Ing. Miguel Angel Sánchez Hernández

# TEMA

# ENTRADA Y SALIDA DE FICHEROS

Si no se tuviera la capacidad de poder almacenar la información que se crea o procesa en los programas creados en java, sería una situación grave ya que cuando se cierra el programa dicha información se pierde.

Se trataran los métodos que nos sirven para poder manipular ficheros y directorios, también los métodos necesarios para leer y escribir datos.

A lo largo de este tema crearemos ejemplos para poder entender el mecanismo que utiliza java para la manipulación de directorios y ficheros.

Objetivos:

- Describir como java reconoce el flujo de E/S
- Manejar el flujo en java a través de bytes

## 1.1 Visión general de los flujos de E/S

Java puede recibir información de cualquier fuente de datos que pueda transmitir un flujo de bytes, como por ejemplo ficheros (archivos), dispositivos (cámaras de video, USB etc.) e información atreves de la red.

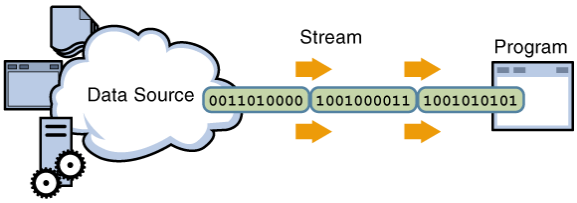


Fig. 1.1 Representación de cómo es el flujo de entrada de datos en java

El mismo concepto se tiene para el flujo de salida de datos, en donde cualquier fuente destino que pueda recibir un flujo de bytes se puede comunicar con nuestros programas.

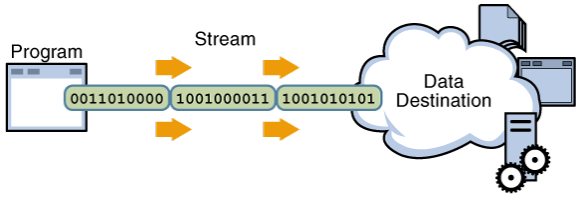


Fig. 1.2 Representación de cómo es el flujo de salida de datos en java

Como se está entendiendo Java maneja un flujo de información, que no será otra cosa que un objeto que será el intermediario entre nuestros programas y el origen o destino de la información. Con esto nosotros podremos leer o escribir en el flujo sin importar de donde viene o adonde se va.

Dado que nosotros no nos importa saber nada de los dispositivos por el nivel de abstracción que ocupares, solo nos tenemos que preocupar por seguir los siguientes pasos en nuestros programas.

| Leer información | Escribir información |
|---|---|
| Abrir el flujo | Abrir el flujo |
| Mientras se tenga información | Mientras se tenga información |
| Leer información | Escribir información |
| Cerrar el flujo | Cerrar el flujo |

Los objetos encargados de realizar estas tareas se trataran mediante las clases abstractas InputStream (secuencia de entrada de bytes), OutputStream (secuencia de salida de bytes). Las clases FileInputStream,FileOutputStream se derivan de las dos clases anteriores, estas nos permitirán poder ocupar un flujo de bytes. Dado que InputStream posee un método abstracto read, nos da la facultad de leer bytes, en cambio OutputStream tiene el método write, que nos da la facilidad de escribir bytes.

El programa 1.1 nos muestra como ocupar un flujo de bytes.

```
package fes.aragon;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopiarBytes {

   public static void main(String[] args) {
      try {
         CopiarBytes cp = new CopiarBytes();
         cp.copiarArchivo();
      } catch (IOException ex) {
         System.out.println(ex.getMessage());
      }
   }

   private void copiarArchivo() throws IOException {
      //objeto para el flujo de entrada
      FileInputStream in = null;
      //objeto para el flujo de salida
      FileOutputStream out = null;
      try {
         //Ubicación del archivo
         in = new FileInputStream("d:/poo/CodeConventions.pdf");
         //Salida del Archivo
         out = new FileOutputStream("d:/poo/CodeConventions2.pdf");
         int c;
         //lectura del flujo de bytes hasta que nos proporcionen -1
         while ((c = in.read()) != -1) {
            //escribiendo el flujo de bytes
            System.out.println("Entero: "+ c +
                  " Binario: " + Integer.toBinaryString(c)+
                  " Octal: " + Integer.toOctalString(c) +
                  " Hexadecimal: " + Integer.toHexString(c));
            out.write(c);
         }

      } finally {
         //cerrando los flujo de entrada y salida
         if (in != null) {
            in.close();
         }
         if (out != null) {
            out.close();
         }
      }
   }
}
```

*Programa 1.1*

Java a través de sus clases nos da la facilidad de poder guardar el flujo de bytes en un arreglo de bytes, para después realizar operaciones con él. El programa 1.2 nos muestra como ocupar el arreglo de bytes para poder sacar 10 copias del mismo archivo.

```
package fes.aragon;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopiarBytes2 {
   public static void main(String[] args) {
      try {
         CopiarBytes2 cp = new CopiarBytes2();
         cp.copiarArchivo();
      } catch (IOException ex) {
         System.out.println(ex.getMessage());
      }
   }
   private void copiarArchivo() throws IOException {
      //objeto para el flujo de entrada
      FileInputStream in = null;
      //objeto para el flujo de salida
      FileOutputStream out = null;
      try {
         String nombreArchivo = "d:/poo/CodeConventions.pdf";
         //Ubicación del archivo
         in = new FileInputStream(nombreArchivo);
         //Se crea un arreglo de bytes, con el tamaño del archivo a copiar
         byte[] datos = new byte[in.available()];
         in.read(datos);
         //Salida del Archivo
         for (int i = 0; i < 10; i++) {
            //Cambio de nombre a los archivos
            String archivoSalida = nombreArchivo.replaceFirst("CodeConventions.pdf",
                                                "CodeConventions" + i + ".pdf");
            System.out.println(archivoSalida);
            out = new FileOutputStream(archivoSalida);
            out.write(datos);
         }

      } finally {
         //cerrando los flujo de entrada y salida
         if (in != null) {
            in.close();
         }
         if (out != null) {
            out.close();
         }
      }
   }
}
```

Programa 1.2

Ahora tomando la idea del Programa 1.2, podemos nosotros crear un programa en donde divida un archivo en 5 partes, para después tomar cada parte en su forma individual y volver a juntar las 5 partes, muy parecido a lo que realiza winrar para dividir los archivo en los bytes deseados.

```
package fes.aragon;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopiarBytes3 {
   public static void main(String[] args) {
      try {
         CopiarBytes3 cp = new CopiarBytes3();
         cp.copiarArchivo();
      } catch (IOException ex) {
         System.out.println(ex.getMessage());
      }
   }
   private void copiarArchivo() throws IOException {
      //clase para el flujo de entrada
      FileInputStream in = null;
      //clase para el flujo de salida
      FileOutputStream out = null;
      try {
         String nombreArchivo = "I:/poo/CodeConventions.pdf";
         //UbicaciÃ³n del archivo
         in = new FileInputStream(nombreArchivo);
         int numeroArchivos=5;
         byte[] datos = new byte[in.available()];
         in.read(datos);
         int numeroBytes = (datos.length / numeroArchivos);
         int contador = 0;
         int indiceArchivo = 1;
         while (contador < datos.length) {
            String archivoSalida =
                  nombreArchivo.replaceFirst("CodeConventions.pdf", "CodeConventions"
                  + indiceArchivo + ".pdf");
            out = new FileOutputStream(archivoSalida);
            out.write(datos, contador, numeroBytes);
            contador += (numeroBytes);
            indiceArchivo++;
            if (indiceArchivo==numeroArchivos) {
               numeroBytes=datos.length-contador;
            }
         }
         out = new FileOutputStream("I:/poo/CodeConventionsNuevo.pdf");
         for (int i = 1; i <=numeroArchivos ; i++) {
            String archivoEntrada =
                        nombreArchivo.replaceFirst("CodeConventions.pdf", "CodeConventions"
                  + i + ".pdf");
            in = new FileInputStream(archivoEntrada);
            System.out.println("Archivo: " + i + " " + in.available());
```

```
                byte[] datosEntrada = new byte[in.available()];
                in.read(datosEntrada);
                out.write(datosEntrada);
            }
            System.out.println("Salida:" + out.getChannel().size());
        } finally {
            //cerrando los flujo de entrada y salida
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```

Programa 1.3