

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра программных систем

**Домашнее задание**

**Иерархия классов учебного центра**

Выполнил: Мороз

Илья Олегович

Группа № K3120

Проверил: Осипов Н.А.

Санкт-Петербург

2019

**Цель работы:**

Целью работы является реализация иерархии классов учебного центра.

**Ход работы:**

1. В самом начале работы были продуманы основные моменты программы. Основными классами программы будут являться Person.cs, Laboratory.cs, Admin.cs, Manager.cs, Student.cs, Teacher.cs.
2. Исходя из названий классов можно догадаться, что в программе будет один абстрактный класс Person, от которого будут наследоваться классы сотрудников лаборатории.
3. Далее на рисунке 1 представлена диаграмма классов в программе.

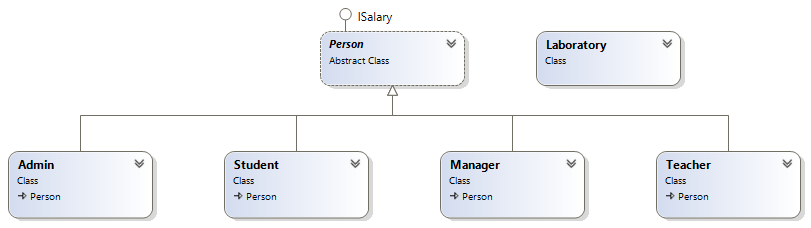


Рисунок 1 – Диаграмма классов

1. С помощью ресурса Creately была создана схема взаимодействия пользователя с программой.

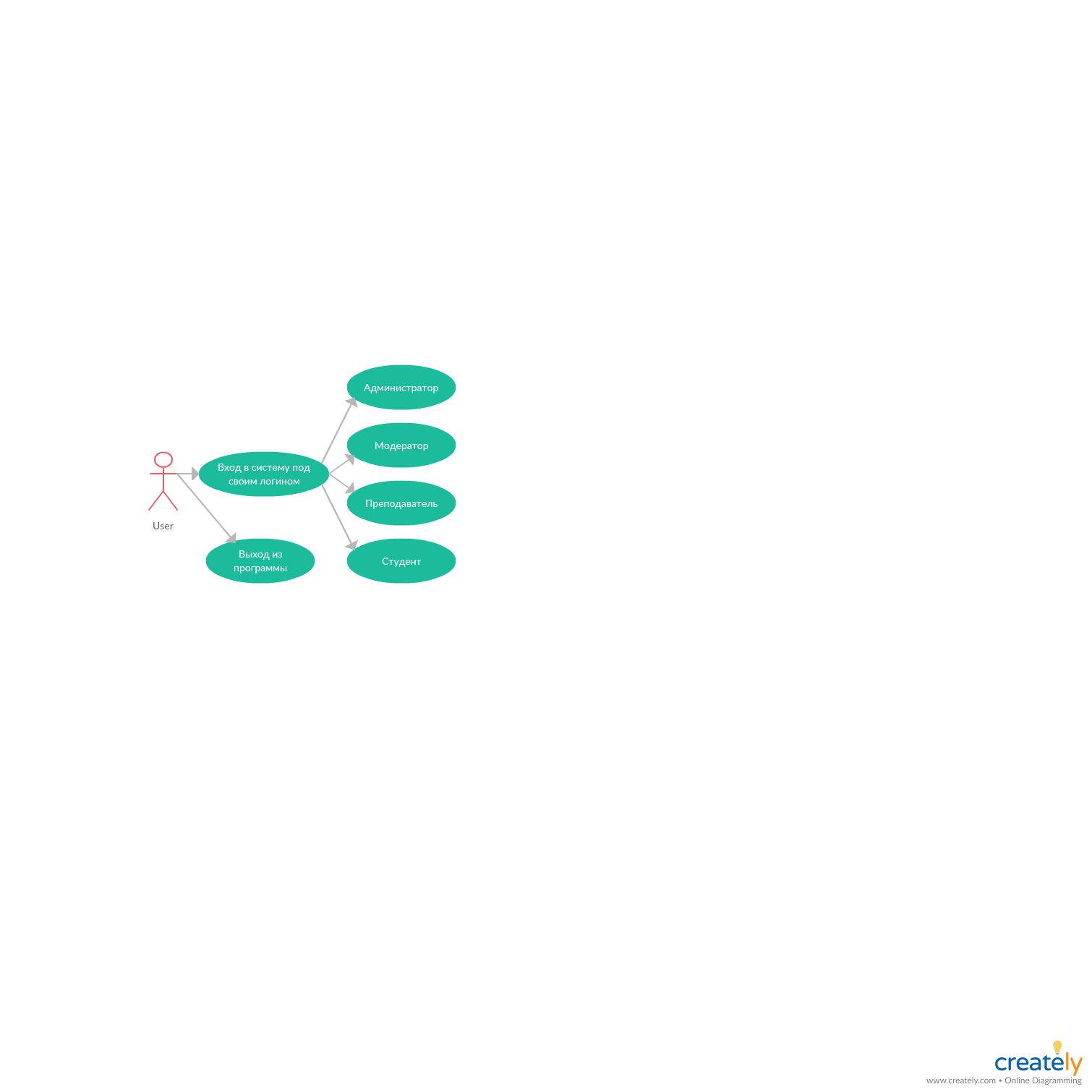


Рисунок 2 – Схема взаимодействия пользователя с программой

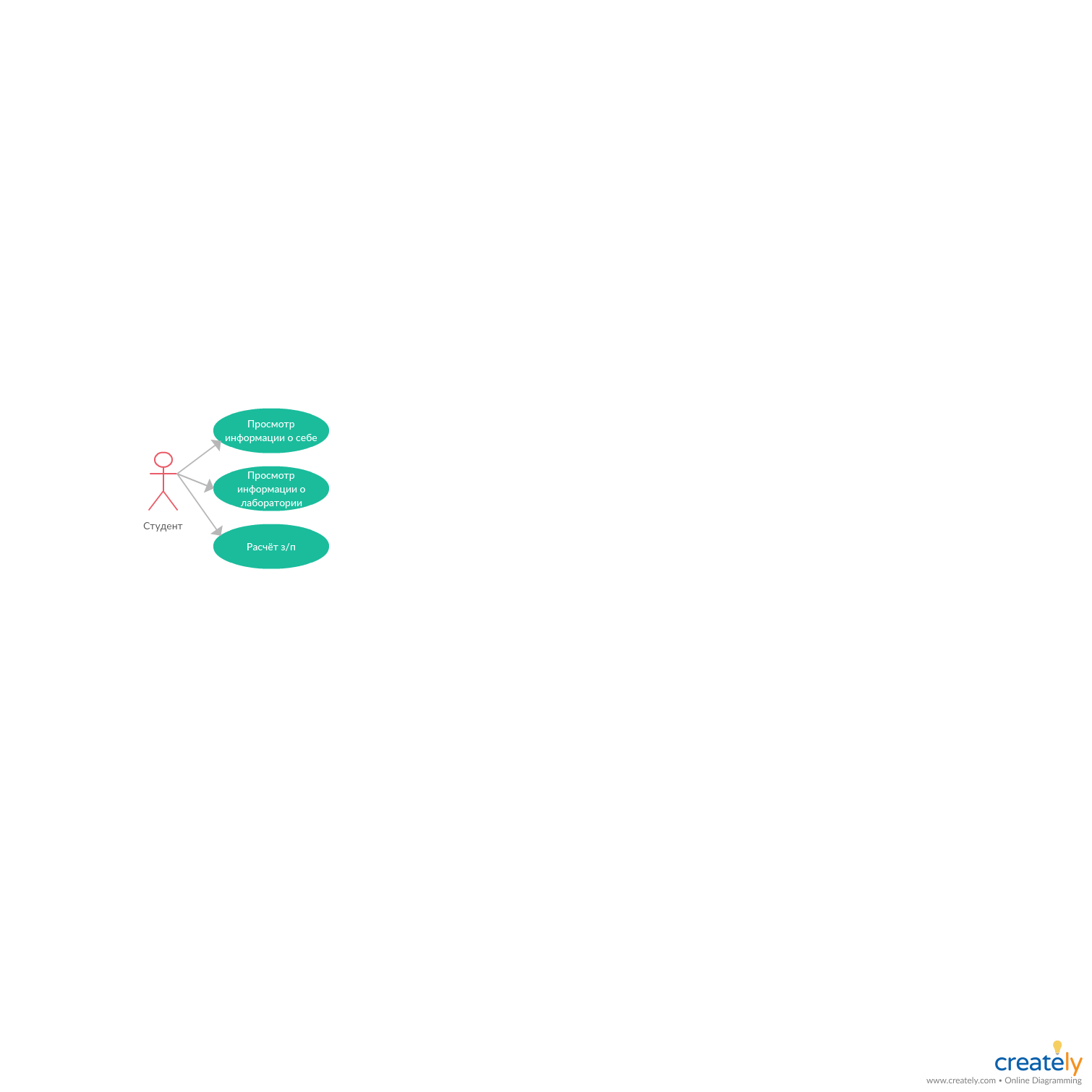


Рисунок 3 – Схема взаимодействия студента с программой

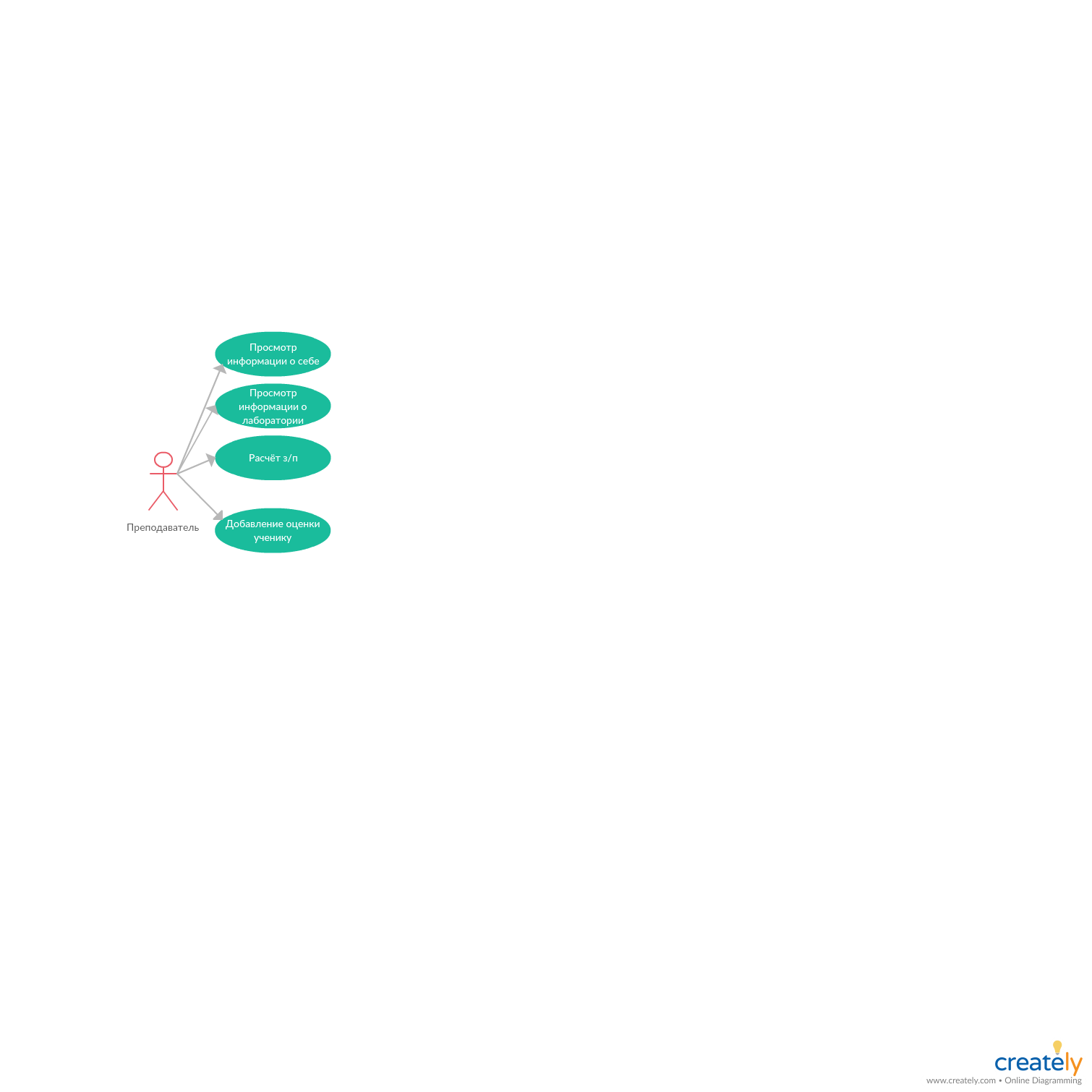


Рисунок 4 – Схема взаимодействия преподавателя с программой

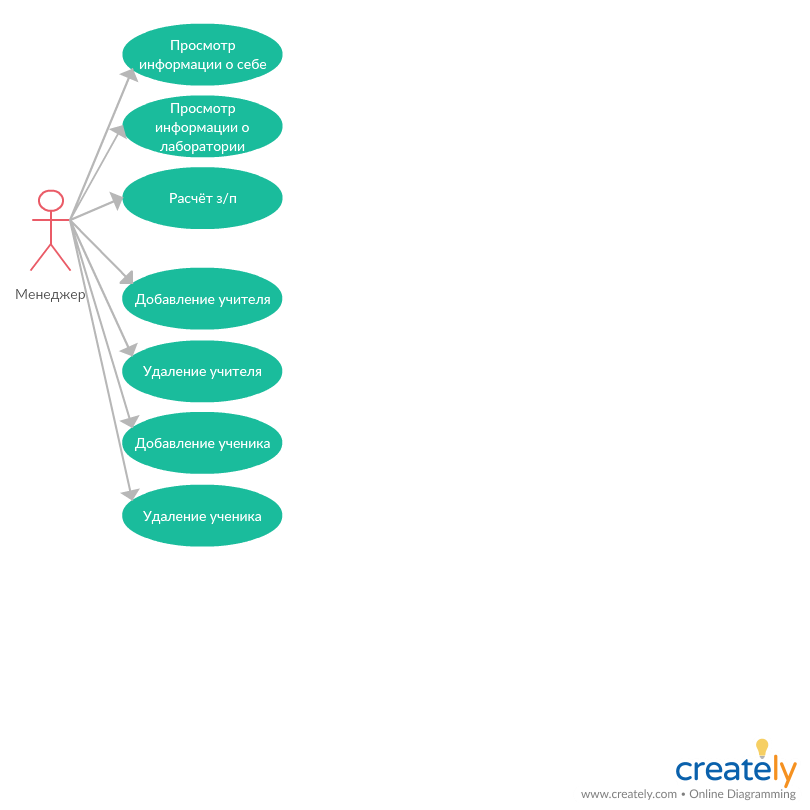


Рисунок 5 – Схема взаимодействия менеджера с программой

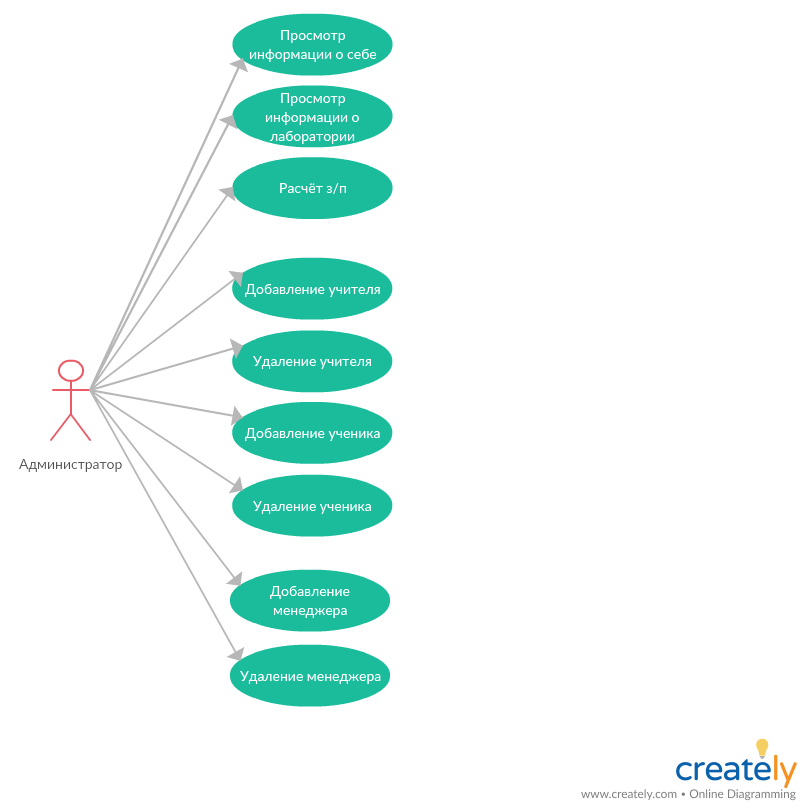


Рисунок 6 – Схема взаимодействия администратора с программой

1. После проектирования программы и обдумывания основного функционала, произошёл переход к написанию кода.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Homework

{

abstract class Person : ISalary

{

public enum Permissions { admin, student, teacher, manager }

public int Id { get; set; }

public string Lastname { get; }

public string Firstname { get; }

public DateTime Birthdate { get; }

public string Username { get; set; }

public string Password { get; set; }

public Permissions CurrentPermissions { get; set; }

protected Person() { }

public Person(int id, string lastname, string firstname, DateTime birthdate, string username, string password, Permissions currentPermissions)

{

Id = id;

Lastname = lastname;

Firstname = firstname;

Birthdate = birthdate;

Username = username;

Password = password;

CurrentPermissions = currentPermissions;

}

public int GetAge()

{

DateTime today = DateTime.Today;

int age = today.Year - this.Birthdate.Year;

if (this.Birthdate > today.AddYears(-age)) age--;

return age;

}

public void Print()

{

Console.WriteLine(this);

}

public void ChangePassword(string newPassword)

{

this.Password = newPassword;

}

public abstract float GetSalary();

}

}

Листинг 1 – Файл Person.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Homework

{

class Student : Person

{

private List<int> Grades = new List<int>();

public string Faculty { get; set; }

public string Group { get; set; }

public Student(int id, string lastname, string firstname, DateTime birthdate, string username, string password, string faculty, string group) : base(id, lastname, firstname, birthdate, username, password, Permissions.student)

{

Faculty = faculty;

Group = group;

}

public void SetGrades(List<int> grades)

{

this.Grades = grades;

}

public void AddGrade(int grade)

{

this.Grades.Add(grade);

}

public void DeleteGrades()

{

this.Grades.Clear();

}

public IReadOnlyCollection<int> GetGrades()

{

return this.Grades.AsReadOnly();

}

public bool IsCool()

{

if (this.Grades.Contains(2) || this.Grades.Contains(3) || this.Grades.Contains(4) || this.Grades.Contains(5))

{

return false;

}

else

{

return true;

}

}

public override string ToString()

{

string gradesS = "";

foreach (int grade in this.Grades)

{

gradesS += grade.ToString() + " ";

}

string bs = String.Format("Студент: \n" +

"\tИмя: {0} {1}\n" +

"\tДата рождения: {2}\n" +

"\tID: {3}\n" +

"\tФакультет: {4}\n" +

"\tГруппа: {5}\n" +

"\tОценки: {6}\n", this.Firstname, this.Lastname, this.Birthdate.ToString("dd.MM.yyyy"), this.Id, this.Faculty, this.Group, (gradesS == "" ? "отсутствуют" : gradesS));

return bs;

}

public override float GetSalary()

{

return 0;

}

}

}

Листинг 2 – Файл Student.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Homework

{

class Teacher : Person

{

public string Faculty { get; set; }

public string Post { get; set; }

public int WorkExperience { get; set; }

public Teacher(int id, string lastname, string firstname, DateTime birthdate, string username, string password, int workExperience, string faculty, string post) : base(id, lastname, firstname, birthdate, username, password, Permissions.teacher)

{

WorkExperience = workExperience;

Faculty = faculty;

Post = post;

}

public bool AddWorkExperience(int workYears)

{

if (workYears > 0)

{

this.WorkExperience += workYears;

return true;

}

else

{

return false;

}

}

public bool RemoveWorkExperience(int workYears)

{

if (workYears <= this.WorkExperience && workYears > 0)

{

this.WorkExperience -= workYears;

return true;

}

else

{

return false;

}

}

public override string ToString()

{

string bs = String.Format("Преподаватель: \n" +

"\tИмя: {0} {1}\n" +

"\tДата рождения: {2}\n" +

"\tID: {3}\n" +

"\tФакультет: {4}\n" +

"\tДолжность: {5}\n", this.Firstname, this.Lastname, this.Birthdate.ToString("dd.MM.yyyy"), this.Id, this.Faculty, this.Post);

return bs;

}

override public float GetSalary()

{

return this.WorkExperience \* 15000;

}

}

}

Листинг 3 – Файл Teacher.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Homework

{

class Manager : Person

{

public string Faculty { get; set; }

public string Post { get; set; }

public Manager(int id, string lastname, string firstname, DateTime birthdate, string username, string password, string faculty, string post) : base(id, lastname, firstname, birthdate, username, password, Permissions.manager)

{

Faculty = faculty;

Post = post;

}

override public float GetSalary()

{

return 45000;

}

}

}

Листинг 4 – Файл Manager.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Homework

{

class Admin : Person

{

public string Laboratory { get; }

public Admin(int id, string lastname, string firstname, DateTime birthdate, string username, string password, string laboratory) : base(id, lastname, firstname, birthdate, username, password, Permissions.admin)

{

this.Laboratory = laboratory;

}

public override string ToString()

{

string bs = String.Format("Администратор: \n" +

"\tИмя: {0} {1}\n" +

"\tДата рождения: {2}\n" +

"\tID: {3}\n" +

"\tМесто работы: {4}\n", this.Firstname, this.Lastname, this.Birthdate.ToString("dd.MM.yyyy"), this.Id, this.Laboratory);

return bs;

}

override public float GetSalary()

{

return 150000;

}

}

}

Листинг 5 – Admin.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Homework

{

class Laboratory

{

public string LaboratoryName { get; set; }

private Admin Admin = null;

private List<Manager> Managers = new List<Manager>();

private List<Teacher> Teachers = new List<Teacher>();

private List<Student> Students = new List<Student>();

public Laboratory(string laboratoryName)

{

LaboratoryName = laboratoryName;

Admin = new Admin(264882, "Мороз", "Илья", new DateTime(2001, 08, 08), "admin", "admin", LaboratoryName);

}

public Admin GetAdmin()

{

return this.Admin;

}

public List<Student> GetStudents()

{

return this.Students;

}

public void AddStudent(Student Student)

{

this.Students.Add(Student);

}

public List<Teacher> GetTeachers()

{

return this.Teachers;

}

public void AddTeacher(Teacher Teacher)

{

this.Teachers.Add(Teacher);

}

public List<Manager> GetManagers()

{

return this.Managers;

}

public void AddManager(Manager Manager)

{

this.Managers.Add(Manager);

}

public List<Person> GetPeople()

{

List<Person> People = new List<Person>();

People.Add(this.Admin);

People.AddRange(this.Students);

People.AddRange(this.Teachers);

People.AddRange(this.Managers);

return People;

} //Не забывать добавлять новые списки

public void PrintInfoAboutLaboratory()

{

Console.Clear();

Console.WriteLine("--- {0} ---", this.LaboratoryName);

Console.WriteLine("Администратор: ");

Console.WriteLine("\t{0} {1}, ID: {2}", this.Admin.Lastname, this.Admin.Firstname, this.Admin.Id);

if (Managers.Count != 0)

{

Console.WriteLine("Менеджеры лаборатории: ");

foreach (Manager manager in Managers)

{

Console.WriteLine("\t{0} {1}, ID: {2}, Факультет: {3}", manager.Lastname, manager.Firstname, manager.Id, manager.Faculty);

}

}

if (Teachers.Count != 0)

{

Console.WriteLine("Преподаватели: ");

foreach (Teacher teacher in Teachers)

{

Console.WriteLine("\t{0} {1}, ID: {2}, Факультет: {3}", teacher.Lastname, teacher.Firstname, teacher.Id, teacher.Faculty);

}

}

if (Students.Count != 0)

{

Console.WriteLine("Студенты: ");

foreach (Student student in Students)

{

Console.WriteLine("\t{0} {1}, ID: {2}, Группа: {3}", student.Lastname, student.Firstname, student.Id, student.Group);

}

}

Console.ReadKey();

}

}

}

Листинг 6 – Файл Laboratory.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Homework

{

class Program

{

public static Person CurrentUser = null;

static void Main(string[] args)

{

Console.SetWindowSize(55, 30);

Laboratory lab1 = new Laboratory("Лаборатория 1");

while (true)

{

PrintMenu();

MakeChoice(ref lab1);

}

}

static void AddStudent(ref Laboratory laboratory)

{

Console.Clear();

if (CurrentUser != null && (CurrentUser.CurrentPermissions == Person.Permissions.admin || CurrentUser.CurrentPermissions == Person.Permissions.manager))

{

Console.Write("Введите ID студента: ");

int id = int.Parse(Console.ReadLine());

Console.Write("Введите фамилию студента: ");

string lastname = Console.ReadLine();

Console.Write("Введите имя студента: ");

string firstname = Console.ReadLine();

Console.Write("Введите дату рождения студента (dd.MM.yyyy): ");

string birthdateS = Console.ReadLine();

DateTime birthdate = DateTime.ParseExact(birthdateS, "dd.MM.yyyy", null);

string username = id.ToString();

string password = "firstpassword";

Console.Write("Введите факультет студента: ");

string faculty = Console.ReadLine();

Console.Write("Введите учебную группу студента: ");

string group = Console.ReadLine();

laboratory.AddStudent(new Student(id, lastname, firstname, birthdate, username, password, faculty, group));

Console.WriteLine("Студент успешно добавлен!");

Console.ReadKey();

}

else

{

Console.WriteLine("У Вас недостаточно прав!");

Console.ReadKey();

}

}

static void AddTeacher(ref Laboratory laboratory)

{

Console.Clear();

if (CurrentUser != null && (CurrentUser.CurrentPermissions == Person.Permissions.admin || CurrentUser.CurrentPermissions == Person.Permissions.manager))

{

Console.Write("Введите ID преподавателя: ");

int id = int.Parse(Console.ReadLine());

Console.Write("Введите фамилию преподавателя: ");

string lastname = Console.ReadLine();

Console.Write("Введите имя преподавателя: ");

string firstname = Console.ReadLine();

Console.Write("Введите дату рождения преподавателя (dd.MM.yyyy): ");

string birthdateS = Console.ReadLine();

DateTime birthdate = DateTime.ParseExact(birthdateS, "dd.MM.yyyy", null);

string username = id.ToString();

string password = "firstpassword";

Console.Write("Введите стаж работы: ");

int workExperience = int.Parse(Console.ReadLine());

Console.Write("Введите факультет преподавателя: ");

string faculty = Console.ReadLine();

Console.Write("Введите должность преподавателя: ");

string post = Console.ReadLine();

laboratory.AddTeacher(new Teacher(id, lastname, firstname, birthdate, username, password, workExperience, faculty, post));

Console.WriteLine("Преподаватель успешно добавлен!");

Console.ReadKey();

}

else

{

Console.WriteLine("У Вас недостаточно прав!");

Console.ReadKey();

}

}

static void AddManager(ref Laboratory laboratory)

{

Console.Clear();

if (CurrentUser != null && (CurrentUser.CurrentPermissions == Person.Permissions.admin || CurrentUser.CurrentPermissions == Person.Permissions.manager))

{

Console.Write("Введите ID менеджера: ");

int id = int.Parse(Console.ReadLine());

Console.Write("Введите фамилию менеджера: ");

string lastname = Console.ReadLine();

Console.Write("Введите имя менеджера: ");

string firstname = Console.ReadLine();

Console.Write("Введите дату рождения менеджера (dd.MM.yyyy): ");

string birthdateS = Console.ReadLine();

DateTime birthdate = DateTime.ParseExact(birthdateS, "dd.MM.yyyy", null);

string username = id.ToString();

string password = "firstpassword";

Console.Write("Введите факультет менеджера: ");

string faculty = Console.ReadLine();

Console.Write("Введите должность менеджера: ");

string post = Console.ReadLine();

laboratory.AddManager(new Manager(id, lastname, firstname, birthdate, username, password, faculty, post));

Console.WriteLine("Менеджер успешно добавлен!");

Console.ReadKey();

}

else

{

Console.WriteLine("У Вас недостаточно прав!");

Console.ReadKey();

}

}

static Person Login(ref Laboratory laboratory)

{

Console.Clear();

bool IsLoginCorrect = false;

bool IsPasswordCorrect = false;

bool IsExit = false;

Person currPerson = null;

while (!IsLoginCorrect && !IsExit)

{

Console.Write("Логин (exit - выход в главное меню): ");

string username = Console.ReadLine();

if (username == "exit")

{

IsExit = true;

return null;

}

else

{

foreach (Person person in laboratory.GetPeople())

{

if (person.Username == username)

{

currPerson = person;

IsLoginCorrect = true;

}

}

if (!IsLoginCorrect)

{

Console.WriteLine("Неверный логин! Попробуйте снова.");

}

}

}

if (currPerson.Password == "firstpassword")

{

Console.Write("Кажется, Вы входите в первый раз. \nВведите свой новый пароль: ");

string newPassword = Console.ReadLine();

currPerson.ChangePassword(newPassword);

}

while (!IsPasswordCorrect)

{

Console.Write("{0}, введите Ваш пароль: ", currPerson.Firstname);

string password = Console.ReadLine();

if (currPerson.Password == password)

{

IsPasswordCorrect = true;

return currPerson;

}

else

{

Console.WriteLine("Неверный пароль! Попробуйте снова.");

}

}

return null;

}

static void Logout()

{

CurrentUser = null;

}

static void PrintMenu()

{

Console.Clear();

if (CurrentUser == null)

{

Console.WriteLine("1 - Вход в систему");

Console.WriteLine("\n0 - Выход");

}

else

{

Console.WriteLine("Здравствуйте, {0}\n", CurrentUser.Firstname);

Console.WriteLine("1 - Посмотреть информацию о себе");

Console.WriteLine("2 - Посмотреть информацию о лаборатории");

Console.WriteLine("3 - Рассчитать свою з/п");

if (CurrentUser.CurrentPermissions == Person.Permissions.teacher)

{

Console.WriteLine("\n4 - Выставить оценки студенту");

}

if (CurrentUser.CurrentPermissions == Person.Permissions.manager || CurrentUser.CurrentPermissions == Person.Permissions.admin)

{

Console.WriteLine("\n4 - Добавить учителя");

Console.WriteLine("5 - Удалить учителя");

Console.WriteLine("6 - Добавить студента");

Console.WriteLine("7 - Удалить студента");

}

if (CurrentUser.CurrentPermissions == Person.Permissions.admin)

{

Console.WriteLine("\n8 - Добавить менеджера");

Console.WriteLine("9 - Удалить менеджера");

}

Console.WriteLine("\n0 - Выход из системы");

}

}

static void MakeChoice(ref Laboratory laboratory)

{

int choice = int.Parse(Console.ReadLine());

bool isCorrect = false;

if (CurrentUser == null)

{

switch (choice)

{

case 1:

isCorrect = true;

CurrentUser = Login(ref laboratory);

break;

case 0:

isCorrect = true;

Environment.Exit(0);

break;

default:

if (!isCorrect)

{

Console.WriteLine("Некорректный ввод! \nНажмите любую клавишу и попробуйте снова.");

Console.ReadKey();

}

break;

}

}

else

{

switch (choice)

{

case 1:

isCorrect = true;

Console.Clear();

CurrentUser.Print();

Console.ReadKey();

break;

case 2:

isCorrect = true;

laboratory.PrintInfoAboutLaboratory();

break;

case 3:

isCorrect = true;

Console.Clear();

if (CurrentUser.CurrentPermissions == Person.Permissions.student)

{

Console.WriteLine("Простите, у Вас нет зарплаты");

}

else

{

Console.WriteLine("Ваша зарплата {0} руб.", CurrentUser.GetSalary());

}

Console.ReadKey();

break;

case 0:

isCorrect = true;

Logout();

break;

default:

if (CurrentUser.CurrentPermissions == Person.Permissions.teacher)

{

switch (choice)

{

case 4:

isCorrect = true;

AddGrade(ref laboratory);

break;

}

}

else if (CurrentUser.CurrentPermissions == Person.Permissions.manager || CurrentUser.CurrentPermissions == Person.Permissions.admin)

{

switch (choice)

{

case 4:

isCorrect = true;

AddTeacher(ref laboratory);

break;

case 5:

isCorrect = true;

DeleteTeacher(ref laboratory);

break;

case 6:

isCorrect = true;

AddStudent(ref laboratory);

break;

case 7:

isCorrect = true;

DeleteStudent(ref laboratory);

break;

default:

if (CurrentUser.CurrentPermissions == Person.Permissions.admin)

{

switch (choice)

{

case 8:

isCorrect = true;

AddManager(ref laboratory);

break;

case 9:

isCorrect = true;

DeleteManager(ref laboratory);

break;

default:

break;

}

}

break;

}

}

if (!isCorrect)

{

Console.WriteLine("Некорректный ввод! \nНажмите любую клавишу и попробуйте снова.");

Console.ReadKey();

}

break;

}

}

}

static int ChoiceStudent(ref List<Student> students)

{

Console.Clear();

int index = 1;

int choice = 0;

foreach (Student student in students)

{

Console.WriteLine("{0}. {1} {2}", index++, student.Lastname, student.Firstname);

}

do

{

Console.Write("Choice ID: ");

choice = Int32.Parse(Console.ReadLine());

} while (!(choice > 0 && choice < index));

return choice;

}

static void DeleteStudent(ref Laboratory laboratory)

{

List<Student> students = laboratory.GetStudents();

int deleteIndex = ChoiceStudent(ref students);

students.RemoveAt(--deleteIndex);

}

static int ChoiceTeacher(ref List<Teacher> teachers)

{

Console.Clear();

int index = 1;

int choice = 0;

foreach (Teacher teacher in teachers)

{

Console.WriteLine("{0}. {1} {2}", index++, teacher.Lastname, teacher.Firstname);

}

do

{

Console.Write("Choice ID: ");

choice = Int32.Parse(Console.ReadLine());

} while (!(choice > 0 && choice < index));

return choice;

}

static void DeleteTeacher(ref Laboratory laboratory)

{

List<Teacher> teachers = laboratory.GetTeachers();

int deleteIndex = ChoiceTeacher(ref teachers);

teachers.RemoveAt(--deleteIndex);

}

static int ChoiceManager(ref List<Manager> managers)

{

Console.Clear();

int index = 1;

int choice = 0;

foreach (Manager manager in managers)

{

Console.WriteLine("{0}. {1} {2}", index++, manager.Lastname, manager.Firstname);

}

do

{

Console.Write("Choice ID: ");

choice = Int32.Parse(Console.ReadLine());

} while (!(choice > 0 && choice < index));

return choice;

}

static void DeleteManager(ref Laboratory laboratory)

{

List<Manager> managers = laboratory.GetManagers();

int deleteIndex = ChoiceManager(ref managers);

managers.RemoveAt(--deleteIndex);

}

static void AddGrade(ref Laboratory laboratory)

{

List<Student> students = laboratory.GetStudents();

int choiceIndex = ChoiceStudent(ref students);

Student student = students.ElementAt(--choiceIndex);

Console.Clear();

Console.Write("Введите новую оценку для ID:{0}: ", student.Id);

student.AddGrade(Int32.Parse(Console.ReadLine()));

}

}

}

Листинг 7 – Файл Program.cs

**Вывод:**

В ходе работы были укреплены навыки работы с классами в языке программирования C#. Были созданы 6 классов, реализованы наследование, переопределение методов, перегрузки конструкторов.