Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №7

по дисциплине «Технологии программирования»

Тема: «Расширенные возможности C# на примере событийных механизмов»

Выполнил:

студент гр. ПРИ-123

А.Ц. Нямаа

Приняла:

Ст. пр. кафедры ИСПИ

В.В. Данилов

Владимир, 2024 г.

ЦЕЛЬ РАБОТЫ:

Познакомиться с возможностями языка программирования C# для организации событийно-ориентированного программирования на основе делегатов и событий

ЗАДАНИЕ 1(вариант 13).

Доработать демонстрационный пример из предыдущей лабораторной работы с учетом добавляемой событийности. Запустить доработанный демонстрационный пример, показывающий работу с событиями. При необходимости дополнить код.

ВЫПОЛНЕНИЕ РАБОТЫ:

Листинг:

namespace DelegateGropuAddressDemo

{

public delegate void RatingHandler(Group group);

enum RatingStatus { Passed, Notpassed };

class StudentGroupActions

{

//печать рейтинговой ведомости

public static void PrintRatingVedomost(Group group)

{

Console.WriteLine($"Рейтинговая ведомость группы {group.GroupTitle}");

for (int i = 0; i < group.GroupSize; i++)

{

Console.WriteLine($"Студент {i+1}. ФИО - {group[i].FIO}");

Console.Write("Оценки: ");

for (int j = 0; j < 4; j++)

{

Console.Write($"{group[i][j]} ");

}

Console.WriteLine();

}

}

//расчет рейтинга студентов

public static void CalculateRatings(Group group)

{

int summBalls;

for (int i = 0; i < group.GroupSize; i++)

{

summBalls = 0;

for (int j = 0; j < 4; j++)

{

summBalls += group[i][j];

}

if (summBalls >= (10 \* 4 / 2)) group[i].Rating = (int)RatingStatus.Passed;

else group[i].Rating = (int)RatingStatus.Notpassed;

}

}

//печать зачетной ведомости

public static void PrintRankingList(Group group)

{

Console.WriteLine($"\nЗачетная ведомость группы {group.GroupTitle}");

for (int i = 0; i < group.GroupSize; i++)

{

Console.Write($"Студент {i+1} - ");

Console.WriteLine((group[i].Rating == (int)RatingStatus.Passed) ? "ЗАЧТЕНО" : "НЕЗАЧТЕНО");

}

}

}

public class Emailer

{

public static void SendSessionNotifyRatingVedomost(Group group)

{

//отправить e-mail уведомление группе с рейтинговой / зачетной ведомостью

Console.WriteLine("\nНа email группы " + group.GroupTitle + " отправлена рейтинговая ведомость...");

}

public static void SendSessionNotifyRankingList(Group group)

{

//отправить e-mail уведомление группе с рейтинговой / зачетной ведомостью

Console.WriteLine("\nВ группу " + group.GroupTitle + " отправлена зачетная ведомость...");

}

}

public class SessionEvent

{

public event RatingHandler activate;

public void FixSemesterResults(Group group)

{

if (activate != null) activate(group);

}

}

public class Student

{

String fio;

public String FIO { set { fio = value; } get { return fio; } }

Byte[] balls;

public Byte this[int i] { get { return balls[i]; } set { balls[i] = value; } }

int rating;

public int Rating { get { return rating; } set { rating = value; } }

public Student(String FIO)

{

rating = 0;

balls = new Byte[4];

fio = FIO;

}

}

public class Group

{

int groupSize;

string groupTitle = "";

public string GroupTitle { get { return groupTitle; } }

public int GroupSize { get { return groupSize; } }

Student[] students;

public Student this[int i] { get { return students[i]; } }

public Group(string Title, int Size)

{

groupTitle = Title;

groupSize = Size;

students = new Student[Size];

}

public void InitRandom()

{

try

{

var rnd = new Random();

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

{

String randomString = "";

for (int j = 0; j < 5; j++)

randomString += (rnd.Next(255).ToString());

students[i] = new Student(randomString);

for (int j = 0; j < 4; j++) students[i][j] = (byte)rnd.Next(10);

}

}

catch (System.ArgumentOutOfRangeException e)

{

Console.WriteLine(e.Message);

}

}

}

class Program

{

static void Main()

{

Group ist110 = new Group("IST110", 10);

ist110.InitRandom();

SessionEvent sessionEvent = new SessionEvent();

sessionEvent.activate += new RatingHandler(StudentGroupActions.PrintRatingVedomost);

sessionEvent.activate += new RatingHandler(StudentGroupActions.CalculateRatings);

sessionEvent.activate += new RatingHandler(StudentGroupActions.PrintRankingList);

sessionEvent.activate += new RatingHandler(Emailer.SendSessionNotifyRatingVedomost);

sessionEvent.activate += new RatingHandler(Emailer.SendSessionNotifyRankingList);

sessionEvent.FixSemesterResults(ist110);

Console.ReadLine();

}

}

}

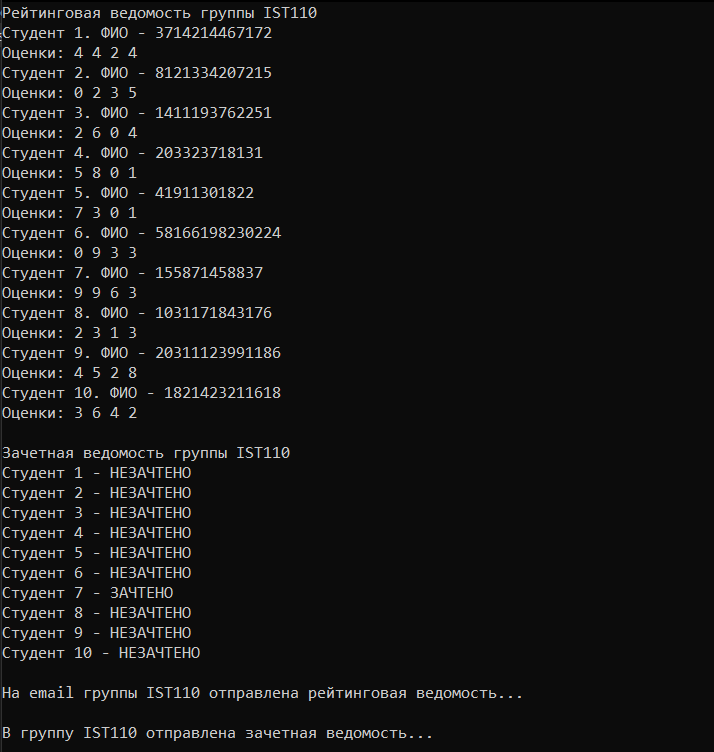


Рисунок 1. Выполнение программы.

ЗАДАНИЕ 2(вариант 13).

Реализовать программу на C# в соответствии с вариантом задания с использованием делегатов и событий.

ВЫПОЛНЕНИЕ РАБОТЫ:

Листинг Program.cs:

using lab\_7;

using System.Diagnostics;

namespace university

{

public delegate void Handler();

class Program

{

static event Handler Handler;

static List<Person> people = new List<Person>();

static List<IDoc> docs = new List<IDoc>();

static List<subjects> subjects = new List<subjects>();

static Stopwatch time = new Stopwatch();

static DateTime startTime = DateTime.Now;

static int[] quantityMethods = new int[3];

static void Main(string[] args)

{

Random rand = new Random();

time.Start();

while (true)

{

Console.WriteLine("\nВыберите действие:");

Console.WriteLine("Для ввода человека - 1");

Console.WriteLine("Для ввода документов - 2");

Console.WriteLine("Для ввода предметов - 3");

Console.WriteLine("Для вывода результатов - 4");

string operationString = Console.ReadLine();

if (!int.TryParse(operationString, out int operation) || operation < 0 || operation > 4)

{

Console.WriteLine("\nВведите корректное число");

continue;

}

if (operation == 1)

{

Console.WriteLine("\nВведите количество людей");

string countpersonsString = Console.ReadLine();

if (!int.TryParse(countpersonsString, out int countpersons) || countpersons < 0)

{

Console.WriteLine("\nВведено некорректное число");

continue;

}

Console.WriteLine("\nЧтобы выбрать профессора - введите 1. \nЧтобы выбрать студента - введите 2");

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

{

Console.WriteLine("\nКого хотите выбрать?");

string typepersonString = Console.ReadLine();

if (!int.TryParse(typepersonString, out int person) || person < 0 || person > 2)

{

Console.WriteLine("\nВведено некорректное число");

continue;

}

Console.WriteLine("\nВведите имя и фамилию");

string nm = Console.ReadLine();

string[] name = nm.Split(' ');

if (person == 1)

{

people.Add(new Proffesor { Id = people.Count + 1, Name = name[0], Surname = name[1], BirthDay = new DateOnly(rand.Next(1950, 2010), rand.Next(1, 12), rand.Next(1, 30)), workBook = rand.Next(100000, 999999) });

}

else if (person == 2)

{

people.Add(new Student { Id = people.Count + 1, Name = name[0], Surname = name[1], BirthDay = new DateOnly(rand.Next(1950, 2010), rand.Next(1, 12), rand.Next(1, 30)), recordBook = rand.Next(1000000, 9999999) });

}

quantityMethods[0]++;

}

}

if (operation == 2)

{

Console.WriteLine("\nСколько документов?");

string countdocsString = Console.ReadLine();

if (!int.TryParse(countdocsString, out int countdocs) || countdocs < 0)

{

Console.WriteLine("\nВведено некорректное число");

continue;

}

Console.WriteLine("\nЧтобы выбрать результаты экзамена - введите 1 \nЧтобы выбрать результаты зачета - введите 2 \nЧтобы выбрать информацию о ведомостях - введите 3");

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

{

Console.WriteLine("\nЧто хотите выбрать?");

string typedocsString = Console.ReadLine();

if (!int.TryParse(typedocsString, out int typedocs) || typedocs < 0 || typedocs > 3)

{

Console.WriteLine("\nВведено некорректное число");

continue;

}

Console.WriteLine("\nВведите группу");

string group = Console.ReadLine();

if (typedocs == 1)

{

docs.Add(new Exam { Id = docs.Count + 1, Document = "экзамена", StudentsQuant = rand.Next(1, 30), Group = group, Date = new DateOnly(rand.Next(2000, 2024), rand.Next(1, 12), rand.Next(1, 30)) });

}

else if (typedocs == 2)

{

docs.Add(new Test { Id = docs.Count + 1, Document = "зачета", StudentsQuant = rand.Next(1, 30), Group = group, Date = new DateOnly(rand.Next(2000, 2024), rand.Next(1, 12), rand.Next(1, 30)) });

}

else if (typedocs == 3)

{

docs.Add(new State { Id = docs.Count + 1, Document = "ведомости", StudentsQuant = rand.Next(1, 30), Group = group, Date = new DateOnly(rand.Next(2000, 2024), rand.Next(1, 12), rand.Next(1, 30)) });

}

quantityMethods[1]++;

}

}

if (operation == 3)

{

Console.WriteLine("\nВведите количество дисциплин");

string countsubjectString = Console.ReadLine();

if (!int.TryParse(countsubjectString, out int countsubject) || countsubject < 0)

{

Console.WriteLine("\nВведено некорректное число");

continue;

}

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

{

Console.WriteLine("\nВведите название предмета");

string subject\_name = Console.ReadLine();

subjects.Add(new subjects { Id = subjects.Count + 1, Subject = subject\_name });

quantityMethods[2]++;

}

}

if (operation == 4)

{

Handler = PrintPeople;

Handler += PrintDocs;

Handler += PrintSubjects;

Handler += AddInfo;

Handler();

}

}

}

static void PrintPeople()

{

Console.WriteLine("\nИнформация о людях:");

for (int i = 0; i < people.Count; i++)

{

if (people[i] != null && people[i] is Proffesor prof)

{

Console.WriteLine($"id преподавателя - {prof.Id}, его имя - {prof.Name}, фамилия - {prof.Surname}, родился {prof.BirthDay.ToShortDateString()}, номер трудовой книжки - {prof.workBook}");

}

else if (people[i] != null && people[i] is Student student)

{

Console.WriteLine($"id студента - {student.Id}, его имя - {student.Name}, фамилия - {student.Surname}, родился {student.BirthDay.ToShortDateString()}, номер зачетки - {student.recordBook}");

}

}

}

static void PrintDocs()

{

Console.WriteLine("\nИнформация о документах:");

for (int i = 0; i < docs.Count; i++)

{

if (docs[i] != null)

{

Console.WriteLine($"id документа - {docs[i].Id}, название документа - {docs[i].WhatADoc()}, необходим для {docs[i].Document}, группа - {docs[i].Group}, {docs[i].results()}, дата подписания документа {docs[i].Date.ToShortDateString()}");

}

}

}

static void PrintSubjects()

{

Console.WriteLine("\nИнформация о предметах:");

for (int i = 0; i < subjects.Count; i++)

{

Console.WriteLine($"id предмета - {subjects[i].Id}, предмет - {subjects[i].Subject}, {subjects[i].Check(subjects[i].Subject)}, преподаватель - {subjects[i].teach(subjects[i].Subject)}");

}

}

static void AddInfo()

{

Console.WriteLine("\nДополнительная информация:");

Console.WriteLine($"Время выполнения программы - {time.ElapsedMilliseconds}ms\nВремя запуска программы - {startTime.ToLongTimeString()}\nКоличество выполнения метода Persons - {quantityMethods[0]} раз\nКоличество выполнения метода IDoc - {quantityMethods[1]} раз\nКоличество выполнения метода Subjects - {quantityMethods[2]} раз\nОперационная система - {Environment.OSVersion}");

}

}

}

Листинг Person.cs:

namespace lab\_7

{

class Person

{

public int Id;

public string Name;

public string Surname;

public DateOnly BirthDay;

}

class Student : Person

{

public int recordBook;

}

class Proffesor : Person

{

public int workBook;

}

}

Листинг docs.cs:

namespace lab\_7

{

interface IDoc

{

int Id { get; set; }

string Document { get; set; }

string Group { get; set; }

DateOnly Date { get; set; }

int StudentsQuant { get; set; }

string WhatADoc();

string results();

}

class Exam : IDoc

{

public int Id { get; set; }

public string Document { get; set; }

public string Group { get; set; }

public DateOnly Date { get; set; }

public int StudentsQuant { get; set; }

public string WhatADoc()

{

return "экзаменационный документ";

}

public string results()

{

return $"количество сдавших экзамен - {StudentsQuant}";

}

}

class Test : IDoc

{

public int Id { get; set; }

public string Document { get; set; }

public string Group { get; set; }

public DateOnly Date { get; set; }

public int StudentsQuant { get; set; }

public string WhatADoc()

{

return "документ о зачетах";

}

public string results()

{

return $"количество сдавших зачет - {StudentsQuant}";

}

}

class State : IDoc

{

public int Id { get; set; }

public string Document { get; set; }

public string Group { get; set; }

public DateOnly Date { get; set; }

public int StudentsQuant { get; set; }

public string WhatADoc()

{

return "ведомость";

}

public string results()

{

return $"количество идеальных посещений - {StudentsQuant}";

}

}

}

Листинг subjects.cs:

namespace lab\_7

{

class subjects

{

private string[] checker = { "Math", "Program", "Eng", "Chem" };

private string[] teachers = { "Иванов", "Петров", "Сидоров", "Зеленов" };

Random rand = new Random();

public int Id;

public string Subject;

public string teach(string Subject)

{

if (checker.Contains(Subject))

{

return $"{teachers[rand.Next(0, 3)]}";

}

else

{

return " такого предмета нет, то и преподавателя тоже нет";

}

}

public string Check(string Subject)

{

if (checker.Contains(Subject))

{

return "такой предмет существует";

}

else

{

return "такого предмета не существует";

}

}

}

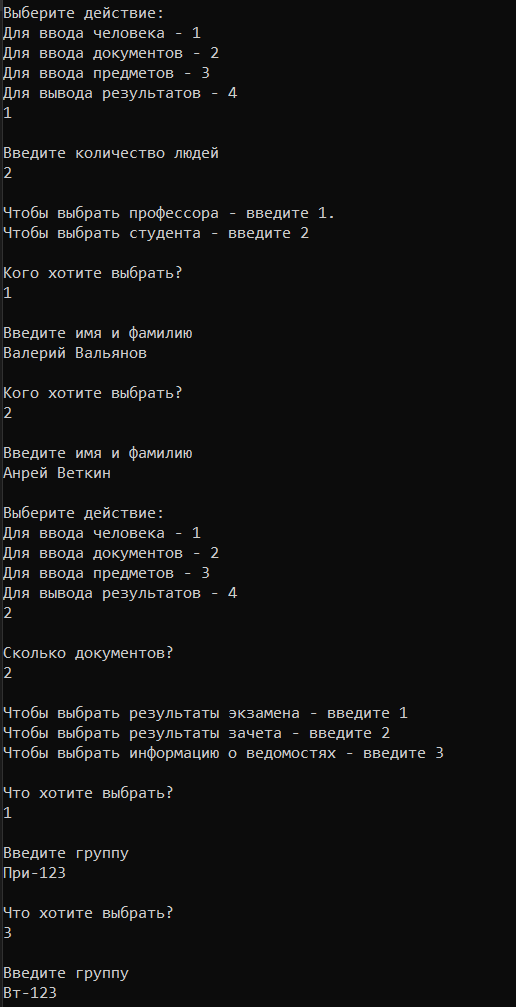


Рисунок 2. Выполнение программы. Часть 1.

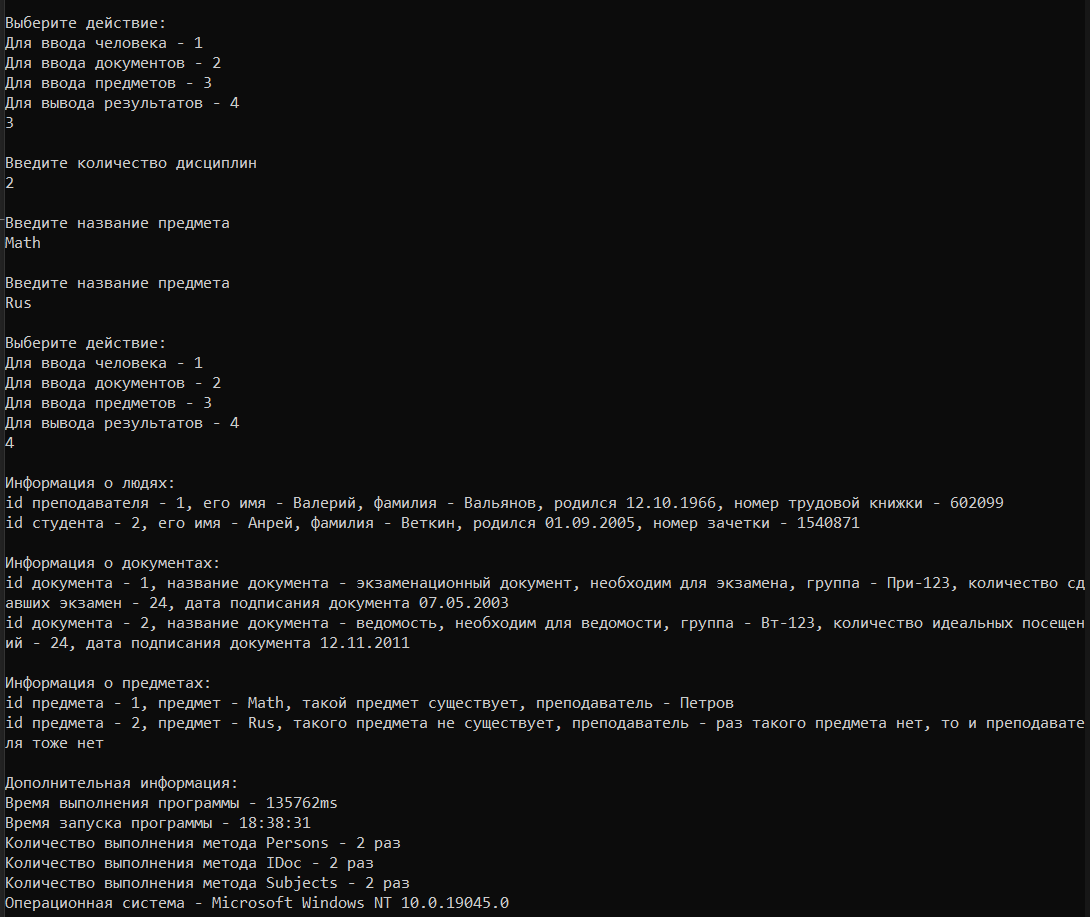


Рисунок 3. Выполнение программы. Часть 2.

ВЫВОД К РАБОТЕ:

Я познакомился с возможностями языка программирования C# для организации событийно-ориентированного программирования на основе делегатов и событий.