GUIA RAPIDA DE FICHEROS DE BINARIOS DE ACCESO SECUENCIAL EN JAVA: UPV-ETSINF-GII-PRG-13/14 GRUPO F

Los Ficheros Binarios Texto en Java tienen un tratamiento similar a los de textos con algunas particularidades. En principio, como en los de texto, se utilizará la clase `File` situada dentro del paquete `Java.io`.
Para crear un objeto básico `File f` basta con crear mediante el `new` de un objeto de esa clase, utilizando como parámetro un `String` con el nombre del fichero en el disco. El fichero creado de este modo deberá estar situado en el mismo directorio donde se encuentre la clase Java en la que estemos trabajando.

Una vez creado el objeto `f`, debemos decidir si lo usaremos para lectura (tomar datos de él) o para escritura (poner datos en el). Para la comunicación con un fichero se utilizan "`Streams`" que son los flujos de entrada o salida (input, output respectivamente) de datos asociados a un fichero. En concreto, para trabajar con ficheros binarios utilizaremos los flujos `FileInputStream` para la entrada o `FileOutputStream` para la salida. Sin embargo será necesario también la utilización de un envoltorio específico para poder utilizar tipos de datos primitivos u objetos del flujo de datos. Estos envoltorios serán respectivamente: `ObjectInputStream` y `ObjectOutputStream`. Además los objetos introducidos en un fichero binario deben implementar la interfaz `Serializable`. (basta con poner `implements Serializable` en la cabecera de la clase)

Si miramos la documentación de la clase `FileInputStream` (o la Output en su caso) veremos que el constructor que tiene como parámetro un objeto de tipo `File` en su cabecera advierte de que lanza una excepción `FileNotFoundException` que es derivada de la clase `IOException` y por tanto es checked. Además la construcción del `ObjectInputStream` (o la Output en su caso) lanza una excepción `IOException` y por lo tanto debe ser siempre incluida dentro de un bloque `try_catch_finally` o de un método que en cuya cabecera tenga la instrucción de propagación de al menos esa excepción. La clase `FileInputStream` (o la Output en su caso) tiene varios constructores entre ellos tenemos aquel que como parámetro lleva un File y otro adicional que lleva como parámetro un String. Éste último lo que hace es directamente construir un File en el constructor y luego invocar al otro. Si en el programa principal no se va a usar el file puede usarse este constructor directamente.

Lo habitual es situar la declaración de la variable de tipo ObjectInputStream (o la Output en su caso) fuera del bloque `try_catch_finally` asignándole un valor inicial igual a `null`. Sin embargo se pondrán dentro de la parte del `try` : La construcción de la variable y las operaciones asociadas a la lectura (escritura en su caso), luego la captura de la excepción ya vista con el `catch`, y en el `finally` otro try catch para el método `close()` pues también este puede lanzar la excepción `IOException`.

Una vez creados los objetos para manipular el flujo de entrada o salida sobre ellos se usan las siguientes operaciones:

- Para leer: (Todas pueden lanzar excepciones IO, en particular las de lectura de datos elementales EOFFException)

void readInt(int v)	Escribe el entero v al stream de salida como 4 bytes
void **readLong**(long v)	Escribe el long v al stream de salida como 8 bytes
void **readUTF**(String str)	Escribe el String str en formato portable (UTF8 mod.)
void **readDouble**(double v)	Escribe el double v al stream de salida como 8 bytes
void **readObject**(Object obj)	Escribe el Object obj al stream de salida. Lo que provoca la escritura de todos los objetos de los que obj esté compuesto (y así recursivamente). Puede lanzar una excepción adicional que habrá que capturar: ClassNotFoundException, si lo que se lee no es un objeto.

- Para escribir: (Todas pueden lanzar excepciones IO)

void writeInt(int v)	Escribe el entero v al stream de salida como 4 bytes
void **writeLong**(long v)	Escribe el long v al stream de salida como 8 bytes
void **writeUTF**(String str)	Escribe el String str en formato portable (UTF8 mod.)
void **writeDouble**(double v)	Escribe el double v al stream de salida como 8 bytes
void **writeObject**(Object obj)	Escribe el Object obj al stream de salida. Lo que provoca la escritura de todos los objetos de los que obj esté compuesto (y así recursivamente).

EJERCICIO SOBRE LECTURA Y ESCRITURA DE FICHEROS DE TEXTO EN JAVA:

```
import java.io.*;
public class Punto2D implements Serializable{
    private int Cx,Cy;
    public Punto2D(int x, int y){ Cx=x; Cy=y; }
    public String toString (){ return String.format("(%04d,%04d)",Cx,Cy); }
    public boolean equals(Object o){
        return ( (o!=null) && !(o instanceof Punto2D) &&
                 (((Punto2D)o).Cx==Cx)  && (((Punto2D)o).Cx==Cx) );
    }
    public void setCoorX(int x){Cx=x;}    public int getCoorX(int x) {return Cx;}
    public void setCoorY(int y){Cy=y;}    public int getCoorY(int y) {return Cy;}
}
```

```
import java.util.*;
import java.io.*;
public class Ejercicio2LecturayEscrituraFicheroBinario{
    // ocultamos el constructor
    private Ejercicio1LecturayEscrituraFicheroBinario(){}

   public static void main (String[] args){
        System.out.println("LEYENDO DATOS DESDE TECLADO Y ESCRIBIENDO EN EL FICHERO");
        leerDesdeTecladoyGuardarEnFichero();
        System.out.println("\nESCRIBIENDO DATOS EN LA PANTALLA DESDE EL FICHERO");
        escribirEnPantallaDesdeFichero();
   }

   /**
    * Ejercicio, implementar los métodos invocados desde el main, el primero leerá
    * dos coordenadas enteras del teclado hasta que escriba el usuario "fin" en la primera
    *  y las escribirá como objetos de tipo Punto2D en un fichero llamado "binario.txt".
    * El segundo leerá el contenido de "binario.txt" y lo mostrará por pantalla.
    */
   private static void leerDesdeTecladoyGuardarEnFichero(){
        Scanner teclado=new Scanner(System.in).useLocale(Locale.US);
        String s1, s2;  Punto2D p;
        File f=new File("binario.txt");
        ObjectOutputStream salida=null;
        try {
            salida=new ObjectOutputStream(new FileOutputStream(f));
             System.out.println("Primera Coordenada: "); s1=teclado.nextLine();
            while(!s1.equals("fin")){
                 System.out.println("Segunda Coordenada: "); s2=teclado.nextLine();
                salida.writeObject(new Punto2D(Integer.parseInt(s1), Integer.parseInt(s2)));
                 System.out.println("Primera Coordenada: "); s1=teclado.nextLine();
            }
        }
        catch (IOException e) {
            System.out.println("El fichero no existe o no se puede crear");}
        finally{ try{  if (salida!=null) salida.close(); }
                 catch(IOException e) {System.out.println("Fichero no cerrado correctamente");}
        }
    }

   private static void escribirEnPantallaDesdeFichero(){
        File f=new File("binario.obj");
        Punto2D p=null;
        ObjectInputStream entrada=null;
        try {
            entrada=new ObjectInputStream(new FileInputStream(f));
            do{
                p=(Punto2D) (entrada.readObject());
                System.out.println(p);
            } while( true );
        }
        catch (EOFException e)          {System.out.println("---> Fin de Fichero");}
        catch (IOException e)           {System.out.println("fich. no existe o no se puede leer ");}
        catch (ClassNotFoundException e){System.out.println("fich. no contiene objs. serializables");}
        catch (ClassCastException e)    {System.out.println("El fichero no contiene objetos Punto2D");}
        finally{ try{  if (entrada!=null) entrada.close(); }
                 catch(IOException e)   {System.out.println("Fichero no cerrado correctamente");}
        }
   }
}
```