**Санкт-петербургский национальный исследовательский университет информационных технологий, механики и оптики**

Факультет инфокоммуникационных технологий

**Домашняя работа**

**Иерархия классов учебного центра**

Выполнил: Труфанов

Егор Владимирович

Группа № К3122

Проверила: Казанова П.П.

Санкт-Петербург

2020

**Цель работы:**

Реализовать иерархию классов учебного центра.

**Ход работы:**

Необходимо реализовать иерархию классов учебного центра. Для этого сначала создадим абстрактный класс **Person**, который будет базовым для остальных классов, у которых и будут создаваться объекты. В нем объявлены поля содержащие информацию об имени, фамилии и даты рождения сотрудника учебного центра, а также два метода: виртуальный метод **Print**, который в дальнейшем будет переопределяться в производных классах, и метод **Age** по определению возраста сотрудника (Рисунок 1). Также создадим интерфейс **IEmployee** с методом определения заработной платы (Рисунок 2). Далее создадим производные классы от базового класса **Person**: **Student**, **Teacher**, **Manager** и **Admin**. В классе **Student** присутсвуют только констуктор и переопределнный метод **Print** (Рисунок 3). В классе **Teacher** есть констуктор (Рисунок 4), переопределенный метод **Print**, а также методы по определению стажа и заработной платы (Рисунок 5). В производном классе **Manager** находятся конструткор класса, переопределенный метод **Print** и метод по определению заработной платы (Рисунок 6). В классе **Admin** все соответсвенно так же, как и в классе **Manager** (Рисунок 7). В главном методе **Main** класса **Program** создаются объекты классов для наглядности и затем сохраняются в списке объектов (Рисунок 8). Далее требуется добавить новых сотрудников в список. Это осуществляется в блоке **try-catch**, чтобы при ошибке пользователь мог снова ввести данные. Пользователь вводит первоначальные данные о сотруднике (имя, фамилия и дата рождения), затем выбирает кем является добавляемый сотрудник (студентом, пеподавателем, менеджером или же администратором) и вводит оставшуюся информацию о нем (Рисунок 9). После этого программа добавляет сотрудника в список. После того, как пользователь закончит добавлять сотрудников, программа выведет весь список сотрудников (Рисунок 10) и затем предложит провести поиск по возрасту, попросив задать границы поиска (Рисунок 11). После этого все классы и интерфейс были перемещены в библиотку **Library** (Рисунок 12). Пример того, как работает программа, можно наблюдать на Рисунках 13-16. Диаграмма классов, задействованных в программе, изображена на Рисунке 17.

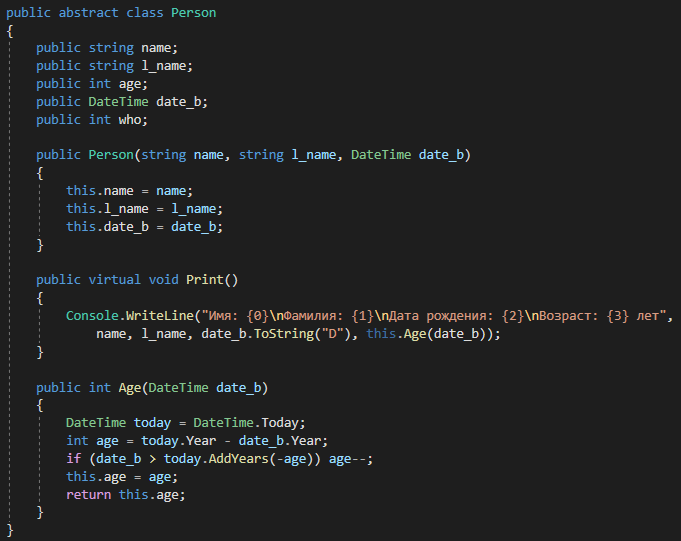


Рисунок 1 - Базовый класс **Person**

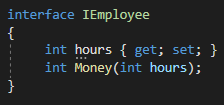


Рисунок 2 - Интерфейс **IEmployee**

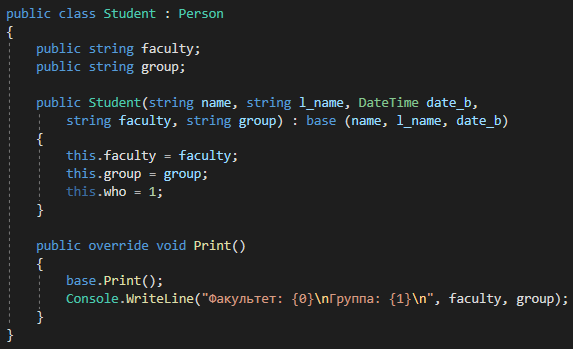


Рисунок 3 - Производный класс **Student**

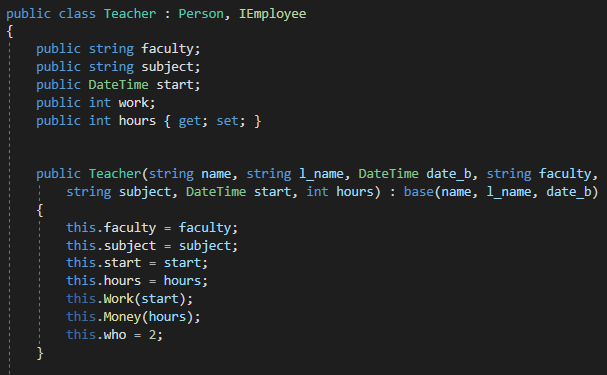


Рисунок 4 - Конструктор класса **Teacher**

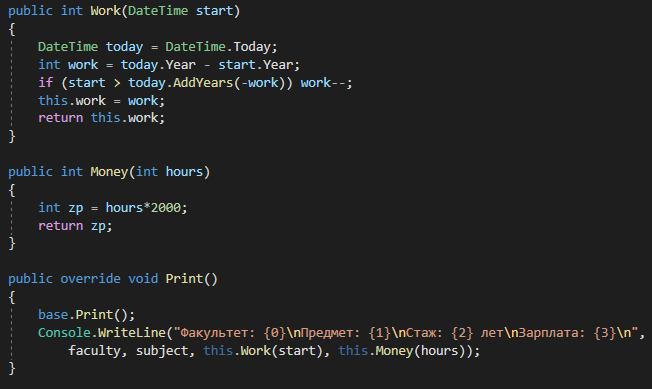


Рисунок 5 - Методы класса **Teacher**

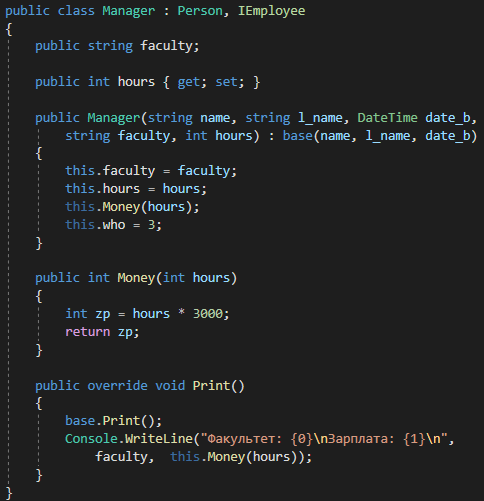


Рисунок 6 - Производный класс **Manager**

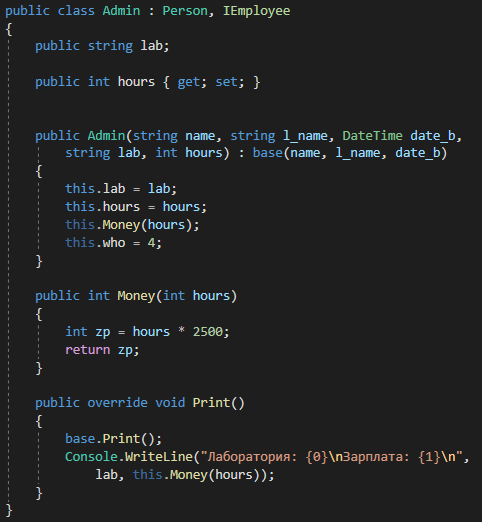


Рисунок 7 - Производный класс **Admin**

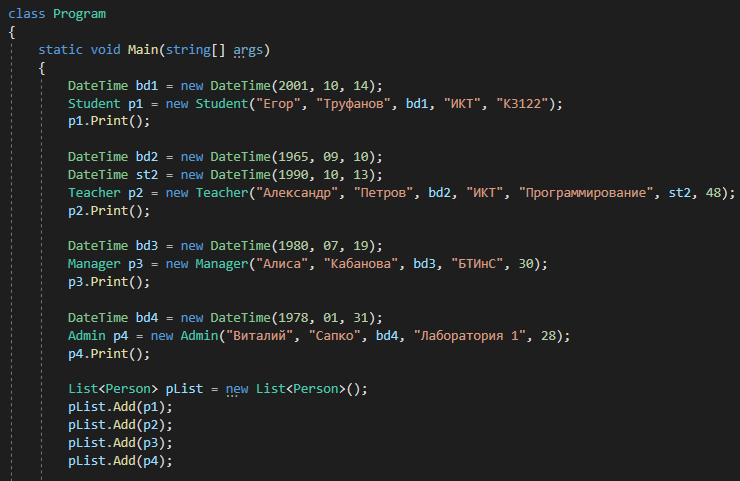


Рисунок 8 - Создание и добавление в список объектов



Рисунок 9 - Добавление новых объектов

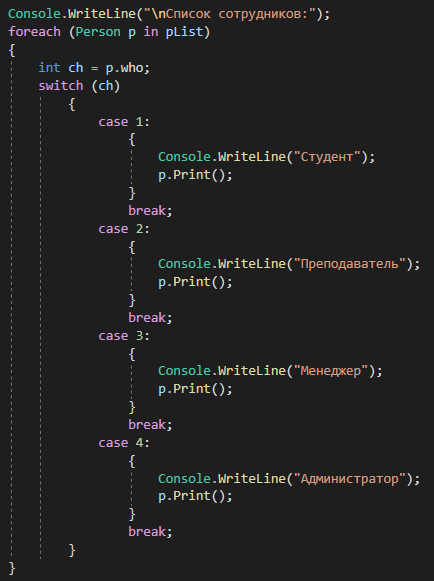


Рисунок 10 – Код, осуществляющий вывод списка

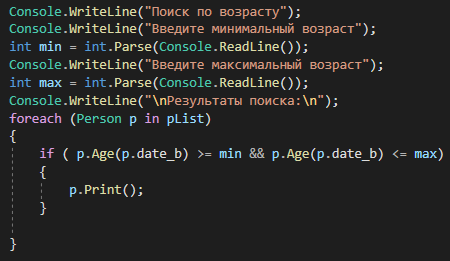


Рисунок 11 - Код, осуществляющий поиск сотрудников по возрасту

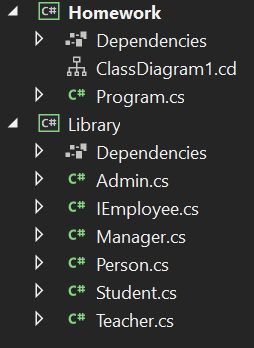


Рисунок 12 - Демонстрация библиотеки **Library**

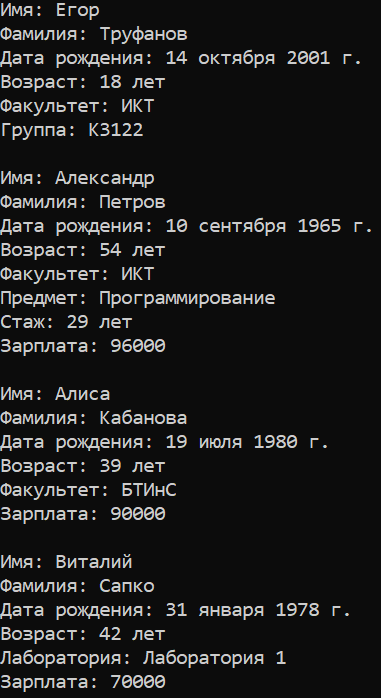


Рисунок 13 - Вывод информации о сотрудниках

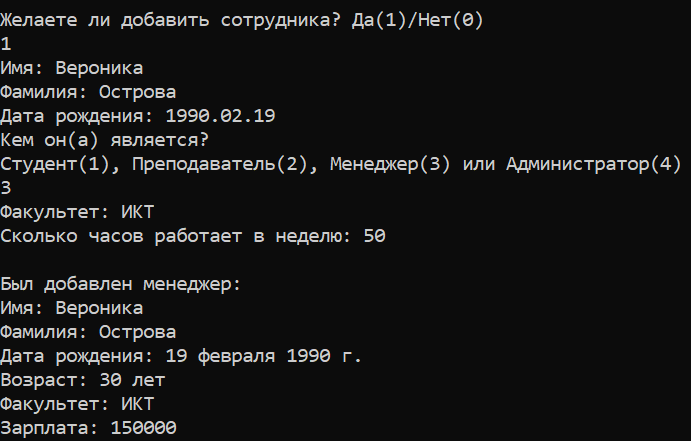


Рисунок 14 - Добавление сотрудника

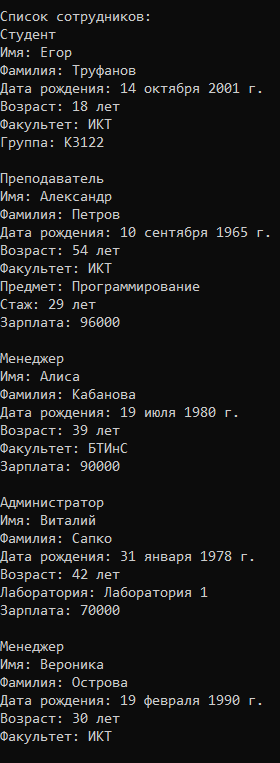


Рисунок 15 - Вывод списка сотрудников

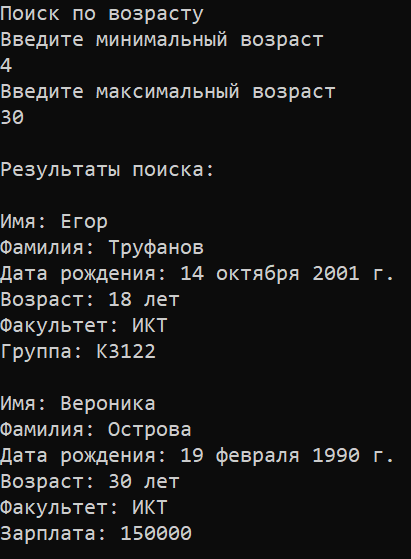


Рисунок 16 - Поиск сотрудников по возрасту

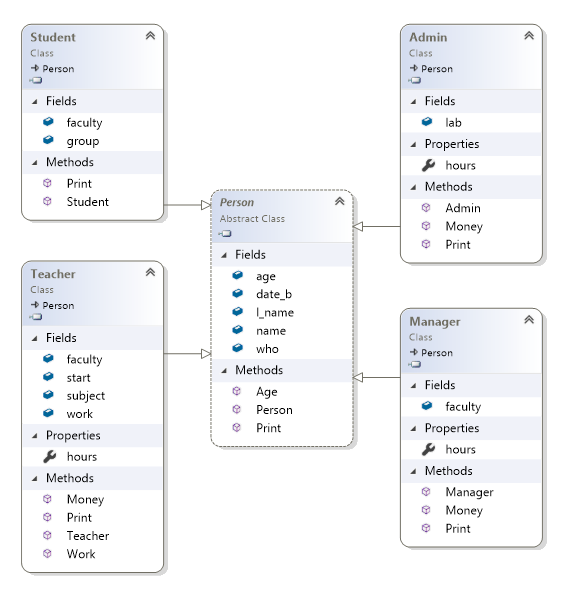


Рисунок 17 - Диаграмма классов

**Вывод:**

В результате работы была реализована иерархия классов учебного центра.