Examen 1º EVA Lucía

# Lo que puede entrar:

* El configuration en un txt y un xml
* Textfiles
* XML Files
* Ejercicios en el main y con Inyección de dependencias.
* SQL Queries y modelos
* JDBC

## TXT y xml:

Hay dos clases en las que se configura, la clase configuration y el archivo txt o xml del que tira:

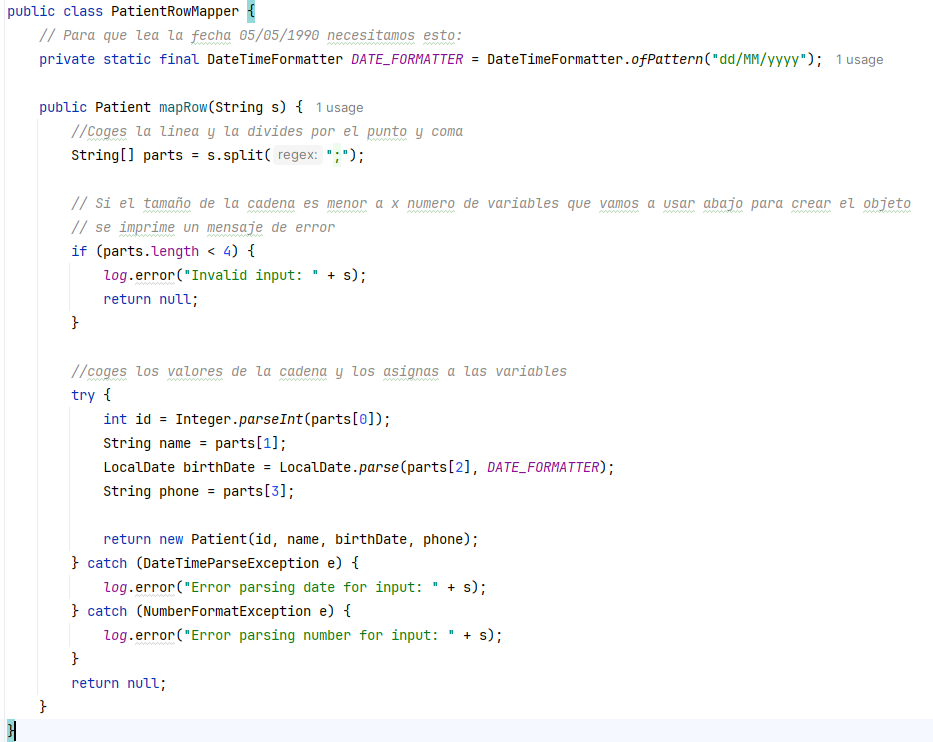
* Clase configuration: en el paquete common/config:
* El txt (tambien puede ser un .properties, es lo mismo) o xml del que se basa, con las variables que luego accede:



## Text Files

Son archivos csv que se guardan en la carpeta data en la ruta raíz, los llamas desde el dao.

* **Cargar contenido a ram desde el archivo de texto**

Antes de nada necesitas una manera de pasar cada línea que veas a un objeto. Se puede hacer de varias maneras, pero lo hacemos con un rowmapper:

Ahora, una vez con el rowmapper configurado:

*// Metodo que usa el rowmapper*private Patient mapToPatient(String line) {  
 return rowMapper.mapRow(line);  
}

*// Metodo que lee el archivo de pacientes*private List<String> readFile() {  
 List<String> lines = new ArrayList<>();

*// en la configuracion*  
 Path filePath = Paths.*get*(config.getPathPatients());

*// Se lee con bufferedReader en este caso porque es mas*

*eficiente para archivos grandes* try (BufferedReader reader =

Files.*newBufferedReader*(filePath))

{  
 String line;  
 while ((line = reader.readLine()) != null) {  
 lines.add(line);  
 }

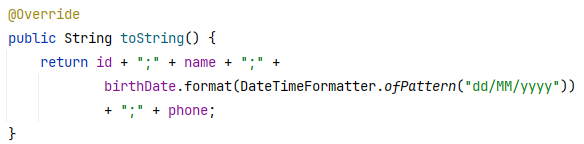
} catch (IOException e) {  
 throw new InternalServerErrorException("Error

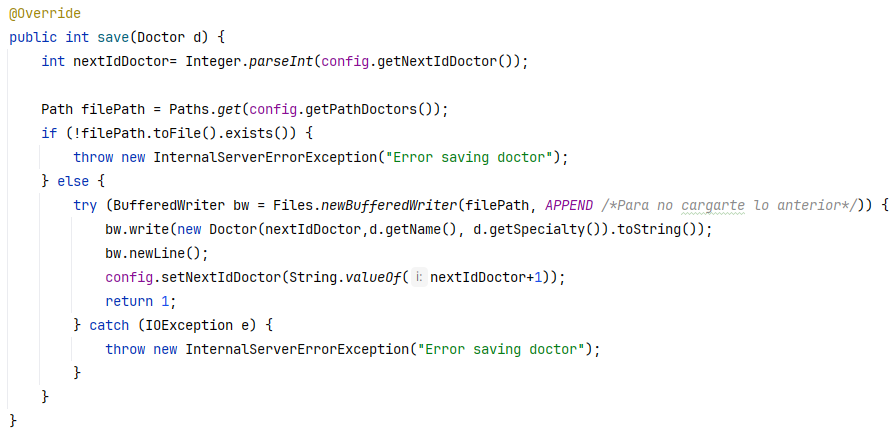
reading the patient file "+ e);

}  
 return lines;  
}

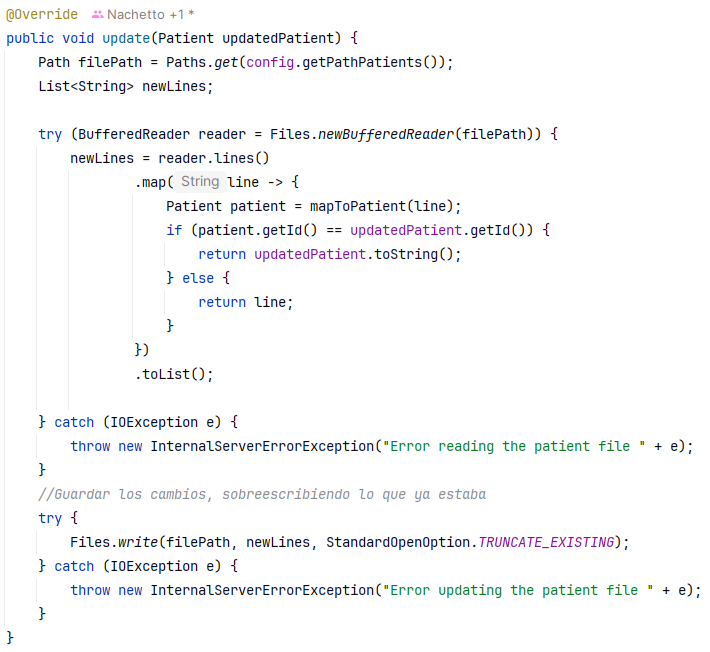
Otro ejemplo que usa readAllLines para archivos pequeños:

* **Guardar contenido a el archivo hacia el archivo de texto**

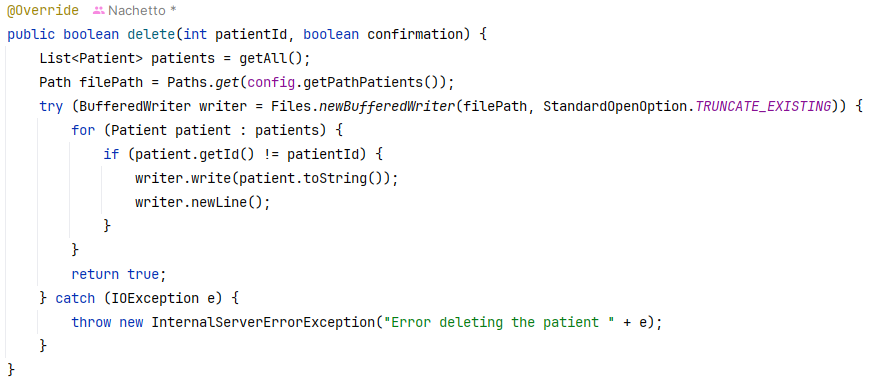
Antes de nada tienes que tener el tostring bien configurado en el objeto que quieras guardar, hazlo con un tostring que no sea el de por defecto por si acaso, aunque lo haga asi yo aqui:

Luego, iteras sobre la lista y vas agregandolos uno a uno, la unica complicacion esá en sacar el ultimo id, pero eso se guarda en una variable cada vez que haces un getAll, llamas al metodo guardarLastId y lo guardas en una variable o algo así.

* **Editar contenido del archivo de texto**

Iteras sobre lo qe hay guardado, para que guando encuentres el mismo objeto que quieres editar, guardes el nuevo en una nueva lista junto con los demas:

* **Eliminar contenido del archivo de texto**

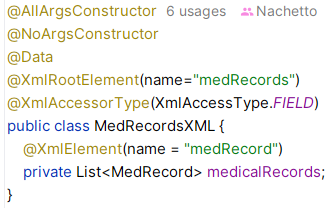
Iteras sobre la lista que te acabas de cargar a RAM del getAll, y si el obejto no es el que quieres eliminar, lo vas guardando:

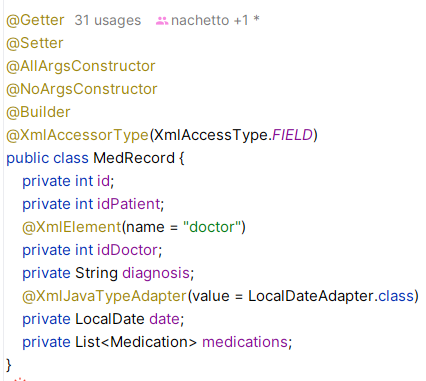
## XML FILES

Antes que nada hay que indicar al motor que se encarga de interpretar todo el xml cómo va a ser la estructura del archivo en nuestro modelo de datos. Para ello, necesitamos indicarle exactamente el mismo nombre y la misma estructura que va a haber. Ella te va a dar el modelo de datos, y probablemente pase que los nombres en su xml son distintos a tus variables de tus objetos, que haya elementos en ru modelo de datos que quieras que tu xml ignore o viceversa. Vamos por partes.

Primero, tu lista parte de una raiz en este ejemplo parte de “medRecords”:

Entonces, tienes que indicar que todo sale de un “elemento” llamado medrecords y que dentro de eso hay una lista de “elementos” llamada “medRecord”, y dentro de “medrecord” hay una lista de elementos “medication” llamada “medication”. Se hace así:



El “name” dentro de la etiqueta XmlElement es para que sepa que hay un nombre distinto en el xml que en el modelo de datos, el resto de variables que no tienen esta notacion cogerá el mismo nombre que tienen ya.

