## **Задание 10**

Распакуйте архив “Задание 10.zip” в ваш рабочий каталог. В Вашем проекте после этого должны появиться классы TestGroup, TestSchool, TestTraineeMap, TestTraineeQueue в каталоге test/java в пакете net.thumbtack.school.ttschool и класс TestMatrixSimilarRows в каталоге test/java в пакете net.thumbtack.school.matrix.

На этом занятии мы будем работать с различными коллекциями из стандартной библиотеки Java. Для этого нам понадобятся следующие классы, которые разместим в пакете net.thumbtack.school.ttschool :

**класс Trainee - студент Школы программиста**

Этот класс у нас уже есть.

**класс Group - группа Школы программиста**

В этом классе должны быть поля “название группы” (текстовая строка) , название аудитории (текстовая строка, предполагается, что аудитория закреплена за группой постоянно) и список студентов типа List<Trainee>. Название группы и номер аудитории не могут быть null или пустой строкой.

Класс должен содержать следующие конструкторы и методы

1. **public Group(String name, String room)**

Создает Group с указанными значениями полей и пустым списком студентов. Для недопустимых значений входных параметров выбрасывает TrainingException с соответствующим TrainingErrorCode

1. **public String getName()**

Возвращает имя группы

1. **public void setName(String name)**

Устанавливает имя группы. Для недопустимого значения входного параметра выбрасывает TrainingException с TrainingErrorCode.GROUP\_WRONG\_NAME (здесь и далее добавляйте новые коды ошибок в TrainingErrorCode)

1. **public String getRoom()**

Возвращает название аудитории

1. **public void setRoom(String room)**

Устанавливает название аудитории. Для недопустимого значения входного параметра выбрасывает TrainingException с TrainingErrorCode.GROUP\_WRONG\_ROOM

1. **public List<Trainee> getTrainees()**

Возвращает список учащихся.

1. **public void addTrainee(Trainee trainee);**

Добавляет Trainee в группу.

1. **public void removeTrainee(Trainee trainee);**

Удаляет Trainee из группы. Если такого Trainee в группе нет, выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND

1. **public void removeTrainee(int index);**

Удаляет Trainee с данным номером в списке из группы. Если номер не является допустимым, выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND

1. **public Trainee getTraineeByFirstName(String firstName);**

Возвращает первый найденный в списке группы Trainee, у которого имя равно firstName. Если такого Trainee в группе нет, выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND

1. **public Trainee getTraineeByFullName(String fullName);**

Возвращает первый найденный в списке группы Trainee, у которого полное имя равно fullName. Если такого Trainee в группе нет, выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND

1. **public void sortTraineeListByFirstNameAscendant();**

Сортирует список Trainee группы, упорядочивая его по возрастанию имени Trainee.

1. **public void sortTraineeListByRatingDescendant();**

Сортирует список Trainee группы, упорядочивая его по убыванию оценки Trainee.

1. **public void reverseTraineeList();**

Переворачивает список Trainee группы, то есть последний элемент списка становится начальным, предпоследний - следующим за начальным и т.д..

1. **public void rotateTraineeList(int positions);**

Циклически сдвигает список Trainee группы на указанное число позиций. Для положительного значения positions сдвигает вправо, для отрицательного - влево на модуль значения positions.

1. **public List<Trainee> getTraineesWithMaxRating();**

Возвращает список тех Trainee группы , которые имеют наивысшую оценку. Иными словами, если в группе есть Trainee с оценкой 5, возвращает список получивших оценку 5, если же таких нет, но есть Trainee с оценкой 4, возвращает список получивших оценку 4 и т.д. Для пустого списка выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND. Желательно сделать этот метод без сортировки и в один проход по списку.

1. **public boolean hasDuplicates();**

Проверяет, есть ли в группе хотя бы одна пара Trainee, для которых совпадают имя, фамилия и оценка.

1. **Методы equals и hashCode**

Не пишите эти методы сами, используйте средства IDEA.

**класс School - Школа программиста**

В этом классе должны быть поле “название школы” (текстовая строка) , год начала обучения (целое число) и множество групп типа Set<Group>. Название школы не может быть null или пустой строкой. В школе не может быть двух и более групп с одинаковым названием.

Класс должен содержать следующие конструкторы и методы

1. **public School(String name, int year)**

Создает School с указанными значениями полей и пустым множеством групп. Для недопустимых значений входных параметров выбрасывает TrainingException с соответствующим TrainingErrorCode

1. **public String getName()**

Возвращает название школы.

1. **public void setName(String name)**

Устанавливает название школы. Для недопустимого значения входного параметра выбрасывает TrainingException с TrainingErrorCode.SCHOOL\_WRONG\_NAME

1. **public int getYear()**

Возвращает год начала обучения.

1. **public void setYear(int year)**

Устанавливает год начала обучения.

1. **public Set<Group> getGroups()**

Возвращает список групп

1. **public void addGroup(Group group);**

Добавляет Group в школу. Если группа с таким именем уже есть, выбрасывает TrainingException с TrainingErrorCode.DUPLICATE\_GROUP\_NAME

1. **public void removeGroup(Group group);**

Удаляет Group из школы. Если такой Group в школе нет, выбрасывает TrainingException с TrainingErrorCode.GROUP\_NOT\_FOUND

1. **public void removeGroup(String name);**

Удаляет Group с данным названием из школы. Если группа с таким названием не найдена, выбрасывает TrainingException с TrainingErrorCode.GROUP\_NOT\_FOUND

1. **public boolean containsGroup(Group group);**

Определяет, есть ли в школе группа с таким названием.

1. **Методы equals и hashCode**

**класс TraineeMap**

Большинство студентов Школы программиста являются также студентами одного из институтов. Данный класс должен хранить информацию о всех парах студент - институт, независимо от того, в какую группу Школы входит студент . Эта информация должна храниться в виде Map<Trainee, String>, где String - название вуза. Если студент Школы не является студентом вуза, вместо названия вуза в этой таблице должна быть пустая строка.

Класс должен содержать следующие конструкторы и методы

1. **public TraineeMap()**

Создает TraineeMap с пустым Map.

1. **public void addTraineeInfo(Trainee trainee, String institute)**

Добавляет пару Trainee - String в Map. Если Map уже содержит информацию об этом Trainee, выбрасывает TrainingException с TrainingErrorCode.DUPLICATE\_TRAINEE.

1. **public void replaceTraineeInfo(Trainee trainee, String institute)**

Если в Map уже есть информация о данном Trainee, заменяет пару Trainee - String в Map на новую пару, иначе выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND.

1. **public void removeTraineeInfo(Trainee trainee)**

Удаляет информацию о Trainee из Map. Если Map не содержит информации о таком Trainee, выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND.

1. **public int getTraineesCount()**

Возвращает количество элементов в Map, иными словами, количество студентов.

1. **public String getInstituteByTrainee(Trainee trainee)**

Возвращает институт, в котором учится данный Trainee. Если Map не содержит информации о таком Trainee, выбрасывает TrainingException с TrainingErrorCode.TRAINEE\_NOT\_FOUND

1. **public Set<Trainee> getAllTrainees()**

Возвращает Set из всех имеющихся в Map Trainee.

1. **public Set<String> getAllInstitutes()**

Возвращает Set из всех институтов.

1. **public boolean isAnyFromInstitute(String institute)**

Возвращает true, если хоть один студент учится в этом institute, иначе false.

**класс TraineeQueue**

Во время обеденного перерыва студенты Школы иногда встают в очередь к имеющейся в офисе кофемашине. Класс TraineeQueue предназначен для работы с этой очередью.

Класс должен содержать следующие конструкторы и методы

1. **public TraineeQueue()**

Создает TraineeQueue с пустой Queue.

1. **public void addTrainee(Trainee trainee)**

Добавляет студента в очередь.

1. **public Trainee removeTrainee()**

Удаляет студента из очереди. Метод возвращает удаленного Trainee. Если в очереди никого нет, выбрасывает TrainingException с TrainingErrorCode.EMPTY\_TRAINEE\_QUEUE.

1. **public boolean isEmpty()**

Возвращает true, если очередь пуста, иначе false

**И на закуску…**

Дана целочисленная матрица, в которой имеется N строк, а число элементов в строке для каждой строки может быть любым, в том числе нулевым. Строки назовем похожими, если совпадают множества чисел, встречающихся в этих строках. Найти список строк этой матрицы максимальной размерности, в котором все строки попарно непохожи друг на друга. Из похожих строк в список включить строку с наибольшим номером. Порядок элементов в списке произвольный.

Пример. Матрица содержит 3 строки:

1 2 2 4 4

4 2 1 4

3 2 4 1 5 8

Первые 2 строки похожи друг на друга и непохожи на 3 строку. Ответом будет список из 2 и 3 строк.

Создайте класс MatrixNonSimilarRows (в пакете net.thumbtack.school.matrix), в котором должны быть следующие конструктор и метод

1. **public MatrixNonSimilarRows(int[][] matrix)**

Создает MatrixNonSimilarRows по заданной матрице.

1. **public List<int[]> getNonSimilarRows()**

Возвращает список непохожих строк.

Проверьте работу тестов в консольном окне, запишите все классы на сервер (не забудьте изменить текст сообщения в git commit!) и убедитесь, что на сервере все тесты также проходят успешно (см. Занятие 1, п.15-19)