**РЕСПУБЛИКА БЕЛАРУСЬ**

**КОМИТЕТ ПО ОБРАЗОВАНИЮ МИНГОРИСПОЛКОМА**

учреждение образования

МИНСКИЙ ГОСУДАРСТВЕННЫЙ КОЛЛЕДЖ ЭЛЕКТРОНИКИ

##### Группа 45 ТП

## Отчет

## по лабораторной работе № 25

**"Создание программной системы,**

**использующей шаблон MVP"**

дисциплина

“Технология разработки программного обеспечения”

**Исполнитель: Сушко А. Ю.**

**Руководитель: Комиссарова Я. О.**

**Минск, 2021**

**Цель работы:** научиться применять шаблон MVP для создания программной системы.

**Вариант 11**

**Выполнение заданий:**

*Задание 1*: Изучить примеры реализации шаблона MVP. Применить шаблон MVP для реализации проекта.

**Модель:**

namespace DemoMVP.DomainModel

{

public class SimpleBankModel : IBankModel

{

private IList<Account> accounts;

public SimpleBankModel(IList<Account> accounts)

{

this.accounts = accounts;

}

/// <summary>

/// Снять деньги со счета

/// </summary>

/// <param name="accountId">Идентификатор счета</param>

/// <param name="sum">сумма</param>

/// <param name="transaction">объект транзакции</param>

/// <returns>True, если успешно. False - если нет такой суммы на балансе</returns>

public bool Withdraw(string accountId, decimal sum, ITransaction transaction)

{

EnsureSum(sum);

Account account = GetAccountById(accountId);

if (account.Balance < sum)

{

return false;

}

(transaction as Transaction).SaveState(account);

account.Balance -= sum;

return true;

}

/// <summary>

/// Внести деньги на счет

/// </summary>

/// <param name="accountId">Идентификатор счета</param>

/// <param name="sum">сумма</param>

/// <param name="transaction">объект транзакции</param>

public void Deposit(string accountId, decimal sum, ITransaction transaction)

{

EnsureSum(sum);

Account account = GetAccountById(accountId);

(transaction as Transaction).SaveState(account);

account.Balance += sum;

}

/// <summary>

/// Получить баланс счета

/// </summary>

/// <param name="accountId">Идентификатор счета</param>

/// <returns>Сумма денег на счету</returns>

public decimal GetBalance(string accountId)

{

Account account = GetAccountById(accountId);

return account.Balance;

}

/// <summary>

/// Создать объект транзакции

/// </summary>

/// <returns></returns>

public ITransaction CreateTransaction()

{

return new Transaction();

}

private void EnsureSum(decimal sum)

{

if (sum <= 0)

{

throw new ApplicationException("Sum must be greater than 0");

}

}

private Account GetAccountById(string accountId)

{

Account account = accounts.Where(x => x.Id == accountId).FirstOrDefault();

if (account == null)

{

throw new ApplicationException("Account not found");

}

return account;

}

}

}

**Представление:**

namespace DemoMVP.Abstract

{

public interface ITransferView

{

/// <summary>

/// Обновить баланс счета-источника

/// </summary>

/// <param name="balance">Сумма на балансе</param>

void UpdateSrcBalance(decimal balance);

/// <summary>

/// Обновить баланс счетаназначения

/// </summary>

/// <param name="balance">Сумма на балансе</param>

void UpdateDestBalance(decimal balance);

/// <summary>

/// Отобразить предупреждение

/// </summary>

/// <param name="text">Текст сообщения</param>

void ShowWarning(string text);

/// <summary>

/// Отобразить ошибку

/// </summary>

/// <param name="text">Текст ошибки</param>

void ShowError(string text);

/// <summary>

/// Событие, срабатывающее при изменении идентификатора счета-источника

/// </summary>

event EventHandler<AccountChangedEventArgs> SrcAccountChanged;

/// <summary>

/// Событие, срабатывающее при изменении идентификатора счета-назначения

/// </summary>

event EventHandler<AccountChangedEventArgs> DestAccountChanged;

/// <summary>

/// Событие, срабатывающее при запросе перевода денег

/// </summary>

event EventHandler<TransferMoneyEventArgs> TransferMoney;

}

}

**Реализация интерфейса:**

namespace DemoMVP

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window, ITransferView

{

public MainWindow()

{

InitializeComponent();

}

private string srcAccountId = string.Empty;

private string destAccountId = string.Empty;

public void UpdateSrcBalance(decimal balance)

{

lblSrcBalance.Content = balance.ToString();

}

public void UpdateDestBalance(decimal balance)

{

lblDestBalance.Content = balance.ToString();

}

public void ShowWarning(string text)

{

MessageBox.Show(text, "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

public void ShowError(string text)

{

MessageBox.Show(text, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

public event EventHandler<TransferMoneyEventArgs> TransferMoney;

protected virtual void OnTransferMoney(TransferMoneyEventArgs e)

{

if (TransferMoney != null)

{

TransferMoney(this, e);

}

}

public event EventHandler<AccountChangedEventArgs> SrcAccountChanged;

protected virtual void OnSrcAccountChanged(AccountChangedEventArgs e)

{

if (SrcAccountChanged != null)

{

SrcAccountChanged(this, e);

}

}

public event EventHandler<AccountChangedEventArgs>DestAccountChanged;

protected virtual void OnDestAccountChanged(AccountChangedEventArgs e)

{

if (DestAccountChanged != null)

{

DestAccountChanged(this, e);

}

}

private void txtSrcAccount\_LostFocus(object sender, RoutedEventArgs e)

{

if (!string.IsNullOrEmpty(txtSrcAccount.Text) && txtSrcAccount.Text != srcAccountId)

{

OnSrcAccountChanged(new AccountChangedEventArgs(txtSrcAccount.Text));

srcAccountId = txtSrcAccount.Text;

}

}

private void txtDestAccount\_LostFocus(object sender, RoutedEventArgs e)

{

if (!string.IsNullOrEmpty(txtDestAccount.Text) && txtDestAccount.Text != destAccountId)

{

OnDestAccountChanged(new AccountChangedEventArgs(txtDestAccount.Text));

destAccountId = txtDestAccount.Text;

}

}

private void btnTransfer\_Click(object sender, RoutedEventArgs e)

{

decimal sum;

if (decimal.TryParse(txtSum.Text, out sum))

{

OnTransferMoney(new TransferMoneyEventArgs(txtSrcAccount.Text, txtDestAccount.Text, sum));

}

else

{

ShowWarning("Ошибка ввода суммы");

}

}

}

public partial class App : Application

{

App()

{

InitializeComponent();

}

[STAThread]

static void Main()

{

IBankModel model = new SimpleBankModel(new List<Account>()

{

new Account() { Id = "1", Balance = 5000 },

new Account() { Id = "2", Balance = 2000 },

new Account() { Id = "3", Balance = 500 },

new Account() { Id = "4", Balance = 5500 }

});

MainWindow view = new MainWindow();

TransferPresenter presenter = new TransferPresenter(model, view);

App app = new App();

app.Run(view);

}

}

}

**Презентор:**

namespace DemoMVP.Presenter

{

public class TransferPresenter

{

private IBankModel model;

private ITransferView view;

public TransferPresenter(IBankModel model, ITransferView view)

{

this.model = model;

this.view = view;

this.view.SrcAccountChanged += view\_SrcAccountChanged;

this.view.DestAccountChanged += view\_DestAccountChanged;

this.view.TransferMoney += view\_TransferMoney;

}

private void view\_SrcAccountChanged(object sender, AccountChangedEventArgs e)

{

try

{

decimal balance = model.GetBalance(e.AccountId);

view.UpdateSrcBalance(balance);

}

catch(ApplicationException ex)

{

view.ShowError(ex.Message);

}

}

private void view\_DestAccountChanged(object sender, AccountChangedEventArgs e)

{

try

{

decimal balance = model.GetBalance(e.AccountId);

view.UpdateDestBalance(balance);

}

catch (ApplicationException ex)

{

view.ShowError(ex.Message);

}

}

private void view\_TransferMoney(object sender, TransferMoneyEventArgs e)

{

ITransaction tran = model.CreateTransaction();

try

{

tran.Begin();

bool success = model.Withdraw(e.SrcAccountId, e.Sum, tran);

if (success)

{

model.Deposit(e.DestAccountId, e.Sum, tran);

tran.Commit();

view.UpdateSrcBalance(model.GetBalance(e.SrcAccountId));

view.UpdateDestBalance(model.GetBalance(e.DestAccountId));

}

else

{

tran.Rollback();

view.ShowWarning("Недостаточно денег для перевода");

}

}

catch(ApplicationException ex)

{

tran.Rollback();

view.ShowError(ex.Message);

}

}

}

}

*Ответы на контрольные вопросы:*

1. Каково назначение шаблона MVP?

MVP — шаблон проектирования, производный от MVC, разработанный в основном для построения пользовательского интерфейса.

2. Из каких элементов состоит шаблон MVP?

В этом шаблоне есть три элемента:

1. View.
2. Presenter.
3. Model (модель).
4. Принцип работы шаблона MVP?

1. Элемент **View** отвечает за показ пользовательских данных и перехват пользовательских действий. Всё это он посылает Presenter.

2. **Presenter** обрабатывает действия пользователя в UI, учитывает изменения данных в Model и посылает эту информацию View. Presenter — это элемент, который содержит всю бизнес-логику работы с пользовательским интерфейсом.

3. **Model** содержит в себе модели из предметной области, которые отображают знания и данные предметной области вашего приложения. Model посылает информацию об изменении данных Presenter и принимает сообщения от Presenter