МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

КАФЕДРА САПР

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Объектно-ориентированное программирование»

Тема: Использование методов

Студенты гр. 9301 _____ **Власов Е.А.**

_____ **Токарев С.В.**

_____ **Служевская А.С.**

Преподаватель _____ **Новакова Н.Е.**

Санкт-Петербург
2021

Цель работы

Научиться работать с классами и с их методами.

Ход работы

Упражнение 1

1.1 Для того, чтобы создать экземпляр класса в методе NewBankAccount, необходимо воспользоваться оператором new, который динамически выделяет память для объекта.

1.2 Модификатор доступа private доступен только из кода в самом в классе. Член класса public доступен в любом месте кода.

1.3 Метод Populate позволяет членам класса private присвоить значение. Методы Number, Balance и Type позволяют вывести значения private полей.

1.4 В методе Write класса CreateAccount экземпляр класса BankAccount должен вывести значения полей, обратившись к методам класса BankAccount.

Упражнение 2

2.1 nextAccNo – это статистический член класса. Он хранит состояние класса в целом, поэтому для обращения к нему создавать экземпляр класса не нужно.

2.2 Метод NextNumber возвращает значение поля nextAccNo, увеличенное на 1. Обращаться к методу нужно через имя класса, так как метод статистический.

Упражнение 3

3.1 Метод Deposit позволяет пополнить баланс. Метод TestDeposit позволяет пользователю ввести сумму для пополнения.

3.2 TestWithdraw позволяет ввести пользователю сумму для вывода денег со счета. Метод Withdraw снимает сумму со счета. При этом тип возвращаемого значения bool. Метод возвращает true, если со счета можно снять, или false, если нельзя.

Текст программы

Упражнение 1

```
using System;

class CreateAccount
{
    enum AccountType
    {
        Checking,
        Deposit
    }

    class BankAccount
    {
        private long accNo;
        private decimal accBal;
        private AccountType accType;

        public void Populate(long number, decimal balance)
        {
            number = accNo;
            balance = accBal;
            accType = AccountType.Checking;
        }

        public long Number()
        {
            return accNo;
        }

        public decimal Balance()
        {
            return accBal;
        }

        public AccountType Type()
        {
            return accType;
        }
    }

    static void Main()
    {
        BankAccount berts = NewBankAccount();
        Write(berts);

        BankAccount freds = NewBankAccount();
        Write(freds);
    }

    static BankAccount NewBankAccount()
    {
        BankAccount created = new BankAccount();

        Console.Write("Enter the account number   : ");
        long number = long.Parse(Console.ReadLine());

        Console.Write("Enter the account balance! : ");
        decimal balance = decimal.Parse(Console.ReadLine());

        created.Populate(number, balance);

       /* created.accNo = number;
        created.accBal = balance;
        created.accType = AccountType.Checking;
       */
        return created;
```

```
    }

    static void Write(BankAccount toWrite)
    {
        Console.WriteLine("Account number is {0}", toWrite.Number());
        Console.WriteLine("Account balance is {0}", toWrite.Balance());
        Console.WriteLine("Account type is {0}", toWrite.Type());
    }
      }
```

Упражнение 2

```
using System;

class CreateAccount
{
    enum AccountType
    {
        Checking,
        Deposit
    }

    class BankAccount
    {
        private long accNo;
        private static long nextAccNo = 123;
        private decimal accBal;
        private AccountType accType;

        public void Populate(decimal balance)
        {
            accNo = NextNumber();
            accBal = balance;
            accType = AccountType.Checking;
        }

        public long Number()
        {
            return accNo;
        }

        private static long NextNumber()
        {
            return nextAccNo++;
        }

        public decimal Balance()
        {
            return accBal;
        }

        public AccountType Type()
        {
            return accType;
        }
    }

    static void Main()
    {
        BankAccount berts = NewBankAccount();
        Write(berts);

        BankAccount freds = NewBankAccount();
        Write(freds);
    }

    static BankAccount NewBankAccount()
    {
        BankAccount created = new BankAccount();
```

```
        //Console.Write("Enter the account number   : ");
        //long number = long.Parse(Console.ReadLine());

        //long number = BankAccount.NextNumber();

        Console.Write("Enter the account balance! : ");
        decimal balance = decimal.Parse(Console.ReadLine());

        created.Populate(balance);

       /* created.accNo = number;
        created.accBal = balance;
        created.accType = AccountType.Checking;
       */
        return created;
    }

    static void Write(BankAccount toWrite)
    {
        Console.WriteLine("Account number is {0}", toWrite.Number());
        Console.WriteLine("Account balance is {0}", toWrite.Balance());
        Console.WriteLine("Account type is {0}", toWrite.Type());
    }
  }
```

Упражнение 3

```
using System;

enum AccountType
{
    Checking,
    Deposit
}

class BankAccount
{
    private long accNo;
    private static long nextAccNo = 123;
    private decimal accBal;
    private AccountType accType;

    public void Populate(decimal balance)
    {
        accNo = NextNumber();
        accBal = balance;
        accType = AccountType.Checking;
    }

    public long Number()
    {
        return accNo;
    }

    private static long NextNumber()
    {
        return nextAccNo++;
    }

    public decimal Balance()
    {
        return accBal;
    }

    public AccountType Type()
    {
        return accType;
    }
```

```
    public decimal Deposit(decimal amount)
    {
        accBal += amount;
        return accBal;
    }

    public bool Withdraw(decimal amount)
    {
        bool sufficientFunds = accBal >= amount;
        if (sufficientFunds)
        {
            accBal -= amount;
        }
        return sufficientFunds;
    }
}

class CreateAccount
{
    static void Main()
    {
        BankAccount berts = NewBankAccount();
        Write(berts);
        TestDeposit(berts);
        Write(berts);
        TestWithdraw(berts);
        BankAccount freds = NewBankAccount();
        TestDeposit(freds);
        Write(freds);
        TestWithdraw(freds);
    }

    public static void TestDeposit(BankAccount acc)
    {
        Console.WriteLine("Enter amount to deposit: ");
        decimal amount = decimal.Parse(Console.ReadLine());
        acc.Deposit(amount);
    }

    public static void TestWithdraw(BankAccount acc2)
    {
        Console.WriteLine("Enter amount to withdraw:");
        decimal amount = decimal.Parse(Console.ReadLine());
        if (acc2.Withdraw(amount) == true)
        {
            Console.WriteLine("successful");
            Console.WriteLine(acc2.Balance());
        }
        else
        {
            Console.WriteLine("unsuccessful");
            Console.WriteLine(acc2.Balance());
        }
    }

    static BankAccount NewBankAccount()
    {
        BankAccount created = new BankAccount();

        //Console.Write("Enter the account number   : ");
        //long number = long.Parse(Console.ReadLine());

        //long number = BankAccount.NextNumber();

        Console.Write("Enter the account balance! : ");
        decimal balance = decimal.Parse(Console.ReadLine());

        created.Populate(balance);
```

```
        /* created.accNo = number;
         created.accBal = balance;
         created.accType = AccountType.Checking;
        */
         return created;
    }

    static void Write(BankAccount toWrite)
    {
        Console.WriteLine("Account number is {0}", toWrite.Number());
        Console.WriteLine("Account balance is {0}", toWrite.Balance());
        Console.WriteLine("Account type is {0}", toWrite.Type());
    }
}
```

Примеры работы программы

На Рис. 1 представлена работа программы упражнения 1.

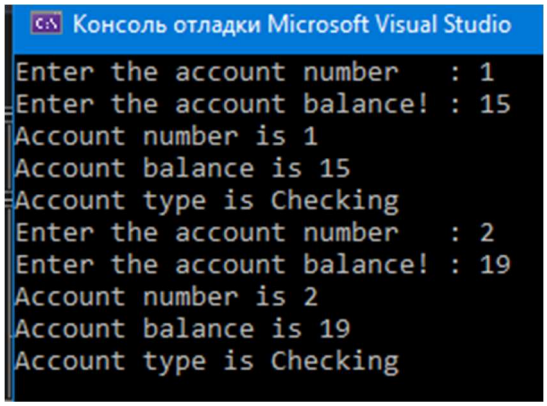

Рис. 1

На Рис. 2 представлена работа программы упражнения 2.

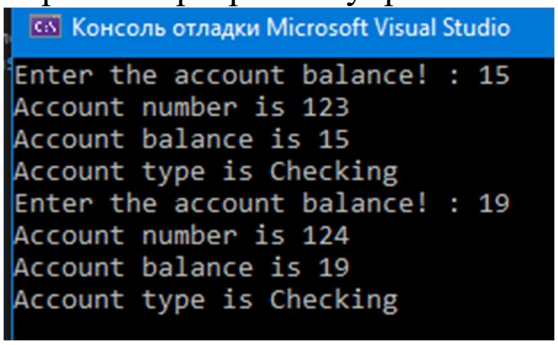

Рис. 2

На Рис. 3 представлена работа программы упражнения 3.

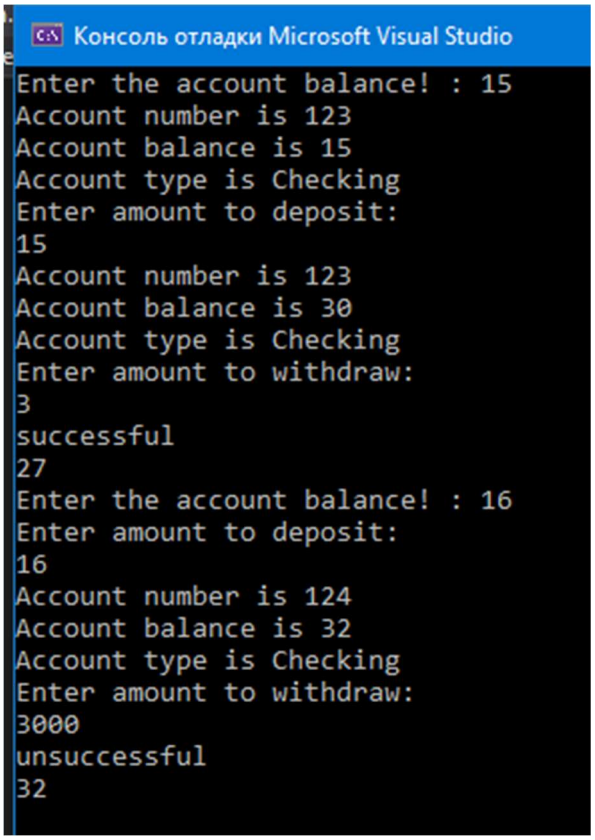

Рис. 3

Вывод

В ходе работы были рассмотрены классы, экземпляры классов, поля и методы, ключевое слово static, модификаторы доступа private и public.