CREACION DE UN ARCHIVO .dat

Para crear un archivo .dat o de la extension que queramos inventar, necesitamos trabajar con clases serializables para poder pasar los objetos a un documento. Solo aquellas de las que pertenecen los objetos. Si en la que creamos el metodo no hay esos objetos no necesitamos serializar. El codigo es:

Necesitamos pasarle una Lista de la clase que queramos mas el nombre o ruta del archivo con la extension que le queramos dar.

Se le pasa como argumento: ArrayList<Pedido> pedidos, String archivo

```
FileOutputStream fos = new FileOutputStream(archivo);
ObjectOutputStream oos = new ObjectOutputStream(fos);

for (Pedido ped: pedidos){
   oos.writeObject(ped);
}

oos.close();
fos.close();
```

LEER UN ARCHIVO .dat, GUARDARLO Y MOSTRARLO POR PANTALLA

```
ArrayList<Pedido> listaPedidos = new ArrayList<>();

FileInputStream fis = new FileInputStream(archivo);
ObjectInputStream ois = new ObjectInputStream(fis);

   Pedido pedido = null;

   while(fis.available() > 0){
      pedido = (Pedido) ois.readObject();
      listaPedidos.add(pedido);

      System.out.println(pedido);
   }

   ois.close();
   fis.close();

return listaPedidos;
```

CREACION DE UN ARCHIVO DOM

Necesitamos un documento DOM para poder trabajar con el archivo. Para esto podemos o tener un archivo y pasarlo a DOM o crearlo desde cero y a partir de ahi trabajar con el. Pero para en ambos para poder trabajar sobre el necesitamos pasarlo a DOM.

-Creacion desde 0 »» Aqui solo creamos el documetno DOM bacio, en este caso para trabajar con un xml posteriormente.

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
DOMImplementation implementacion = db.getDOMImplementation();
doc = implementacion.createDocument(null, "pedidos", null);
doc.setXmlVersion("1.0");
```

-A raiz de otro documento xml ya creado: Se le pasa el nombre del documento o la ruta donde se encuentra.

```
Document doc = null;

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
doc = db.parse(nombre);
```

PASO DE DOM A XML

Una vez que ya hayamos creado el documento y hayamos trabajado con el todo lo que queramos, tendremos que guardarlo en algun lugar del disco ya que ahora mismo solo se encuentra en memoria. En este metodo le pasaremos el nombre o ruta donde se va a guardar y el Documento que vayamos a guardar.

```
File f = new File(nombre);
Transformer transformer = TransformerFactory.newInstance().newTransformer();
transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
StreamResult result = new StreamResult(f);
DOMSource source = new DOMSource(doc);
transformer.setOutputProperty(INDENT, "yes");
transformer.transform(source, result);
```

LEER UN XML

Para leerlo usaremos la recursividad y crearemos un metodo que recivira como argumento un Nodo, ya que como no sabemos la profundidad del xml que vamos a leer con la recursividad los recorreremos todos. Habra que analizar el tipo de Nodo aunque nos centramos en texto y element.
Codigo:

```
switch (nodo.getNodeType()) {
        case Node.ELEMENT_NODE:
           Element elemento = (Element) nodo;
              System.out.println("Etiqueta " + elemento.getTagName());
              NodeList nl = nodo.getChildNodes();
                 for(int i = 0; i < nl.getLength(); i++){
                    mostrarXml(nl.item(i));
                 }
              break;
        case Node.TEXT_NODE:
           Text texto = (Text) nodo;
           System.out.println("Texto: " + texto.getWholeText());
           break;
     }
```

TRABAJAR CON XStream

Para trabajar con Xstream necesitaremos una clase con los objetos que vamos aquerer guardar y otra con una Lista que haga referencia a la clase con los objetos para darle nombre a la raiz del xml. Se puede trabajar con una Lista directamentes o con un archivo .dat.

Si trabajamos con el .dat lo tendremos que pasar a una Lista y despues trabajaremos con el. Si no trabajaremos con la Lista directamente, pasandosela a la clase donde tenemos solo la lista.

Despues trabajaremos con alias para darle el nombre que queramos a los elementos del xml, haciendo referencia a la clase que pertenecen.

Codigo:

```
File file1 = new File("pedidosXStream.xml");

CrearXmlPedidos crearPedidos = new CrearXmlPedidos();
ListaPedido listaPedido = new ListaPedido();
listaPedido.setPedidos(crearPedidos.lePedidos("pedidos.dat"));

XStream xstream = new XStream();
xstream.setMode(XStream.NO_REFERENCES);
xstream.alias("pedido", Pedido.class);
xstream.alias("lista", ListaPedido.class);
xstream.alias("producto", Producto.class);

//Instanciar objeto DcumentoConXStream
xstream.toXML(listaPedido, new FileOutputStream(file));
```

LEER DOCUMENTO XStream

```
XStream xs = new XStream(new DomDriver());
        xs.setMode(XStream.NO_REFERENCES);
        xs.alias("pedido", Pedido.class);
        xs.alias("lista", ListaPedido.class);
        xs.alias("producto", Producto.class);
        xs.addPermission(AnyTypePermission.ANY);
        ListaPedido listaLeida = new ListaPedido();

        try{
           xs.fromXML(new FileInputStream(file1), listaLeida);
           System.out.println("\nLista de pedidos: \n" + listaLeida);
        } catch (FileNotFoundException ex) {
           Logger.getLogger(DocumentoConXStream.class.getName()).log(Level.SEVERE, null,
ex);
        }
```

ESCRIBIR UN FICHERO JSON

Para crear un documento .json debemos pasar el objeto json que vamos a escribir mas la ruta donde queremos guardar el fichero.

```
    public void crearDocumentoJson(JSONArray pedidos, String ruta) {

        try ( FileWriter fw = new FileWriter(ruta)) {
           fw.write(pedidos.toJSONString());
           fw.close();
        } catch (IOException ex) {
           Logger.getLogger(UtilidadesJson.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

LEER UN FICHERO JSON

Para leer un fichero .json, debemos pasarle la ruta donde se encuentra el documento. A continuacion crear el objeto JSONParser y si lo queremos guardar en una lista, la lista de la clase a la que pertenece. Leeremos ese objeto y lo pasaremos a un objeto JSONArray o JSONObject, dependiendo de que sea la raíz del documento. Por ultimo tendremos que ir recuperando cada objeto e ir añadiéndolo a la lista

```
public List<Pedido> leeJsonPedidos(String ruta) {
    JSONParser parser = new JSONParser();
    List<Pedido> pedidos = new ArrayList<>();

    try {
       FileReader fr = new FileReader(ruta);
       JSONArray arrayPedidos = (JSONArray) parser.parse(fr);

       for (int i = 0; i < arrayPedidos.size(); i++) {
          JSONObject objPedido = (JSONObject) arrayPedidos.get(i);

          pedidos.add(this.leeJSonPedidos(objPedido));
       }

    } catch (FileNotFoundException ex) {
       Logger.getLogger(UtilidadesJson.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException | ParseException ex) {
       Logger.getLogger(UtilidadesJson.class.getName()).log(Level.SEVERE, null, ex);
    }

    return pedidos;
}
```

CREAR UN FICHERO JSON CON GSON

Se le pasa la lista que queremos escribir y la ruta. Instanciamos el objeto Gson y pasamos a una cadena de texto con gson.toJson() la lista que queremos escribir. Después pasaremos al FileWriter la ruta y después escribiremos el objeto.

```
public void crearGsonPedidos(List<Pedido> pedidos, String ruta) {
    Gson gson = new Gson();

    String jsonPedido = gson.toJson(pedidos);

    try ( FileWriter fw = new FileWriter(ruta)) {
       fw.write(jsonPedido);
       fw.close();
    } catch (IOException ex) {
       Logger.getLogger(GsonUtilidades.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

LEER UN FICHERO JSON CON GSON

Se le pasa la ruta donde se encuentra. Instanciamos el objeto Gson y creamos un objeto de la clase List ya que el objeto que vamos a recuperar es una Lista de objetos. Despues hacemos un gson.fromJson() y ponemos el objeto leido mas la clase que hace referencia, como en este caso es una lista, ponemos List.class. Si fuese un unico objeto pondriamos la clase a la que pertenece, como por ejemplo, Pedido.class

```
public List<Pedido> leerGsonPedidos(String ruta) {
    Gson gson = new Gson();
    List<Pedido> listaPedidos = new ArrayList<>();
    try {
        FileReader fr = new FileReader(ruta);
        listaPedidos = gson.fromJson(fr, List.class);
        fr.close();
    } catch (FileNotFoundException ex) {
        Logger.getLogger(GsonUtilidades.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(GsonUtilidades.class.getName()).log(Level.SEVERE, null, ex);
    }

    return listaPedidos;
}
```