САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ ИНФОРМАТИКИ

Отчет по лабораторной работе №2

по курсу «объектно-ориентированное программирование»

Тема: Система управления курсами и преподавателями

Выполнила:
Буй Тхук Хуен - К3239
Проверил:
Слюсаренко С. В.

Санкт-Петербург
2025 г.

Содержание отчета

I. Задача

Создайте систему для управления учебными курсами и преподавателями. Система должна:

1. Позволять добавлять и удалять курсы, а также назначать преподавателей на эти курсы. Включать возможность хранения и возможности получения информации о студентах, записанных на каждый курс.
2. Поддерживать различные типы курсов: онлайн-курсы и офлайн-курсы с уникальными характеристиками.
3. Обеспечивать возможность получить все курсы, которые ведет конкретный преподаватель.
4. Ключевая бизнес логика должна быть покрыта юнит тестами. (необходимо использовать xunit)

Рекомендуемые паттерны (в этой лр их можно не писать, задание со звездочкой):

1. Builder
2. Singleton

Вы разрабатываете систему для университета, который хочет улучшить управление своими образовательными программами. Важно, чтобы система была гибкой, расширяемой и легко поддерживаемой.

Цель ЛР заключается в закреплении навыков работы с наследованием, полиморфизмом, абстракцией и инкапсуляцией, а также создание объектов и их взаимодействие в системе.

1. Листинг кода

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace CourseManagement
{
    // Модели
    public class Student
    {
        public int Id { get; set; }
        public string? Name { get; set; }
    }

    public class Teacher
    {
        public int Id { get; set; }
```

```
        public string? Name { get; set; }
    }

    public abstract class Course
    {
        public int Id { get; set; }
        public string? Name { get; set; }
        public Teacher? AssignedTeacher { get; set; }
              public List<Student> EnrolledStudents { get; set; } = new
List<Student>();
        public abstract void DisplayInfo();
    }

    public class OnlineCourse : Course
    {
        public string? Platform { get; set; }

        public override void DisplayInfo()
        {
                    Console.WriteLine($"[Онлайн] Id: {Id}, Название: {Name},
Преподаватель: {AssignedTeacher?.Name ?? "Нет"}, Платформа: {Platform},
Студентов: {EnrolledStudents.Count}");
        }
    }

    public class OfflineCourse : Course
    {
        public string? Location { get; set; }

        public override void DisplayInfo()
        {
                    Console.WriteLine($"[Офлайн] Id: {Id}, Название: {Name},
Преподаватель: {AssignedTeacher?.Name ?? "Нет"}, Место: {Location},
Студентов: {EnrolledStudents.Count}");
        }
    }

    // Класс для управления бизнес-логикой
    public class CourseManager
    {
        private List<Student> _students = new List<Student>();
        private List<Teacher> _teachers = new List<Teacher>();
        private List<Course> _courses = new List<Course>();
        private int _nextStudentId = 1;
```

```
        private int _nextTeacherId = 1;
        private int _nextCourseId = 1;

        // Добавить студента (для тестов или расширения)
        public void AddStudent(string? name)
        {
                _students.Add(new Student { Id = _nextStudentId++, Name = name
});
        }

        // Добавить преподавателя (для тестов или расширения)
        public void AddTeacher(string? name)
        {
                _teachers.Add(new Teacher { Id = _nextTeacherId++, Name = name
});
        }

        // Добавить курс
                public bool AddCourse(string? name, string type, string?
platformOrLocation)
        {
            Course? course = null;
            if (type.ToLower() == "online")
            {
                    course = new OnlineCourse { Id = _nextCourseId++, Name =
name, Platform = platformOrLocation };
            }
            else if (type.ToLower() == "offline")
            {
                    course = new OfflineCourse { Id = _nextCourseId++, Name =
name, Location = platformOrLocation };
            }
            else
            {
                return false; // Неверный тип
            }
            _courses.Add(course);
            return true;
        }

        // Удалить курс
        public bool RemoveCourse(int id)
        {
            var course = _courses.FirstOrDefault(c => c.Id == id);
```

```
            if (course != null)
            {
                _courses.Remove(course);
                return true;
            }
            return false;
        }

        // Назначить преподавателя на курс
        public bool AssignTeacher(int courseId, int teacherId)
        {
            var course = _courses.FirstOrDefault(c => c.Id == courseId);
            var teacher = _teachers.FirstOrDefault(t => t.Id == teacherId);
            if (course != null && teacher != null)
            {
                course.AssignedTeacher = teacher;
                return true;
            }
            return false;
        }

        // Записать студента на курс
        public bool EnrollStudent(int courseId, int studentId)
        {
            var course = _courses.FirstOrDefault(c => c.Id == courseId);
            var student = _students.FirstOrDefault(s => s.Id == studentId);
                            if (course != null && student != null &&
!course.EnrolledStudents.Contains(student))
            {
                course.EnrolledStudents.Add(student);
                return true;
            }
            return false;
        }

        // Получить все курсы преподавателя
        public List<Course> GetCoursesByTeacher(int teacherId)
        {
                        return _courses.Where(c => c.AssignedTeacher?.Id ==
teacherId).ToList();
        }

        // Получить всех студентов на курсе
        public List<Student> GetStudentsInCourse(int courseId)
```

```
        {
            var course = _courses.FirstOrDefault(c => c.Id == courseId);
            return course?.EnrolledStudents ?? new List<Student>();
        }

        // Получить все курсы
        public List<Course> GetAllCourses()
        {
            return _courses;
        }

        // Вспомогательные методы для тестов
        public List<Student> GetStudents() => _students;
        public List<Teacher> GetTeachers() => _teachers;
    }

    // Главная программа
    class Program
    {
        static CourseManager manager = new CourseManager();

        static void Main()
        {
            SeedData(); // пример данных
            while (true)
            {
                Console.WriteLine("\n--- Система управления курсами ---");
                Console.WriteLine("1. Добавить курс");
                Console.WriteLine("2. Удалить курс");
                Console.WriteLine("3. Назначить преподавателя на курс");
                Console.WriteLine("4. Записать студента на курс");
                Console.WriteLine("5. Показать все курсы");
                Console.WriteLine("6. Показать курсы преподавателя");
                Console.WriteLine("7. Показать студентов на курсе");
                Console.WriteLine("0. Выход");
                Console.Write("Выберите действие: ");
                var choice = Console.ReadLine();
                switch (choice)
                {
                    case "1": AddCourse(); break;
                    case "2": RemoveCourse(); break;
                    case "3": AssignTeacher(); break;
                    case "4": EnrollStudent(); break;
                    case "5": ShowAllCourses(); break;
```

```
                    case "6": ShowCoursesByTeacher(); break;
                    case "7": ShowStudentsInCourse(); break;
                    case "0": return;
                    default: Console.WriteLine("Неверный выбор"); break;
                }
            }
        }

        static void SeedData()
        {
            // Добавить преподавателей
            manager.AddTeacher("Алиса");
            manager.AddTeacher("Боб");
            manager.AddTeacher("Марина");
            manager.AddTeacher("Иван");

            // Добавить студентов
            manager.AddStudent("Джон");
            manager.AddStudent("Мария");
            manager.AddStudent("Кира");
            manager.AddStudent("Лиза");

            // Добавить курсы
            manager.AddCourse("Математика", "online", "Zoom");
            manager.AddCourse("Физика", "offline", "Аудитория 101");
                  manager.AddCourse("Программирование", "online", "Microsoft
Teams");

            // Назначить преподавателей на курсы
            manager.AssignTeacher(1, 1); // Алиса на Математика
            manager.AssignTeacher(2, 2); // Боб на Физика
            manager.AssignTeacher(3, 1); // Алиса на Программирование

            // Записать студентов на курсы
            manager.EnrollStudent(1, 1); // Джон на Математика
            manager.EnrollStudent(1, 2); // Мария на Математика
            manager.EnrollStudent(2, 1); // Джон на Физика
            manager.EnrollStudent(3, 2); // Мария на Программирование
        }

        static void AddCourse()
        {
            Console.Write("Название курса: ");
```

```
            string? name = Console.ReadLine();  // Sửa thành string? để tránh
warning
            Console.WriteLine("Выберите тип курса: 1. Онлайн 2. Офлайн");
            Console.Write("Введите номер: ");
              string? typeInput = Console.ReadLine();  // Đã là string?, giữ
nguyên
             string type = typeInput == "1" ? "online" : typeInput == "2" ?
"offline" : "";
               if (string.IsNullOrEmpty(type)) { Console.WriteLine("Неверный
выбор типа!"); return; }
                   Console.Write(type == "online" ? "Платформа: " : "Место
проведения: ");
            string? param = Console.ReadLine();  // Đã là string?, giữ nguyên
            if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(param))
            {
                 Console.WriteLine("Название курса и платформа/место не могут
быть пустыми!");
                return;
            }
            if (manager.AddCourse(name, type, param))
            {
                Console.WriteLine("Курс добавлен успешно!");
                // Автоматически показать информацию о добавленном курсе
                 var newCourse = manager.GetAllCourses().Last(); // Получить
последний добавленный курс
                newCourse.DisplayInfo();
            }
            else
            {
                Console.WriteLine("Ошибка добавления курса!");
            }
        }

        static void RemoveCourse()
        {
            Console.Write("Введите Id курса для удаления: ");
            if (int.TryParse(Console.ReadLine(), out int id))
            {
                if (manager.RemoveCourse(id))
                    Console.WriteLine("Курс удален.");
                else
                    Console.WriteLine("Курс не найден.");
            }
        }
```

```
        static void AssignTeacher()
        {
            Console.Write("Введите Id курса: ");
            int.TryParse(Console.ReadLine(), out int courseId);
            Console.WriteLine("Преподаватели:");
                                foreach   (var   t   in   manager.GetTeachers())
Console.WriteLine($"{t.Id}: {t.Name}");
            Console.Write("Введите Id преподавателя: ");
            int.TryParse(Console.ReadLine(), out int teacherId);
            if (!manager.AssignTeacher(courseId, teacherId))
                Console.WriteLine("Ошибка назначения преподавателя.");
            else
                Console.WriteLine("Преподаватель назначен.");
        }

        static void EnrollStudent()
        {
            Console.Write("Введите Id курса: ");
            int.TryParse(Console.ReadLine(), out int courseId);
            Console.WriteLine("Студенты:");
                                foreach   (var   s   in   manager.GetStudents())
Console.WriteLine($"{s.Id}: {s.Name}");
            Console.Write("Введите Id студента: ");
            int.TryParse(Console.ReadLine(), out int studentId);
            if (!manager.EnrollStudent(courseId, studentId))
                Console.WriteLine("Ошибка записи студента.");
            else
                Console.WriteLine("Студент записан на курс.");
        }

        static void ShowAllCourses()
        {
            foreach (var c in manager.GetAllCourses()) c.DisplayInfo();
        }

        static void ShowCoursesByTeacher()
        {
            Console.WriteLine("Преподаватели:");
                                foreach   (var   t   in   manager.GetTeachers())
Console.WriteLine($"{t.Id}: {t.Name}");
            Console.Write("Введите Id преподавателя: ");
            int.TryParse(Console.ReadLine(), out int teacherId);
            var courses = manager.GetCoursesByTeacher(teacherId);
```

```
                if (courses.Count == 0) Console.WriteLine("Нет назначенных
курсов.");
            else foreach (var c in courses) c.DisplayInfo();
        }

        static void ShowStudentsInCourse()
        {
            Console.Write("Введите Id курса: ");
            int.TryParse(Console.ReadLine(), out int courseId);
            var students = manager.GetStudentsInCourse(courseId);
                if (students.Count == 0) Console.WriteLine("Нет студентов на
курсе.");
            else
            {
                Console.WriteLine("Студенты на курсе:");
                     foreach (var s in students) Console.WriteLine($"{s.Id}:
{s.Name}");
            }
        }
    }
}
```

- Текстовое объяснение решения

  Модели

  + Модели Student, Teacher: содержит Id и Name

  + Модели Course: Абстрактный базовый класс для курсов
    - Содержит общие свойства: Id, Name, назначенный преподаватель (AssignedTeacher), список зачисленных студентов (EnrolledStudents), который по умолчанию инициализируется пустым списком.
    - Абстрактный метод DisplayInfo() – обеспечивает полиморфизм: разные подклассы выводят информацию по-своему.

  + OnlineCourse и OfflineCourse: Оба наследуются от Course и реализуют DisplayInfo()
    - OnlineCourse добавляет свойство Platform.
    - OfflineCourse добавляет свойство Location.
    - В DisplayInfo() используется интерполяция строк и безопасный доступ к AssignedTeacher?.Name ?? "Нет", т.е. если преподаватель не назначен – выводится "Нет". Также показывается число студентов EnrolledStudents.Count.

+ CourseManager: CourseManager хранит три списка сущностей в памяти и счётчики для автогенерации Id. Все данные живут только в памяти (не сохраняются в БД).

+ AddStudent / AddTeacher: Принимают name, создают новый объект с уникальным Id (через _next...Id++) и добавляют в соответствующий список. Метод используется в SeedData() и может использоваться для расширения/тестов.

CourseManager

+ AddCourse(string? name, string type, string? platformOrLocation):
  - type ожидается как "online" или "offline" (перед вызовом код переводит ввод в эти строки).
  - Внутри выбирается соответствующий класс (OnlineCourse или OfflineCourse), инициализируется Id, Name, и соответствующее поле (Platform или Location).
  - Если type не распознан – возвращает false (ошибка).
  - В случае успеха добавляет курс в courses и возвращает true.

+ RemoveCourse(int id):
  Ищет курс по Id через FirstOrDefault.
  Если найден – удаляет и возвращает true, иначе false.

+ AssignTeacher(int courseId, int teacherId)
  - Находит курс и преподавателя по Id.
  - Если оба найдены – устанавливает course.AssignedTeacher = teacher и возвращает true.
  - Иначе – false.

+ EnrollStudent(int courseId, int studentId)
  - Находит курс и студента.
  - Проверяет, что студент ещё не содержится в course.EnrolledStudents (!course.EnrolledStudents.Contains(student)).
  - Если всё ок – добавляет студента в список и возвращает true, иначе false.
  - Замечание: Contains сравнивает объекты по ссылке; это корректно, пока объекты берутся из одного и того же списка _students.

+ GetCoursesByTeacher(int teacherId)
  - Возвращает все курсы где AssignedTeacher?.Id == teacherId.

- Используется LINQ Where(...).ToList().

+ GetStudentsInCourse(int courseId): Находит курс и возвращает course.EnrolledStudents, либо пустой список если курс не найден.

+ GetAllCourses, GetStudents, GetTeachers: Вспомогательные методы для получения внутренних списков (полезно в UI и тестах).

Главная программа

- Используется единый статический экземпляр manager.
- В Main() сначала вызывается SeedData() – заполняет менеджер начальными преподавателями, студентами и курсами, назначает преподавателей и записывает студентов на курсы – чтобы при старте были данные для демонстрации.

+ SeedData()
  - Добавляет 4 преподавателя, 4 студента, 3 курса (2 online, 1 offline), назначает учителей и записывает студентов (через методы менеджера).
  - Пример вызовов: manager.AddTeacher("Алиса"); manager.AddCourse("Математика","online","Zoom"); manager.AssignTeacher(1,1); manager.EnrollStudent(1,1);

+ Главный цикл и меню
  - Бесконечный цикл while(true) показывает меню действий:
    1. Добавить курс
    2. Удалить курс
    3. Назначить преподавателя на курс
    4. Записать студента на курс
    5. Показать все курсы
    6. Показать курсы преподавателя
    7. Показать студентов на курсе
    8. Выход
  - Чтение выбора через Console.ReadLine() и switch по строке.
  - Если введено "0", return – программа завершается.

Вспомогательные статические методы UI: Каждый пункт меню реализован отдельным статическим методом в Program:

+ AddCourse():
  - Спрашивает название, тип (1 или 2) и параметр (платформа/место).
  - Переводит typeInput в "online" или "offline". Если ввод некорректен – сообщает об ошибке.

    - Проверяет, что name и param не пусты.
    - Вызывает manager.AddCourse(...). Если успешно – выводит сообщение и автоматически показывает DisplayInfo() у только что добавленного курса (manager.GetAllCourses().Last()).

+ RemoveCourse(): Считывает Id (через int.TryParse) и вызывает manager.RemoveCourse(id), сообщает результат.

+ AssignTeacher():
    - Считывает courseId.
    - Выводит список преподавателей (manager.GetTeachers()), чтобы пользователь видел доступные Id.
      Считывает teacherId и вызывает manager.AssignTeacher(...).

+ EnrollStudent(): Аналогично: показывает студентов, считывает studentId, вызывает manager.EnrollStudent(...).

+ ShowAllCourses(): Перебирает manager GetAllCourses() и вызывает DisplayInfo() для каждого курса.

+ ShowCoursesByTeacher(): Выводит список преподавателей и ждёт teacherId, затем получает и показывает курсы через manager.GetCoursesByTeacher(teacherId).

+ ShowStudentsInCourse(): Считывает courseId и показывает список студентов (или сообщение, что студентов нет).

- Результат работы код

```
--- Система управления курсами ---
1. Добавить курс
2. Удалить курс
3. Назначить преподавателя на курс
4. Записать студента на курс
5. Показать все курсы
6. Показать курсы преподавателя
7. Показать студентов на курсе
0. Выход
Выберите действие: 5
[Онлайн] Id: 1, Название: Математика, Преподаватель: Алиса, Платформа: Zoom, Студентов: 2
[Офлайн] Id: 2, Название: Физика, Преподаватель: Боб, Место: Аудитория 101, Студентов: 1
[Онлайн] Id: 3, Название: Программирование, Преподаватель: Алиса, Платформа: Microsoft Teams, Студентов: 1

--- Система управления курсами ---
1. Добавить курс
2. Удалить курс
3. Назначить преподавателя на курс
4. Записать студента на курс
5. Показать все курсы
6. Показать курсы преподавателя
7. Показать студентов на курсе
0. Выход
Выберите действие: 1
Название курса: Biography
Выберите тип курса: 1. Онлайн 2. Офлайн
Введите номер: 2
Место проведения: 206
Курс добавлен успешно!
[Офлайн] Id: 4, Название: Biography, Преподаватель: Нет, Место: 206, Студентов: 0
```

```
--- Система управления курсами ---
1. Добавить курс
2. Удалить курс
3. Назначить преподавателя на курс
4. Записать студента на курс
5. Показать все курсы
6. Показать курсы преподавателя
7. Показать студентов на курсе
0. Выход
Выберите действие: 2
Введите Id курса для удаления: 2
Курс удален.

--- Система управления курсами ---
1. Добавить курс
2. Удалить курс
3. Назначить преподавателя на курс
4. Записать студента на курс
5. Показать все курсы
6. Показать курсы преподавателя
7. Показать студентов на курсе
0. Выход
Выберите действие: 3
Введите Id курса: 4
Преподаватели:
1: Алиса
2: Боб
3: Марина
4: Иван
Введите Id преподавателя: 4
Преподаватель назначен.
```

```
--- Система управления курсами ---
1. Добавить курс
2. Удалить курс
3. Назначить преподавателя на курс
4. Записать студента на курс
5. Показать все курсы
6. Показать курсы преподавателя
7. Показать студентов на курсе
0. Выход
Выберите действие: 4
Введите Id курса: 4
Студенты:
1: Джон
2: Мария
3: Кира
4: Лиза
Введите Id студента: 3
Студент записан на курс.
```

```
--- Система управления курсами ---
1. Добавить курс
2. Удалить курс
3. Назначить преподавателя на курс
4. Записать студента на курс
5. Показать все курсы
6. Показать курсы преподавателя
7. Показать студентов на курсе
0. Выход
Выберите действие: 6
Преподаватели:
1: Алиса
2: Боб
3: Марина
4: Иван
Введите Id преподавателя: 1
[Онлайн] Id: 1, Название: Математика, Преподаватель: Алиса, Платформа: Zoom, Студентов: 2
[Онлайн] Id: 3, Название: Программирование, Преподаватель: Алиса, Платформа: Microsoft Teams, Студентов: 1
```

```
--- Система управления курсами ---
1. Добавить курс
2. Удалить курс
3. Назначить преподавателя на курс
4. Записать студента на курс
5. Показать все курсы
6. Показать курсы преподавателя
7. Показать студентов на курсе
0. Выход
Выберите действие: 7
Введите Id курса: 1
Студенты на курсе:
1: Джон
2: Мария
```

- Test

```
PS D:\OOP\OOP2025\Lab2\CourseManagement.Tests> dotnet test
>>
Restore complete (0.6s)
  CourseManagement succeeded (0.3s) → D:\OOP\OOP2025\Lab2\CourseManagement\bin\Debug\net9.0\CourseManagement.dll
  CourseManagement.Tests succeeded (0.2s) → bin\Debug\net9.0\CourseManagement.Tests.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v3.1.5+1b188a7b0a (64-bit .NET 9.0.9)
[xUnit.net 00:00:00.53]   Discovering: CourseManagement.Tests
[xUnit.net 00:00:00.58]   Discovered:  CourseManagement.Tests
[xUnit.net 00:00:00.61]   Starting:    CourseManagement.Tests
[xUnit.net 00:00:00.94]   Finished:    CourseManagement.Tests
  CourseManagement.Tests test succeeded (4.2s)

Test summary: total: 5, failed: 0, succeeded: 5, skipped: 0, duration: 4.2s
Build succeeded in 6.1s
```

II. Вывод

В ходе выполнения лабораторной работы была разработана и протестирована программа – система для управления учебными курсами и преподавателями на языке C# с использованием объектно-ориентированного подхода.

- Код организован по принципу разделения моделей, бизнес-логики и пользовательского интерфейса.
- CourseManager инкапсулирует операции над данными, Program реализует консольное взаимодействие.
- Использованы основы ООП: наследование (Course → OnlineCourse/OfflineCourse), полиморфизм (DisplayInfo()), композиция (курс содержит список студентов и ссылку на преподавателя).
- Программа хороша как учебный пример и базовая основа для расширения (сохранение, валидация, GUI/веб-интерфейс, БД).

В результате работы получена консольная программа, демонстрирующая принципы полиморфизмом, абстракцией и инкапсуляцией, а также основы тестирования программных модулей.