Для всех задач необходимо дополнительно создавать класс Application, который будет содержать статический метод main.

**Задача 1**

Реализовать класс PhoneGenerator, содержащий метод generate. Метод должен возвращать случайно сгенерированную строку, содержащую номер телефона в формате +7(ххх)ххх-хх-хх.

Создать класс NameGenerator, содержащий метод generate. Метод должен возвращать случайно сгенерированную строку, содержащую последовательность символов кириллицы. Первая буква – заглавная, остальные строчные. Длина выбирается случайно от 5 до 12 символов.

Создать класс AgeGenerator, содержащий метод generate. Метод должен возвращать случайно сгенерированное число в диапазоне [20..75].

Создать класс Human с полями, определяющими фамилию, имя, возраст и номер телефона человека.

В методе main необходимо создать список (List) людей из 500 объектов. При заполнении генерировать поля объектов класса Human с помощью 3 классов-генераторов, созданных ранее. Вывести на экран информацию о тех людях, чей возраст равен 35 годам, в формате (каждую запись с новой строки):

<Фамилия> <Имя>, возраст <возраст>, тел. <телефон>.

**Задача 2**

Создать класс PhoneType, являющийся перечислением типа телефона. Телефон может быть: домашним, рабочим, мобильным.

Создать класс Chance, содержащий метод calculateChance(int percent). Метод должен возвращать логическую переменную boolean и принимать целое число percent. В самом методе реализовать логику, которая возвращает true, если шанс выпал положительно, и false в противоположном случае. Например, вызов метода calculateChance(80) должен вернуть true с шансом 80% и false с шансом в 20%

Создать пакет generators, в который необходимо поместить все классы, реализующие генерацию.

Методы PhoneGenerator, NameGenerator и AgeGenerator создать аналогично задаче 1 (рекомендую создать их заново, без копирования☺).

Создать класс Human, содержащий следующие поля: фамилия, имя, возраст. Также предусмотреть в классе Human структуру данных, которая позволит хранить коллекцию связок «Тип телефона» - «номер телефона». У человека может быть одновременно несколько типов телефонов, могут быть все типы телефонов, может быть ни одного. Для каждого конкретного типа предусмотрен максимум 1 телефон.

Реализовать класс HumanGenerator, содержащий метод generate, который генерирует человека. Для генерации использовать созданные ранее классы генераторов. Коллекцию телефонов генерировать по следующему принципу: Шанс наличия каждого типа – 20% независимо друг от друга.

В методе main реализовать создание множества (Set), содержащий 2000 сгенерированных людей. На экран необходимо в произвольном формате вывести информацию о тех людях, у которых есть одновременно три номера телефона (суммарный шанс данного события – 0.8%). После этого удалить из множества всех людей, у которых есть хотя бы один из телефонов. Вывести на экран количество всех людей, которые остались в множестве.

**Задача 3**

Создать класс DayOfWeek, описывающий дни недели. Дни недели должны быть описаны с понедельника по субботу.

Создать класс Lesson, описывающий тип урока. Список возможных уроков – математика, физика, программирование, история, биология, география. Также в классе реализовать метод, генерирующий случайный список уроков (от 4 до 6)

Создать класс Scholar, описывающий некоторого ученика школы, с единственным полем diary. Поле diary должно представлять собой коллекцию, содержащую информацию в виде «День недели» - «Уроки в этот день недели».

В методе main создать объект ученика со случайным расписанием уроков (6 дней в неделю, 4-6 уроков каждый день). Вывести расписание на экран в следующем формате:

<День недели>: <Урок1>, <Урок2>, <Урок3>, <Урок4>, <Урок5(если есть)>, <Урок6(если есть)>

Также в методе main создать коллекцию, содержащую информацию в виде «Тип урока» - «Количество уроков в неделе», содержащий список уроков в порядке убывания их количества за неделю. Вывести на экран эту коллекцию в виде

<Тип урока>: <количество уроков>

**Задача 4**

Создать класс IdentityDocumentType, описывающий тип документа, удостоверяющий личность. Возможные типы документов:

|  |  |  |
| --- | --- | --- |
| Код | Наименование документа | Шаблон серии и номера |
| 07 | Военный билет | ББ 0999999 |
| 09 | Дипломатический паспорт | 99 9999999 |
| 10 | Паспорт иностранного гражданина | SSSSSSSSSSSSSSSSSSSSSSSSS (до 25 символов включительно) |
| 21 | Паспорт гражданина Российской Федерации | 99 99 999999 |
| 22 | Загранпаспорт гражданина Российской Федерации | 99 9999999 |

Используются следующие обозначения:

9 - любая десятичная цифра (обязательная);

0 - любая десятичная цифра (необязательная, может отсутствовать);

Б - любая русская заглавная буква;

S - символ не контролируется (может содержать любую латинскую букву, цифру);

Создать класс IdGenerator с методом, принимающим в качестве параметра IdentityDocument и возвращающий случайную строку согласно шаблона серии и номера.

Создать класс IdentityDocument с полями, описывающими тип документа и серию и номер. Серию и номер описывать одним полем. Также в классе реализовать статический метод random(), возвращающий случайный тип документа со случайными серией и номер.

Создать класс NameGenerator, содержащий метод generate. Метод должен возвращать случайно сгенерированную строку, содержащую последовательность латинских символов. Первая буква – заглавная, остальные строчные. Длина выбирается случайно от 5 до 12 символов.

Создать класс Person с полями, описывающими фамилию, имя, документ, удостоверяющий личность. В классе Person реализовать статический метод random(), возвращающий сгенерированного человека.

В методе main необходимо создать множество из случайно сгенерированных 500 людей. В консоль необходимо вывести информацию в формате:

<Наименование вида документа> (<Код документа>): <Количество человек>

где количество человек соответствует числу людей с данным типом документа. Каждую запись вывести с новой строки, в порядке убывания кода документа.

После вывода оставить в множестве только тех людей, у которых код документа, удостоверяющего личность, равен «21». Вывести в консоль количество всех людей в получившемся множестве