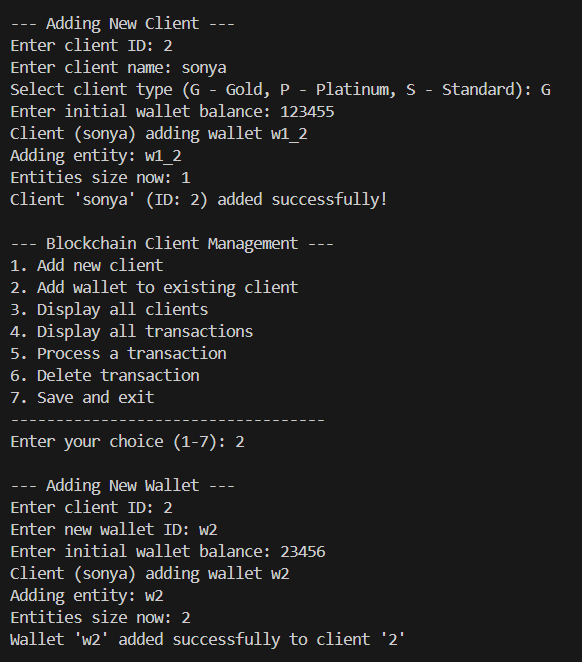
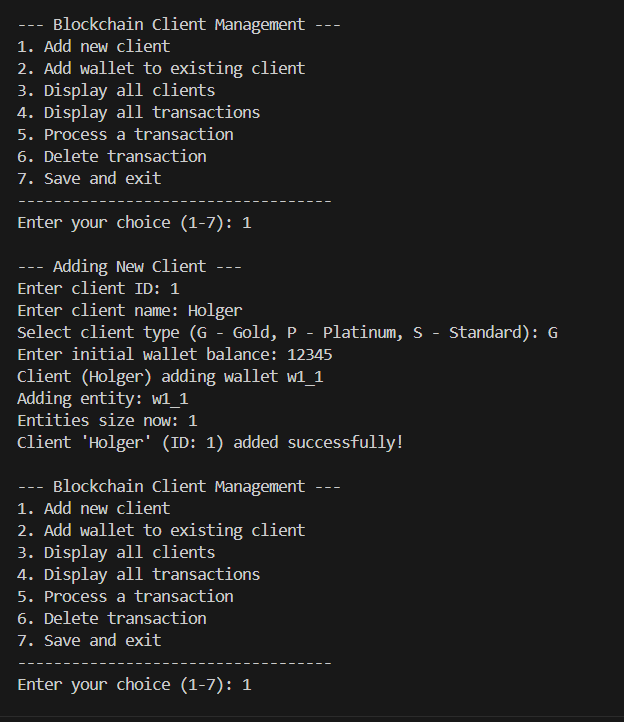
current = current->next;

}

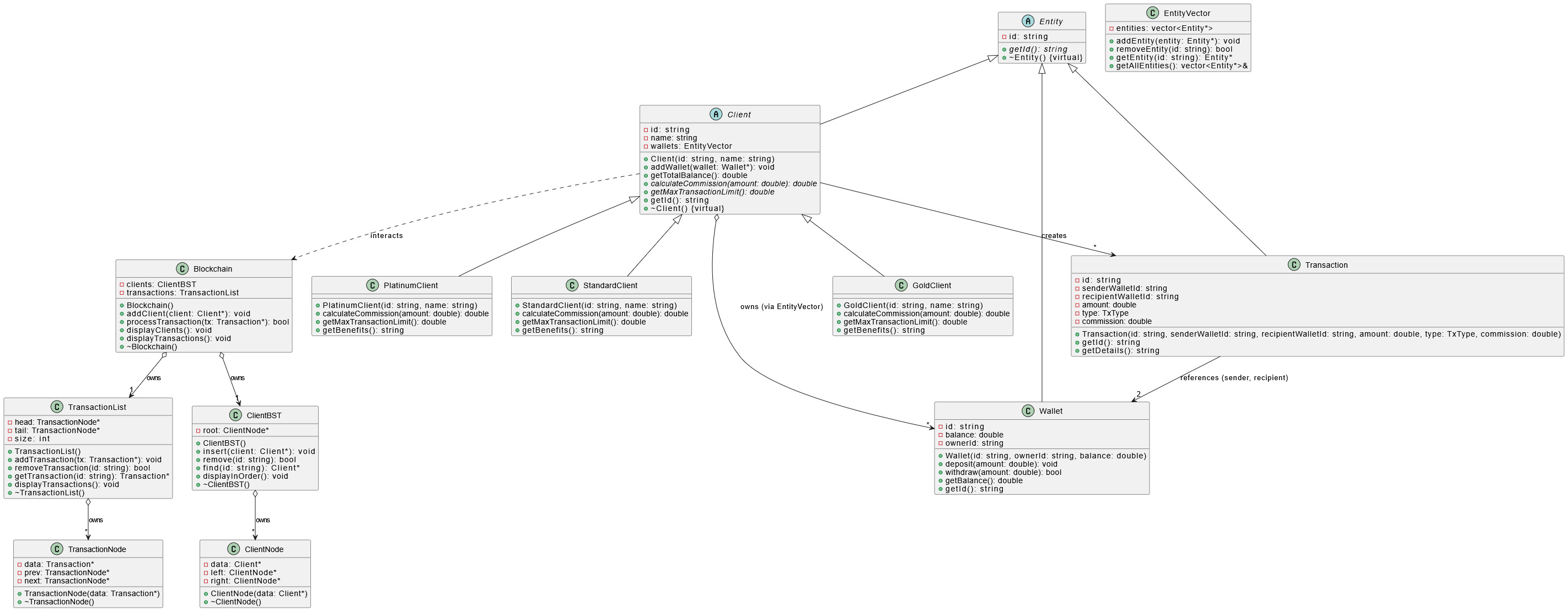
return nullptr;

}

**Описание тестирования разработанного решения**



**Блок-схема:**



**+ методы класса Blockchain:**

void saveClientsToFile(const string& filename) const;

    void loadClientsFromFile(const string& filename);

    void saveTransactionsToFile(const string& filename) const;

    void loadTransactionsFromFile(const string& filename);

    bool removeTransaction(const Transaction& tx);

**Заключение**

**Изучено:**

* Модульное программирование и организация целостной структуры проекта
* Работа с умными указателями
* Наследование классов в **С++**
* Создание виртуальных методов в **С++**

**Освоено:**

* Создание абстрактных классов
* Использование модулей **vector** и **fstream**
* Работа с файлами **С++**

**Реализовано:**

* Запись информации о клиентах и транзакциях в файл
* Дерево клиентов с бинарным поиском по **ID**
* Поддержка трёх категорий клиентов (**Gold**, **Platinum**, **Standart**)
* Пользовательский интерфейс