Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М. Машерова

Отчет

по учебной практике по программированию

«Модуль 3 **Задания по делегатам**»

Выполнил учащийся Близнецов А.А.

группы 3ПОИТ22 02.10.2024 г.

Проверил Алейников М.А.

2024 г.

Орша, 2024

**Задания:**

1. Создайте базовый класс "Фигура" с методом для вычисления площади. Затем создайте производные классы для разных геометрических фигур (круг, прямоугольник, треугольник) и используйте делегат для динамического вызова метода вычисления площади.
2. Реализуйте систему событий для мобильного приложения. Создайте класс "Уведомление" с событиями для отправки уведомлений (сообщения, звонки, электронные письма). Зарегистрируйте обработчики событий для разных типов уведомлений.
3. Создайте приложение для управления задачами с использованием делегатов. Пользователь должен иметь возможность добавлять задачи и выбирать делегата для выполнения каждой задачи (например, отправка уведомления или запись в журнал).
4. Разработайте систему фильтрации данных с использованием делегатов. Пользователь должен иметь возможность выбрать фильтр для списка данных (например, фильтр по дате или по ключевым словам).
5. Создайте приложение для сортировки числовых данных. Пользователь должен иметь возможность выбрать метод сортировки (например, сортировка пузырьком или быстрая сортировка) с помощью делегатов.

Выполнение:

Листинг 1. Код основной логики приложения

public MainWindow()

{

InitializeComponent();

}

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

{

try

{

// Получаем значения из TextBox

double radius = double.Parse(RadiusTextBox.Text);

double width = double.Parse(Rect1TextBox.Text);

double height = double.Parse(Rect2TextBox.Text);

double sideA = double.Parse(Triangle1TextBox.Text);

double sideB = double.Parse(Triangle2TextBox.Text);

double sideC = double.Parse(Triangle3TextBox.Text);

// Создаем фигуры

Figure circle = new Circle(radius);

Figure rectangle = new Rectangle(width, height);

Figure triangle = new Triangle(sideA, sideB, sideC);

// Вычисляем площади

double circleArea = circle.CalculateArea();

double rectangleArea = rectangle.CalculateArea();

double triangleArea = triangle.CalculateArea();

// Отображаем результаты

ResultTextBlock.Text = $"Площадь круга: {circleArea}\nПлощадь прямоугольника: {rectangleArea}\nПлощадь Треугольника: {triangleArea}";

}

catch (FormatException)

{

MessageBox.Show("Введены неверные значения");

}

catch (Exception ex)

{

MessageBox.Show($"Возникла ошибка: {ex.Message}");

}

}

Листинг 2. Интерфейс программы

<Window x:Class="WpfApp1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="600">

<Grid>

<!-- Круг -->

<Label Content="рудиус круга:" HorizontalAlignment="Left" VerticalAlignment="Top"/>

<TextBox x:Name="RadiusTextBox" HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Margin="160,4,0,0"/>

<!-- Прямоугольник -->

<Label Content="сторона прямоугольника 1:" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="0,30,0,0"/>

<TextBox x:Name="Rect1TextBox" HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Margin="160,34,0,0"/>

<Label Content="сторона прямоугольника 2:" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="0,60,0,0"/>

<TextBox x:Name="Rect2TextBox" HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Margin="160,64,0,0"/>

<!-- Треугольник -->

<Label Content="сторона треугольника 1:" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="0,90,0,0"/>

<TextBox x:Name="Triangle1TextBox" HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Margin="160,94,0,0"/>

<Label Content="сторона треугольника 2:" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="0,120,0,0"/>

<TextBox x:Name="Triangle2TextBox" HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Margin="160,124,0,0"/>

<Label Content="сторона треугольника 3:" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="0,150,0,0"/>

<TextBox x:Name="Triangle3TextBox" HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Margin="160,154,0,0"/>

<Button Content="Расчёт площади" HorizontalAlignment="Left" Width="120" Margin="40,180,0,0" Click="CalculateArea\_Click" VerticalAlignment="Top" HorizontalContentAlignment="Center" VerticalContentAlignment ="Center"/>

<TextBlock x:Name="ResultTextBlock" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="25,218,0,0" Width="275" Height="87">

<TextBlock.OpacityMask>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Black"/>

<GradientStop Color="White" Offset="1"/>

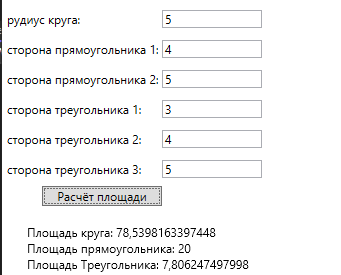
</LinearGradientBrush>

</TextBlock.OpacityMask>

</TextBlock>

</Grid>

</Window>

  
Рисунок 1 – Результат выполнения программы

Листинг 3: Логика работы системы уведомлений

public partial class MainWindow : Window

{

Notification notification = new Notification();

public MainWindow()

{

InitializeComponent();

notification.MessageSent += Notification\_MessageSent;

notification.CallMade += Notification\_CallMade;

notification.EmailSent += Notification\_EmailSent;

}

private void Notification\_MessageSent(object sender, NotificationEventArgs e)

{

MessageBox.Show($"Сообщение отправлено: {e.Message}");

}

private void Notification\_CallMade(object sender, NotificationEventArgs e)

{

MessageBox.Show($"Звонок сделан: {e.Message}");

}

private void Notification\_EmailSent(object sender, NotificationEventArgs e)

{

MessageBox.Show($"Email отправлен: {e.Message}");

}

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

{

notification.SendMessage("Привет! Это тестовое сообщение.");

}

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

{

notification.MakeCall("Звонок на номер 123-456-7890.");

}

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

{

notification.SendEmail("test@example.com");

}

}

Листинг 4: Класс Notification

public class NotificationEventArgs : EventArgs

{

public string Message { get; }

public NotificationEventArgs(string message)

{

Message = message;

}

}

public class Notification

{

public event EventHandler<NotificationEventArgs> MessageSent;

public event EventHandler<NotificationEventArgs> CallMade;

public event EventHandler<NotificationEventArgs> EmailSent;

public void SendMessage(string message)

{

OnMessageSent(new NotificationEventArgs(message));

}

public void MakeCall(string message)

{

OnCallMade(new NotificationEventArgs(message));

}

public void SendEmail(string message)

{

OnEmailSent(new NotificationEventArgs(message));

}

protected virtual void OnMessageSent(NotificationEventArgs e)

{

MessageSent?.Invoke(this, e);

}

protected virtual void OnCallMade(NotificationEventArgs e)

{

CallMade?.Invoke(this, e);

}

protected virtual void OnEmailSent(NotificationEventArgs e)

{

EmailSent?.Invoke(this, e);

}

}

}

Листинг 5: Интерфейс приложения

Title="Система уведомлений" Height="500" Width="600">

<StackPanel Margin="20">

<Button Content="Отправить сообщение" Click="SendMessageButton\_Click" />

<Button Content="Сделать звонок" Click="MakeCallButton\_Click" />

<Button Content="Отправить письмо на электронную почту" Click="SendEmailButton\_Click" />

</StackPanel>

</Window>

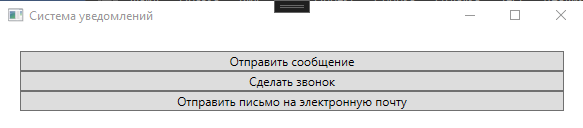


Рисунок 2 – Интерфейс приложения

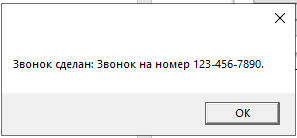


Рисунок 3 – Результат вызова метода класса Notification по нажатию кнопки

Листинг 6 – Интерфейс программы task list

<Window x:Class="WpfApp1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Task Manager" Height="350" Width="525">

<Grid>

<StackPanel Margin="10">

<TextBox x:Name="TaskDescription" Width="400" Margin="0,0,0,10" />

<ComboBox x:Name="DelegateSelector" Margin="0,0,0,10">

<ComboBoxItem Content="Отправить уведомление" Tag="Notify"/>

<ComboBoxItem Content="Отправить в файл" Tag="Log"/>

</ComboBox>

<Button Content="добавить задачу" Click="AddTask\_Click" Width="100"/>

<Button Content="Информация о задаче" Click="ShowTaskInfo\_Click" Width="140" Margin="0,10,0,0"/>

<ListBox x:Name="TaskList" Margin="0,10,0,0"/>

</StackPanel>

</Grid>

</Window>

Листинг 7 – Логика программы (делегат прописан в начале)

public delegate void TaskDelegate(string message);

public class Task

{

public string Name { get; set; }

public TaskDelegate Execute { get; set; }

public Task(string name, TaskDelegate execute)

{

Name = name;

Execute = execute;

}

public void Run()

{

if (Execute != null)

{

Execute($"Executing task: {Name}");

}

else

{

throw new InvalidOperationException("Не выбран делегат");

}

}

}

public partial class MainWindow : Window

{

private List<Task> tasks = new List<Task>();

public MainWindow()

{

InitializeComponent();

}

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

{

var description = TaskDescription.Text;

if (string.IsNullOrWhiteSpace(description)) return;

TaskDelegate selectedDelegate = null;

var selectedItem = DelegateSelector.SelectedItem as ComboBoxItem;

if (selectedItem != null)

{

switch (selectedItem.Tag)

{

case "Notify":

selectedDelegate = Notify;

break;

case "Log":

selectedDelegate = LogToConsole;

break;

}

}

if (selectedDelegate != null)

{

var task = new Task(description, selectedDelegate);

tasks.Add(task);

TaskList.Items.Add(task.Name);

TaskDescription.Clear();

}

else

{

MessageBox.Show("Выберите делегат");

}

}

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

{

var selectedTaskName = TaskList.SelectedItem as string;

if (string.IsNullOrWhiteSpace(selectedTaskName))

{

MessageBox.Show("Выберете задачу из списка");

return;

}

var task = tasks.FirstOrDefault(t => t.Name == selectedTaskName);

if (task != null)

{

string delegateName = task.Execute != null ? task.Execute.Method.Name : "Не добавлен делегат";

MessageBox.Show($"Задача: {task.Name}\nДелегат: {delegateName}");

}

else

{

MessageBox.Show("задача не найдена");

}

}

private void Notify(string message)

{

MessageBox.Show(message);

}

private void LogToConsole(string message)

{

MessageBox.Show(message);

}

}

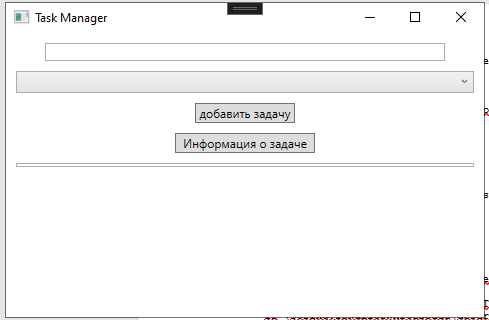


Рисунок 4 – Интерфейс приложения

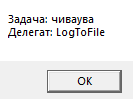


Рисунок 5 – Результат вывода информации о задаче

Листинг 8. Интерфейс программы по фильтрации данных

<Window x:Class="WpfApp1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Фильтр информации" Height="400" Width="600">

<Grid>

<StackPanel Margin="10">

<ComboBox

x:Name="FilterSelector"

Margin="0,0,0,10"

SelectionChanged="FilterSelector\_SelectionChanged">

<ComboBoxItem Content="все" Tag="все"/>

<ComboBoxItem Content="недавние" Tag="недавние"/>

<ComboBoxItem Content="ключевое слово" Tag="ключевое слово"/>

</ComboBox>

<TextBox

x:Name="KeywordTextBox"

Width="200" Margin="0,0,0,10"

Visibility="Collapsed"/>

<ListBox

x:Name="ItemList"

Height="200"

Margin="0,10,0,0"/>

</StackPanel>

</Grid>

</Window>

Листинг 9. Логика программы по фильтрации данных

public partial class MainWindow : Window

{

private List<Item> items;

public MainWindow()

{

InitializeComponent();

LoadItems();

UpdateItemList();

}

private void LoadItems()

{

items = new List<Item>

{

new Item { Name = "Item 1", Date = DateTime.Now.AddDays(-1) },

new Item { Name = "Item 255", Date = DateTime.Now.AddDays(-6) },

new Item { Name = "какой-то элемент", Date = DateTime.Now.AddDays(-3) },

new Item { Name = "Item 4", Date = DateTime.Now.AddDays(-5) },

new Item { Name = "Task", Date = DateTime.Now.AddDays(-4) },

new Item { Name = "чиваува", Date = DateTime.Now.AddDays(-1) }

};

}

private void UpdateItemList(List<Item> filteredItems = null)

{

ItemList.Items.Clear();

var itemsToDisplay = filteredItems ?? items;

foreach (var item in itemsToDisplay)

{

ItemList.Items.Add(item.Name);

}

}

private void FilterSelector\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var selectedItem = FilterSelector.SelectedItem as ComboBoxItem;

if (selectedItem == null) return;

switch (selectedItem.Tag.ToString())

{

case "все":

UpdateItemList();

KeywordTextBox.Visibility = Visibility.Collapsed;

break;

case "недавние":

var recentFilter = new ItemFilter(item => item.Date >= DateTime.Now.AddDays(-2));

var recentItems = FilterManager.FilterItems(items, recentFilter);

UpdateItemList(recentItems);

KeywordTextBox.Visibility = Visibility.Collapsed;

break;

case "ключевое слово":

KeywordTextBox.Visibility = Visibility.Visible;

KeywordTextBox.TextChanged -= KeywordTextBox\_TextChanged;

KeywordTextBox.TextChanged += KeywordTextBox\_TextChanged;

break;

}

}

private void KeywordTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var keyword = KeywordTextBox.Text.ToLower();

var keywordFilter = new ItemFilter(item => item.Name.ToLower().Contains(keyword));

var filteredItems = FilterManager.FilterItems(items, keywordFilter);

UpdateItemList(filteredItems);

}

}

Листинг 10. Интерфейс приложения по сортировке

<Window x:Class="WpfApp1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Sorting App" Height="300" Width="400">

<Grid>

<StackPanel Margin="10">

<TextBox x:Name="InputTextBox" Width="250" Margin="0,0,0,10" />

<ComboBox x:Name="SortMethodComboBox" Margin="0,0,0,10">

<ComboBoxItem Content="Сортировка пузырьком" Tag="BubbleSort"/>

<ComboBoxItem Content="Быстрая сортировка" Tag="QuickSort"/>

</ComboBox>

<Button Content="Сортировать" Click="SortButton\_Click" Width="100"/>

<TextBlock x:Name="ResultTextBlock" Margin="0,10,0,0" TextWrapping="Wrap"/>

</StackPanel>

</Grid>

</Window>

Листинг 11. Логика программы (делегат описан для выбора сортировки)

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

string input = InputTextBox.Text;

if (string.IsNullOrWhiteSpace(input))

{

MessageBox.Show("Введите значения через пробел.");

return;

}

int[] numbers;

try

{

numbers = input.Split(' ')

.Select(int.Parse)

.ToArray();

}

catch (FormatException)

{

MessageBox.Show("Введите корректные значения, разделённые пробелами.");

return;

}

SortDelegate sortMethod = null;

var selectedItem = SortMethodComboBox.SelectedItem as ComboBoxItem;

if (selectedItem != null)

{

switch (selectedItem.Tag.ToString())

{

case "BubbleSort":

sortMethod = Sorter.BubbleSort;

break;

case "QuickSort":

sortMethod = Sorter.QuickSort;

break;

}

}

if (sortMethod != null)

{

var sortedNumbers = sortMethod(numbers);

ResultTextBlock.Text = "Отсортированные числа: " + string.Join(", ", sortedNumbers);

}

else

{

MessageBox.Show("Выберите методы сортировки.");

}

}

}