**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

СТУПИНСКИЙ ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

**(национальный исследовательский университет)»**

**(СТУПИНСКИЙ ФИЛИАЛ МАИ)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра "МСиИТ"

**ЛАБОРАТОРНАЯ РАБОТА №1**

**по дисциплине «Объектно-ориентированное программирование»**

Студент: Ивинский Илья Артемович

Группа: ТСО-305Б-22

Научный руководитель: Новиков Б. Б.

Оценка:\_\_\_\_\_\_ Дата защиты «\_\_\_» \_\_\_\_\_\_\_\_\_\_2025 г.

Ступино 2025 г.

**Введение**

Объектно-ориентированное программирование (ООП) — это парадигма программирования, которая основана на представлении программы в виде совокупности объектов, взаимодействующих друг с другом. ООП позволяет абстрагировать реальные сущности и процессы, описывая их свойства и поведение с помощью классов и наследования. ООП также обеспечивает инкапсуляцию, полиморфизм и модульность кода, что упрощает разработку, отладку и расширение программ.

Целью данной лабораторной работы является изучение основных принципов и механизмов ООП на языке C#. В ходе работы создавался проект учебного процесса с преподавателями, студентами, курсами и дисциплинами. Для демонстрации принципов ООП было разработано консольное приложение с добавлением информации с клавиатуры и ее выводом. В лабораторной работе созданы классы Person, Student, Lecturer, Discipline, Course, DisciplineManagementSystem, LecturersManagementSystem и StudentsManagementSystem которые имеют различные связи и зависимости между собой. Глобальные списки заполнены тестовыми данными по 12 записей для главных классов для проверки корректности и эффективности кода на разных типах ввода данных.

**Исследовательская часть**

В лабораторной работе рассмотрено применение основных принципов и механизмов ООП в C# на примере разработки консольного приложения, которое имитирует учебный процесс. Для создания приложения использовалась среда разработки JetBrains Rider, которая предоставляет удобные инструменты для работы с C#.

Для генерации случайных чисел использовалась переменная random типа Random. Случайное число получалось как итог вызова функции Next() объекта random. Этим числом являются либо напрямую данные, которые используются в свойстве, например, возраст. Либо индексом для массива данных, из которых извлекается случайным образом один элемент.

**Ход работы**

В приложении были созданы классы Person, Lecturer, Student, Discipline, Course, DisciplineManagementSystem, LecturersManagementSystem и StudentsManagementSystem. Класс Lecturer и Student наследуется от класса Person. Листинг класса Person:

***public class Person  
{  
 public int Id { get; set; }  
 public string Surname { get; set; }  
 public string Name { get; set; }  
 public string Patronymic { get; set; }  
 public int Age { get; set; }  
   
 public Person(int id, string surname, string name, string patronymic, int age)  
 {  
 Id = id;  
 Name = name;  
 Surname = surname;  
 Patronymic = patronymic;  
 Age = age;  
 }  
}***

***В классе созданы автосвойства, одинаковые для наследуемых классов.***

***Листинг класса Lecturer:***

***public class Lecturer : Person  
{  
 public string AcademicTitle { get; set; }  
 public List<string> Subjects { get; set; }  
 public List<int> SubjectsId { get; set; }  
   
 public Lecturer(int id, string surname, string name, string patronymic, int age)   
 : base(id, surname, name, patronymic, age)  
 {  
 Subjects = new List<string>();  
 SubjectsId = new List<int>();  
 }  
   
 public static Lecturer AddLecturer(int id, string surname, string name, string patronymic, int age, string academicTitle, List<string> subjects, List<int> diciplineId)  
 {  
 var lecturer = new Lecturer(id, surname, name, patronymic, age)  
 {  
 AcademicTitle = academicTitle,  
 Subjects = subjects,  
 SubjectsId = diciplineId  
 };  
 return lecturer;  
 }  
  
 public static void UpdateLecturer(int id, string surname, string name, string patronymic, int age, string academicTitle, List<string> subjects, List<int> SubjectsId)  
 {  
 Global.Lecturers[id].Surname = surname;  
 Global.Lecturers[id].Name = name;  
 Global.Lecturers[id].Patronymic = patronymic;  
 Global.Lecturers[id].Age = age;  
 Global.Lecturers[id].AcademicTitle = academicTitle;  
 Global.Lecturers[id].Subjects = subjects;  
 Global.Lecturers[id].SubjectsId = SubjectsId;  
 }  
  
 public void DisplayInfo()  
 {  
 Console.WriteLine($"ID: {Id}");  
 Console.WriteLine($"ФИО: {Surname} {Name} {Patronymic}, возраст: {Age}");  
 Console.WriteLine($"Ученое звание: {AcademicTitle}");  
 Console.WriteLine($"ID дисциплин: {string.Join(", ", SubjectsId)}");  
 Console.WriteLine($"Преподаваемые дисциплины: {string.Join(", ", Subjects)}");  
 Console.WriteLine(new string('-', 50));  
 }  
}***

***В этом классе реализован конструктор, метод для добавления нового преподавателя, обновления существующего и вывода всей связанной информации на экран консоли.***

***Класс студента имеет аналогичные методы и специфичные свойства:***

***public class Student : Person  
{  
 public int GroupNumber { get; set; }  
 public int Course { get; set; }  
   
 public Student(int id, string surname, string name, string patronymic, int age, int groupNumber, int course)   
 : base(id, surname, name, patronymic, age)  
 {  
 Id = id;  
 GroupNumber = groupNumber;  
 Course = course;  
 }  
   
 public static Student AddStudent(int id, string surname, string name, string patronymic, int age, int groupNumber, int course)  
 {  
 var student = new Student(id, surname, name, patronymic, age, groupNumber, course)  
 {  
 GroupNumber = groupNumber,  
 Course = course  
 };  
 return student;  
 }  
   
 public static void UpdateStudent(int id, string surname, string name, string patronymic, int age, int groupNumber, int course)  
 {  
 Global.Students[id].Surname = surname;  
 Global.Students[id].Name = name;  
 Global.Students[id].Patronymic = patronymic;  
 Global.Students[id].Age = age;  
 Global.Students[id].GroupNumber = groupNumber;  
 Global.Students[id].Course = course;  
 }  
   
 public void DisplayInfo()  
 {  
 Console.WriteLine($"ID: {Id}");  
 Console.WriteLine($"ФИО: {Surname} {Name} {Patronymic}, возраст: {Age}");  
 Console.WriteLine($"Курс обучения: {Course}");  
 Console.WriteLine($"Изучаемые дисциплины: {string.Join(", ", Global.Courses[Course-1].Subjects)}");  
 Console.WriteLine($"Группа студента: {GroupNumber}");  
 Console.WriteLine(new string('-', 50));  
 }  
}***

***Дисциплины содержат информацию о своем названии, описание для каждого предмета, список преподавателей, идентификатор дисциплины и преподавателя:***

***public class Discipline  
{  
 public int Id { get; set; }  
 public string Title { get; set; }  
 public string Description { get; set; }  
 public List<string> Lecturer { get; set; }  
 public List<int> LecturerId { get; set; }  
  
 public Discipline(int id, string title, string description)  
 {  
 Id = id;  
 Title = title;  
 Description = description;  
 Lecturer = new List<string>();  
 LecturerId = new List<int>();  
 }  
  
 public static Discipline AddDiscipline(int id, string title, string description, List<string> lecturer, List<int> lecturerId)  
 {  
 var discipline = new Discipline(id, title, description)  
 {  
 Lecturer = lecturer,  
 LecturerId = lecturerId  
 };  
 return discipline;  
 }  
  
 public static void UpdateDiscipline(int id, string title, string description, List<string> lecturer, List<int> lecturerId)  
 {  
 Global.Disciplines[id].Title = title;  
 Global.Disciplines[id].Description = description;  
 Global.Disciplines[id].Lecturer = lecturer;  
 Global.Disciplines[id].LecturerId = lecturerId;  
 }  
   
 public void DisplayInfo()  
 {  
 Console.WriteLine($"ID: {Id}");  
 Console.WriteLine($"Название: {Title}");  
 Console.WriteLine($"Описание: {Description}");  
 Console.WriteLine($"ID преподавателей: {string.Join(", ", LecturerId)}");  
 Console.WriteLine($"Преподаватель: {string.Join(", ", Lecturer)}");  
 Console.WriteLine(new string('-', 50));  
 }  
}***

***Класс курса имеет аналогичные методы работы с данными и список предметов курса:***

***public class Course  
{  
 public int Id { get; set; }  
 public int CourseNumber { get; set; }  
 public List<string> Subjects { get; set; }  
  
 public Course(int id, int courseNumber)  
 {  
 Id = id;  
 CourseNumber = courseNumber;  
 Subjects = new List<string>();  
 }  
  
 public static Course AddCourse(int id, int courseNumber, List<string> subjects)  
 {  
 var course = new Course(id, courseNumber)  
 {  
 Subjects = subjects  
 };  
 return course;  
 }  
  
 public static void UpdateCourse(int id, int courseNumber, List<string> subjects)  
 {  
 Global.Courses[id].CourseNumber = courseNumber;  
 Global.Courses[id].Subjects = subjects;  
 }  
  
 public void DisplayInfo()  
 {  
 Console.WriteLine($"Номер курса: {CourseNumber}");  
 Console.WriteLine("Дисциплины:");  
 foreach (var subject in Subjects)  
 {  
 Console.WriteLine($" - {subject}");  
 }  
 Console.WriteLine(new string('-', 50));  
 }  
}***

***В главном методе программы и в методах классов-менеджеров содержится часть кода, реализующая консольный интерфейс и логику работы с данными, а так же начальное заполнение данными:***

***public class Program  
{  
 public static void Main()  
 {  
 Random random = new Random();  
   
 string[] subjects = { "Иностранный язык", "История", "Математика", "Менеджмент", "Программирование", "Психология", "Сети ЭВМ", "Схемотехника", "Физика", "Философия", "Экономика", "Электротехника" };  
 string[] descriptions = { "Язык, на котором говорят жители другой для индивида страны", "Научная дисциплина, предметом изучения которой является человеческое прошлое", "Наука, занимающаяся изучением свойств и отношений чисел, пространства, форм и структур", "Дисциплина, которая исследует законы и особенности управления", "Дисциплина, изучающая программы для ЭВМ и способы их составления, проверки и улучшения", "Наука о закономерностях развития и функционирования психики человека, как особой формы его жизнедеятельности", "Дисциплина, изучающая совокупность средств вычислительной техники, представляющих собой множество ЭВМ, объединённых с помощью средств телекоммуникаций", "Научно-техническое направление, охватывающее проблемы проектирования и исследования схем электронных устройств радиотехники, связи, автоматики, вычислительной техники и других областей техники", "Наука о природе, изучающая простейшие и вместе с тем наиболее общие её закономерности, свойства и строение материи и законы её движения", "Форма общественного сознания, учение об общих принципах бытия и познания, об отношении человека к миру, наука о всеобщих законах развития природы, общества и мышления", "Наука о хозяйстве, способах его ведения и управления им, отношениях между хозяйствующими субъектами в процессе производства, обмена, распределения и потребления товаров и услуг", "Область техники, связанная с получением, распределением, преобразованием и использованием электрической энергии"};  
 string[] surnames = { "Александров", "Артемов", "Антонов", "Васильев", "Дмитриев", "Иванов", "Максимов", "Павлов", "Петров", "Семенов", "Сергеев", "Федоров" };  
 string[] names = { "Александр", "Артем", "Антон", "Василий", "Дмитрий", "Иван", "Максим", "Павел", "Петр", "Семен", "Сергей", "Федор" };  
 string[] patronymics = { "Александрович", "Артемович", "Антонович", "Васильевич", "Дмитриевич", "Иванович", "Максимович", "Павлович", "Петрович", "Семенович", "Сергеевич", "Федорович" };  
 string[] titles = { "Ассистент", "Доцент", "Профессор", "Старший преподаватель", "Заведующий кафедрой" };  
   
 int numOfObjects = 12;   
 for (int i = 0; i < numOfObjects; i++)  
 {  
 string lecturerSurname = surnames[random.Next(surnames.Length)];  
 string lecturerName = names[random.Next(names.Length)];  
 string lecturerPatronymic = patronymics[random.Next(patronymics.Length)];  
 Discipline discipline = Discipline.AddDiscipline(i + 1, subjects[i], descriptions[i], new List<string> { lecturerSurname }, new List<int>{i+1});  
 Global.Disciplines.Add(discipline);  
 Lecturer lecturer = Lecturer.AddLecturer(i + 1, lecturerSurname, lecturerName, lecturerPatronymic, random.Next(25, 61), titles[random.Next(titles.Length)], new List<string> { subjects[i] }, new List<int>{i+1});  
 Global.Lecturers.Add(lecturer);  
 Student student = Student.AddStudent(i + 1, surnames[random.Next(surnames.Length)], names[random.Next(names.Length)], patronymics[random.Next(patronymics.Length)], random.Next(18, 25), random.Next(1, 5), random.Next(1, 4));  
 Global.Students.Add(student);  
 }  
   
 Course firstCourse = Course.AddCourse(1, 1, new List<string>{subjects[0], subjects[1], subjects[2]});  
 Global.Courses.Add(firstCourse);  
 Course secondCourse = Course.AddCourse(2, 2, new List<string>{subjects[3], subjects[4], subjects[5]});  
 Global.Courses.Add(secondCourse);  
 Course thirdCourse = Course.AddCourse(3, 3, new List<string>{subjects[6], subjects[7], subjects[8]});  
 Global.Courses.Add(thirdCourse);  
 Course fourthCourse = Course.AddCourse(4, 4, new List<string>{subjects[9], subjects[10], subjects[11]});  
 Global.Courses.Add(fourthCourse);  
   
 while (true)  
 {  
 Console.WriteLine("1 - дисциплины\n2 - курсы\n3 - преподаватели\n4 - студенты\n5 - поиск");  
 bool isNumber = int.TryParse(Console.ReadLine(), out int selection);  
 Console.Clear();  
 if (isNumber)  
 {  
 switch (selection)  
 {  
 case 1:  
 Console.WriteLine("1 - добавить новую дисциплину\n2 - изменить данные о дисциплине\n3 - удалить дисциплину\n4 - показать преподаваемые дисциплины\n5 - прикрепить дисциплину к преподавателю\n6 - прикрепить дисциплину к курсу");  
 bool isDisciplineNumber = int.TryParse(Console.ReadLine(), out int disciplineSelection);  
 Console.Clear();  
 if (isDisciplineNumber)  
 {  
 DisciplinesManagementSystem.Main(disciplineSelection);  
 }  
 break;  
 case 2:  
 foreach (var course in Global.Courses)  
 {  
 course.DisplayInfo();  
 }  
 break;  
 case 3:  
 Console.WriteLine("1 - добавить нового преподавателя\n2 - изменить данные о преподавателе\n3 - удалить преподавателя\n4 - показать информацию о преподавателях");  
 bool isLecturerNumber = int.TryParse(Console.ReadLine(), out int lecturerSelection);  
 Console.Clear();  
 if (isLecturerNumber)  
 {  
 LecturersManagementSystem.Main(lecturerSelection);  
 }  
 break;  
 case 4:  
 Console.WriteLine("1 - добавить нового студента\n2 - изменить данные о студенте\n3 - удалить студента\n4 - показать информацию о студентах");  
 bool isStudentNumber = int.TryParse(Console.ReadLine(), out int studentSelection);  
 Console.Clear();  
 if (isStudentNumber)  
 {  
 StudentsManagementSystem.Main(studentSelection);  
 }  
 break;  
 case 5:  
 Console.WriteLine("1 - дисциплины\n2 - преподаватели\n3 - студенты");  
 bool isSearch = int.TryParse(Console.ReadLine(), out int search);  
 switch (search)  
 {  
 case 1:  
 Console.WriteLine("Введите ID дисциплины:");  
 bool isSearchDiscipline = int.TryParse(Console.ReadLine(), out int searchDiscipline);  
 Console.WriteLine(new string('-', 50));  
 Console.WriteLine($"Данные дисцпиплины с указанным ID:");  
 Global.Disciplines[searchDiscipline].DisplayInfo();  
 break;  
 case 2:  
 Console.WriteLine("Введите ID преподавателя:");  
 bool isSearchLecturer = int.TryParse(Console.ReadLine(), out int searchLecturer);  
 Console.WriteLine(new string('-', 50));  
 Console.WriteLine($"Данные преподавателя с указанным ID:");  
 Global.Lecturers[searchLecturer].DisplayInfo();  
 break;  
 case 3:  
 Console.WriteLine("Введите ID студента:");  
 bool isSearchStudent = int.TryParse(Console.ReadLine(), out int searchStudent);  
 Console.WriteLine(new string('-', 50));  
 Console.WriteLine($"Данные студента с указанным ID:");  
 Global.Students[searchStudent].DisplayInfo();  
 break;  
 }  
 break;  
 }  
 }  
 }  
 }  
}***

**Вывод**

В результате выполнения лабораторной работы было создано консольное приложение, иллюстрирующее основные принципы ООП на примере системы образовательного процесса из классов, конструкторов и методов вывода информации о преподавателях, студентах и дисциплинах. В приложении реализован принцип наследования. В программном меню реализованы функции добавления, изменения и удаления студентов, преподавателей и дисциплин, а также вывод информации о них через методы соответствующих классов.