Flujos *Streams*

Entrada y salida

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

19 de noviembre de 2025

Definiciones

1 Definiciones

2 Flujos en Java

3 Acceso aleatorio

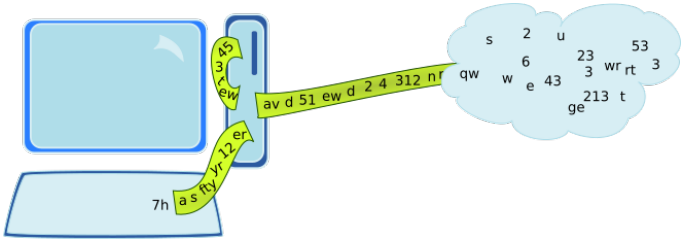
Temas

1. Definiciones
   - Flujos y filtros
   - Uso

Flujo

Definición (Flujo)

Un *flujo* (*stream*) es una secuencia de datos, potencialmente ilimitados, puestos a disposición a través del tiempo.



Filtros

- Los flujos pueden ser procesados por *filtros*, que pueden ser colocados secuencialmente, y cada uno procesa los datos devolviendo, a su vez, otro flujo con los datos procesados.
- Estos filtros permiten: realizar una manejo más eficiente de los datos (`BufferedInputStream`) o aplicar procesamientos intermedios como cifrado (`CipherInputStream`), monitoreo (`ProgressMonitorInputStream`), llevar la cuenta del número de línea en archivos de texto (`LineNumberInputStream`), etc.

Temas

1. Definiciones
   - Flujos y filtros
   - Uso

Uso de flujos

- Los flujos permiten acceder secuencias de datos disponibles a través de medios diversos como:
  - Archivos
  - Sitios de la red
  - Periféricos (impresoras, modems, scanners, etc)
- Para ello se deben establecer y manejar conexiones a estos medios.
- El uso de flujos requiere seguir un protocolo específico, que se explica a continuación.

Flujos de entrada

$$\text{Lectura} = \begin{cases} \text{Inicializar} & \{\text{Abrir el flujo} \\ \text{Procesar información} \\ \quad \text{(mientras haya)} & \{\text{Leer información} \\ \text{Final} & \{\text{Cerrar el flujo} \end{cases} \tag{1}$$

Flujos de salida

Para escritura: Si ya había información en el recurso, se borra e inicia desde cero.

$$\text{Escritura} = \begin{cases} \text{Inicializar} & \begin{cases} \text{Abrir el flujo para escritura}(write) \end{cases} \\ \text{Procesar información} & \begin{cases} \text{Escribir información} \\ \text{Jalar la palanca } (opcional : flush) \end{cases} \\ \text{(mientras haya)} & \\ \text{Final} & \begin{cases} \text{Cerrar el flujo} \end{cases} \end{cases} \tag{2}$$

Para ajuntar: La nueva información se agrega a lo que hubiera previamente.

$$\text{Escritura} = \begin{cases} \text{Inicializar} & \{\text{Abrir el flujo para adjuntar}(\texttt{append}) \\ \text{Procesar información} & \begin{cases} \text{Añadir información} \\ \text{Jalar la palanca } (\texttt{opcional}: \texttt{flush}) \end{cases} \\ \quad \text{(mientras haya)} & \\ \text{Final} & \{\text{Cerrar el flujo} \end{cases} \tag{3}$$

Flujos en Java

1 Definiciones

2 Flujos en Java

3 Acceso aleatorio

Temas

2 Flujos en Java
- Tipos de flujos

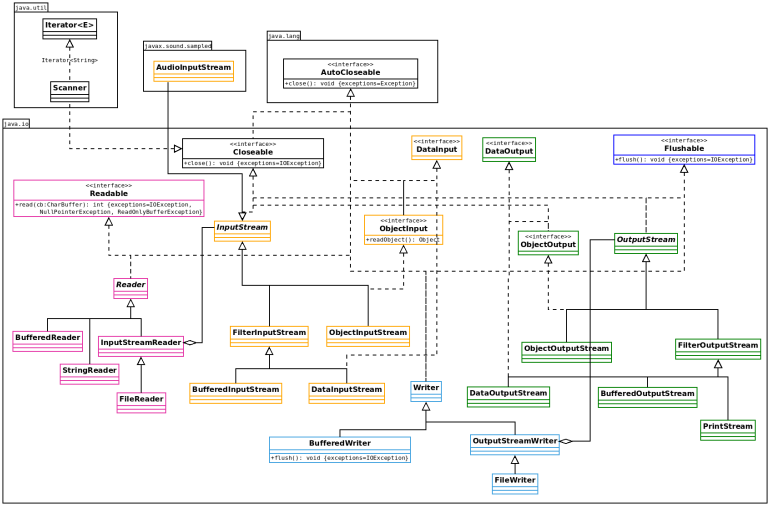
Tipos de flujos

En `Java` se distinguen los siguientes tipos de flujos:

1. Flujos de *bytes*. Trabajan con paquetes de 8-bits (bytes) y corresponden al nivel más bajo de procesamiento de entrada/salida, por lo que se deben pensar como secuencias de unos y ceros. Extienden las clases `InputStream` y `OutputStream`. Son adecuados para imágenes, audio y video.
2. Flujos de *caracteres*. Leen y escriben caracteres en la codificación local y `Java` los maneja, en memoria, como caracteres Unicode. Estas clases heredan de `Reader` y `Writer`.

Objetos de tipo `InputStreamReader` y `OutputStreamWriter` permiten trabajar con caracteres y enviar bytes, tejiendo un puente entre los flujos de caracteres y los de bytes.

Clases para el uso de flujos (ejemplos´)



Entrada y salida al sistema

Obsérvese el caso de los flujos en la clase `System`, utilizados para comunicarse con la entrada y salida por defecto:

- `System.out` es de tipo `PrintStream`.
- `System.in` es de tipo `InputStream`.
- `System.err` es de tipo `PrintStream`.

Fuente: https://docs.oracle.com/javase/8/docs/api/index.html

PrintStream

- Los objetos de tipo `PrintStream` permiten dar formato, con facilidad, a las cadenas que serán impresas.
- Nótese el uso de códigos como `%n`, `%d`, `%f`, `%s` definidos en la clase `Formatter`.
- En la declaración del método `format` también aparece el uso de una nueva notación:

```
1 public PrintStream format(String format, Object... args);
```

Esto permite pasar como parámetros un número indefinido de argumentos, según sean requeridos por la cadena `format`. Dentro del método, `args` es un arreglo de `Object`.

Código: PrintStream

```
1  package entradasalida;
2
3  import java.io.FileNotFoundException;
4  import java.io.PrintStream;
5
6  public class DemoPrintStream {
7    public static void main(String[] args) {
8      String nombreArchivo = "Salida.txt";
9      try (PrintStream fout = new PrintStream(nombreArchivo)) {
10         fout.println("Inicio");
11         fout.format("Línea %d %s\n", 1, "cadenita");
12         fout.println("Fin");
13     } catch (FileNotFoundException fnfe) {
14         System.err.println("No se encontró el archivo " + nombreArchivo + " y no pudo ser creado");
15     } catch (SecurityException se) {
16         System.err.println("No se tiene permiso de escribir en el archivo");
17     }
18   }
19 }
```

Procesando la entrada con Scanner

Código: Scanner e InputStream

```
1  package entradasalida;
2
3  import java.util.Scanner;
4
5  public class DemoScanner {
6
7    public static void main(String[] args) {
8      Scanner s = new Scanner(System.in);
9      while(s.hasNext()) {
10       String linea = s.nextLine();
11       //String linea = s.next();
12       if (linea.equals("")) {
13         // Sólo funciona después de ingresar una línea distinta de ""
14         System.out.println("No␣me␣des␣el␣avión,␣escribe␣algo.");
15       } else {
16         System.out.println("Eco:␣" + linea);
17       }
18       if(linea.equals("Adios")) break;
19     }
20     s.close();
21   }
22 }
```

Acceso a datos en la red

Código: URL

```java
package entradasalida;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DemoURLInputStream {
    public static void main(String[] args) {
        System.out.println("Ejecutando...");
        try {
            URL url = new URL("https://www.google.com/");
            String line;
            try (BufferedReader br = new BufferedReader(new InputStreamReader(url.openStream()))) {
                System.out.println("Abierto");
                while ((line = br.readLine()) != null) { System.out.println(line); }
            } catch (IOException ex) { System.err.println(ex); }
        } catch (MalformedURLException ex) { System.err.println("URL inválida"); }
        System.out.println("Terminado...");
    }
}
```

El buffer

- Leer caracter por caracter o byte por byte no es eficiente.
- Para leer bloques más largos de información se utilizan los *buffers*, secciones de memoria más grandes donde se almacena la información por procesar.
- Los objetos utilizados para este tipo de lectura/escritura son:
  - `BufferedReader` y `BufferedWriter`
  - `BufferedInputStream` y `BufferedOutputStream`

BufferedReader

Código: "BufferedReader"

```
1  package entradasalida;
2
3  import java.io.BufferedReader;
4  import java.io.FileNotFoundException;
5  import java.io.FileReader;
6  import java.io.IOException;
7
8  public class DemoBufferedReader {
9    public static void main(String[] args) {
10     try {
11         BufferedReader in = new BufferedReader(new FileReader("texto.txt"));
12         String line;
13         while((line = in.readLine()) != null) {
14             System.out.println(line);
15         }
16     } catch (FileNotFoundException e) {
17         System.err.println("No se encontró el archivo texto.txt ¿Olvidaste crearlo?");
18     } catch (IOException ioe) {
19         System.err.println("Error al leer el contenido de texto.txt");
20     }
21   }
22 }
```

Acceso aleatorio

1 Definiciones

2 Flujos en Java

3 Acceso aleatorio

Temas

3 Acceso aleatorio
- FileChannel

Acceso aleatorio a los archivos

- Permite acceder secciones específicas de un archivo, sin necesidad de hacerlo en forma secuencial.
- Se realiza con objetos tipo `FileChannel`, que forma parte de `Java NIO`

  https://docs.oracle.com/javase/tutorial/essential/io/rafs.html

FileChannel I

```
1  package entradasalida;
2
3  import java.io.BufferedReader;
4  import java.io.FileNotFoundException;
5  import java.io.FileReader;
6  import java.io.IOException;
7  import java.io.PrintStream;
8  import java.nio.ByteBuffer;
9  import java.nio.channels.FileChannel;
10 import java.nio.file.Path;
11 import java.nio.file.Paths;
12 import static java.nio.file.StandardOpenOption.READ;
13 import static java.nio.file.StandardOpenOption.WRITE;
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16
17 /**
18  * Muestra el acceso aleatorio, intercambiando dos líneas de un archivo de texto.
19  * @author blackzafiro
20  */
21 public class DemoAccesoAleatorio {
22
23     /**
24      * Escribe algunas líneas en el flujo indicado.
25      * @param out
```

FileChannel II

```
26          */
27         public static void escribeEnFlujo(PrintStream out) {
28             out.println("Rellenando archivo muestra");
29             for(int i = 1; i < 7; i++) {
30                 out.format("Linea %d%n", i);
31             }
32         }
33
34         /**
35          * Imprime en pantalla el contenido del archivo indicado.
36          * @param nombreArchivo
37          * @throws java.io.FileNotFoundException
38          */
39         public static void imprimeArchivo(String nombreArchivo) {
40             System.out.format("%n---* Contenido de %s *---%n", nombreArchivo);
41
42             try {
43                 BufferedReader in = new BufferedReader(new FileReader(nombreArchivo));
44                 String line;
45                 while((line = in.readLine()) != null) {
46                         System.out.println(line);
47                 }
48             } catch (FileNotFoundException ex) {
49                 System.err.println("No se encontró el archivo al leer," + ex);
50             } catch (IOException ex) {
```

FileChannel III

```
51              System.err.println("Error␣al␣leer,␣" + ex);
52          }
53
54          System.out.format("---*␣Fin␣del␣contenido␣de␣%s␣*---%n%n", nombreArchivo);
55      }
56
57      /**
58       *
59       * @param args
60       */
61      public static void main(String[] args) {
62          // Detectar directorio desde donde se leen los archivos
63          System.out.format("Directorio␣de␣trabajo:%n\t%s%n%n", System.getProperty("user.dir"));
64          System.out.flush();
65          Path ruta = Paths.get("datos/ArchivoAleatorio.txt"); // Dirección del archivo
66
67          // Vacía y rellena el archivo.
68          try (PrintStream ps = new PrintStream(ruta.toFile())) {
69              escribeEnFlujo(ps);
70          } catch (FileNotFoundException ex) {
71              System.err.println("No␣se␣encontró␣el␣archivo␣al␣escribir,␣" + ex);
72          }
73
74          // Muestra el contenido del archivo en la consola
75          imprimeArchivo(ruta.toString());
```

FileChannel IV

```
76
77
78          final int SKIP = 26;
79          final int BUFFER_SIZE = 13;
80
81          // Almacenar un texto como buffer de bytes.
82          String s = ">>>Inserción<<<";
83          byte data[] = s.getBytes();
84          ByteBuffer out = ByteBuffer.wrap(data);
85
86          // Buffer para los bytes que se leerán "aleatoriamente".
87          ByteBuffer copy = ByteBuffer.allocate(BUFFER_SIZE);
88
89          try (FileChannel fc = (FileChannel.open(ruta, READ, WRITE))) {
90              // Lee BUFFER_SIZE bytes a partir del byte SKIP en el archivo.
91              System.out.println("Leyendo...");
92              fc.position(SKIP);
93              int nread;
94              do {
95                  nread = fc.read(copy);
96                  System.out.format("se␣leyeron␣%d␣bytes%n", nread);
97              } while (nread != -1 && copy.hasRemaining());
98
99              // Regresa e inserta la cadena en s al inicio del archivo.
100             fc.position(0);
```

FileChannel V

```
101             while (out.hasRemaining()) {
102                 fc.write(out);
103             }
104             out.rewind();
105
106             // Se mueve a la mitad del archivo.
107             // Copia ahí los BUFFER_SIZE bytes que leyó, borrando lo que había.
108             long length = fc.size()/2;
109             fc.position(length);
110             copy.flip();
111             while (copy.hasRemaining()) {
112                 fc.write(copy);
113             }
114
115             // Se mueve al final del archivo.
116             // Agrega ahí el contenido de s otra vez.
117             length = fc.size();
118             fc.position(length);
119             while (out.hasRemaining()) {
120                 fc.write(out);
121             }
122         } catch (java.nio.file.NoSuchFileException x) {
123             System.err.println("No se encontró el archivo: " + x);
124         }
125         catch (IOException x) {
```

FileChannel VI

```
126                 System.err.println("I/O Exception: " + x);
127             }
128
129             // Muestra el contenido del archivo después de las modificaciones.
130             imprimeArchivo(ruta.toString());
131         }
132     }
```

```
Directorio de trabajo:
        /media/blackzafiro/.../Ejemplitos


---* Contenido de datos/ArchivoAleatorio.txt *---
Rellenando archivo muestra
Línea 1
Línea 2
Línea 3
Línea 4
Línea 5
Línea 6
---* Fin del contenido de datos/ArchivoAleatorio.txt *---

Leyendo...
se leyeron 13 bytes

---* Contenido de datos/ArchivoAleatorio.txt *---
>>>Inserción<<<vo muestra
Línea 1
Lín
Línea 1
Lí
Línea 4
Línea 5
Línea 6
>>>Inserción<<<
---* Fin del contenido de datos/ArchivoAleatorio.txt *---
```

Más allá

Hay muchas otras cosas que se pueden hacer con entrada/salida, especialmente para el manejo de archivos. Para ello, revisa el tutorial oficial en Oracle 2020.

Bibliografía I

- Oracle (16 de dic. de 2020). *Lesson: Basic I/O*. URL: https://docs.oracle.com/javase/tutorial/essential/io/index.html.
- Viso, Elisa y Canek Peláez V. (2012). *Introducción a las ciencias de la computación con Java*. 2a. Temas de computación. Las prensas de ciencias. 571 págs. ISBN: 978-607-02-3345-6.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual