## **Задание 9**

Распакуйте архив “Задания 9.zip” в Ваш рабочий каталог. В Вашем проекте после этого должен появиться класс TestFileService в каталоге test/java в пакете net.thumbtack.school.file.

На этом занятии мы создадим сервисный класс FileService для операций ввода-вывода, с помощью которого будем записывать различные данные в двоичные и текстовые файлы. Разместим этот класс в пакете net.thumbtack.school.file (создайте такой пакет). Этот класс не будет иметь своих данных и экземпляры этого класса нам не понадобятся, поэтому все методы этого класса будут иметь атрибут static. В случае каких-либо ошибок методы будут выбрасывать IOException (класс из стандартной библиотеки).

Для сериализации и десериализации в/из Json нам понадобится библиотека Gson (<https://github.com/google/gson/blob/master/UserGuide.md>). Для того, чтобы мы могли ее использовать, необходимо добавить в pom.xml следующую зависимость (в раздел <dependencies>)

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.5</version>

</dependency>

Постарайтесь при написании методов не дублировать код - вызывайте уже написанный метод, если это возможно.

1. **public static void writeByteArrayToBinaryFile(String fileName, byte[] array);**

Записывает массив байтов в двоичный файл, имя файла задается текстовой строкой.

1. **public static void writeByteArrayToBinaryFile(File file, byte[] array);**

Записывает массив байтов в двоичный файл, имя файла задается экземпляром класса File.

1. **public static byte[] readByteArrayFromBinaryFile(String fileName);**

Читает массив байтов из двоичного файла, имя файла задается текстовой строкой.

1. **public static byte[] readByteArrayFromBinaryFile(File file);**

Читает массив байтов из двоичного файла, имя файла задается экземпляром класса File.

1. **public static byte[] writeAndReadByteArrayUsingByteStream( byte[] array);**

Записывает массив байтов в ByteArrayOutputStream, создает на основе данных в полученном ByteArrayOutputStream экземпляр ByteArrayInputStream и читает из ByteArrayInputStream байты с четными номерами. Возвращает массив прочитанных байтов.

1. **public static void writeByteArrayToBinaryFileBuffered(String fileName, byte[] array);**

Записывает массив байтов в двоичный файл, используя буферизованный вывод, имя файла задается текстовой строкой.

1. **public static void writeByteArrayToBinaryFileBuffered(File file, byte[] array);**

Записывает массив байтов в двоичный файл, используя буферизованный вывод, имя файла задается экземпляром класса File.

1. **public static byte[] readByteArrayFromBinaryFileBuffered(String fileName);**

Читает массив байтов из двоичного файла, используя буферизованный ввод, имя файла задается текстовой строкой.

1. **public static byte[] readByteArrayFromBinaryFileBuffered(File file);**

Читает массив байтов из двоичного файла, используя буферизованный ввод, имя файла задается экземпляром класса File.

1. **public static void writeRectButtonToBinaryFile(File file, RectButton rectButton);**

Записывает RectButton в двоичный файл, имя файла задается экземпляром класса File. Состояние окна записывается в виде текстовой строки в формате UTF. Используйте DataOutputStream.

1. **public static RectButton readRectButtonFromBinaryFile(File file);**

Читает данные для RectButton из двоичного файла и создает на их основе экземпляр RectButton, имя файла задается экземпляром класса File. Предполагается, что данные в файл записаны в формате предыдущего упражнения. Используйте DataInputStream.

1. **public static void writeRectButtonArrayToBinaryFile(File file, RectButton[] rects );**

Записывает массив из RectButton в двоичный файл, имя файла задается экземпляром класса File. Поле состояния и текст не записываются.

1. **public static void modifyRectButtonArrayInBinaryFile(File file);**

В файле массива данных RectButton из предыдущего упражнения увеличивает на 1 значение x каждой точки каждого RectButton. Имя файла задается экземпляром класса File.

1. **public static RectButton[] readRectButtonArrayFromBinaryFile(File file);**

Читает данные, записанные в формате предыдущего упражнения и создает на их основе массив RectButton c с состоянием “ACTIVE” и текстом “OK”.

1. **public static void writeRectButtonToTextFileOneLine(File file, RectButton rectButton);**

Записывает RectButton в текстовый файл в одну строку, имя файла задается экземпляром класса File. Поля в файле разделяются пробелами.

1. **public static RectButton readRectButtonFromTextFileOneLine(File file);**

Читает данные для RectButton из текстового файла и создает на их основе экземпляр RectButton, имя файла задается экземпляром класса File. Предполагается, что данные в файл записаны в формате предыдущего упражнения.

1. **public static void writeRectButtonToTextFileSixLines(File file, RectButton rectButton);**

Записывает RectButton в текстовый файл В первые 4 строки записываются координаты (каждое число в отдельной строке) , в следующие 2 - состояние и текст. Имя файла задается экземпляром класса File.

1. **public static RectButton readRectButtonFromTextFileSixLines(File file);**

Читает данные для RectButton из текстового файла и создает на их основе экземпляр RectButton, имя файла задается экземпляром класса File. Предполагается, что данные в файл записаны в формате предыдущего упражнения.

1. **public static void writeTraineeToTextFileOneLine(File file, Trainee trainee);**

Записывает Trainee в текстовый файл в одну строку в кодировке UTF-8, имя файла задается экземпляром класса File. Имя, фамилия и оценка в файле разделяются пробелами.

1. **public static Trainee readTraineeFromTextFileOneLine(File file);**

Читает данные для Trainee из текстового файла и создает на их основе экземпляр Trainee, имя файла задается экземпляром класса File. Предполагается, что данные в файл записаны в формате предыдущего упражнения.

1. **public static void writeTraineeToTextFileThreeLines(File file, Trainee trainee);**

Записывает Trainee в текстовый файл в кодировке UTF-8, каждое поле в отдельной строке, имя файла задается экземпляром класса File.

1. **public static Trainee readTraineeFromTextFileThreeLines(File file);**

Читает данные для Trainee из текстового файла и создает на их основе экземпляр Trainee, имя файла задается экземпляром класса File. Предполагается, что данные в файл записаны в формате предыдущего упражнения.

1. **public static void serializeTraineeToBinaryFile(File file, Trainee trainee);**

Сериализует Trainee в двоичный файл, имя файла задается экземпляром класса File.

1. **public static Trainee deserializeTraineeFromBinaryFile(File file);**

Десериализует Trainee из двоичного файла, имя файла задается экземпляром класса File. Предполагается, что данные в файл записаны в формате предыдущего упражнения.

1. **public static String serializeTraineeToJsonString(Trainee trainee);**

Сериализует Trainee в формате Json в текстовую строку.

1. **public static Trainee deserializeTraineeFromJsonString(String json);**

Десериализует Trainee из текстовой строки с Json-представлением Trainee.

1. **public static void serializeTraineeToJsonFile(File file, Trainee trainee);**

Сериализует Trainee в формате Json в файл, имя файла задается экземпляром класса File.

1. **public static Trainee deserializeTraineeFromJsonFile(File file);**

Десериализует Trainee из файла с Json-представлением Trainee, имя файла задается экземпляром класса File.

Проверьте работу тестов в консольном окне, запишите все классы **и измененный pom.xml** 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)