Ejercicios IV – Lectura y escritura de ficheros

Realiza los siguientes programas usando el lenguaje de programación Orientado a Objetos Java.

1. Crea una aplicación que pida la **ruta** de **dos ficheros de texto** y de una **ruta** de **destino** (solo la ruta, sin fichero al final). Debes copiar el contenido de los dos ficheros en uno, este tendrá el nombre de los dos ficheros separados por un guion bajo, este se guardará en la ruta donde le hayamos indicado por teclado.

   Para unir los ficheros en uno, crea un método donde le pases como parámetro todas las rutas. En este método, aparte de copiar debe comprobar si existe el fichero de destino, nos muestre un mensaje informándonos de si queremos sobrescribir el fichero. Te recomiendo usar la clase File

   Por ejemplo, si tengo un fichero **A.txt** con **"ABC"** como contenido, un fichero **B.txt** con **"DEF"** y una ruta de destino (por ejemplo, **D:\)**, el resultado será un fichero llamado **A_B.txt** en la ruta **D:** con el contenido **"ABCDEF"**.

   Puedes crear métodos para realizar la copia de ficheros, piensa también como podrías optimizar esta copia, si los ficheros tuvieran mucho contenido.

   Recuerda que debes controlar las excepciones que puedan aparecer. En caso de error, mostrar una ventana de dialogo con información del error.

2. ~~Crea una aplicación que copie un **fichero binario** a otra localización. En lugar de leer y escribir byte a byte, crea un array de bytes con el tamaño del fichero de origen (utiliza el método available()), copia el contenido del fichero a este array y escribe a partir de este array.~~

   ~~Recuerda que debes controlar las excepciones que puedan aparecer. En caso de error, mostrar una ventana de dialogo con información del error.~~

Investiga como se usan los DataInputStream/DataOutputStream y los ObjectOutputStream/ObjectInputStream y realiza los siguientes tres ejercicios.

3. Crea una aplicación que pida por teclado un número de números aleatorios enteros positivos y la ruta de un fichero. Este fichero contendrá la cantidad de números aleatorios enteros positivos que se ha pedido por teclado.

   Escribe los números usando un **DataOutputStream** y muestra por pantalla estos números leyéndolos con un **DataInputStream**.

   El rango de los números aleatorios estará entre 0 y 100, incluyendo el 100.

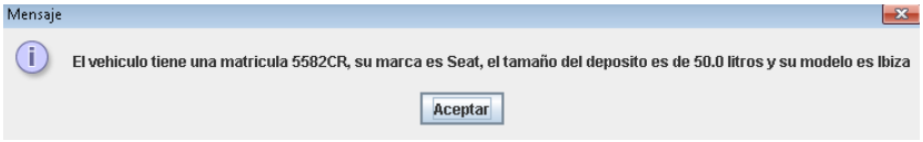
   Cada vez que ejecutemos la aplicación añadiremos números al fichero sin borrar los anteriores, es decir, si cuando creo el fichero añado 10 números y después añado otros 10 al mismo, en el fichero habrá 20 números que serán mostrados por pantalla.

4. Crea una aplicación que almacene los datos básicos de un vehículo como la matricula(String), marca (String), tamaño de deposito (double) y modelo (String) en ese orden y de uno en uno usando la clase **DataInputStream**.

   Los datos anteriores datos se pedirán por teclado y se irán añadiendo al fichero (no se sobrescriben los datos) cada vez que ejecutemos la aplicación.

   El fichero siempre será el mismo, en todos los casos.

   Muestra todos los datos de cada coche en un cuadro de dialogo, es decir, si tenemos 3 vehículos mostrara 3 cuadros de dialogo con sus respectivos datos. Un ejemplo de salida de información puede ser este:



5. Vamos a realizar el mismo ejercicio pero con serialización, para ello, crea una simple clase Vehiculo con los atributos matricula, marca, tamaño del depósito y modelo, con sus respectivos métodos get y el constructor se invocara con todos los atributos.

   El atributo tamanioDeposito no se incluirá en el fichero (aun así debemos pedirlo), debemos indicarlo en la clase (recuerda el uso de transient).

   Recuerda que al usar la clase ObjectOutputStream, si vamos a añadir varios objetos en distintas ejecuciones, debemos crear nuestra propia versión de ObjectOutputStream.