Пермский филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский университет «Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Шульжик Кирилл Владимирович

**Отчёт по практической работе №9 (Вариант 1)**

студента образовательной программы бакалавриата «Программная инженерия» по направлению подготовки 09.03.04 Программная инженерия

Преподаватель

В.Д Марквирер

**Оглавление**

[Формулировка задания 3](#_Toc95146961)

[Часть 1 3](#_Toc95146962)

[Часть 2 3](#_Toc95146963)

[Часть 3 4](#_Toc95146964)

[Диаграмма классов 6](#_Toc95146965)

[Код программы 7](#_Toc95146966)

[Код класса Program 7](#_Toc95146967)

[Код класса Time 7](#_Toc95146968)

[Код класса TimeArray 7](#_Toc95146969)

[Код unit-тестов 8](#_Toc95146970)

[Анализ покрытия кода 9](#_Toc95146971)

[ПРИЛОЖЕНИЕ А *Код класса Program* 10](#_Toc95146972)

[ПРИЛОЖЕНИЕ Б *Код класса Time* 12](#_Toc95146973)

[ПРИЛОЖЕНИЕ В Код класса *TimeArray* 15](#_Toc95146974)

[Приложение Г *Код unit-тестов* 17](#_Toc95146975)

Формулировка задания

Часть 1

Задание 1 формулируется следующим образом:

1. Реализовать (в отдельном файле) определение нового класса/типа данных (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:
   1. статическую функцию;
   2. метод класса.
4. В основной функции продемонстрировать работу функции.
5. Используя статическую компоненту класса подсчитать количество созданных в программе объектов.

Таблица 1 – Формулировка 1 задания

| № | Название класса | Атрибуты (поля) | Методы |
| --- | --- | --- | --- |
| 1 | Time | int hours,  int minutes | Добавление минут (int) к объекту типа Time (учесть, что минут не может быть больше 59).  Результат должен быть типа Time. |

В таблице 1 представлена формулировка 1 задания.

Часть 2

Задание 2 формулируется следующим образом:

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

Таблица 2 – Формулировка 2 задания

| № | Название класса | Методы |
| --- | --- | --- |
| 1 | Time | Унарные операции:  ++ добавление минуты к объекту типа Time (учесть, что минут не может быть больше 59).  -- вычитание минуты из объекта типа Time (учесть, что часов и минут не может быть меньше 0).  Операции приведения типа:  int (явная) – результатом является количество часов (минуты отбрасываются);  bool (неявная) – результатом является true, если часы и минуты не равны нулю и false в противном случае.  Бинарные операции:  + Time t, целое число (минуты) (лево- и правосторонние операции). Результат должен быть типа Time.  – Time t, целое число (минуты) (лево- и право- сторонние операции). Результат должен быть типа Time. |

В таблице 2 представлена формулировка 2 задания.

Часть 3

Задание 3 формулируется следующим образом:

1. Реализовать класс-коллекцию (в отдельном файле), полем которого является одномерный массив (не использовать стандартные коллекции C#!) из элементов заданного в варианте типа. Например, для класса Fraction нужно создать класс FractionArray следующим образом:

class FractionArray

{

Fraction[] arr;//одномерный массив элементов типа Fraction

int size;

. . . .

}

В классе реализовать:

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

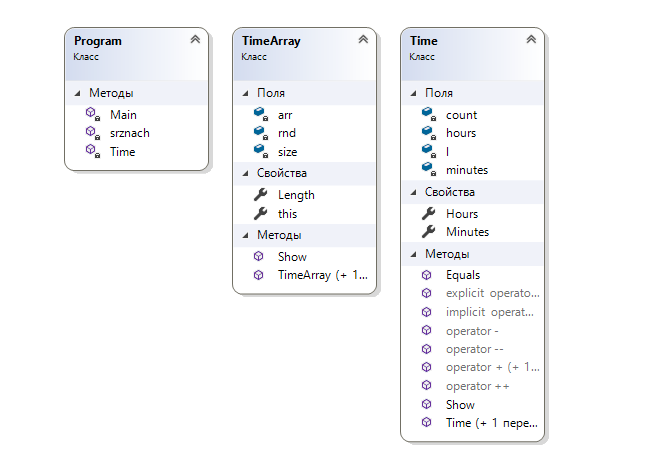
1. Реализовать индексатор для доступа к элементам коллекции. Предусмотреть проверку при выходе индекса за пределы массива.
2. Написать демонстрационную программу, позволяющую создать массив разными способами и распечатать элементы массива. Подсчитать количество созданных объектов.
3. Написать функцию в классе Program для выполнения указанного в варианте задания (использовать индексатор и, если необходимо, перегрузить нужные для выполнения задачи операции).

Таблица 3 – Формулировка 3 задания

| № | Название класса | Задание |
| --- | --- | --- |
| 1 | Time | Среднее арифметическое. |

В таблице 3 представлена формулировка 3 задания.

Диаграмма классов

****Диаграмма классов представлена на рисунке 1

**Рисунок 1 – Диаграмма классов**

Код программы

Код класса Program

См. [Приложение А](#Приложение)

Код класса Time

См. [Приложение Б](#Приложение2)

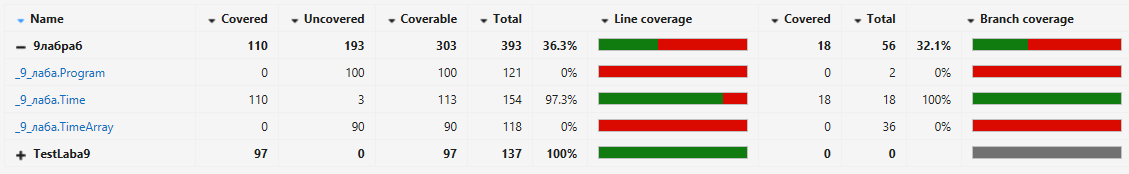
**Код класса TimeArray**

См. [Приложение В](#Приложение2)

Код unit-тестов

См. [Приложение Г](#Приложение3)

Анализ покрытия кода

Анализ покрытия кода представлен на рисунке 2

**Рисунок 2 – Анализ покрытия кода**

ПРИЛОЖЕНИЕ А

**Код класса Program**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_9\_лаба

{

class Program

{

static int srznach(TimeArray k)

{

int count = 0;

int summa = 0;

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

{

summa += k[i].Hours;

summa += k[i].Minutes;

count += 1;

}

return summa / count;

}

static void Main(string[] args)

{

Time f = new Time(1, 50);

f.Show();

Time f1 = new Time();

f1.Show();

Time f2 = new Time(-4, 50);

f2.Show();

Time f3 = new Time(-7, -5);

f3.Show();

Time f4 = new Time(7, -444);

f4.Show();

Time f5 = new Time(0, 100);

Time f7 = new Time(1, 10);

Console.WriteLine("Добавление минут с помощью метода класса (30 минут). Было:");

f7.Show();

f7.Add(30);

Console.WriteLine("Стало:");

f7.Show();

Console.WriteLine("Добавление минут с помощью статического метода (30 минут). Стало:");

Time.Add(f7, 30);

f7.Show();

Console.WriteLine("Постфиксный инкремент. Было:");

f5.Show();

f5++;

Console.WriteLine("Стало: ");

f5.Show();

Time f6 = new Time(0, 59);

Console.WriteLine("Постфиксный инкремент. Было:");

f6.Show();

f6++;

Console.WriteLine("Стало:");

f6.Show();

Console.WriteLine("Постфиксный декремент. Было:");

f4.Show();

f4--;

Console.WriteLine("Постфиксный декремент. Стало:");

f4.Show();

Console.WriteLine("Постфиксный декремент. Было:");

f5.Show();

f5--;

Console.WriteLine("Постфиксный декремент. Стало:");

f5.Show();

Console.WriteLine("Перегрузка бинарной операции + (31 минута). Было:");

f6.Show();

f6 += 31;

Console.WriteLine("Стало: ");

f6.Show();

Console.WriteLine("Прибавление ещё 30 минут:");

f6 = 30 + f6;

f6.Show();

Console.WriteLine("Перегрузка бинарной операции - (30 минут)");

f6 = f6 - 30;

f6.Show();

Console.WriteLine("Перегрузка бинарной операции - (100 минут). Было:");

f5.Show();

f5 -= 100;

Console.WriteLine("Стало: ");

f5.Show();

Console.WriteLine("Перегрузка бинарной операции - (200 минут). Было:");

f.Show();

f -= 200;

Console.WriteLine("\nСтало:");

f.Show();

Console.WriteLine("Приведение к типу int. Было:");

f4.Show();

int y = (int)f4;

Console.WriteLine("Стало:\n" + y);

Console.WriteLine("\nПриведение к типу bool. Было:");

f5++;

f5.Show();

bool y1 = f5;

Console.WriteLine("Стало:\n" + y1 + "\n");

y1 = f6;

Console.WriteLine("Было:");

f6.Show();

Console.WriteLine("Стало:\n" + y1 + "\n");

TimeArray a = new TimeArray(4);

a.Show();

Console.WriteLine("1 элемент:");

a[0].Show();

Console.WriteLine("5 элемент:");

try

{

a[4].Show();

}

catch (ArgumentException)

{

Console.WriteLine("\nНеправильно задан индекс");

}

}

}

}

ПРИЛОЖЕНИЕ Б

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_9\_лаба

{

public class Time

{

int hours, minutes;

static bool isMinusOverload = false;

static int count = 0;

public int Hours

{

get { return hours; }

set

{

if (value < 0)

{

Console.WriteLine("Ошибка. Отрицательное кол-во часов");

hours = 0;

}

else

{

hours = value;

}

}

}

public int Minutes

{

get { return minutes; }

set

{

if (value < 0)

{

if (!isMinusOverload)

{

Console.WriteLine("Ошибка. Отрицательное кол-во минут\n");

minutes = 0;

}

else

{

if (hours != 0)

{

hours -= 1;

minutes = 59;

isMinusOverload = false;

}

else

{

Console.WriteLine("Ошибка. Отрицательное время\n");

isMinusOverload = false;

}

}

}

else if (value > 59)

{

minutes = value % 60;

hours += value / 60;

}

else

{

minutes = value;

}

}

}

public Time()

{

Hours = 0;

Minutes = 0;

count += 2;

}

public Time(int r, int k)

{

Hours = r;

Minutes = k;

count += 2;

}

public void Show()

{

Console.WriteLine($"часов: {Hours}, минут: {Minutes}\nСоздано объектов: {count}\n");

}

public static Time operator ++(Time f)

{

f.Minutes++;

return f;

}

public static Time operator --(Time f)

{

isMinusOverload = true;

f.Minutes--;

return f;

}

public static explicit operator int(Time f)

{

return f.Hours;

}

public static implicit operator bool(Time f)

{

return f.Hours != 0 && f.Minutes != 0;

}

public static Time operator +(int min, Time f)

{

Time temp = new Time();

temp.Hours = f.Hours;

temp.Minutes = f.Minutes + min;

return temp;

}

public static Time operator +(Time f, int min)

{

Time temp = new Time();

temp.Hours = f.Hours;

temp.Minutes = f.Minutes + min;

return temp;

}

public static Time operator -(Time f, int min)

{

Time temp = new Time();

if (f.Minutes < min)

{

if (f.Hours \* 60 + f.Minutes < min)

{

Console.WriteLine("Ошибка. Отрицательное время");

}

else

{

temp.Hours = (f.Hours \* 60 + f.Minutes - min) / 60;

temp.Minutes = (f.Hours \* 60 + f.Minutes - min) % 60;

}

}

else

{

temp.Hours = f.Hours;

temp.Minutes = f.Minutes - min;

}

return temp;

}

public static void Add(Time time, int moneyAmount)

{

time.Minutes += moneyAmount;

}

public void Add(int moneyAmount)

{

this.Minutes += moneyAmount;

}

public override bool Equals(object obj)

{

Time f = (Time)obj;

return this.Hours == f.Hours && this.Minutes == f.Minutes;

}

}

}

ПРИЛОЖЕНИЕ В

**Код класса TimeArray**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_9\_лаба

{

public class TimeArray

{

Time[] arr;

int size;

static Random rnd = new Random();

public TimeArray(params Time[] list)

{

arr = new Time[list.Length];

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

{

arr[i] = list[i];

}

}

public void Show()

{

if (arr.Length == 0)

{

Console.WriteLine("Пустой массив");

}

else

{

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

{

arr[i].Show();

}

}

}

public int Length

{

get { return arr.Length; }

}

public TimeArray(int size)

{

int count = 1; int hours; int minutes;

Console.WriteLine($"1. Заполнить массив длины {size} случайными числами"

+ $"\n2. Заполнить массив длины {size} с клавиатуры");

string a = Console.ReadLine();

while (a != "1" && a != "2")

{

Console.WriteLine("Введите 1 или 2");

a = Console.ReadLine();

}

if (a == "1")

{

arr = new Time[size];

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

{

Time f = new Time(rnd.Next(0, 100), rnd.Next(0, 60));

arr[i] = f;

}

}

else

{

arr = new Time[size];

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

{

Console.WriteLine($"Введите {count} кол-во часов");

a = Console.ReadLine();

while (!int.TryParse(a, out hours) || int.Parse(a) < 0)

{

if (!int.TryParse(a, out hours))

{

Console.WriteLine("Введите целое число!");

}

else

{

Console.WriteLine("Введите положительное целое число!");

}

a = Console.ReadLine();

}

hours = int.Parse(a);

Console.WriteLine($"Введите {count} кол-во минут");

a = Console.ReadLine();

while (!int.TryParse(a, out minutes) || int.Parse(a) < 0)

{

if (!int.TryParse(a, out minutes))

{

Console.WriteLine("Введите целое число!");

}

else

{

Console.WriteLine("Введите положительное целое число!");

}

a = Console.ReadLine();

}

minutes = int.Parse(a);

count += 1;

Time f = new Time(hours, minutes);

arr[i] = f;

}

}

}

public Time this[int index]

{

get

{

if (index >= 0 && index < arr.Length)

return arr[index];

throw new ArgumentException();

}

set

{

if (index >= 0 && index < arr.Length)

arr[index] = value;

else Console.WriteLine("Выход за границу массива");

}

}

}

}

Приложение Г

**Код unit-тестов**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using \_9\_лаба;

namespace TestLaba9

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

{

Time f1 = new Time(0, 0);

Time f2 = new Time();

Assert.AreEqual(f1, f2);

}

[TestMethod]

public void TestMethod2()

{

Time f1 = new Time(0, 0);

Time f2 = new Time(-6, 0);

Assert.AreEqual(f1, f2);

}

[TestMethod]

public void TestMethod3()

{

Time f2 = new Time(0, 59);

Time f1 = new Time(1, 0);

f1--;

Assert.AreEqual(f1, f2);

}

[TestMethod]

public void TestMethod4()

{

Time f2 = new Time(0, 0);

Time f1 = new Time(0, 0);

f1--;

Assert.AreEqual(f1, f2);

}

[TestMethod]

public void TestMethod5()

{

Time f2 = new Time(0, 0);

Time f1 = new Time(0, -7);

Assert.AreEqual(f1, f2);

}

[TestMethod]

public void TestMethod6()

{

Time f1 = new Time(1, 51);

Time f2 = new Time(1, 50);

f2++;

Assert.AreEqual(f1, f2);

}

[TestMethod]

public void TestMethod7()

{

Time f1 = new Time(1, 0);

int y1 = 1;

Time f2 = new Time(1, 51);

int y = (int)f2;

Assert.AreEqual(y, y1);

}

[TestMethod]

public void Explicit()

{

bool z = true;

Time f1 = new Time(1, 7);

Assert.AreEqual(z, f1);

bool z1 = false;

Time f2 = new Time(0, 0);

Assert.AreEqual(z1, f2);

}

[TestMethod]

public void TimeSum()

{

Time f1 = new Time(1, 15);

Time f2 = new Time(1, 0);

f2 += 15;

Assert.AreEqual(f1, f2);

Time f3 = new Time(1, 15);

Time f4 = new Time(1, 0);

f4 = 15 + f4;

Assert.AreEqual(f3, f4);

}

[TestMethod]

public void TimeMinus()

{

Time f1 = new Time(1, 3);

Time f2 = new Time(1, 15);

f2 = f2 - 12;

Assert.AreEqual(f1, f2);

Time f3 = new Time(0, 0);

Time f4 = new Time(1, 15);

f4 -= 85;

Assert.AreEqual(f3, f4);

Time f5 = new Time(4, 40);

Time f6 = new Time(6, 20);

f6 -= 100;

Assert.AreEqual(f5, f6);

}

[TestMethod]

public void Length()

{

}

[TestMethod]

public void MinutesOver()

{

Time f = new Time(1, 40);

Time f1 = new Time(0, 100);

Assert.AreEqual(f, f1);

}

[TestMethod]

public void StaticAdd()

{

Time f = new Time(1, 41);

Time f1 = new Time(1, 40);

Time.Add(f1, 1);

Assert.AreEqual(f, f1);

}

[TestMethod]

public void NotStaticAdd()

{

Time f = new Time(1, 41);

Time f1 = new Time(1, 40);

f1.Add(1);

Assert.AreEqual(f, f1);

}

[TestMethod]

public void AreNotEqualsCheck()

{

Time f = new Time(1, 41);

Time f1 = new Time(1, 40);

f1.Add(56);

Assert.AreNotEqual(f, f1);

}

}

}