Wyższa Szkoła Bankowa Merito

Programowanie Obiektowe

Ćwiczenia 4 - sprawozdanie

Imię i nazwisko: Bazyliak Stanislav
Adres email: darkplay084@gmail.com
Nr. albumu: 143067
Data: 20.05.2023
Link na git: https://github.com/darkp111/BankApplication

Po pierwsze, postanowiłem wykonać zadanie w nieco inny sposób i stworzyłem nie tylko klasę BankAccount, ale także interfejs IUser i klasę User:

```
namespace BankApplication.Data
{
    9 references
    public class BankAccount
    {
        public Users Users;
        private UserDefinitionService userDefinitionService;
        1 reference
        public BankAccount()...

        2 references
        public void AddUser(User user)...

        5 references
        public User? FindUser(Func<User, bool> predicate)...
        1 reference
        public Users GetUsers()...

        /// <summary>
        /// Function that deposit ammount to user which take as param accountNumber of user nad amount of deposit
        /// </summary>
        /// <param name="accountNumber"></param>
        /// <param name="amount"></param>
        1 reference
        public void Deposit(string accountNumber, double amount)...

        /// <summary>
        /// Function that withdraw ammount from user bank acc and check, if user doesn't have enough money
        /// </summary>
        /// <param name="accountNumber"></param>
        /// <param name="amount"></param>
        /// <returns></returns>
        1 reference
        public bool Withdraw(string accountNumber, double amount)...

        0 references
        public void DisplayBalance(User user)...
    }
}
```

```
namespace BankApplication.Interfaces
{
    1 reference
    public interface IUser
    {
        9 references
        string AccountNumber { get; set; }
        10 references
        double Balance { get; set; }
        4 references
        string Login { get; set; }
        3 references
        string Password { get; set; }
        4 references
        void DisplayBalance();
    }
}
```

```
16 references
public class User : IUser
{
    9 references
    public string AccountNumber { get; set; }
    10 references
    public double Balance { get; set; }
    4 references
    public string Login { get; set; }
    3 references
    public string Password { get; set; }

    1 reference
    public User(string accountNumber, double balance, string login, string password)
    {
        AccountNumber = accountNumber;
        Balance = balance;
        Login = login;
        Password = password;
    }

    4 references
    public void DisplayBalance()...

    2 references
    public void Deposit(double amount)...

    1 reference
    public bool Withdraw(double amount)...
}
```

Następnie, zgodnie z zadaniem, utworzyłem klasę SavingsAccount, która dziedziczy po klasie BankAccount

```
namespace BankApplication.Data
{
    3 references
    public class SavingsAccount : BankAccount
    {
        1 reference
        public double InterestRate { get; set; } = 0.05;

        1 reference
        public void AddInterest()...

        0 references
        public void DisplayBalance()...
    }
}
```

```
3 references
public class SavingsAccount : BankAccount
{
    1 reference
    public double InterestRate { get; set; } = 0.05;

    1 reference
    public void AddInterest()
    {
        foreach (var user in Users.users)
        {
            double interest = user.Balance * (InterestRate / 100);
            user.Deposit(interest);
        }
    }

    0 references
    public void DisplayBalance()
    {
        Console.WriteLine("Savings Account Balance:");
        foreach (var user in Users.users)
        {
            Console.WriteLine($"Account Number: {user.AccountNumber}, Balance: {user.Balance:C}");
        }
    }
}
```

Stworzyłem również klasę pomocniczą BankApp, oprócz tego stworzyłem klasy Authentication i UserDefinitionService:

```
public class BankApp
{
    private BankAccount bankAccount;
    private SavingsAccount savingsAccount;
    private Authentication authentication;
    private UserDefinitionService userDefinitionService;
    1 reference
    public BankApp()...

    2 references
    public void LoginningUser(BankAccount account)...

    1 reference
    public void Run()...

    1 reference
    public void CreateNewUser()...

    1 reference
    private string GenerateRandomAccountNumber()...
}
```

```
public class Authentication
{
    private BankAccount bankAccount;

    1 reference
    public Authentication(BankAccount bankAccount)
    {
        this.bankAccount = bankAccount;
    }

    1 reference
    public User? Login(string login, string password)
    {
        return bankAccount.FindUser(user => user.Login == login && user.Password == password);
    }
}
```

```
5 references
public class UserDefinitionService : IUserDefinitionService
{
    private string filePath;

    2 references
    public UserDefinitionService(string filePath)
    {
        this.filePath = filePath;
    }
    2 references
    public void LoadUsersFromJson(BankAccount bankAccount)...

    1 reference
    public void AddUsersToJson(Users users)...

    5 references
    public void SaveUsersToJson(Users users)...
}
```

Utworzyłem również osobny folder dla wyliczeń. Cały projekt jest uporządkowany, zawiera komentarze i ukończone zadania: