

Flujos de E/S y serialización de objetos. RMS - Ejercicios

Índice

1 Gestión de productos que implementan Serializable.....	2
2 Almacén de notas con serialización para JavaME.....	2

1. Gestión de productos que implementan Serializable

Vamos a hacer una aplicación Java para gestionar una lista de productos que vende nuestra empresa. Escribiremos la información de estos productos en un fichero, para almacenarlos de forma persistente. A partir del proyecto `java-serializacion` de las plantillas de la sesión 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. Debemos codificar los datos de cada producto (título, 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 de Java para almacenar y recuperar objetos `ProductoTO` del fichero.

2. Almacén de notas con serialización para JavaME

Vamos a implementar un almacén de notas. En el directorio `Notas` tenemos la base de esta aplicación. Cada nota será un objeto de la clase `Mensaje`, que tendrá un asunto y un texto. Además incorpora métodos de serialización y deserialización.

Vamos a almacenar estos mensajes de forma persistente utilizando RMS. Para ello vamos a utilizar el patrón de diseño adaptador, completando el código de la clase `AdaptadorRMS` de la siguiente forma:

- En el constructor se debe abrir el almacén de registros de nombre `RS_DATOS` (creándolo si es necesario). En el método `cerrar` cerraremos el almacén de registros abierto.
- En `listaMensajes` utilizaremos un objeto `RecordEnumeration` para obtener todos los mensajes almacenados. Podemos utilizar la deserialización definida en el objeto `Mensaje` para leerlos. Conforme leamos los mensajes los añadiremos a un vector.

El índice de cada registro en el almacén nos servirá para luego poder modificar o eliminar dicho mensaje. Por esta razón deberemos guardarnos este valor en alguna parte. Podemos utilizar para ello el campo `rmsID` de cada objeto `Mensaje` creado.

En `getMessage` deberemos introducir el código para obtener un mensaje dado su

identificador de RMS.

- En `addMensaje` deberemos introducir el código necesario para insertar un mensaje en el almacén. Con esto ya podremos probar la aplicación, añadiendo mensajes y comprobando que se han añadido correctamente. Probar a cerrar el emulador y volverlo a abrir para comprobar que los datos se han guardado de forma persistente.
- En `removeMensaje` deberemos eliminar un mensaje del almacén dado su identificador.
- (*) En `updateMensaje` modificaremos un mensaje del almacén sobrescribiéndolo con el nuevo mensaje proporcionado. Para ello obtendremos el identificador correspondiente a al mensaje antiguo de su campo `rmsID`.
- (*) Obtener la información de la cantidad de memoria ocupada y disponible a partir del objeto `RecordStore`. Deberemos hacer que los métodos `getLibre` y `getOcupado` de la clase `AdaptadorRMS` devuelvan esta información.

