Serialización y Deserialización

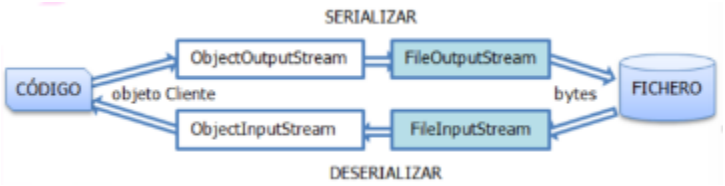
Java proporciona clases predefinidas para convertir de forma automática objetos a formatos que son portables, o fácilmente transportables a otras localizaciones.

Este proceso de conversión de objetos a un formato portable se denomina serialización. Cuando realizamos el proceso inverso de reconstrucción del objeto hablaremos de deserialización.

Para poder almacenar los objetos de una aplicación en una localización necesitamos convertirlos a un formato común, como a una serie de bytes. El proceso de convertir objetos y datos a un formato común para su almacenaje y transporte es llamado serialización. Java proporciona la clase **ObjectOutputSteam** para gestionar la serialización de objetos y datos.

Cuando los datos son recuperados, el código que recupera los datos serializados debe identificar cómo están representados. Si los datos serializados representan un objeto, el código que los recupera debe poder convertirlos al objeto original. Este proceso es denominado deserialización y Java proporciona la clase **ObjectInputStream** para gestionarla.

Para que un objeto sea **serializable** a un formato binario, su clase (o una de sus superclases en la jerarquía de herencia) debe implementar la interfaz **java.io.Serializable**, la cual no define ningún método.



Acción	Clases
Serializar	FileOutputStream ObjectOutputStream
Deserializar	FileInputStream ObjectInputStream

Al serializar un objeto automáticamente se serializan todas sus variables y objetos miembro. A su vez se serializan los que estos objetos miembros puedan tener (todos deben ser serializables). También se reconstruyen de igual manera.

Restricciones a la serialización / deserialización

• Para que funcione la serialización de un objeto, es necesario que todas las variables miembros de su clase sean serializables.

• La versión de la clase compilada cuando se serializa debe ser igual a la versión de la clase compilada cuando se deserializa. Esto quiere decir que, si serializamos un objeto con la versión 4 de java y lo intentamos deserializar con la versión 5 de java, se produciría un error.

• Se pueden serializar vectores y colecciones. Si dos elementos referencian el mismo objeto, el objeto se guarda una sola vez y se mantienen las referencias.

• El modificador **transient** permite indicar que un objeto o variable miembro no sea serializable. Al recuperar un objeto, lo marcado como **transient** quedará asignado al valor por defecto de su tipo de dato.

• Las variables y objetos **static** no se serializan.

Serialización y Deserialización en XML

Se puede realizar desde Java 2. XML fue diseñado para describir datos y actualmente tiene mucha importancia en el intercambio de una gran variedad de datos en el Web.

Java ofrece la clase **java.beans.XMLEncoder** para permitir la persistencia de objetos como documentos XML. Esta clase se encarga de convertir el objeto y todos sus datos (incluidos los campos que también son objetos) a un documento XML. Por su parte, la clase **java.beans.XMLDecoder** se encarga de deserializar los documentos XML generados por **XMLEncoder**.

Acción	Clases
Serializar	FileOutputStream XMLEncoder
Deserializar	FileInputStream XMLDecoder

Restricciones a la serialización / deserialización en XML

Para poder serializar una clase a formato XML no es necesario que implemente ninguna interfaz, pero si es necesario que cumpla las condiciones de un JavaBean: debe ser una clase pública, debe tener un constructor sin argumentos, y usar métodos accesores para acceder a las propiedades que definen su estado.

El modificador **transient** no se tiene en cuenta, y las propiedades que lo incorporan se serializan/deserializan como cualquier otra propiedad.

Serialización y Deserialización en JSON

Para poder serializar una clase a JSON debe tener el constructor por defecto y sus propiedades getters y setters.

Hay varias librerias para serializar/deserializar a JSON, siendo **GSON** una de las más populares:

1. Descarga GSON:

https://repo1.maven.org/maven2/com/google/code/gson/gson/2.6.2/gson-2.6.2.jar

2. Agrega GSON a tu proyecto:

    Pulsa con el botón derecho sobre tu proyecto, y en **Build Path > Add External Archives**, selecciona el jar descargado.

3. Impórtalo en la clase que requieras usarlo:

    ```
    import com.google.gson.*;
    ```

Jackson es otra librería que podéis usar para serializar/deserializar mediante JSON:

https://www.baeldung.com/jackson-object-mapper-tutorial