Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №9

По дисциплине

“Объектно-ориентированное программирование”

# Тема “Классы и объекты”

Вариант 15

Выполнил работу

студент группы РИС-19-1б

Миннахметов Э.Ю.

Проверила

доцент кафедры ИТАС

Викентьева О.Л.

Работу выполнил:

Пермь 2020

**1 ПОСТАНОВКА ЗАДАЧИ**

**Постановка задачи 1.**

1. Реализовать (в отдельном файле) определение нового класса (закрытые атрибуты, свойства, конструкторы, инициализация и вывод атрибутов).
2. Для демонстрации работы с объектами написать главную функцию, в которой создаются объекты класса и выводится информация, которая содержится в атрибутах.
3. Написать функцию, реализующую указанное в варианте действие. Рассмотреть два варианта:

1) статическую функцию;

2) метод класса.

В основной функции продемонстрировать работу функции.

1. Используя статическую компоненту класса подсчитать количество созданных в программе объектов.

|  |  |  |
| --- | --- | --- |
| Название класса | Атрибуты | Методы |
| Money | int rubles, int kopeks | Сравнение двух объектов типа Money. Результат должен быть типа bool. |

**Постановка задачи 2.**

1. Добавить к реализованному классу указанные в варианте перегруженные операции.
2. Написать демонстрационную программу, в которой создаются объекты пользовательских классов и выполняются указанные операции.

|  |  |
| --- | --- |
| Название класса | Методы |
| Money | Унарные операции:  -- вычитание копейки из объекта типа Money (учесть, что рублей и копеек не может быть меньше 0).  ++ добавление копейки к объекту типа Money (учесть, что копеек не может быть больше 99).  Операции приведения типа:  int (неявная) результатом является количество рублей (копейки отбрасываются);  double (явная) результатом является копейки, рубли отбрасываются, результат <1.  Бинарные операции:  - Money m, целое число (лево- и правосторонние операции) вычитание копеек из денежной суммы, учесть, что результат не может быть меньше 0. |

**Постановка задачи 3.**

1. Реализовать класс (в отдельном файле), полем которого является одномерный массив из элементов заданного в варианте типа.
2. В классе реализовать

* конструктор без параметров,
* конструктор с параметрами, заполняющий элементы случайными значениями,
* конструктор с параметрами, позволяющий заполнить массив элементами, заданными пользователем с клавиатуры,
* индексатор (для доступа к элементам массива),
* метод для просмотра элементов массива.

1. Написать демонстрационную программу, позволяющую создать массив разными способами и распечатать элементы массива. Подсчитать количество созданных объектов.
2. Выполнить указанное в варианте задание (если необходимо, перегрузить нужные для выполнения задачи операции или функции).

|  |  |
| --- | --- |
| Название класса | Возврат функции |
| Money | Номер минимального элемента |

**2 ОПИСАНИЕ ЭТАПА ПРОЕКТИРОВАНИЯ**

**2.1 Класс Money**

**2.1 Класс MoneyArray**

**2.1 Модуль Core**

**2.1 Модуль CLI**

**2.1 Класс CLIObject**

**2.1 Модуль с точкой входа Program**

**3 ИСХОДНЫЙ КОД ПРОГРАММЫ**

**3.1 Класс Money**

using System;

namespace Lab9

{

class Money

{

private const int c\_iDevide = 100;

private static int s\_iCount = 0;

private int m\_iRuble = 0;

private int m\_iPenny = 0;

public Money(int p) : this(0, p) { }

public Money(int r, int p)

{

if (r < 0 || p < 0)

throw new ArgumentException();

++s\_iCount;

m\_iRuble = r + p / c\_iDevide;

m\_iPenny = p % c\_iDevide;

}

~Money() { --s\_iCount; }

public static int Count

{

get { return s\_iCount; }

set { s\_iCount = value; }

}

public int Ruble

{

get { return m\_iRuble; }

set { m\_iRuble = value; }

}

public int Penny

{

get { return m\_iPenny; }

set { m\_iPenny = value; }

}

public override string ToString()

{

return "" + m\_iRuble + "," + ((m\_iPenny < 10) ? "0" : "") + m\_iPenny;

}

public static bool operator ==(Money left, Money right)

=> left.m\_iRuble == right.m\_iRuble

&& left.m\_iPenny == right.m\_iPenny;

public static bool operator !=(Money left, Money right)

=> !(left == right);

public static bool operator <(Money left, Money right)

=> left.m\_iRuble \* c\_iDevide + left.m\_iPenny

< right.m\_iRuble \* c\_iDevide + right.m\_iPenny;

public static bool operator >(Money left, Money right)

=> left != right && !(left < right);

public static bool operator <=(Money left, Money right)

=> left == right || left < right;

public static bool operator >=(Money left, Money right)

=> left == right || left > right;

public static Money operator ++(Money money)

{

Money result = new Money(money.m\_iRuble, money.m\_iPenny);

if (++result.m\_iPenny == c\_iDevide)

{

result.m\_iPenny = 0;

if (result.m\_iRuble == Core.MaxInt)

throw new InvalidOperationException();

else

++result.m\_iRuble;

}

return result;

}

public static Money operator --(Money money)

{

Money result = new Money(money.m\_iRuble, money.m\_iPenny);

if(--result.m\_iPenny < 0)

{

result.m\_iPenny = 99;

if (--result.m\_iRuble < 0)

throw new InvalidOperationException();

}

return result;

}

public static implicit operator int(Money money)

=> money.m\_iRuble;

public static explicit operator double(Money money)

=> (double)money.m\_iPenny / c\_iDevide;

public static Money operator -(Money money, int penny)

=> new Money(money.m\_iRuble \* c\_iDevide + money.m\_iPenny - penny);

public static Money operator -(int penny, Money money)

=> new Money(penny - (money.m\_iRuble \* c\_iDevide + money.m\_iPenny));

}

}

**3.2 Класс MoneyArray**

using System;

namespace Lab9

{

class MoneyArray

{

private Money[] m\_maWads = null;

public MoneyArray(Money[] array)

{

int len = array.Length;

m\_maWads = new Money[len];

for (int i = 0; i < len; ++i)

m\_maWads[i] = array[i];

}

public Money this [int index]

{

get

{

if (m\_maWads == null || index < 0 || index >= m\_maWads.Length)

throw new ArgumentException();

return m\_maWads[index];

}

set

{

if (m\_maWads == null || index < 0 || index >= m\_maWads.Length)

throw new ArgumentException();

m\_maWads[index] = value;

}

}

public void Minimum(out int index, out Money money)

{

if (m\_maWads == null)

throw new InvalidOperationException();

int len = m\_maWads.Length;

if (len == 0)

throw new InvalidOperationException();

money = m\_maWads[0];

index = 0;

for (int i = 1; i < len; ++i)

if (money > m\_maWads[i])

{

money = m\_maWads[i];

index = i;

}

}

public override string ToString()

{

string result = "";

for (int i = 0, n = m\_maWads.Length; i < n; ++i)

result += " " + m\_maWads[i];

return result;

}

}

}

**3.3 Модуль Core**

using System;

namespace Lab9

{

delegate void GetNumber(out int number, string thing, int i = -1);

delegate void GetValidate(out int number, GetNumber GetNum, IsValidate IsValid, string thing);

delegate GetNumber ModeGetNumber();

delegate bool IsValidate(int x);

delegate void Task();

delegate Task GetTask();

static class Core

{

public static IsValidate IsValidRuble = n => n >= 0;

public static IsValidate IsValidPenny = n => n >= 0 && n < 100;

public static IsValidate IsValidNum = n => n >= 0 && n < MaxInt;

public const int MinArray = 10;

public const int MaxArray = 20;

public const int MaxRubles = 999;

public const int MaxPennies = 99;

public const int MaxInt = 2147483647;

public static Random Rand = new Random();

}

}

**3.4 Модуль CLI**

using System;

namespace Lab9

{

static class CLI

{

private const string c\_sCount = "Количество элементов";

private const string c\_sRuble = "Рубли: ";

private const string c\_sPenny = "Копейки: ";

private const string c\_sReadNumber = "{0}) {1}";

private const string c\_sResult = "Результат: {0}\n";

private const string c\_sMinimum = "Минимум: Array[{0}] == {1}\n";

private const string c\_sIncorrectlyValue = "Некорректное значение!\n";

private const string c\_sInvalidArgument = "Некорректные аргументы!\n";

private const string c\_sInvalidOperation = "Некорректная операция!\n";

private const string c\_sEmptyArray = "Массив пуст.\n";

private const string c\_sException = "Неизвестная ошибка!\n";

private const string c\_sModus =

"Введите способ получения чисел (1 - ввод, - 2 случайное): ";

public static void IncorrectValue()

=> Console.WriteLine(c\_sIncorrectlyValue);

public static void Result<T>(T arg)

=> Console.WriteLine(c\_sResult, arg);

public static void Result<T, P>(T arg1, P arg2)

=> Console.WriteLine(c\_sMinimum, arg1, arg2);

public static void ReadNum(out int number, string thing, int i = -1)

{

number = 0;

for (bool flag = false; !flag;)

{

if (i == -1)

Console.Write(thing);

else

Console.Write(c\_sReadNumber, i, thing);

string sNum = Console.ReadLine();

flag = int.TryParse(sNum, out number);

if (!flag)

IncorrectValue();

}

}

public static void RandNum(out int number, string thing, int i)

{

if (thing == c\_sCount)

number = Core.Rand.Next(Core.MinArray, Core.MaxArray);

else if (thing == c\_sRuble)

number = Core.Rand.Next(0, Core.MaxRubles);

else

number = Core.Rand.Next(0, Core.MaxPennies);

}

public static GetNumber GetMode()

{

string key = "";

do

{

Console.Write(c\_sModus);

key = Console.ReadLine();

} while (key != "1" && key != "2");

if (key == "1")

return ReadNum;

else

return RandNum;

}

public static void GetValid(out int number, GetNumber GetNum, IsValidate IsValid, string thing, int i = -1)

{

while (true)

{

GetNum(out number, thing);

if (IsValid(number))

break;

else

IncorrectValue();

}

}

public static int GetValidNum(GetNumber GetNum, string text)

{

GetValid(out int num, GetNum, Core.IsValidNum, text);

return num;

}

public static Money GetMoney(GetNumber GetNum, int i = -1)

{

GetValid(out int ruble, GetNum, Core.IsValidRuble, c\_sRuble, i);

GetValid(out int penny, GetNum, Core.IsValidPenny, c\_sPenny, i);

return new Money(ruble, penny);

}

public static void Run(GetTask task)

{

while (true)

{

try

{

task()();

}

catch (ArgumentNullException)

{

Console.WriteLine(c\_sEmptyArray);

}

catch (ArgumentException)

{

Console.WriteLine(c\_sInvalidArgument);

}

catch (InvalidOperationException)

{

Console.WriteLine(c\_sInvalidOperation);

}

catch (ApplicationException)

{

break;

}

catch (Exception)

{

Console.WriteLine(c\_sException);

}

}

}

}

}

**3.5 Класс CLIObject**

using System;

namespace Lab9

{

class CLIObject

{

private const string c\_sCount = "Количество элементов: ";

private string m\_sMenu;

private Task[] m\_tTasks;

public CLIObject(string menu, params Task[] tasks)

{

m\_sMenu = menu;

m\_tTasks = tasks;

}

public void Run()

=> CLI.Run(GetTask);

public void Run(string menu, params Task[] tasks)

{

string sTemp = m\_sMenu;

Task[] tTemp = m\_tTasks;

m\_sMenu = menu;

m\_tTasks = tasks;

CLI.Run(GetTask);

Console.WriteLine();

m\_sMenu = sTemp;

m\_tTasks = tTemp;

}

public Task GetTask()

{

while (true)

{

CLI.ReadNum(out int i, m\_sMenu);

if (m\_tTasks.Length >= i && i > 0)

return m\_tTasks[i - 1];

else if (i == 0)

break;

else

continue;

}

throw new ApplicationException();

}

public Money GetMoney(GetNumber GetNum, int i = -1)

=> CLI.GetMoney(GetNum, i);

public MoneyArray GetMoneyArray(GetNumber GetNum)

{

int n = CLI.GetValidNum(GetNum, c\_sCount);

Money[] array = new Money[n];

for(int i = 0; i < n; ++i)

array[i] = GetMoney(GetNum, i);

return new MoneyArray(array);

}

}

}

**3.6 Модуль с точкой входа Program**

using System;

using System.Collections.Generic;

namespace Lab9

{

static class Program

{

private const string c\_sEnterPenny = "Введите копейки: ";

private const string c\_sRuble = "рублей";

private const string c\_sPenny = "копеек";

private const string c\_sThanksForJob = "Спасибо за работу!";

private const string c\_sMainMenu =

"Главное меню\n" +

"1. Money (1 и 2 части)\n" +

"2. MoneyArray (3 часть)\n" +

"0. Выход\n" +

"Выберете действие: ";

private const string c\_sMoneyMenu =

"Меню Money\n" +

"1. Добавить объект в список\n" +

"2. Удалить первый объект из списка\n" +

"3. Сравнение двух первых объектов в списке (1 часть)\n" +

"4. Операции над первым объектом в списке (2 часть)\n" +

"5. Вывести количество созданных объектов\n" +

"6. Вывести список объектов\n" +

"0. Назад\n" +

"Выберете действие: ";

private const string c\_sFirstPart =

"МЕНЮ ОПЕРАЦИЙ\n" +

"1. Меньше\n" +

"2. Меньше или равно\n" +

"3. Больше\n" +

"4. Больше или равно\n" +

"5. Равно\n" +

"6. Не равно\n" +

"0. Назад\n" +

"Выберете операцию: ";

private const string c\_sSecondPart =

"Меню Money (2 часть)\n" +

"1. Инкрементировать первый объект\n" +

"2. Декрементировать первый объект\n" +

"3. Привести первый объект неявно к int\n" +

"4. Привести первый объект явно к double\n" +

"5. Вычесть int (копейки) из Money (=> Money)\n" +

"6. Вычесть Money из int (копейки) (=> Money)\n" +

"0. Назад\n" +

"Выберете действие: ";

private const string c\_sThirdPart =

"Меню MoneyArray\n" +

"1. Создать и вывести\n" +

"2. Найти минимальное значение\n" +

"0. Назад\n" +

"Выберете действие: ";

private static CLIObject Obj;

private static List<Money> St;

private static MoneyArray Ar;

static void Main(string[] args)

{

Obj = new CLIObject(c\_sMainMenu, MoneyMenu, ThirdPart);

St = new List<Money>();

Obj.Run();

Console.WriteLine(c\_sThanksForJob);

Console.ReadKey();

}

static void MoneyMenu()

{

Console.WriteLine();

Obj.Run(c\_sMoneyMenu, AddStack, PopStack,

FirstPart, SecondPart, ElemCount, EnterElems);

}

static void FirstPart()

{

if (St.Count < 2)

throw new InvalidOperationException();

Console.WriteLine();

Obj.Run(c\_sFirstPart, OperatitonLess, OperatitonLessEq,

OperatitonOver, OperatitonOverEq, OperatitonEq, OperatitonNEq);

}

static void SecondPart()

{

if (St.Count == 0)

throw new InvalidOperationException();

Console.WriteLine();

Obj.Run(c\_sSecondPart, Increment, Decrement,

ToInt, ToDouble, MoneyInt, IntMoney);

}

static void ThirdPart()

{

Console.WriteLine();

Obj.Run(c\_sThirdPart, CreateEnter, FindMinimum);

}

static void AddStack()

{

St.Add(Obj.GetMoney(CLI.GetMode()));

Console.WriteLine();

}

static void PopStack()

{

St.RemoveAt(0);

Console.WriteLine();

}

static void OperatitonLess()

=> CLI.Result(St[0] < St[1]);

static void OperatitonLessEq()

=> CLI.Result(St[0] <= St[1]);

static void OperatitonOver()

=> CLI.Result(St[0] > St[1]);

static void OperatitonOverEq()

=> CLI.Result(St[0] >= St[1]);

static void OperatitonEq()

=> CLI.Result(St[0] == St[1]);

static void OperatitonNEq()

=> CLI.Result(St[0] != St[1]);

static void Increment()

=> CLI.Result(++St[0]);

static void Decrement()

=> CLI.Result(--St[0]);

static void ToInt()

{

int money = St[0];

CLI.Result(money);

}

static void ToDouble()

=> CLI.Result((double)St[0]);

static void MoneyInt()

{

CLI.GetValid(out int num, CLI.GetMode(), Core.IsValidNum, c\_sEnterPenny);

CLI.Result($"{St[0]} {c\_sRuble} - {num} {c\_sPenny} = {St[0] - num} {c\_sRuble}");

}

static void IntMoney()

{

CLI.GetValid(out int num, CLI.GetMode(), Core.IsValidNum, c\_sEnterPenny);

CLI.Result($"{num} {c\_sPenny} - {St[0]} {c\_sRuble} = {num - St[0]} {c\_sRuble}");

}

static void ElemCount()

=> CLI.Result(Money.Count);

static void EnterElems()

{

if (St.Count == 0)

throw new ArgumentNullException();

CLI.Result(new MoneyArray(St.ToArray()));

}

static void CreateEnter()

{

Ar = Obj.GetMoneyArray(CLI.GetMode());

CLI.Result(Ar);

}

static void FindMinimum()

{

if (Ar == null)

return;

Ar.Minimum(out int index, out Money money);

CLI.Result(index, money);

}

}

}

**4 СКРИНШОТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**