МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Автоматизированных систем управления

Отчет по курсовой работе по дисциплине «программирование»

**Разработка системы для составления** **расписания сессии**

Выполнил: Проверил:

Студент гр. АВТ-813 к.т.н., доцент кафедры

Пустовских Дмитрий Александрович Достовалов Д.Н.

Новосибирск

2019

Оглавление

[Постановка задачи 2](#_Toc28022802)

[Описание классов 3](#_Toc28022803)

[Класс Audience 4](#_Toc28022804)

[Класс Consultation 4](#_Toc28022805)

[Класс Date 5](#_Toc28022806)

[Класс Directions 5](#_Toc28022807)

[Класс Discipline 5](#_Toc28022808)

[Класс Exam 6](#_Toc28022809)

[Класс Group 6](#_Toc28022810)

[Класс ILesson 6](#_Toc28022811)

[Класс Session 7](#_Toc28022812)

[Класс Syllabus 7](#_Toc28022813)

[Класс Teacher 7](#_Toc28022814)

[Класс Time 8](#_Toc28022815)

[Взаимосвязь классов 8](#_Toc28022816)

[Описание приложения 8](#_Toc28022817)

[Тестирование 8](#_Toc28022818)

[Заключение 8](#_Toc28022819)

[Источники 8](#_Toc28022820)

[Приложение 8](#_Toc28022821)

# Постановка задачи

Необходимо разработать приложение, которое поможет составить расписание сессии. Работа программы должна начаться с заполнения базы данных: преподавателей, аудитории, потоков, кафедры. Данные о количестве экзаменов будут взяты из учебного плана. По необходимости реализовать возможность добавления экзамена не из учебного плана, но при создании необходимо будет указать кафедру.

Расписание сессии будет выводится в виде таблицы. При добавлении экзамена необходимо будет указать группу, исходя из нее будет формироваться список экзаменов, преподавателей. Также надо будет указать аудиторию и время начала экзамена. В аудитории ограниченное количество мест, поэтому если группа не будет входить в аудиторию, нужно составить список аудиторий, в которые вместится группа. Необходимо реализовать проверку: занятости аудитории в указанные время, занятость преподавателя (или при выборе исключить варианты, которые нам не подходят). Надо учитывать то, что в один день у преподавателя не должно стоять больше 1 экзамена. Максимально у преподавателя может быть 1 консультация и 1 экзамен.

Составляя расписание, необходимо исключить ситуации, когда у группы в один день ставят несколько экзаменов. Промежуток между экзаменами и консультацией должен быть не менее 4-х часов в 1 день.

Учебный план будет хранится в Json-файле. Также необходимо чтобы приложение могло сохранять расписание в файл (для дальнейшей печати).

Приложение должно быть лояльным к ошибкам пользователя, поэтому необходимо при ошибки указать пользователю что пошло не так (реализовать обработчик ошибок).

# Описание классов

## Класс Audience:

Класс представляет из себя данные о аудитории. В данном классе хранится информация о количестве мест в аудитории и ее номере.

Данный класс является одной из составляющей Класса Экзамен и Консультация.

## Класс Consultation

Класс с реализованным интерфейсом ILesson. Хранит в себе данные о консультации. В данный класс входят такие поля:

* Аудитория
* Дата консультации
* Предмет
* Преподавателя
* Время начала

Также переопределен метод getName и свойства Audience, Date, NameLesson, Teacher и Time.

Объекты данного класса являются элементами класса Session.

## Класс Date:

Класс: Дата. Хранит в себе день и месяц. При создании объекта проверяет на правильность ввода данных (на соответствие календарным дням)

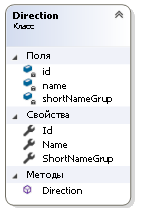
Функция-утилита check проверяет на правильность данных.

Переопределенный метод ToString преобразует дату в сток вида :

” day. month”

Перегружены операторы сравнения.

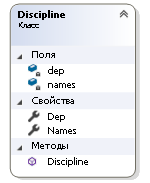
## Класс Directions



Класс представляет из себя Направление. Хранит в себе ID направления, полное название и короткое имя для группы.

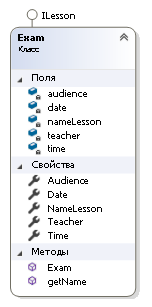
Является элементом объекта Группа, и также по данному направлению можно найти информацию о предметах в учебном плане.

## Класс Discipline

Класс представляет из себя Предмет. Хранит в себе кафедру, полное название предмета.

Является элементом объекта Экзамен и Консультация, и также хранится в учебном плане.

## Класс Exam

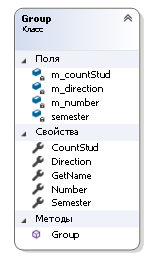
Класс с реализованным интерфейсом ILesson. Хранит в себе данные о экзамене. В данный класс входят такие поля:

* Аудитория
* Дата консультации
* Предмет
* Преподавателя
* Время начала

Также переопределен метод getName и свойства Audience, Date, NameLesson, Teacher и Time.

Объекты данного класса являются элементами класса Session.

## Класс Group



Класс, хранящий в себе информацию о Группе. В данном классе имеется информация о:

* Количестве студентов в группе
* Направление
* Номер
* Семестр

Метод GetName возвращает имя группы: АВТ-813:

Где АВТ это короткое наименование направления, а 813 номер группы.

## Класс ILesson

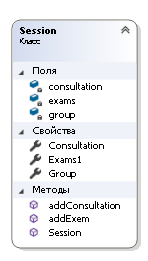
Интерфейс класса Экзамен и Консультация.

Данный интерфейс представляет реализует работу и взаимодействие с данными имеющие отношение к сессии(через данный интерфейс также можно реализовать работу с зачётными дисциплинами ).

В нем имеется виртуальный метод GetName, виртуальные свойства:

Audience, Date, NameLesson, Teacher и Time

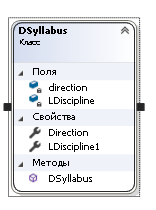
## Класс Session



Данный класс хранит в себе информацию о Группе и экзаменах которые она будет сдавать.

В данном классе содержится список экзаменов. А также список консультаций. Разбитие на 2 списка обосновывается тем что для записи в файл json необходимо четко знать какой тип у объекта (то есть json не может работать с абстрактными классами)

## Класс Syllabus

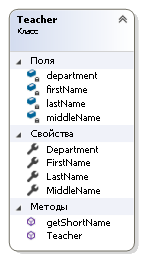


Данный класс представляет из себя учебный план. Хранит в себе направление и экзаменационные предметы.

Поля:

* Направление
* Список предметы

## Класс Teacher

Представляет из себя класс учителя.

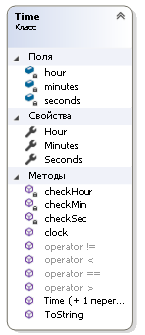
Хранит в поля :

* Имя
* Фамилию
* Отчество
* Кафедра

Является частью класса Экзамен и Консультация.

Метод getShortName возвращает строку, где указана Фамилия инициалы преподавателя.

## Класс Time



Класс: Время. Хранит в себе минуты, часы и секунды. При создании объекта проверяет на правильность ввода данных.

Функция-утилита checkHour, checkMinr, checkSec проверяет на правильность данных.

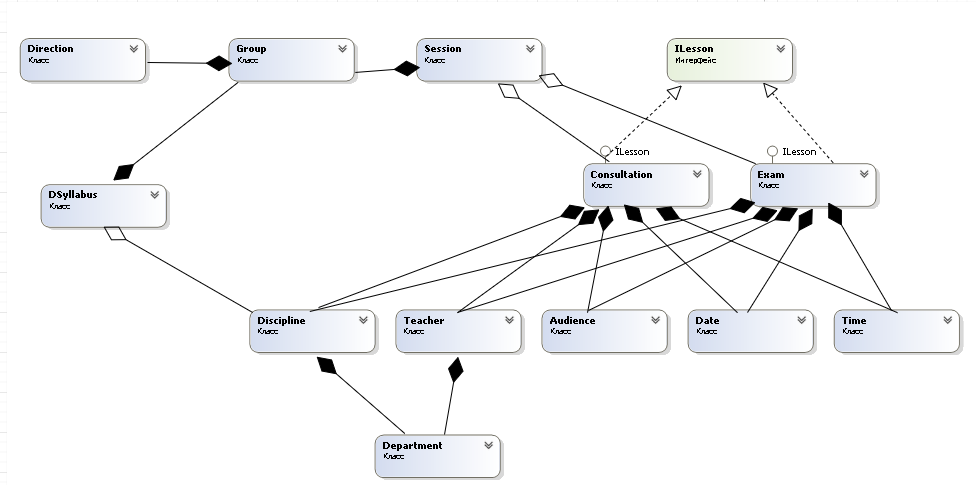
Переопределенный метод ToString преобразует дату в сток вида :

” 14.:00”

Перегружены операторы сравнения.

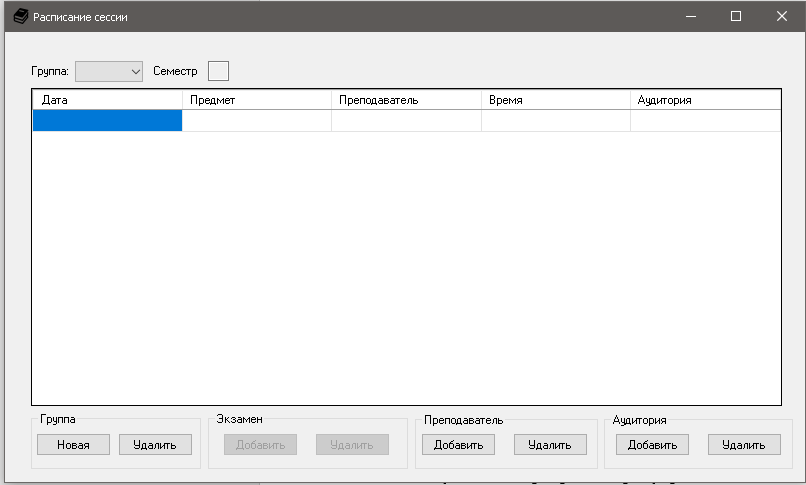
# Взаимосвязь классов

Ниже представлена UML-диаграмма иерархии классов

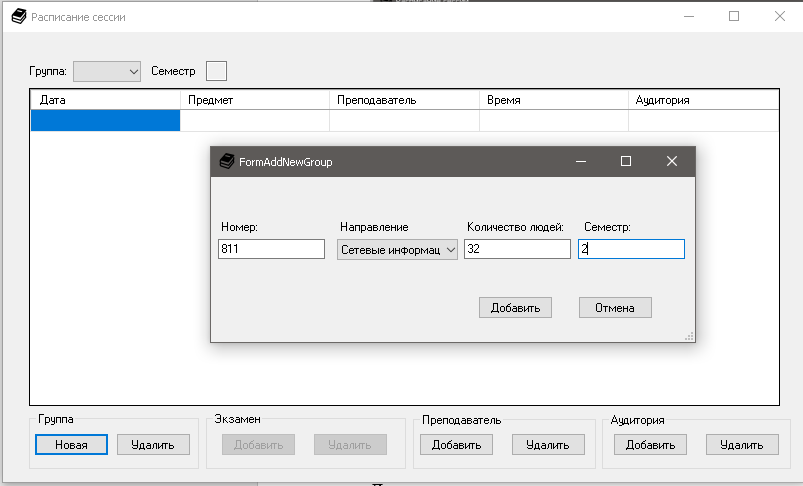


# Описание приложения

При запуске программы запускается стартовое окно, где пользователь может взаимодействовать с данными расписания сессии.

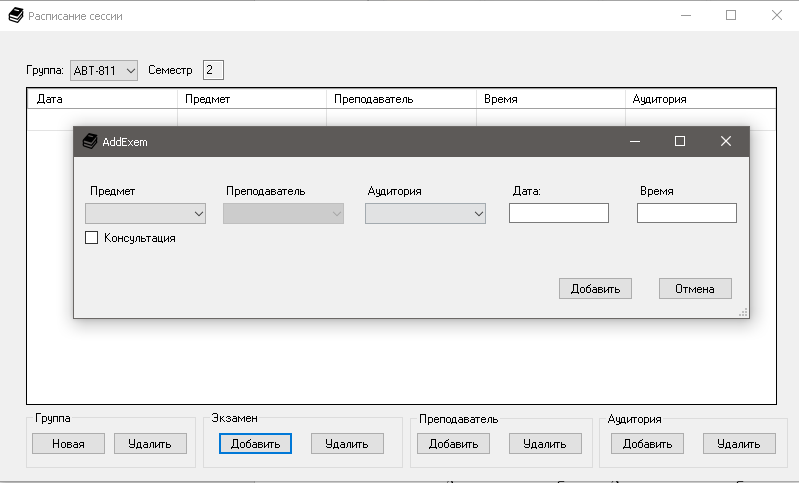


Прежде всего в начале работы программы необходимо создать группу. При создании группы выдается вспомогательное окно куда пользователь может ввести данные о группе: номер группы, количество студентов в группе, семестр, и направление.

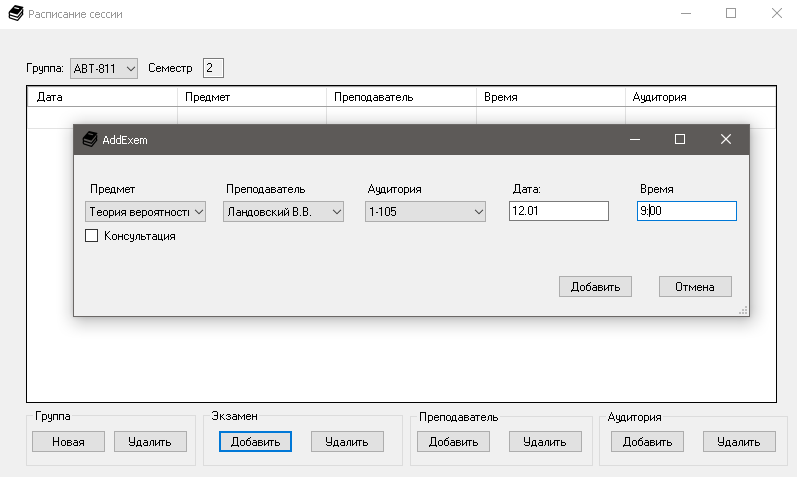


Дальше при нажатии на кнопку довить происходит проверка введенных данных, если данные верны, то вспомогательное окно закроется и пользователю будет доступен выбор группы. Если же данные оказались неверными, то программа сообщает об ошибке.

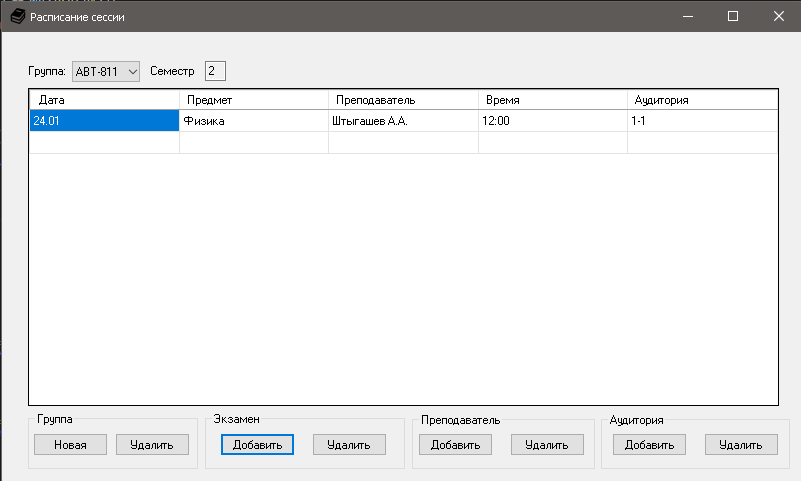
Дальше необходимо выбрать группу, и именно после этого будет доступна возможность добавление экзаменов.



При добавлении экзамена тоже выводится вспомогательное окно. В данном окне есть 3 поля для выбора и 2 для ввода: выбор предмета, выпор преподавателя и выбор аудитории (данные формируются относительно выбранной группы, преподаватели при выборе предмета, а аудитории относительно вместимости). Также если произойдёт наложение экзаменов или консультаций программа предупредит об этом. В поля для ввода необходима ввести дату и время проведения экзамена. Формат ввода для времент” 12:00”, а для даты “день: месяц». Если ввод произведен неверно, то выведется сообщение об ошибке.



При нажатии на кнопку добавить, в основное окно в таблицу добавиться информация об экзамене.



Также есть возможность удалять Группы, Экзамены, Преподавателей, и Аудитории.

# Тестирование

Для тестирования был использован инструмент отладки в среде разработки Visual Studio. Во время отладки были испытаны все возможные действия пользователя в приложении.

В ходе тестирования были найдены несколько фатальных исходов. Но в следствии того что они были обнаружены они были исправлены и при дальнейшем тестировании не возникали. Также была обнаружена ошибка при добавлении новой группы, а в дальнейшем добавлении экзамена, при нажатии на кнопу добавить ничего не происходило. При закрытии этого окна и повторного ввода экзамена, экзамен был успешно добавлен. Этот баг не был исправлен связи с тем, что при дальнейшем тестировании данный баг больше не появлялся.

# Заключение

В процессе работы было изучено программирование с использованием Windows Forms и Json и использованы основные концепции объектно-ориентированного подхода в языке C#. Несмотря на то, что на данный момент приложение работает исправно, в нём оставлено много возможностей для дальнейшего развития. В дальнейшем можно добавить работу с базами данных, также можно разработать алгоритм автоматического заполнения расписания. Добавить гибкое окно. И реализовать возможность конвертировать расписание из Excel.

# Источники

* Шилдт, Герберт. Ш57 C# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. — 1056 с.: ил. — Парал. тит. Англ
* Сайт абота с JSON Сериализация в JSON. JsonSerializerРежим доступа: <https://metanit.com/sharp/tutorial/6.5.php>
* C# 7.0. Карманный справочник.Авторы - Джозеф Албахари, Бен Албахари

# Приложение

* Sourse-master.zip

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Audience

{

private int CountSeats;

private string number;

public Audience(int countSeats, string number)

{

CountSeats = countSeats;

this.number = number;

}

public int CountSeats1 { get => CountSeats; set => CountSeats = value; }

public string Number { get => number; set => number = value; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Consultation : ILesson

{

private Date date;

private Discipline nameLesson;

private Teacher teacher;

private Time time;

private Audience audience;

public Consultation(Date date, Discipline nameLesson, Teacher teacher, Time time, Audience audience)

{

this.date = date;

this.nameLesson = nameLesson;

this.teacher = teacher;

this.time = time;

this.audience = audience;

}

public Date Date { get => date; set => date = value; }

public Discipline NameLesson { get => nameLesson; set => nameLesson = value; }

public Teacher Teacher { get => teacher; set => teacher = value; }

public Time Time { get => time; set => time = value; }

public Audience Audience { get => audience; set => audience = value; }

public string getName()

{

return nameLesson.Names + ". Консультация";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Date

{

private ushort dey ;

private ushort month ;

private static ushort[] calendar = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 } ;

public ushort Dey { get => dey; set => dey = value; }

public ushort Month { get => month; set => month = value; }

public override string ToString()

{

if (month > 9)

return (dey.ToString() + "." + month.ToString());

else

return (dey.ToString() + ".0" + month.ToString());

}

public Date(ushort dey, ushort month)

{

if (check(dey, month))

{

this.dey = dey;

this.month = month;

}

else

throw new IndexOutOfRangeException();

}

public void setDate(ushort dey, ushort month)

{

if (check(dey, month))

{

this.dey = dey;

this.month = month;

}

else

throw new IndexOutOfRangeException();

}

private bool check(ushort dey, ushort month)

{

if (month > 12)

return false;

else

if (dey > calendar[month - 1])

return false;

return true;

}

public static bool operator<(Date tis , Date other)

{

if (tis.month < other.month)

return true;

else if(tis.month == other.month)

{

if (tis.dey <= other.dey)

return true;

else

return false;

}

return false;

}

public static bool operator >(Date tis, Date other)

{

if (tis.month > other.month)

return true;

else if (tis.month == other.month)

{

if (tis.dey > other.dey)

return true;

else

return false;

}

return false;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Department

{

public String Name

{

get

{

return name;

}

set

{

name = value;

}

}

public UInt32 Id

{

get

{

return id;

}

set

{

id = value;

}

}

public Department(string name, uint id)

{

this.name = name;

this.id = id;

}

private String name;

private UInt32 id;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Direction

{

public String Name

{

get

{

return name;

}

set

{

name = value;

}

}

public UInt32 Id

{

get

{

return id;

}

set

{

id = value;

}

}

public string ShortNameGrup { get => shortNameGrup; set => shortNameGrup = value; }

public Direction(string shortNameGrup, string name, uint id)

{

this.shortNameGrup = shortNameGrup;

this.name = name;

this.id = id;

}

private String shortNameGrup;

private String name;

private UInt32 id;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Discipline

{

private string names;

private Department dep;

public Discipline(string names, Department dep)

{

this.names = names;

this.dep = dep;

}

public string Names { get => names; set => names = value; }

public Department Dep { get => dep; set => dep = value; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Exam : ILesson

{

private Date date;

private Discipline nameLesson;

private Teacher teacher;

private Time time;

private Audience audience;

public Exam(Date date, Discipline nameLesson, Teacher teacher, Time time, Audience audience)

{

this.date = date;

this.nameLesson = nameLesson;

this.teacher = teacher;

this.time = time;

this.audience = audience;

}

public Date Date { get => date; set => date = value; }

public Discipline NameLesson { get => nameLesson; set => nameLesson = value; }

public Teacher Teacher { get => teacher; set => teacher = value; }

public Time Time { get => time; set => time = value; }

public Audience Audience { get => audience; set => audience = value; }

public string getName()

{

return nameLesson.Names ;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Group

{

public Group(UInt32 countStud, Direction direction, short m\_number, short semester)

{

this.m\_countStud = countStud;

this.m\_number = m\_number;

this.m\_direction = direction;

this.semester = semester;

}

public UInt32 CountStud

{

get

{

return m\_countStud;

}

set

{

m\_countStud = value;

}

}

public Direction Direction

{

get

{

return m\_direction;

}

set

{

m\_direction = value;

}

}

public short Number {

get

{

return m\_number;

}

set

{

m\_number = value;

}

}

public string GetName

{

get

{

return m\_direction.ShortNameGrup + "-" + m\_number;

}

}

public short Semester { get => semester; set => semester = value; }

private UInt32 m\_countStud;

private Direction m\_direction;

private short m\_number;

private short semester;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public interface ILesson

{

Date Date { get; set; }

Discipline NameLesson { get; set; }

Teacher Teacher { get; set; }

Time Time { get; set; }

Audience Audience { get; set; }

string getName();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Session

{

private List<Exam> exams = new List<Exam>();

private List<Consultation> consultation = new List<Consultation>();

private Group group;

public Session(Group group)

{

this.group = group;

}

public Group Group { get => group; set => group = value; }

public List<Exam> Exams1 { get => exams; set => exams = value; }

public List<Consultation> Consultation { get => consultation; set => consultation = value; }

public void addExem(Exam exem)

{

exams.Add(exem);

}

public void addConsultation(Consultation exem)

{

consultation.Add(exem);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class DSyllabus

{

private Direction direction;

private List<Discipline> LDiscipline = new List<Discipline>();

public DSyllabus(Direction direction, List<Discipline> lDiscipline)

{

this.direction = direction;

LDiscipline = lDiscipline;

}

public Direction Direction { get => direction; set => direction = value; }

public List<Discipline> LDiscipline1 { get => LDiscipline; set => LDiscipline = value; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Teacher

{

private string firstName;

private string lastName;

private string middleName;

private Department department;

public Teacher(string lastName , string firstName, string middleName, Department department)

{

this.FirstName = firstName;

this.lastName = lastName;

this.middleName = middleName;

this.department = department;

}

public string getShortName()

{

return (lastName + " " + FirstName[0] + "." + middleName[0] + ".");

}

public string FirstName { get => firstName; set => firstName = value; }

public string LastName { get => lastName; set => lastName = value; }

public string MiddleName { get => middleName; set => middleName = value; }

public Department Department { get => department; set => department = value; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SsessionWriting.Class

{

public class Time

{

private ushort minutes;

private ushort hour;

private ushort seconds;

public Time(ushort minutes, ushort hour, ushort seconds = 0)

{

this.Minutes = minutes;

this.Hour = hour;

this.Seconds = seconds;

}

public Time()

{

Seconds = 0; ;

hour = 0;

}

public ushort Minutes

{

get => minutes;

set

{

if (checkMin(value))

minutes = value;

else

throw new IndexOutOfRangeException();

}

}

public ushort Hour

{

get => hour;

set

{

if (checkHour(value))

hour = value;

else

throw new IndexOutOfRangeException();

}

}

public ushort Seconds

{

get => seconds;

set

{

if (checkSec(value))

seconds = value;

else

throw new IndexOutOfRangeException();

}

}

public Time clock(ushort min, ushort hous)

{

Time tim = new Time();

if (this.hour >= hous)

tim.hour = (ushort)(this.hour - hous);

else

tim.hour = (ushort)(24 - (hous - this.hour));

if (this.minutes >= min)

tim.minutes = (ushort)(this.minutes - min);

else

{

tim.minutes = (ushort)(60 - (min - this.minutes));

tim = tim.clock(0, 1);

}

return tim;

}

public static bool operator<(Time th,Time other)

{

if (th.hour < other.hour)

return true;

else if (th.hour == other.hour)

{

if (th.minutes <= other.minutes)

return true;

else

return false;

}

else

return false;

}

public static bool operator>(Time th,Time other)

{

if (th.hour > other.hour)

return true;

else if (th.hour == other.hour)

{

if (th.minutes >= other.minutes)

return true;

else

return false;

}

else

return false;

}

public static bool operator==(Time th, Time other)

{

if (th.hour == other.hour && th.minutes == other.minutes)

return true;

else

return false;

}

public static bool operator !=(Time th, Time other)

{

return !(th == other);

}

public override string ToString()

{

return (hour.ToString() + ":" + ((minutes > 9) ? minutes.ToString() : "0" + minutes.ToString())) ;

}

private bool checkMin(ushort minutes)

{

return !(minutes < 0 || minutes > 60);

}

private bool checkHour(ushort hour)

{

return !(hour < 0 || hour > 24);

}

private bool checkSec(ushort seconds)

{

return !(seconds < 0 || seconds > 60);

}

}

}