**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: Создание и уничтожение объектов**

|  |  |  |
| --- | --- | --- |
| **Студенты гр. 9301** |  | **Власов Е.А.** |
|  |  | **Токарев С.В.** |
|  |  | **Служевская А.С.** |
| **Преподаватель** |  | **Новакова Н.Е.** |

**Санкт-Петербург**

**2021**

# Цель работы

Научиться создавать конструкторы и деконструкторы, ознакомиться с классом Queue, ознакомиться с обработчиком даты и времени.

# Ход работы Упражнение 1

* 1. Отредактировать файл BankAccount.cs, удалив метод Populate.
  2. Добавить конструктор BankAccount, не имеющий параметров, задающий номер аккаунта с помощью метода NextNumber, присваивающий аккаунту тип Checking и задающий баланс 0.
  3. Добавить конструктор BankAccount с параметром типа AccountType, задающий номер аккаунта с помощью NextNumber, присваивающий аккаунту тип из передаваемого параметра и задающий баланс 0.
  4. Добавить конструктор BankAccount с параметром типа decimal, задающий номер аккаунта с помощью NextNumber, присваивающий аккаунту тип Checking и задающий баланс из передаваемого параметра.
  5. Добавить конструктор BankAccount с двумя параметрами: первый типа AccountType, а второй — decimal. Конструктор задает номер аккаунта с помощью NextNumber, присваивает тип из первого параметра и задает баланс из второго параметра.

**Упражнение 2**

* 1. Добавить класс BankTransaction.
  2. Класс имеет две переменные типа readonly, позволяющего только считывать нформацию.
  3. Переменная, отвечающая за дату и время транзакции задается с помощью структуры DateTime, которая считывает текущие дату и время на устройстве, на котором запущена программа.
  4. Для запоминания очередности операций используется очередь Queue. С помощью метода Enqueue операция добавляется в конец очереди

# Упражнение 3

* 1. Добавить деконструктор ~BankTransaction() в класс BankTransaction.
  2. StreamWriter открывает файловый поток на запись в существующий файл или создает новый файл, если изначально файл отсутсвует.
  3. Далее в файл записывается информация о транзакции.

# Диаграмма классов

# Текст программы

# Упражнение 1

using System;

namespace main

{

enum AccountType

{

Checking,

Deposit

}

class BankAccount

{

private long accNo;

private decimal accBal;

private AccountType accType;

private static long nextNumber = 123;

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;

}

return sufficientFunds;

}

public decimal Deposit(decimal amount)

{

accBal += amount;

return accBal;

}

public long Number()

{

return accNo;

}

public decimal Balance()

{

return accBal;

}

public string Type()

{

string accountType = accType.ToString();

return accountType;

}

private static long NextNumber()

{

return nextNumber++;

}

public void TransferFrom(ref BankAccount accForm, decimal ammount)

{

if (accForm.Withdraw(ammount))

{

Deposit(ammount);

}

}

public static void Reverse(ref string s)

{

string sRev = "";

for (int i = s.Length - 1; i >= 0; i--)

{

sRev += s[i];

}

s = sRev;

}

}

public class Test

{

public 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);

Write(ref acc1);

Write(ref acc2);

Write(ref acc3);

Write(ref acc4);

}

static void Write(ref BankAccount getInfo)

{

Console.WriteLine("Account number is {0}", getInfo.Number());

Console.WriteLine("Account balance is {0}", getInfo.Balance());

Console.WriteLine("Account type is {0}", getInfo.Type());

Console.WriteLine(" ");

}

}

}

# Упражнение 2

using System;

using System.Collections;

namespace main

{

enum AccountType

{

Checking,

Deposit

}

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++;

}

}

public class CreateAccount

{

public 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(500);

acc1.Withdraw(100);

acc2.Deposit(100);

acc2.Withdraw(50);

acc3.Withdraw(30);

acc3.Deposit(50);

acc4.Deposit(500);

acc4.Withdraw(100);

Write(acc1);

Write(acc2);

Write(acc3);

Write(acc4);

}

static void Write(BankAccount getInfo)

{

Console.WriteLine("Account number is {0}", getInfo.Number());

Console.WriteLine("Account balance is {0}", getInfo.Balance());

Console.WriteLine("Account type is {0}", getInfo.Type());

Console.WriteLine("Transactions:");

foreach (BankTransaction tran in getInfo.Transactions())

{

Console.WriteLine("Date/Time: {0}\tAmount: {1}", tran.When(), tran.Amount());

}

Console.WriteLine(" ");

}

}

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;

}

}

# }

# Упражнение 3

using System;

using System.Collections;

using System.IO;

namespace main

{

enum AccountType

{

Checking,

Deposit

}

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 transaction in acc.Transactions())

{

Console.WriteLine("Date;Time: {0}\tAmount: {1}", transaction.When(), transaction.Amount());

}

Console.WriteLine();

}

}

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++;

}

}

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);

}

}

}

# Примеры работы программы

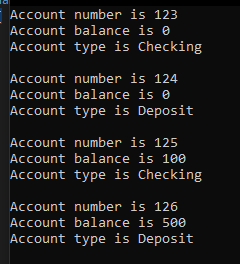
На Рис. 1 представлена работа программы упражнения 1.

Рис. 1

На Рис. 2 представлена работа программы упражнения 2.

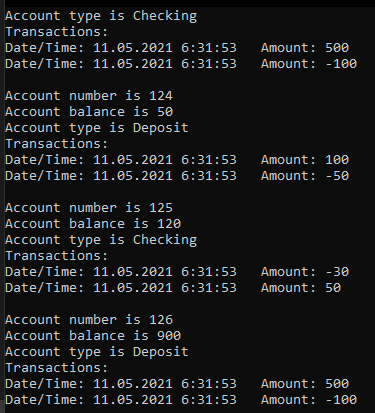


Рис. 2

На Рис. 3 представлена работа программы упражнения 3.

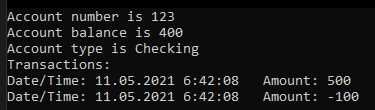


Рис. 3

# Вывод

В ходе работы были изучены разные варианты создания конструкторов и деконструктор. Так же были изучены структура DateTime, класс Queue и параметр readonly.