Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №4

на тему:

**«Конструкторы. Статические члены класса.**

**Шаблон проектирования Singleton.»**

БГУИР 6-05-0612-02 05

|  |
| --- |
| Выполнила студент группы 353504  АНТОНОВА Лидия Сергеевна |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 задание

* Спроектировать классы для выбранной предметной области: класс-контейнер, управляющий контейнеризируемым классом, и контейнеризируемый класс.
* Предусмотреть у классов наличие полей, методов и свойств.
* Предусмотреть перегрузку методов.
* Предусмотреть наличие статических методов.
* Применить к классу-контейнеру шаблон проектирования Singleton.

Предметная область: Банк-Вклад. В классе хранить информацию о наименовании банка, число вкладов, размер вклада (одинаковый для всех вкладчиков - класс Вклад), размер процентной ставки. Реализовать метод для подсчета общей выплаты по процентам. Реализовать возможность изменения (увеличения и уменьшения) размера вклада.

# Выполнение работы

Ниже представлен листинг кода.

// Deposit.cs

namespace Classes

{

public class Deposit

{

public double Amount { get; private set; }

public static double InterestRate { get; set; }

public Deposit(double amount)

{

Amount = amount;

}

public void ChangeAmount(double change)

{

Amount += change;

}

public void ChangeAmount()

{

Amount += 100;

}

public static double CalculateInterest(double amount)

{

return amount \* InterestRate / 100;

}

}

}

// Bank.cs  
namespace Classes

{

public class Bank

{

private static Bank \_instance;

public string Name { get; private set; }

public int DepositCount { get; private set; }

public double DepositSize { get; private set; }

private Bank(string name)

{

Name = name;

DepositCount = 0;

DepositSize = 0;

}

public static Bank GetInstance(string name)

{

if (\_instance == null)

{

\_instance = new Bank(name);

}

return \_instance;

}

public void AddDeposit(Deposit deposit)

{

DepositCount++;

DepositSize += deposit.Amount;

}

public void UpdateDepositSize(double change)

{

DepositSize += change;

}

public double CalculateTotalInterest()

{

return Deposit.CalculateInterest(DepositSize);

}

}

}

// Program.cs

using System;

using Classes;

class Program

{

static void Main()

{

Console.WriteLine("Enter name of bank: ");

string s = Console.ReadLine();

Bank bank = Bank.GetInstance(s);

Console.WriteLine("Enter InterestRate: ");

double Is;

while (!Double.TryParse(Console.ReadLine(), out Is))

{

Console.Write("Invalid input. Please enter a valid number for InterestRate: ");

}

Deposit.InterestRate = Is;

Deposit deposit1 = new Deposit(10000);

Deposit deposit2 = new Deposit(15000);

bank.AddDeposit(deposit1);

bank.AddDeposit(deposit2);

Console.WriteLine($"Общая выплата по процентам: {bank.CalculateTotalInterest()}");

Console.WriteLine("Enter change amount of deposit1: ");

int change;

while (!Int32.TryParse(Console.ReadLine(), out change))

{

Console.Write("Invalid input. Please enter a valid number for ChangeAmount: ");

}

deposit1.ChangeAmount(5000);

deposit2.ChangeAmount();

bank.UpdateDepositSize(change + 100);

Console.WriteLine($"Общая выплата по процентам после изменения вкладов: {bank.CalculateTotalInterest()}");

}

}

В рамках разработки программного решения для автоматизации банковских операций с вкладами, были спроектированы два ключевых класса: Deposit и Bank. Эти классы являются основой для создания объектов, которые будут управлять данными о вкладах и выполнять расчеты по процентным ставкам.

Класс Deposit представляет собой модель вклада и содержит информацию о сумме вклада. Он обладает свойством Amount, которое позволяет получить текущую сумму вклада, но изменить ее может только сам класс, что обеспечивает контроль над операциями с вкладом. Также класс имеет статическое свойство InterestRate, которое хранит процентную ставку, применимую ко всем вкладам. Это свойство является общим для всех экземпляров класса Deposit и может быть изменено непосредственно через класс, что позволяет легко управлять процентной ставкой для всех вкладов.

Конструктор класса Deposit принимает начальную сумму вклада и инициализирует свойство Amount. Это обеспечивает удобство создания вклада с заданной суммой при создании объекта класса. Класс также предоставляет два метода ChangeAmount для изменения суммы вклада: один принимает параметр с суммой изменения, а другой — перегруженный вариант — увеличивает вклад на фиксированную сумму. Это демонстрирует гибкость класса в управлении вкладами.

Статический метод CalculateInterest используется для расчета выплаты по процентам на основе текущей суммы вклада и процентной ставки. Этот метод может быть вызван без создания экземпляра класса и применяется ко всем вкладам.

Класс Bank реализует шаблон проектирования Singleton, который гарантирует, что в рамках приложения существует только один объект банка. Это достигается за счет использования приватного конструктора и статического метода GetInstance. Приватный конструктор предотвращает создание экземпляров класса напрямую, а метод GetInstance возвращает единственный экземпляр класса, создавая его при первом вызове и возвращая существующий при последующих.

Класс Bank хранит информацию о банке, такую как название, количество вкладов и общую сумму вкладов. Методы AddDeposit и UpdateDepositSize позволяют управлять вкладами, добавляя новые или изменяя сумму существующих. Метод CalculateTotalInterest используется для расчета общей выплаты по процентам на основе общей суммы вкладов.

Основная программа (Program.cs). Файл Program.cs содержит точку входа в программу и демонстрирует использование классов Bank и Deposit. В нем реализован пользовательский интерфейс для ввода данных и вывода результатов работы программы.

# Вывод

В ходе выполнения Лабораторной работы №4 были приобретены навыки работы с шаблоном проектирования Singleton. Также были рассмотрены возможности членов класса с модификатором доступа static и назначение конструкторов в классе.