Práctica 1 - Sesión 5
Grabación y recuperación de datos en Java

Programación Avanzada
(EI1017)

Guillermo Peris Ripollés

Índice

Serialización

Java proporciona un mecanismo, denominado **Serialización**, con el que un objeto puede representarse como una secuencia de bytes. En esta secuencia se almacena tanto los datos como el tipo de objeto, como los tipos de datos de que se compone el objeto.

En cuanto un objeto *serializado* se escribe en un fichero, puede leerse y *deserializarse*, es decir, recrear en memoria el objeto y sus datos.

Además, esta función es independiente de la JVM, de forma que un objeto puede *serializarse* en una plataforma y *deserializarse* en otra completamente distinta.

Serialización

Para que un objeto pueda *serializarse* ha de implementar la interfaz **Serializable** (del paquete **java.io**):

```
public class Agenda implements Serializable {
     private List<Contacto> contactos;
     ...
}
```

- No es necesario sobreescribir ningún método de la interfaz **Serializable**.
- Los atributos de la clase deben implementar también **Serializable**. En este caso, los objetos implicados son **List** (que ya implementa esta interfaz) y **Contacto** (que debería implementarla).

Grabación de datos

La clase **ObjectOutputStream** permite almacenar un objeto serializado. Para ello, dispone del método **writeObject**:

```
FileOutputStream fos = new FileOutputStream("agenda.bin");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(agenda);
oos.close();
```

Recuperación de datos

De forma semejante, la clase **ObjectInputStream** permite cargar en memoria un objeto serializado. Para ello, dispone del método **readObject**:

```
FileInputStream fis = new FileInputStream("agenda.bin");
ObjectInputStream ois = new ObjectInputStream(fis);
agenda = (Agenda)ois.readObject();
ois.close();
```

Fíjate en el casting que se realiza para pasar del objeto recuperado (**Object**) al objeto de la clase requerida (**Agenda**).

Excepciones

Además de las excepciones habituales que debes manejar en casos de lectura/escritura de ficheros, has de tener en cuenta que en la recuperación de datos JVM necesita conocer los tipos de datos que ha de *deserializar*. Si JVM no puede encontrar una clase durante la recuperación de un objeto, lanza la excepción **ClassNotFoundException**.

Número de serie

Un problema de la serialización consiste en que puede que la clase haya cambiado entre la generación y la recuperación del fichero de datos.

Para evitar este problema, es aconsejable añadir un número de serie aleatorio para cada versión distinta que compilemos de la clase:

```
private static final long serialVersionUID = -106534185022584846 4L
    ;
```

Eclipse nos avisa si no incluimos este número de serie con un *warning*, y nos permite generar un número aleatorio.