APÉNDICE A

Incompatibilidades en la serialización

Según la especificación de la serialización de Java [Sun Java 1.3 serialización, 1999] estas son las posibles causas de incompatibilidad entre distintas versiones de una clase Java al ser serializadas o recuperadas.

NOTA: En este contexto, versión se ha de entender como una implementación diferente de una misma clase.

1. Borrado de campos: si un campo es borrado en una clase, el flujo escrito no contendrá su valor. Cuando el flujo es nuevamente leído por una clase previa, el valor del campo será establecido al valor por defecto porque el valor no está disponible en el flujo. Sin embargo, este valor por defecto puede impedir que la versión previa de la clase cumpla su contrato.

2. Movimientos de clases hacia arriba o hacia abajo en la jerarquía de clases: esto no puede ser permitido ya que los datos en el flujo aparecerían en una secuencia equivocada.

3. Cambio de un campo no estático a estático o no *transient* a *transient*: cuando se usa la serialización que ofrece Java por defecto, este cambio es equivalente a borrar el campo de la clase. Esta versión de la clase no escribirá el dato en el flujo, por tanto, no estará disponible para su lectura para versiones anteriores de

la clase. Respecto al borrado de un campo, el campo de la versión previa de la clase será inicializado a su valor por defecto, el cual puede causar que la clase falle de manera inesperada.

4. Cambio del tipo declarado de un campo primitivo: cada versión de la clase escribe los datos con su tipo declarado. Las versiones previas de la clase intentarán leer el campo y fallarán porque el tipo del dato en el flujo no concuerda con el tipo del campo.

5. Cambio de los métodos *writeObject* o *readObject* de forma que ya no pueda leer o escribir los campos por defecto o cambios que intente escribirlos o leerlos cuando las versiones previas no lo hacían. Los campos por defecto deben aparecer o no aparecer de forma consistente en el flujo.

6. Cambio de una clase de *Serializable* a *Externalizable* o viceversa: es un cambio incompatible ya que el flujo contendrá datos que son incompatibles con la implementación en la clase disponible.

7. Borrado de *Serializable* o *Externalizable* es un cambio incompatible ya que cuando se escriba no suministrará los campos necesitados por versiones previas de la clase.

8. Añadir los métodos *writeReplace* o *readResolve* a una clase es un cambio incompatible si el comportamiento produjera un objeto que es incompatible con cualquier versión anterior de la clase.