Tristan Izlar

COP2362

TUTORIAL 4-2: Using Indexers

I worked alone.

Graphical user interface, text

Description automatically generated with medium confidence

Name.cs

namespace Indexers

{

struct Name

{

private string name;

public Name(string text)

{

this.name = text;

}

public string Text

{

get { return this.name; }

}

public override int GetHashCode() => this.name.GetHashCode();

public override bool Equals(object other) => (other is Name) && Equals((Name)other);

public bool Equals(Name other) => this.name == other.name;

}

}

PhoneBook.cs

using System;

namespace Indexers

{

sealed class PhoneBook

{

private int used;

private Name[] names;

private PhoneNumber[] phoneNumbers;

public PhoneBook()

{

int initialSize = 0;

this.used = 0;

this.names = new Name[initialSize];

this.phoneNumbers = new PhoneNumber[initialSize];

}

public void Add(Name name, PhoneNumber number)

{

enlargeIfFull();

this.names[used] = name;

this.phoneNumbers[used] = number;

this.used++;

}

public Name this[PhoneNumber number]

{

get

{

int i = Array.IndexOf(this.phoneNumbers, number);

if(i!=-1)

{

return this.names[i];

}

else

{

return new Name();

}

}

}

public PhoneNumber this[Name name]

{

get

{

int i = Array.IndexOf(this.names, name);

if(i!=-1)

{

return this.phoneNumbers[i];

}

else

{

return new PhoneNumber();

}

}

}

private void enlargeIfFull()

{

if (this.used == this.names.Length)

{

int bigger = used + 16;

Name[] moreNames = new Name[bigger];

this.names.CopyTo(moreNames, 0);

PhoneNumber[] morePhoneNumbers = new PhoneNumber[bigger];

this.phoneNumbers.CopyTo(morePhoneNumbers, 0);

this.names = moreNames;

this.phoneNumbers = morePhoneNumbers;

}

}

}

}

PhoneNumber.cs

namespace Indexers

{

struct PhoneNumber

{

private string number;

public PhoneNumber(string text)

{

this.number = text;

}

public string Text

{

get { return this.number; }

}

public override int GetHashCode() => this.number.GetHashCode();

public override bool Equals(object other) => (other is PhoneNumber) && Equals((PhoneNumber)other);

public bool Equals(PhoneNumber other) => this.number == other.number;

}

}

MainPage.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.WindowsRuntime;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

namespace Indexers

{

public sealed partial class MainPage : Page

{

private PhoneBook phoneBook = new PhoneBook();

public MainPage()

{

this.InitializeComponent();

}

private void findByNameClick(object sender, RoutedEventArgs e)

{

string text = name.Text;

if(!String.IsNullOrEmpty(text))

{

Name personsName = new Name(text);

PhoneNumber personsPhoneNumber = this.phoneBook[personsName];

phoneNumber.Text = String.IsNullOrEmpty(personsPhoneNumber.Text) ? "Not Found" : personsPhoneNumber.Text;

}

}

private void findByPhoneNumberClick(object sender, RoutedEventArgs e)

{

string text = phoneNumber.Text;

if(!String.IsNullOrEmpty(text))

{

PhoneNumber personsPhoneNumber = new PhoneNumber(text);

Name personsName = this.phoneBook[personsPhoneNumber];

name.Text = String.IsNullOrEmpty(personsName.Text) ? "Not Found" : personsName.Text;

}

}

private void addClick(object sender, RoutedEventArgs e)

{

if (!String.IsNullOrEmpty(name.Text) && !String.IsNullOrEmpty(phoneNumber.Text))

{

phoneBook.Add(new Name(name.Text),

new PhoneNumber(phoneNumber.Text));

name.Text = "";

phoneNumber.Text = "";

}

}

}

}