Serialización + Sockets en Java

Serialización

La serialización es el proceso mediante el cual transformamos objetos a un formato que se puede escribir en un fichero, enviar por red ... para que posteriormente se pueda reconstruir el objeto.

En Java podemos obtener una representación en formato binario de un objeto si implementa la interfaz `Serializable`. Es una interfaz sin métodos pero para poder implementarla, los atributos del objeto han de ser serializables. En general declarar una clase como serializable es todo lo que tenemos que hacer y Java se encarga del resto.

Serialización y Streams

Al igual que existen objetos para utilizar fácilmente entradas y salidas de texto utilizando Streams (e.g. `BufferedReader`, `PrintWriter`), existen objetos que permiten utilizar streams para leer y escribir objetos *Serializables*: `ObjectInputStream`, `ObjectOutputStream`

Ejemplo serialización a fichero

```
import java.io.Serializable;

class Person implements Serializable {
    private static final long serialVersionUID = 1L; // Recommended for Serializable classes
    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

```
String filename = "person.ser";
Person person = new Person("Alice", 30);
//Serializar y escribir en un fichero
try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
    oos.writeObject(person);
    System.out.println("Object has been serialized: " + person);
} catch (IOException e) {
    e.printStackTrace();
}

//Deserializar leyendo de un fichero
try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
    Person deserializedPerson = (Person) ois.readObject();
    System.out.println("Object has been deserialized: " + deserializedPerson);
} catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
}
```

Sockets y serialización

El ejemplo dado utiliza `FileInputStream` y `FileOutputStream` pero de manera similar podemos utilizar cualquier `InputStream` / `OutputStream`, ¡como aquellos provenientos de un socket!

La idea es poder reconstruir y hacer uso de un objeto en otra máquina de manera sencilla.