Ejercicios de Serialización

Índice

1. Leer un fichero de texto (0.5 puntos)

Vamos a realizar un programa que lea un fichero de texto ASCII, y lo vaya mostrando por pantalla. El esqueleto del programa se encuentra en el fichero `Ej1.java` dentro del proyecto `lja-serializacion` de las plantillas de la sesión. Se pide:

a) ¿Qué tipo de flujo de datos utilizaremos para leer el fichero? Añadir al programa la creación del flujo de datos adecuado (variable `in`), compilar y comprobar su correcto funcionamiento.

b) Podemos utilizar un flujo de procesamiento llamado `BufferedReader` que mantendrá un *buffer* de los caracteres leídos y nos permitirá leer el fichero línea a línea además de utilizar los métodos de lectura a más bajo nivel que estamos usando en el ejemplo. Consultar la documentación de la clase `BufferedReader` y aplicar la transformación sobre el flujo de entrada (ahora `in` deberá ser un objeto `BufferedReader`). Compilar y comprobar que sigue funcionando correctamente el método de lectura implementado.

c) Ahora vamos a cambiar la forma de leer el fichero y lo vamos a hacer línea a línea aprovechando el `BufferedReader`. ¿Qué método de este objeto nos permite leer líneas de la entrada? ¿Qué nos devolverá este método cuando se haya llegado al final el fichero? Implementar un bucle que vaya leyendo estas líneas y las vaya imprimiendo, hasta llegar al final del fichero. Compilar y comprobar que sigue funcionando de la misma forma.

2. Lectura de una URL (0.5 puntos)

El fichero `Ej2.java` es un programa que tomará una URL como parámetro, accederá a ella y leerá su contenido mostrándolo por pantalla. Deberemos añadir código para:

a) Crear un objeto URL para la `url` especificada en el método `creaURL`, capturando las posibles excepciones que se pueden producir si está mal formada y mostrando el mensaje de error correspondiente por la salida de error. Compilar y comprobar que ocurre al pasar URLs correctas e incorrectas.

b) Abrir un flujo de entrada desde la URL indicada en el método `leeURL`. Deberemos obtener un `InputStream`de la URL, y convertirlo a un objeto `BufferedReader`, aplicando las transformaciones intermedias necesarias, para poder leer de la URL los caracteres línea a línea. Comprobar que lee correctamente algunas URLs conocidas. Descomentar el bloque de código que realiza la lectura de la URL.

3. Gestión de productos (1 punto)

Vamos a hacer una aplicación para gestionar una lista de productos que vende nuestra empresa. Escribiremos la información de estos productos en un fichero, para

almacenarlos de forma persistente. Se pide:

a) Introducir el código necesario en el método `almacenar` de la clase `GestorProductos` para guardar la información de los productos en el fichero definido en la constante `FICHERO_DATOS`. Guardaremos esta información codificada en un fichero binario. Deberemos codificar los datos de cada producto (titulo, autor, precio y disponibilidad) utilizando un objeto `DataOutputStream`.

b) Introducir en el método `recuperar` el código para cargar la información de este fichero. Para hacer esto deberemos realizar el procedimiento inverso, utilizando un objeto `DataInputStream` para leer los datos de los productos almacenados. Leeremos productos hasta llegar al final del fichero, cuando esto ocurra se producirá una excepción del tipo `EOFException` que podremos utilizar como criterio de parada.

c) Modificar el código anterior para, en lugar de codificar manualmente los datos en el fichero, utilizar la serialización de objetos para almacenar y recuperar objetos `ProductoTO` del fichero.

4. Guardar datos de la filmoteca (1 punto)

Vamos a implementar la clase `FilePeliculaDAO` de nuestra aplicación de gestión de filmotecas. De esta forma, nuestra lista de películas quedará guardada de forma persistente. En el fichero guardaremos directamente una lista de objetos `PeliculaTO` serializados. Se pide:

a) Implementar la operación de consulta del listado de películas. En este caso, si no existiese todavía el fichero devolveríamos una lista de películas vacía.

> **Nota**
>
> Siempre que las operaciones de acceso a ficheros puedan lanzar alguna excepción *checked*, deberemos capturarla y lanzarla como *nested exception* dentro de una excepción de tipo `DAOException`.

b) Implementar las operaciones para añadir y eliminar películas. En estas operaciones primero obtendremos el listado de películas actuales del fichero, realizaremos con esta lista la operación correspondiente, y volveremos a guardar la lista resultante en el fichero. La especificación de estos métodos será la misma que en el caso de `MemoryPeliculaDAO`, pero en este caso en lugar de tener la lista en memoria, tendremos que recuperarla del fichero, realizar la operación correspondiente con ella, y volverla a guardar.

> **Nota**
>
> Utiliza los bloques `finally` para asegurarnos de que los ficheros se cierren.

c) ¿Hay algún bloque de código común en las operaciones de añadir y eliminar películas?

Si tienes código repetido utiliza las opciones de refactorización de Eclipse para extraer el código duplicado a un nuevo método y de esta forma evitar este problema.