Serialización en Java

Serialización de un objeto: Implementar *Serializable*

Si queremos enviar un objeto a través de un flujo de datos, deberemos convertirlo en una serie de *bytes*. Esto es lo que se conoce como serialización de objetos, que nos permitirá leer y escribir objetos directamente.

Para leer o escribir objetos podemos utilizar los objetos ObjectInputStream y ObjectOutputStream que incorporan los métodos readObject y writeObject respectivamente. Los objetos que escribamos en dicho flujo deben tener la capacidad de ser *serializables*.

Serán *serializables* aquellos objetos que implementan la interfaz Serializable. Cuando queramos hacer que una clase definida por nosotros sea *serializable* deberemos implementar dicho interfaz, que no define ninguna función, sólo se utiliza para identificar las clases que son *serializables*. Para que nuestra clase pueda ser *serializable*, todas sus propiedades deberán ser de tipos de datos básicos o bien objetos que también sean *serializables*.

Tomamos como ejemplo una clase que sería la siguiente:

```
package Serializar;

import java.io.Serializable;

public class Datos implements Serializable
{
   public int a;
   public String b;
   public char c;

   public Datos(int A, String B, char C){
             a = A;
             b = B;
             c = C;
   }

   public int getA() {
             return a;
   }

   public String getB() {
             return b;
   }

   public char getC() {
             return c;
   }
}
```

Si dentro de la clase hubiera atributos que son objetos de otras clases, éstos a su vez también deben ser **Serializable**. Con los tipos de *java* (**String**, **Integer**, etc.) no hay problema porque lo son.

Lectura y escritura de un objeto serializado en un fichero

Si queremos enviar un objeto a través de un flujo de datos, deberemos convertirlo en una serie de *bytes*. Esto es lo que se conoce como serialización de objetos, que nos permitirá leer y escribir objetos directamente.

Para leer o escribir objetos podemos utilizar los
objetos **ObjectInputStream y ObjectOutputStream** que incorporan
los métodos **readObject y writeObject** respectivamente. Los objetos que escribamos en dicho flujo deben tener la capacidad de ser ***serializables***.

```
package Coche;
import java.io.*;


public class CopiaObjetos implements Serializable {

          public static void main(String[] args) {
          try
                    {
                      //Objeto que vamos a serializar
                      Datos dato = new Datos(2,"Hola",'c');

                      //Abrimos el flujo de salida del objeto y del fichero
                ObjectOutputStream salida=new ObjectOutputStream(new
FileOutputStream("media.obj"));

                //Escribimos el objeto en el fichero
                salida.writeObject(dato);

                //cerramos el fichero
                salida.close();

                //Abrimos el flujo de entrada del fichero y el objeto
                ObjectInputStream entrada=new ObjectInputStream(new
FileInputStream("media.obj"));

                //Leemos el objeto del fichero
                Datos obj=(Datos)entrada.readObject();

                //Mostramos por pantalla el objeto para ver que lo hemos serializado
correctamente
                System.out.println("Objeto "+ obj.getA() + obj.getB() + obj.getC());

                //Cerramos el fichero
                entrada.close();
                    }
                    catch (Exception e)
                    {
                              e.printStackTrace();
                    }
          }

}
```

Serial Version UID

Cuando pasamos objetos **Serializable** de un lado a otro tenemos un pequeño problema. Si la clase que queremos pasar es *objeto_serializable*, lo normal es que en ambos lados (el que envía y el que recibe la clase), tengan su propia copia del fichero (en nuestro ejemplo Datos.class*)*. Es posible que en distintas versiones de nuestro programa la clase *objeto_serializable* cambie, de forma que es posible que un lado tenga una versión más antigua que en el otro lado. Si sucede esto, la reconstrucción de la clase en el lado que recibe es imposible.

Para evitar este problema, se aconseja que la clase *objeto_serializable* tenga un atributo privado de esta forma

```
 private static final long serialVersionUID = 8799656478674716638L;
```

de forma que el numerito que ponemos al final debe ser distinto para cada versión de compilado que tengamos.

De esta forma, *java* es capaz de detectar rápidamente que las versiones de *objeto_serializable.class* en ambos lados son distintas.

Algunos entornos de desarrollo, como Eclipse, dan un *warning* si una clase que implementa **Serializable** (o hereda de una clase que a su vez implementa **Serializable**) no tiene definido este campo. Es más, puede generarlo automáticamente, número incluido, si se lo pedimos. En eclipse basta con hacer *click* con el ratón sobre el símbolo de *warning* para que nos de las posibles soluciones al *warning*. Una de ellas genera el número automáticamente.