

Bootcamp Java Profesional

Serialización y Deserialización de Objetos

Javier Ramírez

Software Professional since 2001

Java Developer / Tech Lead

@javaMexico Founder/Co-Leader



@_benek | @javamexico

linktree.com/javamexico



Source Code

- El código fuente de esta clase estará disponible en:
- <https://github.com/benek/Bootcamp-Java-CodigoFacilito>

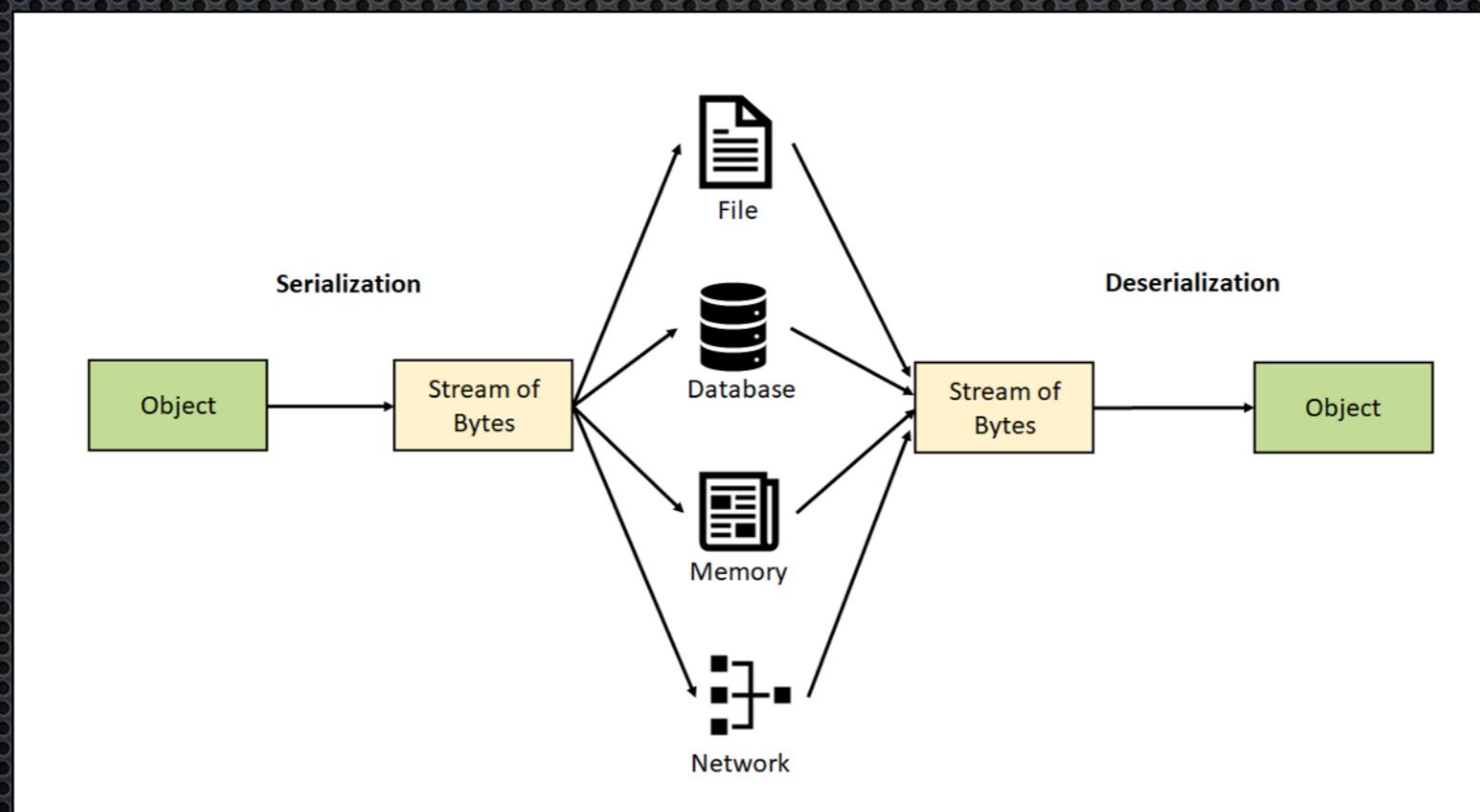
Serialización

Serialization

- Mecanismo que provee Java para convertir un Objeto y su estado a una secuencia de bytes, incluyendo la información que contiene
- *Deserialization* es el proceso inverso, en donde una secuencia de bytes es convertida de vuelta en un Objeto

Propósito

- Almacenar el estado de un Objeto a un archivo, enviarlo por medio de la red, o guardararlo en una BD



¿Para qué se usa?

- Para transportar objetos de una JVM a otra
- Ejemplos de uso:
 - Hibernate, JPA, JMS, RMI, EJB, etc...

Ventajas y beneficios

- Data Persistence
- Interoperability
- State sharing

Drawbacks

- Performance overhead (CPU, memoria)
- Security risks
- Compatibilidad de versiones entre clases

Serializable Interface

- `java.io.Serializable`
- “Marker” Interface
- Habilita la posibilidad de que Java serialice objetos de esa clase
- Es necesario implementarla para hacer uso de Serialización/Deserialización

SerialVersionUID

- **serialVersionUID**

- Identificador único para cada clase
- Sirve para que Java se asegure durante la deserialización de que la clase corresponde al objeto a deserializar
- También a verificar que sea la misma versión de la clase
- Es opcional para nosotros definirlo, Java generará uno en runtime si no está presente

¿Cómo implementar?

- Para crear una clase serializable, solo se debe implementar la interfaz Serializable:
- ```
public class MiClase implements Serializable {}
```
- Con esto se indica a la JVM que los objetos de esta clase podrán ser serializados/deserializados

# Transient

- En Java, existe la palabra reservada `transient`
  - `transient String name;`
- Sirve para indicar que una propiedad no debe ser serializada
- En su lugar, al momento de la deserialización, las propiedades `transient` se setearán con valores por default, dependiendo del tipo de dato

# Proceso de Deserialización

java.io.InputStream

ObjectInputStream

- ObjectInputStream puede leer un stream de bytes
- ```
public final Object readObject() throws  
IOException, ClassNotFoundException;
```

Proceso de Serialización

java.io.OutputStream



ObjectOutputStream

- ObjectOutputStream puede escribir variables primitivas y grafos de objetos hacia un stream de bytes
- `public final void writeObject(Object o) throws IOException;`

Clase Persona

```
8 public class Person implements Serializable {  
7     private static final long serialVersionUID = 1L;  
6     static String country = "ITALY";  
5     private int age;  
4     private String name;  
3     transient int height;  
2  
1     // getters and setters  
9 }
```

Ejemplo de Serialización

```
18 public class MyClass {  
17     public void serialize() {  
16         Person person = new Person();  
15         person.setAge(20);  
14         person.setName("Joe");  
13         ---  
12         FileOutputStream fileOutputStream = new FileOutputStream("yourfile.txt");  
11         ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);  
10         objectOutputStream.writeObject(person);  
9         objectOutputStream.flush();  
8         objectOutputStream.close();  
7         ---  
6         FileInputStream fileInputStream = new FileInputStream("yourfile.txt");  
5         ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);  
4         Person p2 = (Person) objectInputStream.readObject();  
3         objectInputStream.close();  
2     }  
1 }  
19
```

Consideraciones

- Si una clase implementa *Serializable* y contiene referencias a otras clases, las otras clases serán serializadas también
- Sin embargo, deben implementar *Serializable* también, de lo contrario provocarán *NotSerializableException*

Ejercicio Práctico

- Conjuntando conocimientos de clases de OOP, Gradle y Serialización
- Crearemos un sistema básico para controlar una Biblioteca
- Después de la clase, el código fuente del ejercicio quedará en el repositorio Github mencionado al inicio

Q&A