МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**ЛАБОРАТОРНАЯ РАБОТА № 5**

по дисциплине

**«Объектно-ориентированное программирование»**

на тему:

**«Классы, объекты, наследование в C#»**

*Вариант № 9*

Выполнил:

Студент группы

Ктбо2-8

Жалнин Д. И.

Проверил:

Тарасов С. А.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Таганрог 2020

# Цель работы

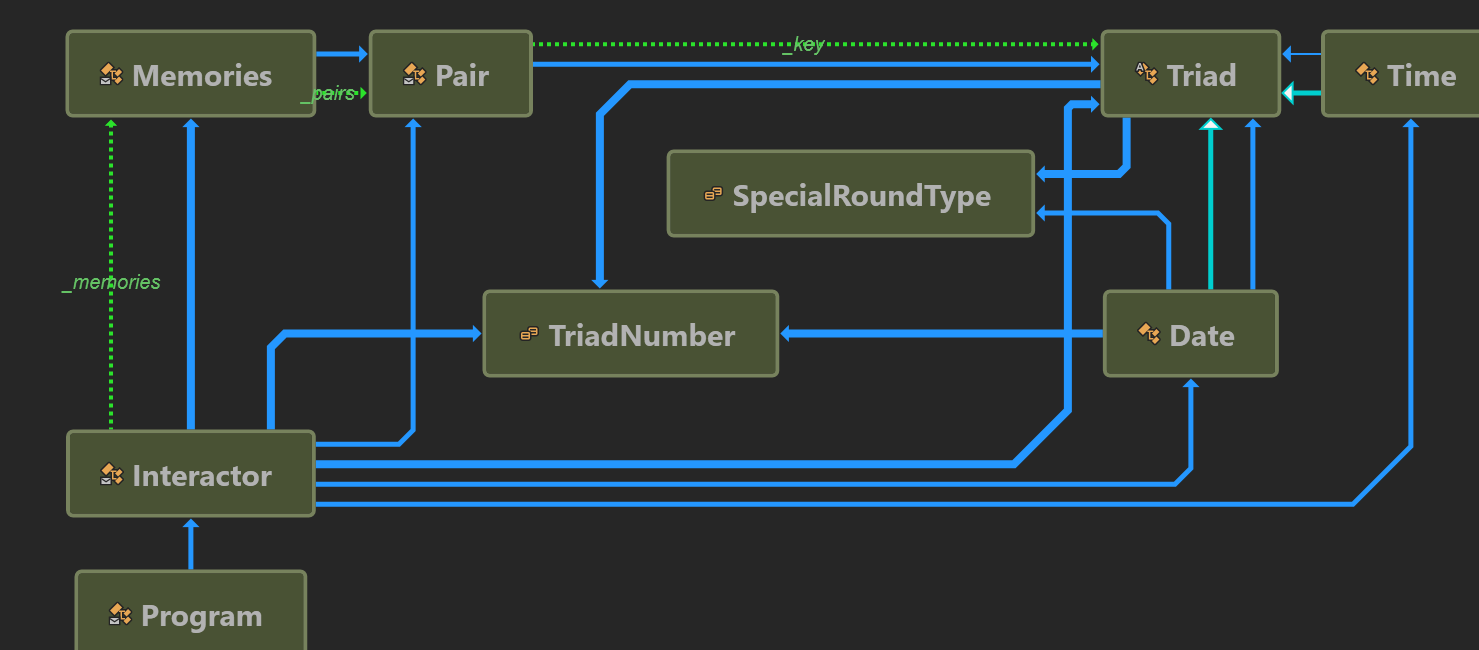
Ознакомление с основами объектно-ориентированного программирования на С#

# Вариант задания

Создать класс Triad (тройка) и на его основе классы Date (дата) и Time (время) с методами увеличения и уменьшения на 1 каждого из значений тройки: год/час, месяц/минуты, день/секунды.

В дополнительном классе Memories создать массив пар (дата-время) объектов этих классов: 01.09.2020 – 8.00.00, …. Количество пар задается конструктором класса Memories.

# Схема классов



*Рисунок 1 – Схема зависимостей классов*

# Листинг программы

## Triad.cs

using System;

namespace Lab5

{

public abstract class Triad

{

private readonly int \_maxA;

private readonly int \_maxB;

private readonly int \_maxC;

private int \_a,\_b,\_c;

protected Triad(int maxA, int maxB, int maxC, int a, int b, int c)

{

this.\_maxA = maxA;

this.\_maxB = maxB;

this.\_maxC = maxC;

this.\_a = a;

this.\_b = b;

this.\_c = c;

}

public int IncrementAndGet(TriadNumber number)

{

Change(number, 1);

return Get(number);

}

public int DecrementAndGet(TriadNumber number)

{

Change(number, -1);

return Get(number);

}

public int GetAndIncrement(TriadNumber number)

{

int temp = Get(number);

Change(number, 1);

return temp;

}

public int GetAndDecrement(TriadNumber number)

{

int temp = Get(number);

Change(number, 1);

return temp;

}

public int Get(TriadNumber number)

{

return number switch

{

TriadNumber.FIRST => \_a,

TriadNumber.SECOND => \_b,

TriadNumber.THIRD => \_c,

\_ => throw new ArgumentOutOfRangeException(nameof(number), number, "Invalid triad number")

};

}

private void Change(TriadNumber number, int x)

{

switch (number)

{

case TriadNumber.FIRST:

\_a += x;

break;

case TriadNumber.SECOND:

\_b += x;

break;

case TriadNumber.THIRD:

\_c += x;

break;

default:

throw new ArgumentOutOfRangeException(nameof(number), number, "Invalid triad number");

}

Round();

}

protected void Set(TriadNumber number, int x)

{

switch (number)

{

case TriadNumber.FIRST:

\_a = x;

break;

case TriadNumber.SECOND:

\_b = x;

break;

case TriadNumber.THIRD:

\_c = x;

break;

default:

throw new ArgumentOutOfRangeException(nameof(number), number, "Invalid triad number");

}

}

protected virtual void SpecialRound(SpecialRoundType specialRound)

{

}

protected void Round()

{

if (\_a > \_maxA)

{

SpecialRound(SpecialRoundType.INCREMENT);

\_b++;

\_a = 0;

SpecialRound(SpecialRoundType.NULL\_INCREMENT);

}

if (\_b > \_maxB)

{

SpecialRound(SpecialRoundType.INCREMENT);

\_c++;

\_b = 0;

SpecialRound(SpecialRoundType.NULL\_INCREMENT);

}

if (\_c > \_maxC)

{

SpecialRound(SpecialRoundType.INCREMENT);

\_c = 0;

SpecialRound(SpecialRoundType.NULL\_INCREMENT);

}

SpecialRound( SpecialRoundType.NULL\_DECREMENT);

if (\_a < 0)

{

\_b--;

\_a = \_maxA;

SpecialRound(SpecialRoundType.DECREMENT);

}

SpecialRound(SpecialRoundType.NULL\_DECREMENT);

if (\_b < 0)

{

\_c--;

\_b = \_maxB;

SpecialRound(SpecialRoundType.DECREMENT);

}

SpecialRound(SpecialRoundType.NULL\_DECREMENT);

if (\_c < 0)

{

SpecialRound(SpecialRoundType.DECREMENT);

\_c = \_maxC;

}

}

}

}

## Date.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab5

{

public class Date : Triad

{

public Date(int a, int b, int c) : base(31, 12, 10000, a, b, c)

{

}

protected override void SpecialRound(SpecialRoundType specialRound)

{

int a = Get(TriadNumber.FIRST);

int b = Get(TriadNumber.SECOND);

int c = Get(TriadNumber.THIRD);

switch (specialRound)

{

case (SpecialRoundType.DECREMENT):

if (a >= 31)

switch (b)

{

case 2:

if (c % 4 == 0 && (c % 100 != 0 || c % 100 == 0))

{

a = 29;

}

else

{

a = 28;

}

break;

case 4:

case 6:

case 9:

case 11:

a = 30;

break;

}

break;

case (SpecialRoundType.NULL\_DECREMENT):

if (a <= 0)

{

a = -1;

}

if (b <= 0)

{

b = -1;

}

if (c <= 0)

{

c = -1;

}

break;

case (SpecialRoundType.INCREMENT):

switch (b)

{

case 2:

if (a == 29 && !(c % 4 == 0 && (c % 100 != 0 || c % 100 == 0)))

{

a += 1;

}

break;

case 4:

case 6:

case 9:

case 11:

if (a == 30)

{

(a) += 2;

}

break;

}

break;

case (SpecialRoundType.NULL\_INCREMENT):

if (a <= 0)

{

a = 1;

}

if (b <= 0)

{

b = 1;

}

if (c <= 0)

{

c = 1;

}

break;

default:

throw new ArgumentOutOfRangeException(nameof(specialRound), specialRound, null);

}

Set(TriadNumber.FIRST, a);

Set(TriadNumber.SECOND, b);

Set(TriadNumber.THIRD, c);

}

}

}

## Time.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab5

{

public class Time : Triad

{

public Time(int a, int b, int c) : base(59, 59, 23, a, b, c)

{

}

}

}

## Pair.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab5

{

class Pair

{

private Triad \_key = null;

private Triad \_value = null;

public Pair() {}

public Pair(Triad key, Triad value)

{

\_key = key;

\_value = value;

}

public Triad GetKey()

{

return \_key;

}

public Triad GetValue()

{

return \_value;

}

public void SetKey(Triad key)

{

\_key = key;

}

public void SetValue(Triad value)

{

\_value = value;

}

}

}

## Memories.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab5

{

class Memories

{

private readonly Pair[] \_pairs;

private readonly int \_maxSize;

public Memories(int size)

{

\_maxSize = size;

\_pairs = new Pair[size];

}

public Pair GetPair(int index)

{

return \_pairs[index];

}

public void SetPair(int index, Pair pair)

{

\_pairs[index] = pair;

}

public void RemovePair(int index)

{

\_pairs[index] = null;

}

public int GetSize()

{

return \_maxSize;

}

}

}

## Interactor.cs

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Text;

namespace Lab5

{

class Interactor

{

private Memories \_memories;

private bool \_runningFlag = true;

private int \_type;

private void Menu()

{

Console.WriteLine("1 - Create \"Memories\"");

Console.WriteLine("2 - Create \"Pare\" and put into \"Memories\"");

Console.WriteLine("3 - Increment");

Console.WriteLine("4 - Decrement");

Console.WriteLine("0 - Exit");

}

private void Execute(int choice)

{

switch (choice)

{

case 1:

{

Console.WriteLine("Size: ");

var size = Convert.ToInt32(Console.ReadLine());

\_memories = new Memories(size);

break;

}

case 2:

{

if (\_memories != null)

{

Console.WriteLine("Date, DD-MM-YYYY:");

Console.WriteLine("Day:");

var a = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Month:");

var b = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Year:");

var c = Convert.ToInt32(Console.ReadLine());

var date = new Date(a, b, c);

Console.WriteLine("Time, HH-MM-SS:");

Console.WriteLine("Hours:");

c = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Minutes:");

b = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Seconds:");

a = Convert.ToInt32(Console.ReadLine());

var time = new Time(a, b, c);

var p = new Pair(date, time);

Console.WriteLine("Index - ");

var index = Convert.ToInt32(Console.ReadLine());

\_memories.SetPair(index, p);

Console.WriteLine("Successful!");

}

else

{

Console.WriteLine("Memories == null!");

}

break;

}

case 3:

WriteTriadPair(true);

break;

case 4:

WriteTriadPair(false);

break;

case 0:

\_runningFlag = false;

break;

default:

break;

}

}

private void WriteTriadPair(bool incrementFlag)

{

if (\_memories != null)

{

Console.WriteLine("Index:");

var index = Convert.ToInt32(Console.ReadLine());

if (\_memories.GetPair(index) == null)

{

Console.WriteLine("Pair with this index is null!");

return;

}

Console.WriteLine("1 - Date, 2 - Time");

\_type = Convert.ToInt32(Console.ReadLine());

switch (\_type)

{

case 1:

Console.WriteLine("1 - Day, 2 - Month, 3 - Year");

break;

case 2:

Console.WriteLine("1 - Seconds, 2 - Minutes, 3 - Hours");

break;

default:

Console.WriteLine("Incorrect type!");

return;

}

var position = Convert.ToInt32(Console.ReadLine());

TriadNumber enumPosition;

switch (position)

{

case 1:

enumPosition = TriadNumber.FIRST;

break;

case 2:

enumPosition = TriadNumber.SECOND;

break;

case 3:

enumPosition = TriadNumber.THIRD;

break;

default:

Console.WriteLine("Incorrect position!");

return;

}

if (incrementFlag)

{

if (\_type == 1)

{

\_memories.GetPair(index).GetKey().IncrementAndGet(enumPosition);

}

else

{

\_memories.GetPair(index).GetValue().IncrementAndGet(enumPosition);

}

}

else

{

if (\_type == 1)

{

\_memories.GetPair(index).GetKey().DecrementAndGet(enumPosition);

}

else

{

\_memories.GetPair(index).GetValue().DecrementAndGet(enumPosition);

}

}

}

else

{

Console.WriteLine("Memories == null!");

}

}

private void PrintData()

{

if (\_memories == null)

{

Console.WriteLine("Memories == NULL");

}

else

{

for (var i = 0; i < \_memories.GetSize(); i++)

{

var p = \_memories.GetPair(i);

if (p != null)

{

var t1 = p.GetKey();

var t2 = p.GetValue();

if (t1 != null && t2 != null)

{

Console.Write(i + ") ");

Console.Write(t1.Get(TriadNumber.FIRST).ToString("D2") + "-" + t1.Get(TriadNumber.SECOND).ToString("D2") + "-" + t1.Get(TriadNumber.THIRD).ToString("D2") + " ");

Console.Write(t2.Get(TriadNumber.THIRD).ToString("D2") + "-" + t2.Get(TriadNumber.SECOND).ToString("D2") + "-" + t2.Get(TriadNumber.FIRST).ToString("D2") + "\n");

}

else

{

Console.WriteLine(i + ") NULL");

}

}

else

{

Console.WriteLine(i + ") NULL");

}

}

}

}

public void Start()

{

while (\_runningFlag)

{

PrintData();

Console.WriteLine();

Menu();

Console.WriteLine();

try

{

var choice = Convert.ToInt32(Console.ReadLine());

Execute(choice);

Console.WriteLine();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

}

## SpecialRoundType.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab5

{

public enum SpecialRoundType

{

INCREMENT, DECREMENT, NULL\_INCREMENT, NULL\_DECREMENT

}

}

## TriadNumber.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab5

{

public enum TriadNumber

{

FIRST, SECOND, THIRD

}

}

## Programs.cs

using System;

namespace Lab5

{

class Program

{

static void Main()

{

var interactor = new Interactor();

interactor.Start();

}

}

}