

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ
по лабораторной работе №8
по дисциплине «Объектно-ориентированное
программирование»
Тема: «Применение конструкторов»

Студенты гр. 1302

_____ Романова О.В.

_____ Новиков Г.В.

Преподаватель:

_____ Васильев А.А.

Санкт-Петербург

2023

1. Цель работы

Изучение конструкторов в языке C# с помощью программного продукта компании Microsoft VS 2022.

2. Анализ задачи

Необходимо:

- 1) Написать программу, которая изменяет программу, написанную в прошлой лабораторной, нужно добавить конструктор по умолчанию и еще три конструктора с различной комбинацией параметров;
- 2) Написать программу, которая добавляет к прошлой программе новый класс BankDeposit;
- 3) Написать программу, которая добавляет в предыдущую программу деструктор.

3. Ход выполнения работы

3.1 Упражнение 1

В ходе выполнения данного упражнения написана программа, которая преобразовывает программу из прошлой лабораторной работы, добавляя в нее четыре различных конструктора (один по умолчанию, три с различной комбинацией параметров).

3.1.1 Пошаговое описание алгоритма

Банковским аккаунтам присваиваются значения с помощью конструкторов.

С помощью методов Deposit, Withdraw изменяется баланс аккаунтов.

С помощью метода Write выводятся все значения аккаунтов.

3.1.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- `System.Console.WriteLine()` – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

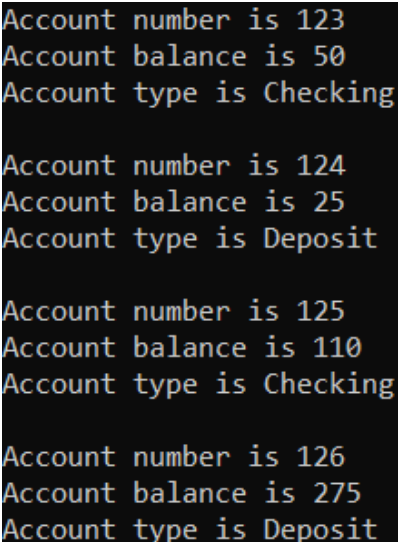
- `System.Console.ReadKey()` – ожидает следующего нажатия клавиши пользователем;

- `Main()` – служит для запуска программы.

- `BankAccount` – класс, который описывает банковский аккаунт с помощью методов: `Number` (возвращает номер аккаунта), `Balance` (возвращает текущий баланс), `Type` (возвращает тип аккаунта), `Withdraw` (позволяет произвести снятие денег) и `Deposit` (позволяет пополнить счет), `Transaction` (возвращает информацию обо всех транзакциях, произведенных с данного аккаунта).

3.1.3 Контрольный пример

На рис.3.1 представлены результаты выполнения программы.



```
Account number is 123
Account balance is 50
Account type is Checking

Account number is 124
Account balance is 25
Account type is Deposit

Account number is 125
Account balance is 110
Account type is Checking

Account number is 126
Account balance is 275
Account type is Deposit
```

Рис.3.1 Контрольный пример для программы

Как видно из рисунка, на экран выводится вся информация об аккаунтах.

3.2 Упражнение 2

В ходе выполнения данного упражнения, написана программа, которая вносит изменения в прошлую программу, добавляя один класс, описывающий транзакцию.

3.2.1 Пошаговое описание алгоритма

Банковским аккаунтам присваиваются значения с помощью конструкторов.

С помощью методов `Deposit`, `Withdraw` изменяется баланс аккаунтов.

С помощью метода `Write` выводятся все значения аккаунтов.

3.2.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- `System.Console.WriteLine()` – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- `System.Console.ReadKey()` – ожидает следующего нажатия клавиши пользователем;

- `Main()` – служит для запуска программы.

- `BankAccount` – класс, который описывает банковский аккаунт с помощью методов: `Number` (возвращает номер аккаунта), `Balance` (возвращает текущий баланс), `Type` (возвращает тип аккаунта), `Withdraw` (позволяет произвести снятие денег) и `Deposit` (позволяет пополнить счет), `Transaction` (возвращает информацию обо всех транзакциях, произведенных с данного аккаунта).

- `BankTransaction` – класс, который описывает транзакцию, включая методы `Amount` и `When`, возвращающие количество переведенных денег и дату произведения операции.

3.2.3 Контрольный пример

На рис.3.2 представлены результаты выполнения программы.

```
Date/Time: 10.05.2023 16:12:01 Amount: -50

Account number is 124
Account balance is 25
Account type is Deposit
Transactions:
Date/Time: 10.05.2023 16:12:01 Amount: 75
Date/Time: 10.05.2023 16:12:01 Amount: -50

Account number is 125
Account balance is 110
Account type is Checking
Transactions:
Date/Time: 10.05.2023 16:12:01 Amount: -30
Date/Time: 10.05.2023 16:12:01 Amount: 40

Account number is 126
Account balance is 275
Account type is Deposit
Transactions:
Date/Time: 10.05.2023 16:12:01 Amount: 200
Date/Time: 10.05.2023 16:12:01 Amount: -450
Date/Time: 10.05.2023 16:12:01 Amount: 25
```

Рис.3.2 Контрольный пример для программы

Как видно из рисунка, на экран выводятся вся информация об аккаунтах, дата и время.

3.3 Упражнение 3

В ходе выполнения данного упражнения, написана программа, которая дополняет прошлую программу, добавляя деструктор.

3.3.1 Пошаговое описание алгоритма

Банковским аккаунтам присваиваются значения с помощью конструкторов.

С помощью методов Deposit, Withdraw изменяется баланс аккаунтов.

С помощью метода Write выводятся все значения аккаунтов.

С помощью деструктора значения аккаунтов выводятся в файл.

3.3.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- System.Console.WriteLine() – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- System.Console.ReadKey() – ожидает следующего нажатия клавиши пользователем;

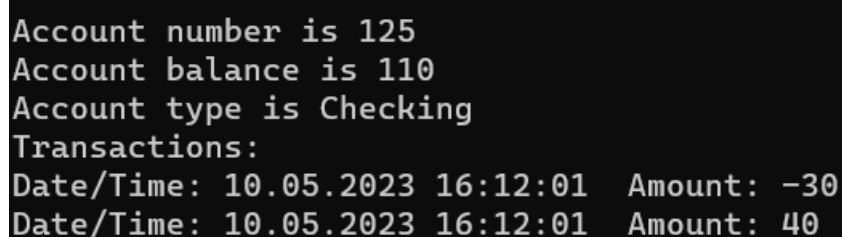
- Main() – служит для запуска программы.

- BankAccount – класс, который описывает банковский аккаунт с помощью методов: Number (возвращает номер аккаунта), Balance (возвращает текущий баланс), Type (возвращает тип аккаунта), Withdraw (позволяет произвести снятие денег) и Deposit (позволяет пополнить счет), Transaction (возвращает информацию обо всех транзакциях, произведенных с данного аккаунта).

- BankTransaction – класс, который описывает транзакцию, включая методы Amount и When, возвращающие количество переведенных денег и дату произведения операции, и деструктор.

3.3.3 Контрольный пример

На рис.3.3 представлены результаты выполнения программы.



```
Account number is 125
Account balance is 110
Account type is Checking
Transactions:
Date/Time: 10.05.2023 16:12:01 Amount: -30
Date/Time: 10.05.2023 16:12:01 Amount: 40
```

Рис.3.3 Контрольный пример для программы

Как видно из рисунка, на экран выводятся вся информация об аккаунтах, дата и время.

4. Листинг программы

AccountType.cs

```
enum AccountType
{
    Checking,
    Deposit
}
```

BankAccount.cs

```
using System.Collections;

class BankAccount
{
    private long accNo;
    private decimal accBal;
    private AccountType accType;
    private Queue tranQueue = new Queue();

    private static long nextNumber = 123;

    // Constructors
    public BankAccount()
    {
        accNo = NextNumber();
        accType = AccountType.Checking;
        accBal = 0;
    }

    public BankAccount(AccountType aType)
    {
        accNo = NextNumber();
        accType = aType;
        accBal = 0;
    }

    public BankAccount(decimal aBal)
    {
        accNo = NextNumber();
        accType = AccountType.Checking;
        accBal = aBal;
    }

    public BankAccount(AccountType aType, decimal aBal)
    {
        accNo = NextNumber();
        accType = aType;
        accBal = aBal;
    }

    public bool Withdraw(decimal amount)
    {
        bool sufficientFunds = accBal >= amount;
        if (sufficientFunds)
```

```

        {
            accBal -= amount;
            BankTransaction tran = new BankTransaction(-amount);
            tranQueue.Enqueue(tran);
        }
        return sufficientFunds;
    }

    public decimal Deposit(decimal amount)
    {
        accBal += amount;
        BankTransaction tran = new BankTransaction(amount);
        tranQueue.Enqueue(tran);
        return accBal;
    }

    public Queue Transactions()
    {
        return tranQueue;
    }

    public long Number()
    {
        return accNo;
    }

    public decimal Balance()
    {
        return accBal;
    }

    public string Type()
    {
        return accType.ToString();
    }

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

```

BankTransaction.cs

```

using System.IO;
using System;

/// <summary>
///   A BankTransaction is created every time a deposit or withdrawal occurs on a
BankAccount
///   A BankTransaction records the amount of money involved, together with the
current date and time.
/// </summary>
public class BankTransaction
{
    private readonly decimal amount;
    private readonly DateTime when;

    public BankTransaction(decimal tranAmount)
    {
        amount = tranAmount;
        when = DateTime.Now;
    }

    public decimal Amount()
    {

```



```

        return amount;
    }

    public DateTime When()
    {
        return when;
    }
    ~BankTransaction()
    {
        StreamWriter swFile = File.AppendText("Transactions.Dat");
        swFile.WriteLine("Date/Time: {0}\tAmount: {1}", when, amount);
        swFile.Close();
        GC.SuppressFinalize(this);
    }
}

```

CreateAccount.cs

```

using System;

class CreateAccount
{
    // Test Harness
    static void Main()
    {
        BankAccount acc1, acc2, acc3, acc4;

        acc1 = new BankAccount();
        acc2 = new BankAccount(AccountType.Deposit);
        acc3 = new BankAccount(100);
        acc4 = new BankAccount(AccountType.Deposit, 500);

        acc1.Deposit(100);
        acc1.Withdraw(50);
        acc2.Deposit(75);
        acc2.Withdraw(50);
        acc3.Withdraw(30);
        acc3.Deposit(40);
        acc4.Deposit(200);
        acc4.Withdraw(450);
        acc4.Deposit(25);

        Write(acc1);
        Write(acc2);
        Write(acc3);
        Write(acc4);
    }

    static void Write(BankAccount acc)
    {
        Console.WriteLine("Account number is {0}", acc.Number());
        Console.WriteLine("Account balance is {0}", acc.Balance());
        Console.WriteLine("Account type is {0}", acc.Type());
        Console.WriteLine("Transactions:");
        foreach (BankTransaction tran in acc.Transactions())
        {
            Console.WriteLine("Date/Time: {0}\tAmount: {1}", tran.When(),
            tran.Amount());
        }
        Console.WriteLine();
    }
}

```

5. Полученные результаты

В ходе выполнения данной лабораторной работы нами были получены следующие результаты:

- В ходе работы программы были созданы методы, конструкторы, деструкторы и классы для работы с банковскими аккаунтами.

6. Выводы

В ходе выполнения данной лабораторной работы:

- были изучены конструкторы и деструкторы в языке C#;

7. Список использованной литературы

1. MSDN – сеть разработчиков Microsoft. URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/constructors> (дата обращения: 19.04.2023)