Serialización

Contenido

Serialización

- ¿Qué es la serialización?
- Serialización binaria.
- Serialización JSON.
- Consideraciones.

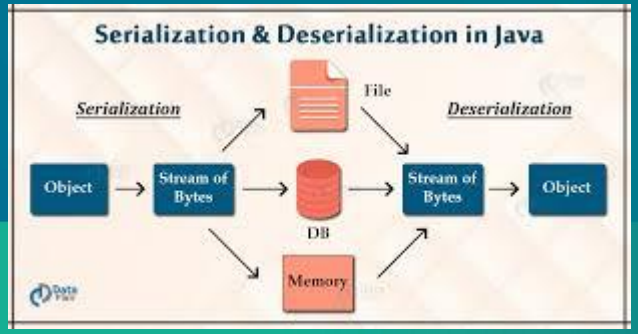
¿Qué es la serialización?

La serialización el proceso de convertir el **estado** de un objeto en una secuencia de bytes con un formato específico.

El objetivo es que pueda ser fácilmente almacenado en un archivo, en memoria o en una base de datos.

Su propósito principal es guardar el ***estado*** del objeto para recrearlo en algún momento en el futuro.

El proceso contrario, recuperar un objeto que fue serializado, se lo conoce como **deserialización.**



Serialización binaria

Serialización binaria

Para serializar un objeto en Java, éste debe implementar la interfaz ***Serializable***.

Los objetos serializables se pueden escribir en un archivo o enviarlos a través de una red, y luego reconstruirlos en la misma o en otra *JVM*.

La **serialización binaria** (con Serializable) es más eficiente para almacenamiento y transferencia de datos en redes, ya que ocupa menos espacio y suele ser más rápida.

Consideraciones

Modificadores de visibilidad:

- Los atributos de **cualquier visibilidad** (public, protected, private, o sin modificador) pueden ser serializados, siempre que no sean ***static*** o ***transient***.

Atributos transient:

- Los atributos marcados como ***transient*** no se serializan. Esto es útil para datos sensibles o temporales (por ejemplo, contraseñas, claves de sesión, etc.) que no se quiere incluir en la serialización.

Consideraciones

Atributos static:

Los atributos ***static*** pertenecen a la clase, no a instancias individuales, por lo que **no se serializan**.

Atributo serialVersionUID (recomendado):

El atributo estático ***serialVersionUID*** asegura la compatibilidad en distintas versiones.
Si la clase es modificada después de una serialización, las instancias deserializadas funcionarán, siempre y cuando las modificaciones sean compatibles.

Serialización JSON

Serialización JSON

*JavaScript Object Notation (**JSON**) es un estándar abierto que usa texto de fácil lectura para almacenar y transferir objetos.*

Se suele utilizar para transferir datos a través de la web (APIs).

Posee una manera organizada y fácil de acceder a sus propiedades.

Puede interactuar con sistemas **no** Java.

Está compuesto de pares **propiedad-valor** y **arrays**.

Java no tiene una implementación nativa para serializar en formato JSON, pero se pueden usar librerías como **Gson** de Google.

Consideraciones

Modificadores de visibilidad:

- Los atributos de **cualquier visibilidad** (public, protected, private, o sin modificador) pueden ser serializados.

  Sin embargo, solo convierte automáticamente en JSON los atributos públicos o privados sin el modificador ***transient***.

Serialización binaria

- **Ventajas**:
  - **Eficiencia en espacio y velocidad**: Al ser binaria, ocupa menos espacio que otros formatos de texto y, en general, es más rápida para serializar y deserializar.
  - **Soporte nativo en Java**: No se necesita librerías externas, ya que es parte del JDK.
- **Desventajas**:
  - **Compatibilidad limitada**: La serialización binaria de Java es específica para JVMs, lo que significa que no es adecuada para sistemas no-Java.
  - **No es legible para humanos**: Al ser un formato binario, no puede leerse ni editarse manualmente.
- **Uso ideal**:
  - En aplicaciones Java internas, donde se requiere eficiencia y no se necesita compartir los datos con otras plataformas.

Serialización JSON

- **Ventajas**:
  - **Interoperabilidad**: JSON es un estándar ampliamente compatible con casi cualquier lenguaje y plataforma.
  - **Legible para humanos**: Al estar en formato de texto, es fácil de leer y modificar.
- **Desventajas**:
  - **Tipos de datos complejos**: Las estructuras como listas de listas o datos anidados pueden ser difíciles de manejar.
  - **Mayor tamaño en comparación con binario**: Ocupa más espacio que la serialización binaria.
- **Uso ideal**:
  - En aplicaciones de comunicación entre sistemas, especialmente en entornos web y ***APIs RESTful***. JSON es también el formato preferido para la mayoría de las aplicaciones móviles y web modernas.

Serialización binaria vs serialización JSON

- La **serialización binaria** (con Serializable) es más eficiente para almacenamiento y transferencia de datos en redes, ya que ocupa menos espacio y suele ser más rápida.

- La **serialización en JSON** es más legible y puede interactuar con sistemas **no** Java, lo cual la hace más común para APIs y aplicaciones web.

Ejercitación