МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

Лабораторная работа №5

по дисциплине: ООП

тема: «Классы, виды отношений. Наследование.»

Выполнил: студент группы
ПВ-233
Мороз Роман Алексеевич

Проверили:
Морозов Данила Александрович

Белгород 2025

Лабораторная работа 5

Классы, виды отношений. Наследование.

Цель работы: Получение теоретических знаний в области разработки классов, получение практических навыков реализаций классов и отношений между ними.

Задание к работе:

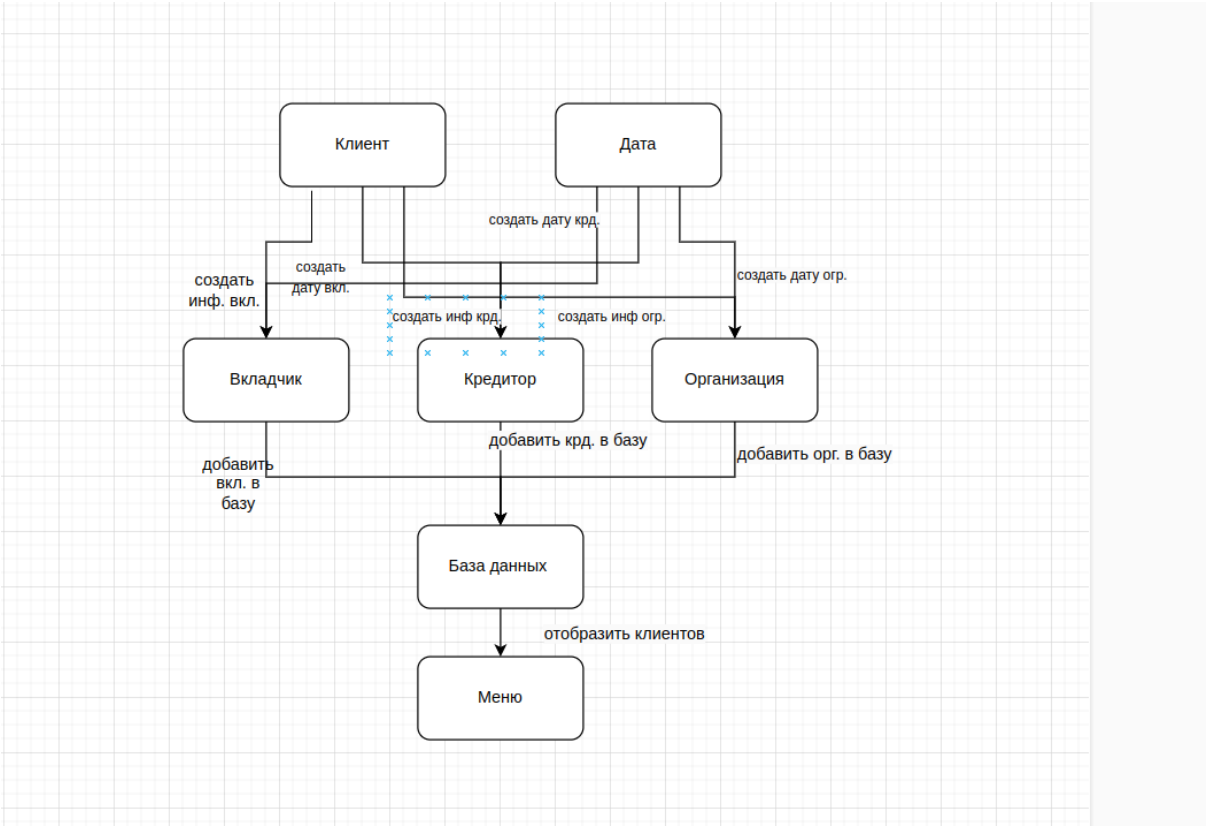
Задание1. В соответствии с вариантом (((номер по списку + 5)%10)+1) выполнить построение объектной модели (использовать не менее 5 объектов) заданной предметной области и разработать диаграмму классов для описанной объектной модели (не менее 7 классов).

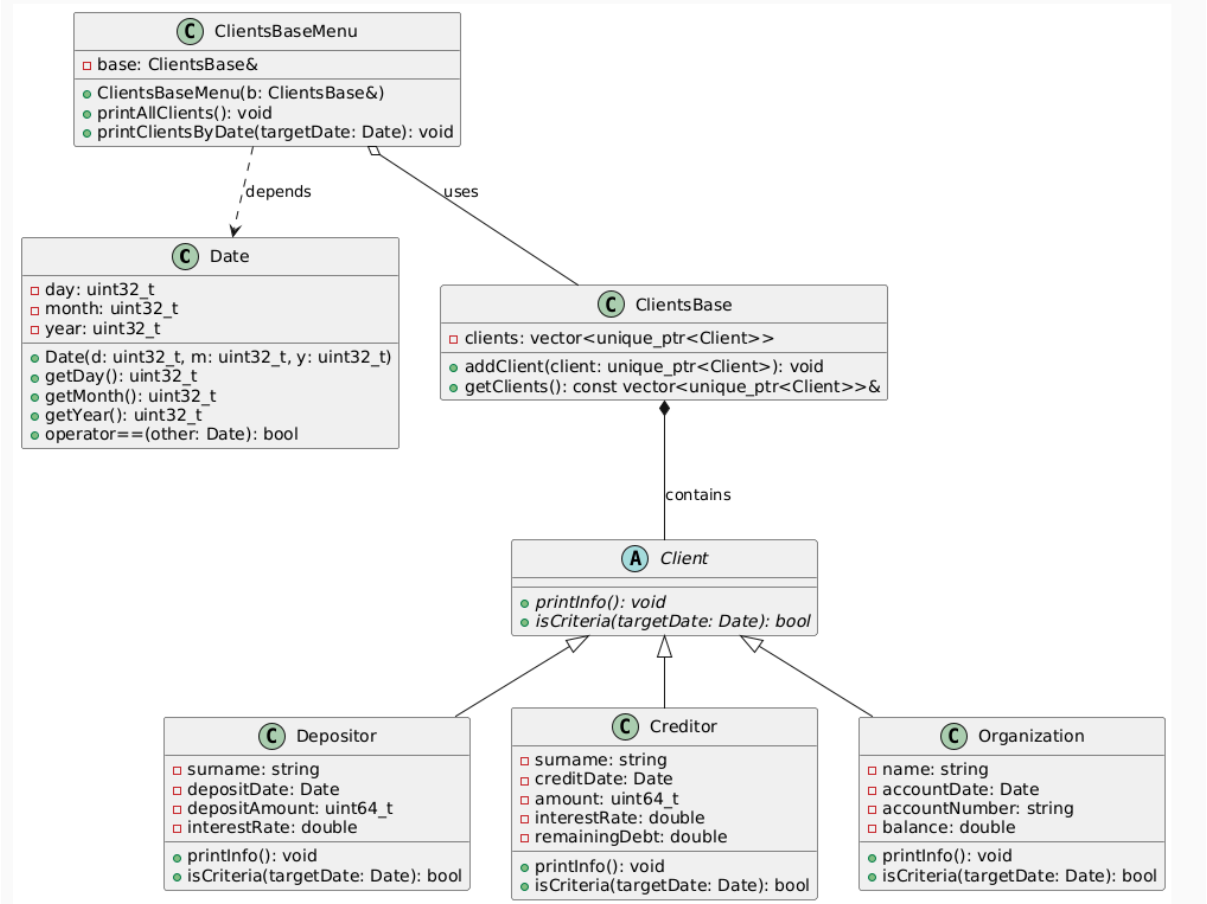
Задание 2. Реализовать некоторые предложенные классы из задания 1, соответствующие постановке задачи задания 2.

Вариант 9

Задание 1

Система кредитования граждан.





Задание 2

1. Создать абстрактный класс Клиент с методами, позволяющими вывести на экран информацию о клиентах банка, а также определить соответствие клиента критерию поиска.
2. Создать производные классы: Вкладчик (фамилия, дата открытия вклада, размер вклада, процент по вкладу), Кредитор (фамилия, дата выдачи кредита, размер кредита, процент по кредиту, остаток долга), Организация (название, дата открытия счёта, номер счёта, сумма на счету) со своими методами вывода информации на экран, и определения соответствия дате (открытия вклада, выдаче кредита, открытия счёта).
3. Создать базу (массив) из n клиентов, вывести полную информацию из базы на экран, а также организовать поиск клиентов, начавших сотрудничать с банком в заданную дату.

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
#include <memory>
```

```
class Date {
private:
   u_int32_t day, month, year;

public:
   Date(u_int32_t d, u_int32_t m, u_int32_t y) : day(d), month(m),
year(y) {}

   inline u_int32_t getDay() const { return day; }
   inline u_int32_t getMonth() const { return month; }
   inline u_int32_t getYear() const { return year; }

   inline bool operator==(const Date& other) const {
       return day == other.day && month == other.month && year ==
other.year;
   }
};

class Client {
public:
   virtual ~Client() = default;
   virtual void printInfo() const = 0;
   virtual bool isCriteria(const Date& targetDate) const = 0;
};

class Depositor: public Client {
private:
   std::string surname;
   Date depositDate;
   u_int64_t depositAmount;
   _Float64 interestRate;

public:
   Depositor(const std::string& n, const Date& d, u_int64_t a, _Float64
r) :
           surname(n), depositDate(d), depositAmount(a),
interestRate(r) {}

   void printInfo() const override {
       std::cout << "Тип: Вкладчик\nФамилия: " << surname
       << "\nДата открытия: " << std::setfill('0')
       << std::setw(2) << depositDate.getDay() << "."
```

```
        << std::setw(2) << depositDate.getMonth() << "."
        << depositDate.getYear() << "\nСумма: " << depositAmount
        << "\nПроцентная ставка: " << interestRate << "%\n\n";
    }

    inline bool isCriteria(const Date& targetDate) const override {
return depositDate == targetDate; }
};

class Creditor : public Client {
private:
    std::string surname;
    Date creditDate;
    u_int64_t amount;
    _Float64 interestRate;
    _Float64 remainingDebt;

public:
    Creditor(const std::string& s, const Date& d, u_int64_t a, _Float64
ir, _Float64 rd) :
        surname(s), creditDate(d), amount(a), interestRate(ir),
remainingDebt(rd) {}

    void printInfo() const override {
        std::cout << "Тип: Кредитор\nФамилия: " << surname
            << "\nДата выдачи: " << std::setfill('0')
            << std::setw(2) << creditDate.getDay() << "."
            << std::setw(2) << creditDate.getMonth() << "."
            << creditDate.getYear() << "\nСумма: " << amount
            << "\nПроцентная ставка: " << interestRate << "%"
            << "\nОстаток долга: " << remainingDebt << "\n\n";
    }

    inline bool isCriteria(const Date& targetDate) const override {
return creditDate == targetDate; }
};

class Organization : public Client {
    std::string name;
    Date accountDate;
    std::string accountNumber;
    _Float64 balance;
```

```
public:
    Organization(std::string n, Date d, std::string an, double b)
        : name(n), accountDate(d), accountNumber(an), balance(b) {}

    void printInfo() const override {
        std::cout << "Тип: Организация\nНазвание: " << name
             << "\nДата открытия: " << std::setfill('0')
             << std::setw(2) << accountDate.getDay() << "."
             << std::setw(2) << accountDate.getMonth() << "."
             << accountDate.getYear() << "\nНомер счёта: " <<
accountNumber
             << "\nБаланс: " << balance << "\n\n";
    }

    inline bool isCriteria(const Date& targetDate) const override {
return accountDate == targetDate; }
};

class ClientsBase {
private:
    std::vector<std::unique_ptr<Client>> clients;

public:
    ClientsBase() = default;

    inline void addClient(std::unique_ptr<Client> client) {
clients.push_back(std::move(client)); }

    inline const std::vector<std::unique_ptr<Client>>& getClients()
const {  return clients;  }
};

class ClientsBaseMenu {
private:
    const ClientsBase& base;

public:
    explicit ClientsBaseMenu(const ClientsBase& b) : base(b) {}

    void printAllClients() const {
        std::cout << "Все клиенты:\n";
        for (const auto& client : base.getClients()) {
            client->printInfo();
```

```
        }
    }

    void printClientsByDate(const Date& targetDate) const {
        std::cout << "\nКлиенты с датой "
                  << std::setfill('0') << std::setw(2) <<
targetDate.getDay() << "."
                  << std::setw(2) << targetDate.getMonth() << "."
                  << targetDate.getYear() << ":\n";

        for (const auto& client : base.getClients()) {
            if (client->isCriteria(targetDate)) {
                client->printInfo();
            }
        }
    }
};

int main() {
    ClientsBase base;

    base.addClient(std::make_unique<Depositor>("Иваныч", Date{15, 5,
2025}, 510, 5.5));
    base.addClient(std::make_unique<Creditor>("Петрова", Date{15, 10,
2023}, 4000000, 7.0, 50000));
    base.addClient(std::make_unique<Depositor>("Сидорова", Date{10, 8,
2024}, 3500000, 6.2));
    base.addClient(std::make_unique<Creditor>("Левина", Date{10, 7,
2022}, 3000000, 6.8, 100000));
    base.addClient(std::make_unique<Creditor>("Мороз", Date{20, 6,
2021}, 1000000, 7.2, 250000));
    base.addClient(std::make_unique<Organization>("ООО 'БНАЛ'", Date{10,
1, 2022}, "40702810500000000001", 2500000));
    base.addClient(std::make_unique<Depositor>("Мавроди", Date{15, 5,
2020}, 750000, 6.0));

    ClientsBaseMenu menu(base);
    menu.printAllClients();

    Date targetDate(15, 5, 2025);
    menu.printClientsByDate(targetDate);

    return 0;
```

```
}
```

```
❯ ./a.out
Все клиенты:
Тип: Вкладчик
Фамилия: Иваныч
Дата открытия: 15.05.2025
Сумма: 510
Процентная ставка: 5.5%

Тип: Кредитор
Фамилия: Петрова
Дата выдачи: 15.10.2023
Сумма: 4000000
Процентная ставка: 7%
Остаток долга: 50000

Тип: Вкладчик
Фамилия: Сидорова
Дата открытия: 10.08.2024
Сумма: 3500000
Процентная ставка: 6.2%

Тип: Кредитор
Фамилия: Левина
Дата выдачи: 10.07.2022
Сумма: 3000000
Процентная ставка: 6.8%
Остаток долга: 100000

Тип: Кредитор
Фамилия: Мороз
Дата выдачи: 20.06.2021
Сумма: 1000000
Процентная ставка: 7.2%
Остаток долга: 250000

Тип: Организация
Название: ООО 'БНАЛ'
Дата открытия: 10.01.2022
Номер счёта: 40702810500000000001
Баланс: 2.5e+06

Тип: Вкладчик
Фамилия: Мавроди
Дата открытия: 15.05.2020
Сумма: 750000
Процентная ставка: 6%


Клиенты с датой сотрудничества 15.05.2020:
Тип: Вкладчик
Фамилия: Мавроди
Дата открытия: 15.05.2020
Сумма: 750000
Процентная ставка: 6%
```

Вывод: Получили теоретические знания в области разработки классов, получили практические навыки реализации классов и отношений между ними.