Clase Persistencia

**Persistencia:** trascender en el tiempo y/o espacio. En la informática hace referencia a la característica que puede tener un dato o estado que le permite sobrevivir al proceso que lo creó.

**Persistencia en POO**

Un ambiente orientado a objetos debe permitir que los objetos persistan, para mantener su vida más allá de la vida de la aplicación.

Un objeto **persistente** es aquel que conserva su estado en un medio de almacenamiento permanente, pudiendo ser reconstruido por el mismo proceso que lo generó u otro, de modo tal que al reconstruirlo se encuentre en el mismo estado en que se lo guardó.

Al objeto no persistente lo llamamos **efímero** o **transitorio.**

Tipos de Persistencia:

**Nativa:** provista por la plataforma. Ej: Java, Smalltalk

**No nativa:** a través de una biblioteca externa. Programada a mano.

El ideal en cuanto a persistencia sería que cada clase tenga un método para guardar objetos y otro para recuperarlos consistentes entre sí. Sin embargo, hay una contradicción entre este ideal y el principio de separación de interéses.

La **persistencia** en **Smalltalk** está **basada en imágenes.**

**Serialización:** proceso que consiste en convertir la representación de un objeto en un *stream* (flujo o secuencia) de bytes. Reconstruir u objeto a partir de un stream de bytes se denomina **deserialización**.

**Formatos de Serialización:**

* **Formato propietario:** más eficiente en términos de uso de almacenamiento y tiempo de traducción. Solo sirve para comunicar aplicaciones basadas en la misma plataforma
* **Uso de lenguaje estándar:** Ej: JSON. Es el más popular lenguaje de intercambio de información. Al ser formato de texto ocupa más espacio y su estructura no propietaria precisa hacer transformaciones en términos de computo.

Java pose **serialización automática** en formato propietario, para trabajar con JSON hay bibliotecas.

**Persistencia y Serialización**

Para persistir debo primero serializar. Serializar no implica necesariamente persistir.

**Serialización en Java**

Para definir una clase serializada, debe implementarse la interfaz **Serializable**

* No tiene métodos.
* Sirve para avisarle a la máquina virtual que la clase puede serializarse (marker interface).
* Todas las subclases de una clase serializable son serializables también.
* Por defecto, se serializan todos los atributos de un objeto que no posean el modificador transient (transitorio).
* Si un objeto serializable tiene referencias a otro objeto serializable, también lo serializa.
* Si algún objeto del “árbol de objetos” a serializar no es serializable se lanza la excepción NonSerializableException.
* Los objetos serializables deben tener un constructor sin parámetros para poder ser deserializados correctamente

**Persistencia en Java**

Persistencia Nativa

**Ventajas**

* **Es nativa del lenguaje (casi no hay que programar)**
* **Resuelve referencias circulares**

**Desventajas**

* **No es portable a otros lenguajes de manera sencilla**
* **No es óptima en cuanto a tamaño (tiene overhead alto)**
* **La información en el archivo es binaria**
* **No es extensible ni reparable.**