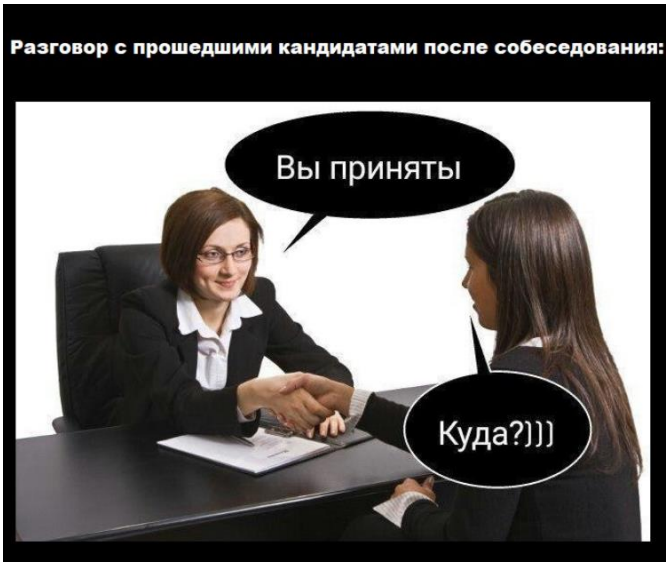

Практическая работа № 6

В вашем городе решили открыть новый завод. Что именно он будет производить, пока не ясно, но ясно, что руководству завода срочно нужны сотрудники в подразделения трех типов: энергетический тип, ремонтно-механический тип и диспетчерский тип.

У каждого типа подразделений свои собственные требования к кандидатам на должности:

- энергетические подразделения: специальность «электрик», средний балл не ниже 4.5;
- ремонтно-механические подразделения: специальность «механик», средний балл не ниже 4.0, не старше 35 лет;
- диспетчерские подразделения: специальность «математик» или «программист», средний балл не ниже 4.8, не моложе 22 лет.

На завод уже поступило достаточное количество резюме, поэтому следует провести процесс отбора кандидатов.

Описание классов

Класс `Person` содержит следующие автосвойства:

- `string` *Name (имя)*;
- `int` *Age (возраст)*;
- `Speciality` *PersonSpeciality (специальность)*;
- `double` *Score (средний балл)*.

Здесь `Specialty` – это справочник (`enum`), который содержит ряд специальностей:

- *Electrician*
- *Mechanic*
- *Mathematician*
- *Programmer*
- *Lawyer*

Класс `Factory` содержит автосвойства:

- `List<Department>` *Departments (коллекция подразделений);*
- `List<Person>` *Candidates (коллекция кандидатов на должности).*

Класс `Department` содержит автосвойства:

- `string` *Title (название подразделения);*
- `List<Person>` *Employees (список сотрудников);*
- `int` *NumberOfVacancies (количество вакансий).*

Также этот класс содержит виртуальный метод `void` *StaffSelection(*`List<Person>` *candidates)*. В базовой реализации этот метод помещает в коллекцию *Employees* объект коллекции *Candidates* со средним баллом не ниже 3.0. При этом из коллекции *Candidates* “трудоустроенный” объект удаляется.

У класса `Department` есть три наследника:

- `class ElectricianDepartment`,
- `class MechanicDepartment`,
- `class InformDepartment`.

Каждый класс-наследник переопределяет метод `void` *StaffSelection(*`List<Person>` *candidates)* в соответствии с описанными ранее требованиями к сотрудникам.

В работе создайте 10 объектов класса `Person` с различными характеристиками и поместите их в коллекцию `List<Person>` *Candidates* объекта класса `Factory`**.** Создайте по одному объекту классов `ElectricDepartment`, `MechanicDepartment`, `InformDepartment` и поместите их в коллекцию `List<Department>` *Departments* объекта класса `Factory`**.** У элементов коллекции `List<Department>` *Departments* поочередно вызовите метод `void` *staffSelection(*`List<Person>` *candidates).* Для демонстрации результатов выведите на консоль сотрудников каждого подразделения и нетрудоустроенных кандидатов.