Пермский филиал федерального государственного автономного образовательного учреждения высшего

 “Национальный исследовательский университет

“Высшая школа экономики”

*Факультет экономики, менеджмента и бизнес-информатики*

Дмитриев Арсений Алексеевич

Лабораторная работа №9. «Классы и объекты»

*Отчет по практической работе*

студента образовательной программы «Программная инженерия»

по направлению подготовки 09.03.04 Программная инженерия

руководитель

к. тех. н., доцент каф.

инф. техн. в биз.

Викентьева О.Л.

Пермь, 2020 г.

Постановка задачи

Общая

Часть 1.

1. Реализовать (в отдельном файле) определение нового класса (закрытые атрибуты, свойства, конструкторы, инициализация и вывод атрибутов).
2. Для демонстрации работы с объектами написать главную функцию, в которой создаются объекты класса и выводится информация, которая содержится в атрибутах.
3. Написать функцию, реализующую указанное в варианте действие. Рассмотреть два варианта:
4. статическую функцию;
5. метод класса.

В основной функции продемонстрировать работу функции.

1. Используя статическую компоненту класса подсчитать количество созданных в программе объектов.

Часть 2.

1. Добавить к реализованному классу указанные в варианте перегруженные операции.
2. Написать демонстрационную программу, в которой создаются объекты пользовательских классов и выполняются указанные операции.

Часть 3.

1. Реализовать класс (в отдельном файле), полем которого является одномерный массив из элементов заданного в варианте типа. В классе реализовать

* конструктор без параметров,
* конструктор с параметрами, заполняющий элементы случайными значениями,
* конструктор с параметрами, позволяющий заполнить массив элементами, заданными пользователем с клавиатуры,
* индексатор (для доступа к элементам массива),
* метод для просмотра элементов массива.

1. Написать демонстрационную программу, позволяющую создать массив разными способами и распечатать элементы массива. Подсчитать количество созданных объектов.
2. Выполнить указанное в варианте задание (если необходимо, перегрузить нужные для выполнения задачи операции или функции).

Вариант 7

Часть 1.

|  |  |  |
| --- | --- | --- |
| Название класса | Атрибуты | Методы |
| Time | int hours, int minutes | Вычитание переменной типа Time. Результат должен быть типа Time. |

Часть 2.

|  |  |
| --- | --- |
| Название класса | Методы |
| Time | Унарные операции:  ++ добавление минуты к объекту типа Time.  -- вычитание минуты из объекта типа Time.  Операции приведения типа:  int (неявная) – результатом является количество минут (время переводится в минуты);  bool (явная) – результатом является true, если минуты и секунды не равны нулю и false в противном случае.  Бинарные операции:  < Time t1, Time t2 – время переводится в минуты, результатом является true, если количество минут в левом операнде меньше, чем количество минут в правом операнде и false – в противном случае.  > Time t1, Time t2 - время переводится в минуты, результатом является true, если количество минут в левом операнде больше, чем количество минут в правом операнде и false – в противном случае. |

Часть 3.

|  |  |
| --- | --- |
| Название класса | Задание |
| TimeArray | Максимальное значение |

Диаграмма классов

Изображение выглядит как снимок экрана

Автоматически созданное описание

Листинг программы

1. Class Time

using System;

namespace Lab9

{

public class Time

{

// Part 1

private int hours;

private int minutes;

private static int count = 0;

private int negHoursCheck = 0;

public int Hours

{

get { return hours; }

private set

{

if (value < 0)

{

hours = 0;

minutes = 0;

negHoursCheck = value;

}

else hours = value;

}

}

public int Minutes

{

get { return minutes; }

private set

{

if (value < 0)

{

value = -value;

if (value < hours \* 60)

{

hours -= (value - 1) / 60 + 1;

minutes = 60 - value % 60;

}

else

{

minutes = 0;

hours = 0;

}

}

else

{

minutes = value;

if (negHoursCheck < 0)

{

minutes -= Math.Abs(negHoursCheck) \* 60;

}

hours += minutes / 60;

minutes %= 60;

}

negHoursCheck = 0;

}

}

public int Count

{

get { return count; }

private set { count = value; }

}

public Time() : this(0, 0) { }

public Time(int hours, int minutes)

{

Hours = hours;

Minutes = minutes;

Count++;

}

public void Show()

{

Console.WriteLine($"{Hours} ч {Minutes} мин.");

}

public static Time DeductTime(Time t1, Time t2)

{

Time tmp = new Time();

tmp.Hours = t1.Hours - t2.Hours;

tmp.Minutes = t1.Minutes - t2.Minutes;

return tmp;

}

public Time DeductTime(Time t)

{

Time tmp = new Time(this.hours - t.hours, this.minutes - t.minutes);

return tmp;

}

public override string ToString()

{

return $"{Hours} ч {Minutes} мин.";

}

// Part 2

public static Time operator ++(Time t)

{

t.Minutes++;

return t;

}

public static Time operator --(Time t)

{

t.Minutes--;

return t;

}

public static implicit operator int(Time t)

{

return t.Hours \* 60 + t.Minutes;

}

public static explicit operator bool(Time t)

{

return (t.hours != 0 && t.minutes != 0);

}

public static bool operator <(Time left, Time right)

{

return (int)left < (int)right;

}

public static bool operator >(Time left, Time right)

{

return (int)left > (int)right;

}

// Test Required

public override bool Equals(Object obj)

{

// Проверка на null и соответствие типов

if (obj == null || !this.GetType().Equals(obj.GetType())) return false;

else

{

Time time = (Time)obj;

return (minutes == time.minutes && hours == time.hours);

}

}

public override int GetHashCode()

{

return hours \* 60 + minutes;

}

}

}

1. Class TimeArray

using System;

namespace Lab9

{

public class TimeArray

{

//Part 3

private Time[] arr;

private int length;

public int Length

{

get { return length; }

private set {

if (value > 0) length = value;

else length = 0;

}

}

public Time this[int index]

{

get {

if (length > 0 && (index < 0 || index >= length) || length == 0) throw new IndexOutOfRangeException();

return arr[index];

}

set

{

if (length > 0 && (index < 0 || index >= length) || length == 0) throw new IndexOutOfRangeException();

arr[index] = value;

}

}

public TimeArray()

{

Length = 0;

arr = new Time[0];

}

public TimeArray(int size)

{

Length = size;

arr = new Time[size];

Random rand = new Random();

for (int i = 0; i < Length; i++)

{

arr[i] = new Time(rand.Next(0, 24), rand.Next(0, 60));

}

}

public TimeArray(int size, Time[] times)

{

Length = size;

arr = new Time[size];

for (int i = 0; i < Length; i++)

{

arr[i] = times[i];

}

}

public void Show()

{

foreach (Time t in arr) t.Show();

}

public Time MaxValue()

{

Time max = arr[0];

for (int i = 1; i < Length; i++)

{

if (arr[i] > max) max = arr[i];

}

return max;

}

}

}

1. Class Program

using System;

namespace Lab9

{

class Program

{

public static void Part1()

{

int h1 = int.Parse(Console.ReadLine()), m1 = int.Parse(Console.ReadLine());

int h2 = int.Parse(Console.ReadLine()), m2 = int.Parse(Console.ReadLine());

Time t1 = new Time(h1, m1);

Time t2 = new Time(h2, m2);

Time t3 = Time.DeductTime(t1, t2);

Time t4 = t1.DeductTime(t2);

Console.WriteLine(t3.ToString());

Console.WriteLine(t4.ToString());

}

public static void Part2()

{

int h1 = int.Parse(Console.ReadLine()), m1 = int.Parse(Console.ReadLine());

int h2 = int.Parse(Console.ReadLine()), m2 = int.Parse(Console.ReadLine());

Time t1 = new Time(h1, m1);

Time t2 = new Time(h2, m2);

t1--;

t2++;

t1.Show();

t2.Show();

Console.WriteLine((bool)t1);

Console.WriteLine($"{t1 - 10}");

if (t1 > t2) Console.WriteLine($"{t1.ToString()} > {t2.ToString()}");

else Console.WriteLine($"{t2.ToString()} >= {t1.ToString()}");

}

public static void Part3()

{

TimeArray array = new TimeArray(10);

array.Show();

Console.WriteLine();

array = new TimeArray();

array.Show();

Console.WriteLine();

int N = 5;

Time[] arr = new Time[N];

for(int i = 0; i<=N; i++)

{

int h = int.Parse(Console.ReadLine()), m = int.Parse(Console.ReadLine());

try

{

arr[i] = new Time(h, m);

}

catch (IndexOutOfRangeException)

{

Console.WriteLine("Индекс вне диапазона");

}

}

array = new TimeArray(arr.Length, arr);

Time max = array[0];

for (int i = 0; i < array.Length; i++) if (array[i] > max) max = array[i];

array.MaxValue().Show();

max.Show();

}

static void Main(string[] args)

{

Part1();

//Part2();

//Part3();

}

}

}

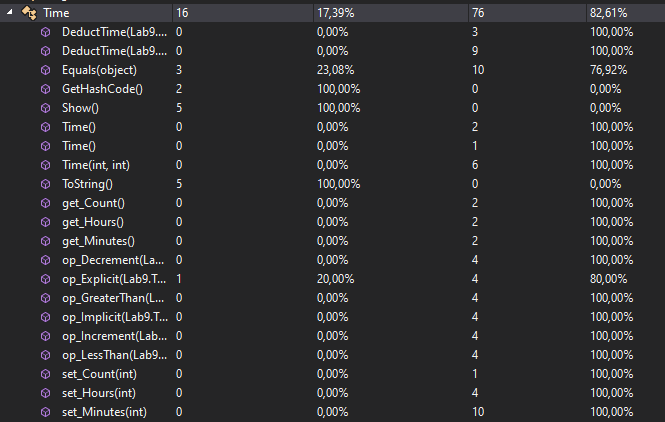
Тестирование

Общее покрытие:

Изображение выглядит как внутренний

Автоматически созданное описание

Покрытие Time:



Покрытие TimeArray:

Изображение выглядит как черный, фотография, компьютер, дорога

Автоматически созданное описание

Листинг тестов

using Lab9;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace UnitTestProject

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

{

// Arrange

Time expected = new Time(0, 29);

// Act

Time t1 = new Time(5, 3);

Time t2 = new Time(4, 34);

Time actual = t1.DeductTime(t2);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod2()

{

// Arrange

Time expected = new Time(0, 0);

// Act

Time t1 = new Time(3, 12);

Time t2 = new Time(5, 80);

Time actual = Time.DeductTime(t1, t2);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod3()

{

// Arrange

Time expected = new Time(0, 20);

// Act

Time actual = new Time(-1, 80);

//Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod4()

{

// Arrange

Time expected = new Time(5, 0);

// Act

Time actual = new Time(5, 0);

actual--;

actual++;

//Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod5()

{

// Arrange

Time expected = new Time(0, 1);

// Act

Time t1 = new Time(0, 0);

Time actual;

if ((bool)t1) actual = t1;

else actual = new Time(0, 1);

//Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod6()

{

// Arrange

Time expected = new Time(0,0);

// Act

TimeArray array = new TimeArray(10);

array[0] = new Time(0, 0);

Time min = array[0];

for (int i = 0; i < array.Length; i++) if (array[i] < min) min = array[i];

// Assert

Assert.AreEqual(expected, min);

}

[TestMethod]

public void TestMethod7()

{

// Arrange

Time expected = new Time(16, 30);

// Act

Time[] tmp = new Time[]

{

new Time(0, 5),

new Time(3, 19),

new Time(10, 14),

expected,

new Time(9, 49)

};

TimeArray array = new TimeArray(5, tmp);

Time actual = array.MaxValue();

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod8()

{

// Arrange

bool expected = false;

// Act

TimeArray array = new TimeArray();

bool actual;

try

{

array[0] = new Time();

actual = true;

}

catch (IndexOutOfRangeException)

{

actual = false;

}

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod9()

{

// Arrange

bool expected = false;

// Act

TimeArray array = new TimeArray(3);

bool actual;

try

{

Time time = array[3];

time.Show();

actual = true;

}

catch (IndexOutOfRangeException)

{

actual = false;

}

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod10()

{

// Arrange

bool expected = false;

// Act

TimeArray array = new TimeArray();

bool actual;

try

{

Time time = array[0];

time.Show();

actual = true;

}

catch (IndexOutOfRangeException)

{

actual = false;

}

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMethod11()

{

// Arrange

bool expected = false;

// Act

TimeArray array = new TimeArray(5);

bool actual;

try

{

array[5] = new Time();

actual = true;

}

catch (IndexOutOfRangeException)

{

actual = false;

}

// Assert

Assert.AreEqual(expected, actual);

}

}

}