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 №5

на тему:

**«Использование коллекций»**

БГУИР 6-05-0612-02 124

|  |
| --- |
| Выполнил студент группы 453504  ЯРЦЕВ Александр Александрович |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2025

# 1 Индивидуальное задание

**Задание 1. Вариант 5.** Предметная область: Банк.

Информационная система банка хранит описание процентов по различным вкладам. Система хранит информацию о вкладчиках и сделанных ими вкладах. Каждый клиент может поместить в банк только один вклад.

Система должна позволять выполнять следующие задачи:

– хранить информацию о процентах по вкладам.

– хранить информацию о клиентах;

– пополнять клиенту величину вклада;

– вычислять общую сумму выплат по процентам для всех вкладов.

# 2 Выполнение работы

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

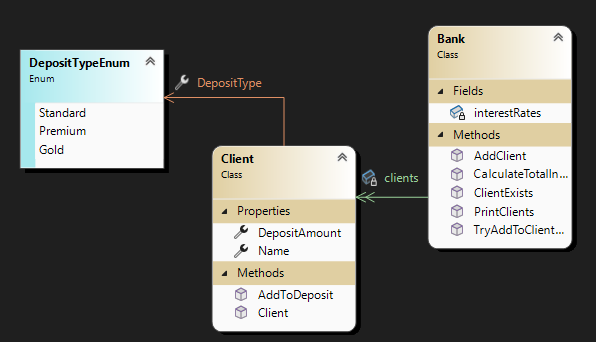


Рисунок 1 – Диаграмма классов

Для выполнения задания в проект были добавлены классы Bank, Client, BankOperations (cм. рисунок 2).

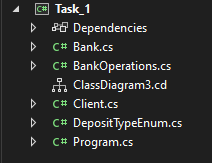


Рисунок 2 – Обозреватель решений

Рассмотрим реализацию класса Bank, где имеются поля clients, interestRares. Эти поля отвечают за хранение списка клиентов, а также имеется словарь ставок по вкладам, берущихся из DepositTypeEnum.cs (где реализовано перечисление(enum)). Имеются ключевые методы: AddClient(), TryAddToClientDeposit(), CalculateTotalInterestPayments(), PrintClients(). Они реализуют логику добавления новых клиентов, пополнение баланса клиента, вычисление общей суммы процентов по всем вкладам, вывод информации о клиентах и их вкладах.

public class Bank

{

private List<Client> clients = new List<Client>();

private Dictionary<DepositTypeEnum, decimal> interestRates = new Dictionary<DepositTypeEnum, decimal>

{

{ DepositTypeEnum.Standard, 0.05m },

{ DepositTypeEnum.Premium, 0.07m },

{ DepositTypeEnum.Gold, 0.10m }

};

public bool ClientExists(string name) =>

clients.Any(c => c.Name.Equals(name, StringComparison.OrdinalIgnoreCase));

public bool AddClient(string name, DepositTypeEnum depositType, decimal initialDeposit)

{

if (ClientExists(name)) return false;

clients.Add(new Client(name, depositType, initialDeposit));

return true;

}

public bool TryAddToClientDeposit(string name, decimal amount, out string message)

{

var client = clients.FirstOrDefault(c => c.Name.Equals(name, StringComparison.OrdinalIgnoreCase));

if (client == null)

{

message = $"Клиент '{name}' не найден.";

return false;

}

client.AddToDeposit(amount);

message = null;

return true;

}

public decimal CalculateTotalInterestPayments() =>

clients.Sum(c => c.DepositAmount \* interestRates[c.DepositType]);

public void PrintClients(Bank bank)

{

if (!clients.Any())

{

Console.WriteLine("Список клиентов пуст.");

return;

}

foreach (var client in clients)

{

decimal rate = bank.interestRates[client.DepositType];

decimal income = client.DepositAmount \* rate;

Console.WriteLine(

$"Клиент: {client.Name}\n" +

$"Депозит: {client.DepositType} ({rate \* 100}%)\n" +

$"Сумма: {client.DepositAmount:N2}\n" +

$"Доход: {client.DepositAmount} \* {rate} = {income:N2}\n" +

new string('-', 30)

);

}

}

}

Взглянем на реализацию класса Client, у которого есть поля Name и DepositType, DepositAmount которые отвечают за хранение имени, типа вклада и текущего баланса.

public class Client

{

public string Name { get; }

public DepositTypeEnum DepositType { get; }

public decimal DepositAmount { get; private set; }

public Client(string name, DepositTypeEnum depositType, decimal initialDeposit)

{

Name = name;

DepositType = depositType;

DepositAmount = initialDeposit;

}

public void AddToDeposit(decimal amount) => DepositAmount += amount;

}

Рассмотрим реализацию класса BankOperations, в котором обрабатывается пользовательский ввод через консоль. В нём определены методы: AddClient(), AddToDeposit(), AddInitialDeposite(). Которые запрашивают данные и передают их в класс Bank, а также обрабатывают пополнение вклада на валидные данные.

public static class BankOperations

{

public static void AddClient(Bank bank)

{

Console.Write("Введите имя клиента: ");

string name = Console.ReadLine()?.Trim();

if (string.IsNullOrEmpty(name))

{

Console.WriteLine("Имя клиента не может быть пустым.");

return;

}

if (bank.ClientExists(name))

{

Console.WriteLine($"Клиент '{name}' уже существует.");

return;

}

Console.WriteLine("Выберите тип вклада:");

Console.WriteLine("1. Standard (5%)");

Console.WriteLine("2. Premium (7%)");

Console.WriteLine("3. Gold (10%)");

Console.Write("Выберите опцию: ");

string typeChoice = Console.ReadLine()?.Trim();

if (!int.TryParse(typeChoice, out int type) || type < 1 || type > 3)

{

Console.WriteLine("Неверный выбор типа вклада.");

return;

}

DepositTypeEnum depositType = (DepositTypeEnum)(type - 1);

AddInitialDeposit(bank, name, depositType);

}

private static void AddInitialDeposit(Bank bank, string name, DepositTypeEnum depositType)

{

Console.Write("Введите начальную сумму вклада: ");

string depositInput = Console.ReadLine()?.Trim();

if (!decimal.TryParse(depositInput, out decimal initialDeposit) || initialDeposit <= 0)

{

Console.WriteLine("Неверная сумма вклада.");

return;

}

Console.WriteLine(bank.AddClient(name, depositType, initialDeposit)

? "Клиент успешно добавлен."

: $"Ошибка при добавлении клиента '{name}'");

}

public static void AddToDeposit(Bank bank)

{

Console.Write("Введите имя клиента: ");

string name = Console.ReadLine()?.Trim();

if (string.IsNullOrEmpty(name))

{

Console.WriteLine("Имя клиента не может быть пустым.");

return;

}

if (!bank.ClientExists(name))

{

Console.WriteLine($"Клиент '{name}' не найден.");

return;

}

Console.Write("Введите сумму для пополнения: ");

string amountInput = Console.ReadLine()?.Trim();

if (!decimal.TryParse(amountInput, out decimal amount) || amount <= 0)

{

Console.WriteLine("Неверная сумма для пополнения.");

return;

}

Console.WriteLine(bank.TryAddToClientDeposit(name, amount, out string message)

? $"Счёт '{name}' успешно пополнен на {amount}."

: message);

}

}

Покажем принцип работы этой программы в классе Program, используя и связывая между собой все предыдущие классы и их реализацию.

class Program

{

static void Main(string[] args)

{

Bank bank = new Bank();

while (true)

{

Console.WriteLine("\n[Банковская система]");

Console.WriteLine("1. Добавить клиента");

Console.WriteLine("2. Пополнить вклад");

Console.WriteLine("3. Список клиентов");

Console.WriteLine("4. Расчёт выплат по процентам");

Console.WriteLine("5. Выход");

Console.Write("Выберите действие: ");

switch (Console.ReadLine()?.Trim())

{

case "1":

BankOperations.AddClient(bank);

break;

case "2":

BankOperations.AddToDeposit(bank);

break;

case "3":

bank.PrintClients(bank);

break;

case "4":

Console.WriteLine($"Общая сумма выплат: {bank.CalculateTotalInterestPayments()}");

break;

case "5":

Console.WriteLine("Работа завершена.");

return;

default:

Console.WriteLine("Неверный выбор!");

break;

}

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 3).

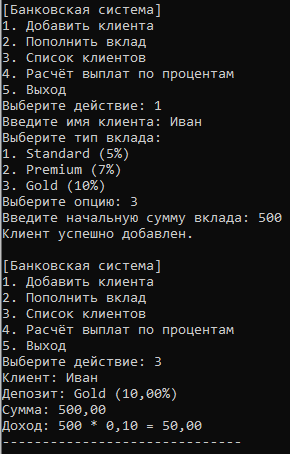


Рисунок 3 – Результат работы программы

# Вывод

В ходе лабораторной работы были изучены принципы построения диаграмм классов, использования коллекций List<T>, Dictionary<TKey, TValue> и получены навыки проектирования приложения, состоящих из нескольких взаимосвязанных классов.