Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №8

на тему:

**«ПОЛИМОРФИЗМ»**

БГУИР 6-05-0612-02 35

|  |
| --- |
| Выполнил студент группы 353504  ГУСЕНЦОВА Екатерина Андреевна |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 5.** **Предметная область: Банк.** Система хранит информацию о вкладчиках и сделанных ими вкладах. Класс “вкладчик” содержит имя вкладчика и величину вклада. Некоторым вкладчикам при создании вклада на счет может дополнительно перечисляться фиксированная сумма. В классе банк реализовать методы добавления нового вкладчика и метод вычисления общей суммы вкладов.

# 2 Выполнение работы

Перед выполнением работы следует разработать диаграмму классов для наглядного выполнения поставленной задачи (см. рисунок 1).

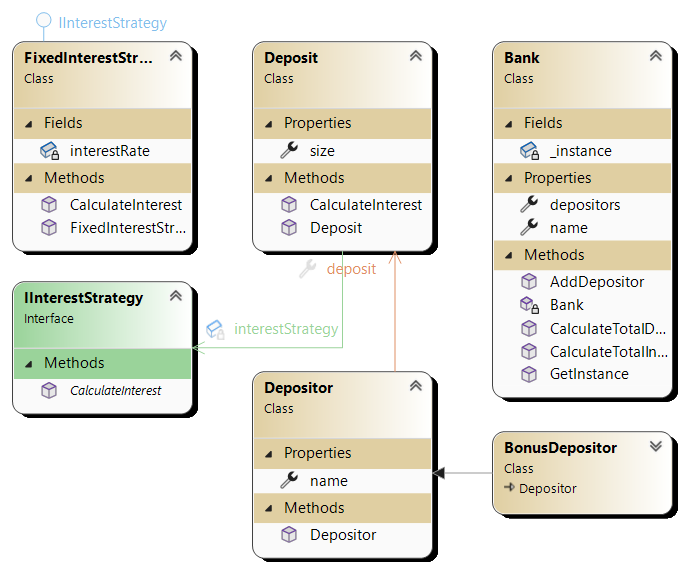


Рисунок 1 – Диаграмма классов

Класс Bank: это основной класс, который представляет банк. Он использует паттерн Singleton, что означает, что может быть создан только один экземпляр этого класса. Это полезно, когда нужно иметь единую точку доступа к ресурсу, такому как банк. В этом классе есть два метода: AddDepositor(), который добавляет нового вкладчика в банк, и CalculateTotalDeposits(), который вычисляет общую сумму всех вкладов в банке. Ниже представлен листинг кода.

using System.Collections.Generic;

namespace task1

{

public class Bank

{

private static Bank \_instance;

public string name { get; set; }

public List<Depositor> depositors { get; set; }

private Bank()

{

name = "";

depositors = new List<Depositor>();

}

public static Bank GetInstance()

{

if (\_instance == null)

{

\_instance = new Bank();

}

return \_instance;

}

public void AddDepositor(Depositor depositor)

{

depositors.Add(depositor);

}

public double CalculateTotalDeposits()

{

double total = 0;

foreach (var depositor in depositors)

{

total += depositor.deposit.size;

}

return total;

}

public double CalculateTotalInterest()

{

double total = 0;

foreach (var depositor in depositors)

{

total += depositor.deposit.CalculateInterest();

}

return total;

}

}

}

Класс Depositor: Этот класс представляет вкладчика, который имеет имя и вклад. Ниже представлен листинг кода.

namespace task1

{

public class Depositor

{

public string name { get; set; }

public Deposit deposit { get; set; }

public Depositor(string value, Deposit deposit\_value)

{

name = value;

deposit = deposit\_value;

}

}

}

Класс Deposit: Этот класс представляет вклад, который имеет размер и стратегию начисления процентов. Стратегия начисления процентов определяется через интерфейс IInterestStrategy. Ниже представлен листинг кода.

namespace task1

{

public class Deposit

{

public double size { get; set; }

private IInterestStrategy interestStrategy;

public Deposit(double value, IInterestStrategy interestStrategy)

{

size = value;

this.interestStrategy = interestStrategy;

}

public double CalculateInterest()

{

return interestStrategy.CalculateInterest(size);

}

}

}

Интерфейс IInterestStrategy: Это интерфейс, который определяет метод CalculateInterest(). Этот метод должен быть реализован в любом классе, который использует эту стратегию. В данном случае он реализован в классе FixedInterestStrategy. Ниже представлен листинг кода.

namespace task1

{

public interface IInterestStrategy

{

double CalculateInterest(double depositSize);

}

}

Класс FixedInterestStrategy: Этот класс реализует интерфейс IInterestStrategy и определяет, как вычисляются проценты для вклада. Ниже представлен листинг кода.

namespace task1

{

public class FixedInterestStrategy : IInterestStrategy

{

private double interestRate;

public FixedInterestStrategy(double interestRate)

{

this.interestRate = interestRate;

}

public double CalculateInterest(double depositSize)

{

return depositSize \* interestRate;

}

}

}

Класс BonusDepositor: Этот класс наследуется от класса Depositor и представляет вкладчика, которому при создании вклада на счет дополнительно перечисляется фиксированная сумма. Это достигается путем добавления нового поля bonus и переопределения метода CalculateInterest(). Ниже представлен листинг кода.

namespace task1

{

public class BonusDepositor : Depositor

{

private double bonus;

public BonusDepositor(string value, Deposit deposit\_value, double bonus) : base(value, deposit\_value)

{

this.bonus = bonus;

}

public double CalculateInterest()

{

return base.deposit.CalculateInterest() + bonus;

}

}

}

Покажем принцип работы этой программы в классе Program.

using System;

namespace task1

{

class Program

{

static void Main(string[] args)

{

Bank bank = Bank.GetInstance();

Depositor depositor1 = new Depositor("Alice", new Deposit(1000, new FixedInterestStrategy(0.05)));

Depositor depositor2 = new Depositor("Bob", new Deposit(2000, new FixedInterestStrategy(0.03)));

bank.AddDepositor(depositor1);

bank.AddDepositor(depositor2);

Console.WriteLine("Total deposits: " + bank.CalculateTotalDeposits());

Console.WriteLine("Total interest: " + bank.CalculateTotalInterest());

Depositor depositor3 = new BonusDepositor("Charlie", new Deposit(1500, new FixedInterestStrategy(0.04)), 100);

bank.AddDepositor(depositor3);

}

}

}

# Вывод

В ходе лабораторной работы были изучены принципы построения диаграмм классов и изучены механизмы реализации полиморфизма в C#. Также ознакомились с основными подходами при использовании интерфейсов.