|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Ассоциации между классами. Распределение обязанностей»**

**ДИСЦИПЛИНА: «Объектно-ориентированное программирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Былинка М.И. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2022

**Цель:** формирование практических навыков реализации объектно-ориентированной модели и распределения обязанностей между объектами модели.

**Задачи:**

1. Изучить интерфейс класса List<T> (конструкторы, свойства, методы)
2. Изучить часто используемые классы библиотеки FCL
3. Получить навыки реализации зависимостей между классами в объектно-ориентированной модели

**Вариант №7**

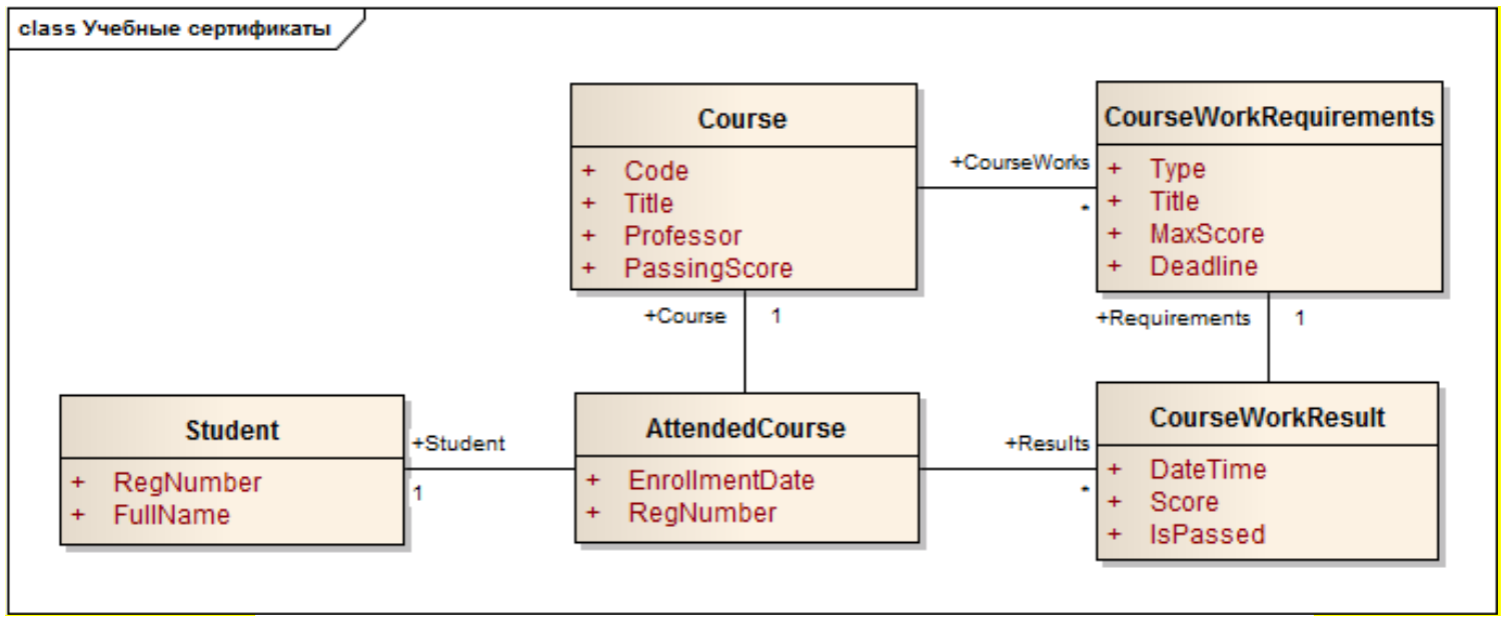


Рис. 1. Учебные сертификаты

В ходе обучения студенты выполняют разные виды работ: рубежные тесты, домашние задания, сдают экзамены. За каждую успешно сданную работу студент получает рейтинговые баллы. После завершения курса студент получает сертификат, если сумма рейтинговых баллов по всем выполненным баллам равна или превышает балл, заданный профессором для курса. При подсчете баллов учитываются следующие правила:

– успешно принятые работы не могут пересдаваться (с целью повышения балла);

– если работа сдана не в срок, то при подсчете учитываются только 75% набранных за нее баллов (округление вниз).

Сущность Student представляет студента и содержит атрибуты «ФИО» (FullName) и регистрационный номер в Университете (RegNumber).

Предоставляемый для изучения курс моделируется сущностью Course, имеющей следующие атрибуты: шифр (Code), наименование (Title), ФИО профессора (Professor), балл, который необходимо набрать для получения сертификата о прохождении курса (PassingScore).

Требования к работе, которую необходимо сдать в ходе обучения, описываются сущностью CourseWorkRequirements с атрибутами: тип работы (Type, рубежный тест/домашнее задание/экзамен), название (Title), максимальный балл, который можно получить за работу (MaxScore), срок сдачи (Deadline).

Факт выбора студентом для изучения курса моделируется сущностью AttendedCourse, хранящей дату зачисления студента на курс (EnrollmentDate) и регистрационный номер студента на курсе (RegNumber).

Результат сдачи студентом работы представляется сущностью CourseWorkResult. Она содержит следующие атрибуты: дата/время сдачи (DateTime), признак, успешно ли сдана работа (IsPassed) и набранный рейтинговый балл.

Необходимо реализовать классы Student, Course, CourseWorkRequirements, AttendedCourse, CourseWorkResult. Реализованная модель должна позволять:

– добавлять информацию о требованиях к работам по курсу;

– добавлять информацию о результатах сдачи работ студентами;

– определять результат изучения студентом курса (получает сертификат или нет).

**Листинг:**

***Program.cs***

using System;

using System.Collections.Generic;

namespace LW\_2

{

public class Student

{

public Student(uint regNumber, string fullName)

{

RegNumber = regNumber;

FullName = fullName;

}

public uint RegNumber { get; }

public string FullName { get; }

}

public class AttendedCourse

{

public AttendedCourse(DateTimeOffset enrollmentDate, Student student, Course course)

{

EnrollmentDate = enrollmentDate;

Student = student;

Course = course;

Results = new List<CourseWorkResult>();

}

public void CreateResult(DateTimeOffset dateTime, bool isPassed, ushort score, CourseWorkRequirements work)

{

foreach (CourseWorkResult result in Results)

{

if (result.Requirements == work)

{

if (result.IsPassed)

{

return;

}

}

}

Results.Add(new CourseWorkResult(dateTime, isPassed, score, work));

}

public bool IsPassed()

{

ushort sum = 0;

foreach (CourseWorkResult result in Results)

{

if (result.IsPassed)

{

sum += result.Score;

}

}

return sum >= Course.PassingScore;

}

public DateTimeOffset EnrollmentDate { get; }

public uint RegNumber { get { return Student.RegNumber; } }

public Student Student { get; }

public Course Course { get; }

public List<CourseWorkResult> Results { get; }

}

public class Course

{

public Course(uint code, string title, string professor, ushort passingScore)

{

Code = code;

Title = title;

Professor = professor;

PassingScore = passingScore;

CourseWorks = new List<CourseWorkRequirements>();

}

public void CreateWork(CourseWorkRequirements.WorkType type, string title, ushort maxScore, DateTimeOffset deadline)

{

CourseWorks.Add(new CourseWorkRequirements(type, title, maxScore, deadline));

}

public uint Code { get; }

public string Title { get; }

public string Professor { get; }

public ushort PassingScore { get; }

public List<CourseWorkRequirements> CourseWorks { get; }

}

public class CourseWorkRequirements

{

public CourseWorkRequirements(WorkType type, string title, ushort maxScore, DateTimeOffset deadline)

{

Type = type;

Title = title;

MaxScore = maxScore;

Deadline = deadline;

}

public enum WorkType

{

Test,

Homework,

Exam

}

public WorkType Type { get; }

public string Title { get; }

public ushort MaxScore { get; }

public DateTimeOffset Deadline { get; }

}

public class CourseWorkResult

{

public CourseWorkResult(DateTimeOffset dateTime, bool isPassed, ushort score, CourseWorkRequirements requirements)

{

DateTime = dateTime;

IsPassed = isPassed;

if (dateTime > requirements.Deadline)

{

Score = (ushort)(score \* 0.75);

}

else

{

Score = score;

}

if (Score > requirements.MaxScore)

{

Score = requirements.MaxScore;

}

Requirements = requirements;

}

public DateTimeOffset DateTime { get; }

public bool IsPassed { get; }

public ushort Score { get; }

public CourseWorkRequirements Requirements { get; }

}

public class LW\_2

{

public static void Main() { }

}

}

***UnitTest.cs***

using Microsoft.VisualStudio.TestTools.UnitTesting;

using LW\_2;

using System;

namespace Testing

{

[TestClass]

public class Testing

{

[TestMethod]

public void TestWorkCreating()

{

Course course = new Course(123, "Math", "Ramazanov A.K.", 70);

course.CreateWork(CourseWorkRequirements.WorkType.Homework, "DZ", 20, DateTimeOffset.Parse("06/03/2022"));

Assert.AreEqual(1, course.CourseWorks.Count);

Assert.AreEqual("DZ", course.CourseWorks[0].Title);

}

[TestMethod]

public void TestResultCreating()

{

Student student = new Student(111, "Ivanov I.I.");

Course course = new Course(123, "Math", "Ramazanov A.K.", 70);

course.CreateWork(CourseWorkRequirements.WorkType.Homework, "DZ", 20, DateTimeOffset.Parse("11/03/2022"));

AttendedCourse attendedCourse = new AttendedCourse(DateTimeOffset.Parse("01/03/2022"), student, course);

attendedCourse.CreateResult(DateTimeOffset.Parse("10/03/2022"), false, 5, course.CourseWorks[0]);

attendedCourse.CreateResult(DateTimeOffset.Parse("12/03/2022"), true, 18, course.CourseWorks[0]);

attendedCourse.CreateResult(DateTimeOffset.Parse("13/03/2022"), true, 20, course.CourseWorks[0]);

Assert.AreEqual(2, attendedCourse.Results.Count);

Assert.AreEqual(5, attendedCourse.Results[0].Score);

Assert.AreEqual(13, attendedCourse.Results[1].Score);

}

[TestMethod]

public void TestCoursePassing()

{

Student student = new Student(111, "Ivanov I.I.");

Course course = new Course(123, "Math", "Ramazanov A.K.", 70);

course.CreateWork(CourseWorkRequirements.WorkType.Homework, "DZ", 70, DateTimeOffset.Parse("11/03/2022"));

AttendedCourse attendedCourse = new AttendedCourse(DateTimeOffset.Parse("01/03/2022"), student, course);

Assert.AreEqual(false, attendedCourse.IsPassed());

attendedCourse.CreateResult(DateTimeOffset.Parse("10/03/2022"), true, 70, course.CourseWorks[0]);

Assert.AreEqual(true, attendedCourse.IsPassed());

}

}

}

**Результат:**

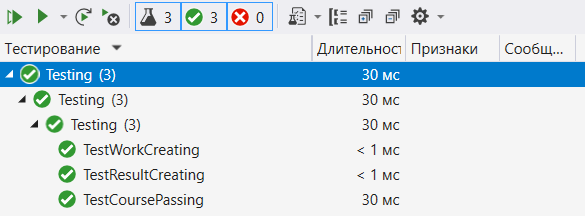


Рис. 1. Результат

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки реализации зависимостей и распределения обязанностей между различными классами, использования контейнеров.