Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА**

О Т Ч Е Т

по лабораторной работе № 3

Название: Классы

Дисциплина: Разработка приложений на языке C#

Студент ИУ6-75Б _____ И.Ю. Жосан
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ А.М. Минитаева
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Цель работы: изучить основы работы с классами в языке C# на практической задаче.

Задание: Создать консольное приложение. Программа представляет собой автоматизированную систему учета банковских сведений.

На каждого клиента банка хранятся следующие сведения:

— Ф.И.О.;

— Возраст;

— Место работы;

— Номера счетов.

На каждом счете хранится информация о текущем балансе и история прихода, расхода. Для каждого клиента может быть создано неограниченное количество счетов. С каждым счетом можно производить следующие действия: открытие, закрытие, вклад денег, снятие денег, просмотр баланса, просмотр истории.

Вся информация должна хранится в массивах. Рекомендуется объекты клиента и счета реализовать в виде классов. Баланс счета организовать в виде свойства только для чтения.

Выполнение задания:

Алгоритм основной программы предусматривает создание новых клиентов и поиск в базе уже существующих. Клиент может создавать новые счета, закрывать и открывать счета, пополнять баланс и снимать деньги, просматривать баланс и историю.

Код разработанной программы:

```
using System;
using System.Linq;

namespace lab3_classes
{
    class Client
    {
        private string first_name, last_name, father_name, job;
        private int year_of_birth;
        private int[] bank_accounts;
        public Client(string firstName, string lastName, string
                          fatherName, int yearOfBirth, string job_)
        {
```

```
            first_name = firstName;
            last_name = lastName;
            father_name = fatherName;
            year_of_birth = yearOfBirth;
            job = job_;
            bank_accounts = new int[0];
        }
        public override string ToString()
        {
            return first_name + ' ' + last_name + ' ' + father_name +
                        ' ' + year_of_birth.ToString() + ' ' + job;
        }
        public void addAccount(int num)
        {
            Array.Resize(ref bank_accounts, bank_accounts.Length + 1);
            bank_accounts[bank_accounts.Length - 1] = num;
        }
        public int[] GetAccounts() { return bank_accounts; }
    }
    class Transaction
    {
        private double sum;
        private string date;
        private bool direction;
        public Transaction(double s, string d, bool dir)
        {
            sum = s;
            date = d;
            direction = dir;    // 1 - in   0 - out
        }
        public override string ToString()
        {
            return direction ?  date + "     IN:     " +
                          sum.ToString() :  date + "    OUT:     " +
                          sum.ToString();
        }
    }
    class Bank_Account
    {
        private static int NumbersForAccounts = 0;
        private int number_of_account;
        private double balance;
        private bool state;
        private Transaction[] history_in, history_out, history;
        public Bank_Account()
        {
            number_of_account = NumbersForAccounts;
            NumbersForAccounts += 1;
            balance = 0.0;
            state = true;
            history_in = new Transaction[0];
            history_out = new Transaction[0];
            history = new Transaction[0];
        }
```

```
    public bool GetState() { return state; }
    public void OpenAccount()
    {
        state = true;
        Console.WriteLine($"### ACCOUNT {number_of_account} WAS
                    OPENED ###");
    }
    public void CloseAccount()
    {
        state = false;
        Console.WriteLine($"### ACCOUNT {number_of_account} WAS
                    CLOSED ###");
    }
    public int GetNumber() { return number_of_account; }
    public void in_money(double sum)
    {
        Transaction t = new Transaction(sum, "18.03.2022", true);
        Array.Resize(ref history_in, history_in.Length + 1);
        history_in[history_in.Length - 1] = t;
        Array.Resize(ref history, history.Length + 1);
        history[history.Length - 1] = t;
        balance += sum;
        Console.WriteLine($"### TRANSACTION IN: {sum} ###");
    }
    public void out_money(double sum)
    {
        Transaction t = new Transaction(sum, "18.03.2022", false);
        Array.Resize(ref history_out, history_out.Length + 1);
        history_out[history_out.Length - 1] = t;
        Array.Resize(ref history, history.Length + 1);
        history[history.Length - 1] = t;
        balance -= sum;
        Console.WriteLine($"### TRANSACTION OUT: {sum} ###");
    }
    public double getBalance() { return balance; }
    public string getHistory()
    {
        string result = "";
        for (int i = 0; i < history.Length; i++)
            result += history[i].ToString() + '\n';
        return result;
    }
}
class Program
{
    static Bank_Account[] accounts = new Bank_Account[0];
    static Client[] clients = new Client[0];
    static void Main(string[] args)
    {
        string readLine = "start";
        while (readLine != "end")
        {
            Console.Write("Enter 1 to create new client OR 2 if
                    client exists: ");
```

```
string code = Console.ReadLine();
while ((code != "1") && (code != "2"))
{
    Console.Write("ERROR: Wrong code. Enter 1 to
         create new client OR 2 if client exists: ");
    code = Console.ReadLine();
}
Console.Write("Enter first name: ");
string first_name = Console.ReadLine();
Console.Write("Enter last name: ");
string last_name = Console.ReadLine();
Console.Write("Enter father name: ");
string father_name = Console.ReadLine();
Console.Write("Enter year of birth: ");
int year = 0;
while (!(int.TryParse(Console.ReadLine(), out year)))
    Console.WriteLine("ERROR: Wrong number. Enter year
         of birth again:");
Console.Write("Enter job: ");
string job = Console.ReadLine();

switch (code)
{
    case "1": {
            Client cl = new Client(first_name,
          last_name, father_name, year, job);
            Array.Resize(ref clients, clients.Length +
          1);
            clients[clients.Length - 1] = cl;
            Console.WriteLine("### CLIENT HAS CREATED
          ###");
            work_with_client(clients.Length - 1);
            break;
        }
    case "2":
        {
            int i;
            for (i = 0; i < clients.Length; i++)
            {
                if (clients[i].ToString() ==
          (first_name + ' ' + last_name + ' ' +
          father_name + ' ' + year.ToString() + job))
                    break;
            }
            if (i == clients.Length)
                Console.WriteLine("### CLIENT DIDN'T
          EXIST ###");
            else
            {
                Console.WriteLine("### CLIENT HAS
          FOUNDED ###");
                work_with_client(i);
            }
            break;
```

```
                }
            }
            Console.WriteLine("\nEnter 'end' to finish OR
                    something else to change client.");
            readLine = Console.ReadLine();
        }
    }
    static void work_with_client(int num)
    {
        string readLine = "start";
        while (readLine != "end")
        {
            int[] accountsArray = clients[num].GetAccounts();
            int numOfAccs = accountsArray.Length;
            string menuWithAccounts = $"Client has {numOfAccs}
                    accounts. Enter:\n1 - to create new account";
            int max_i = 0, code = 0;
            for (max_i = 0; max_i < numOfAccs; max_i++)
            {
                menuWithAccounts += $"\n{max_i + 2} - to work with
                      account {accountsArray[max_i]}";
            }
            Console.WriteLine(menuWithAccounts);
            while (true)
            {
                while (!(int.TryParse(Console.ReadLine(), out
                      code)))
                    Console.WriteLine("### ERROR: WRONG NUMBER
                      ###");
                if ((code < 0) || (code >= max_i + 2))
                    Console.WriteLine("### ERROR: WRONG CODE
                      ###\n" + menuWithAccounts);
                else
                    break;
            }
            int accNum = 0;
            if (code == 1)
            {
                Bank_Account ba = new Bank_Account();
                Array.Resize(ref accounts, accounts.Length + 1);
                accounts[accounts.Length - 1] = ba;
                clients[num].addAccount(ba.GetNumber());
                Console.WriteLine("### BANK ACCOUNT HAS CREATED
                      ###");
                accNum = ba.GetNumber();
            }
            else
                accNum = accountsArray[code - 2];

            Console.WriteLine("Enter 'end' to change account OR
                    something else to continue.");
            while (Console.ReadLine() != "end")
            {
```

```
string menuWithFunctions = "Enter: \n1 - to open
     bank account\n2 - to close bank account\n3 -
     to put in\n4 - to put out\n5 - to see
     balance\n6 - to see history";
Console.WriteLine(menuWithFunctions);
string str = Console.ReadLine();
switch(str)
{
    case "1": accounts[accNum].OpenAccount();
     break;
    case "2": accounts[accNum].CloseAccount();
     break;
    case "3":
        {
            if (accounts[accNum].GetState())
            {
                Console.WriteLine("Enter sum to
     input: ");
                double sum;
                while (true)
                {
                    while
     (!(double.TryParse(Console.ReadLine(), out
     sum)))
                        Console.WriteLine("###
     ERROR: WRONG NUMBER ###\nEnter sum to input
     again: ");
                    if (sum < 0)
                        Console.WriteLine("###
     ERROR: WRONG SUM ###\nEnter sum to input
     again: ");
                    else
                        break;
                }
                accounts[accNum].in_money(sum);
            }
            else
                Console.WriteLine("### BANK
     ACCOUNT IS CLOSED ###");
            break;
        }
    case "4":
        {
            if (accounts[accNum].GetState())
            {
                Console.WriteLine("Enter sum to
     output: ");
                double sum;
                while (true)
                {
                    while
     (!(double.TryParse(Console.ReadLine(), out
     sum)))
```

```
                                    Console.WriteLine("###
                        ERROR: WRONG NUMBER ###\nEnter sum to output
                        again: ");
                                    if ((sum > 0) && (sum <
                        accounts[accNum].getBalance()))
                                        break;
                                    if (sum < 0)
                                        Console.WriteLine("###
                        ERROR: WRONG SUM ###\nEnter sum to output
                        again: ");
                                    if (sum >
                        accounts[accNum].getBalance())
                                        Console.WriteLine("###
                        ERROR: NOT ENOUGH MONEY ON BALANCE ###\nEnter
                        sum to output again: ");
                                }
                                accounts[accNum].out_money(sum);
                            }
                            else
                                Console.WriteLine("### BANK
                        ACCOUNT IS CLOSED ###");
                            break;
                        }
                    case "5": Console.WriteLine($"### BALANCE:
                      {accounts[accNum].getBalance()} ###"); break;
                    case "6": Console.WriteLine($"### HISTORY:
                      ###\n{accounts[accNum].getHistory()}");
                      break;
                }
                Console.WriteLine("Enter 'end' to change account
                      OR something else to continue.");
            }
            Console.WriteLine("\nEnter 'end' to change client OR
                      something else to continue.");
            readLine = Console.ReadLine();
        }
    }
}
}
```

Рисунки, отображающие работу программы, приведены ниже:

— рисунок 1 – создание нового клиента;

— рисунок 2 – создание банковского счета;

— рисунок 3 – закрытие счета;

— рисунок 4 – открытие счета;

— рисунок 5 – пополнение счета;

— рисунок 6 – снятие денег со счета с попыткой снять больше, чем есть на балансе;

— рисунок 7 – проверка баланса;

— рисунок 8 – просмотр истории;

— рисунок 9 – выбор счета клиента;

```
Enter 1 to create new client OR 2 if client exists: 1
Enter first name: Paliy
Enter last name: Julia
Enter father name: Yaroslavovna
Enter year of birth: 2001
Enter job: programmer
### CLIENT HAS CREATED ###
```

Рисунок 1 – Создание нового клиента

```
Client has 0 accounts. Enter:
1 - to create new account
1
### BANK ACCOUNT HAS CREATED ###
```

Рисунок 2 – Создание банковского счета

```
Enter:
1 - to open bank account
2 - to close bank account
3 - to put in
4 - to put out
5 - to see balance
6 - to see history
2
### ACCOUNT 0 WAS CLOSED ###
```

Рисунок 3 – Закрытие счета

```
Enter:
1 - to open bank account
2 - to close bank account
3 - to put in
4 - to put out
5 - to see balance
6 - to see history
1
### ACCOUNT 0 WAS OPENED ###
```

Рисунок 4 – Открытие счета

```
Enter:
1 - to open bank account
2 - to close bank account
3 - to put in
4 - to put out
5 - to see balance
6 - to see history
3
Enter sum to input:
126
### TRANSACTION IN: 126 ###
```

Рисунок 5 – Пополнение счета

```
Enter:
1 - to open bank account
2 - to close bank account
3 - to put in
4 - to put out
5 - to see balance
6 - to see history
4
Enter sum to output:
567
### ERROR: NOT ENOUGH MONEY ON BALANCE ###
Enter sum to output again:
100
### TRANSACTION OUT: 100 ###
```

Рисунок 6 – Снятие денег со счета с попыткой снять больше, чем есть на балансе

```
Enter:
1 - to open bank account
2 - to close bank account
3 - to put in
4 - to put out
5 - to see balance
6 - to see history
5
### BALANCE: 26 ###
```

Рисунок 7 – Проверка баланса

```
Enter:
1 - to open bank account
2 - to close bank account
3 - to put in
4 - to put out
5 - to see balance
6 - to see history
6
### HISTORY: ###
18.03.2022     IN:     126
18.03.2022    OUT:     100
18.03.2022     IN:     250
```

Рисунок 8 – Просмотр истории

```
Client has 3 accounts. Enter:
1 - to create new account
2 - to work with account 0
3 - to work with account 1
4 - to work with account 2
```

Рисунок 9 – Выбор счета клиента

Вывод: в процессе выполнения лабораторной работы были изучены основы классов в языке программирование C# (поля и методы разной доступности public/private). В результате практической работы была получена консольная программа, выполняющие функции банковской системы: создание клиентов, создание банковских счетов, выполнение операций со счетами.