Flujos de Entrada y Salida

Introducción

Programación

Abril 2018

Objetivo

- Comprender los flujos de entrada y salida en lenguajes de POO

Introducción

- Dispositivos de Almacenamientos
  - Son componentes para la lectura y escritura de datos (temporal o permanentemente)
  - Las computadoras poseen almacenamiento principal (RAM, ROM) y secundario (DD)
- Archivos
  - Son colecciones de datos almacenados permanentemente en dispositivos de almacenamiento
  - Son formados por un conjunto de registros
  - Tienen fin (eof)
  - Tradicionalmente son organizados en registros, los registros en campos, los campos en bytes y los bytes en bits
  - Existen archivos de texto y archivos binarios

Archivo

- Jerarquía de Datos

Nombre	Edad
Ana	10
Hugo	12
Verónica	14
María	9
Luis	10
Juan	13

Archivo

eof

Hugo	12

Registro

H u g o ← **Campo**

0100 1000 - **Byte** (carácter H en código ASCII - 48 hexadecimal)

1 **Bit**

Flujos

- Para la manipulación de un archivo almacenado permanentemente es necesario la creación de un **flujo**
- Los **flujos** son conductos a través del cual se transportan datos hacia o desde un dispositivo
- Cuando los datos van desde la memoria de un programa hacia el dispositivo de almacenamiento es **un flujo de salida (output)**
- Cuando los datos van desde el dispositivo de almacenamiento hacia la memoria de un programa es un **flujo de entrada (input)**

Flujos de Datos para Entrada y Salida Estándar

- Son flujos que actúan como canales de comunicación permitiendo la interacción entre un programa y su entorno en el sistema
- El acceso a estos flujos de entrada es desde la clase java.lang.System
- Salida Estándar
  - Está relacionada directamente con la terminal del sistema, de modo que los resultados enviados a este flujo son mostrados en la pantalla (aunque el destino puede ser modificado)
  - Existen dos tipos de salida: i) La regular y ii) la destinada para errores ocurridos en un programa
  - En Java, la salida estándar (StdOut) es representada por un objeto ***PrintStream*** llamado *out* en la clase ***System***
  - La clase ***PrintStream*** es una implementación de ***FilterOutputStream*** y por ende también de la clase base ***OutputStream***.
- Ejemplo de Salida Estándar:
  - System.out.println("Hola Mundo!"); //La clase System tiene una propiedad estática llamada out del tipo ***// PrintStream*** y que define el método println

Salida de Errores (System.err)

- Es otra salida estándar pero con el fin de ser utilizada para errores (StdErr)
- Al igual que StdOut es representada por un objeto **PrintStream** llamado err
- Los métodos que pueden ser invocados son los mismo que out
- Si el destino es el mismo para out y err no tendría mucha utilidad si en ambas salidas el color en la consola es el mismo

- System.out.println("Hola!");
- System.err.println("Hola!");

```
C:\Programacion\práctica 5 - Flujos>javac Main.java

C:\Programacion\práctica 5 - Flujos>java Main
Hola!
Hola!
```

- Se puede aprovechar de mejor forma si se re-define la salida estándar para out con el archivo "salida.txt" y de err con el archivo "errores.txt"

Redefinición de Salida Estándar

- En Java, para modificar el destino de **err** y **out** en dos archivos diferentes, se utilizan los métodos estáticos **void setOut(PrintStream out)** y **void setErr(PrintStream err)** de la clase **System**

- Para ello, es necesario abrir flujos de datos hacia los nuevos destinos de cada salida, para cada uno se crea un objeto **FileOutputStream(File file)** con un objeto **PrintStream(OutputStream out)**

  - System.setOut(new PrintStream(new FileOutputStream("salida_normal.txt")));
  - System.setErr(new PrintStream(new FileOutputStream("salida_error.txt")));

Entrada Estándar

- La entrada estándar de datos (StdIn) es representada por un objeto **InputStream**
- La clase **InputStream** es la clase base en el paquete java.io para manejar los flujos entrantes de bytes

- Lectura simple en bytes
- Para leer datos provenientes del teclado de un usuario se utiliza la variable **in** de la clase **System** y el método **int read()** de su correspondiente objeto InputStream
  - System.out.println(“Escribe una letra: “);
  - int in = System.in.read();
  - System.out.println(“Escribiste: “ + in);

```
C:\Programacion\práctica 5 - Flujos>javac Main.java

C:\Programacion\práctica 5 - Flujos>java Main
Escribe una letra:
l
Escribiste: 108
```

Entrada Estándar

- El método **int read()** lee concretamente un byte y lo devuelve representado como un **int** (número entero) entre 0 – 255
- Si la entrada de datos proviniera desde un archivo se podría obtener también -1 en caso de que se llegue al EOF (End Of File: fin de un archivo)
- En el ejemplo el 108 correspondería a la l minúscula en formato ASCII

```
C:\Programacion\práctica 5 - Flujos>javac Main.java

C:\Programacion\práctica 5 - Flujos>java Main
Escribe una letra:
l
Escribiste: 108
```

Lectura de Caracteres

- System.out.println(“Escribe una letra:");
- InputStreamReader isr = new InputStreamReader(System.in);
- int in = isr.read();
- char c = (char) in;
- System.out.println(“Escribiste: “ + c);

Otro Ejemplo

```
int in = 0;
while (in != -1){
        System.out.println("Escribe una letra: ");
        in = System.in.read();
        System.out.println("Escribiste: " + in);
}
```

Ejemplo con Arreglo de Bytes

- System.out.println(“Escriba 5 letras:");
-
- byte[] bufferIn = new byte[5];
- System.in.read(bufferIn);

- for (int i = 0 ; i < bufferIn.length ; i++) {
  System.out.println(“Escribiste: " + bufferIn[i]);
- }

Ejemplo con Arreglo de Char

- System.out.println(“Escribe 5 letras:”);
-
- InputStreamReader isr = new InputStreamReader(System.in);
- char[] bufferIn = new char[5];
- isr.read(bufferIn);
-
- for (int i = 0 ; i < bufferIn.length ; i++) {
- System.out.println(“Escribiste: " + bufferIn[i]);
- }

Lectura de una Cadena (1/2)

- Lectura de una línea de texto:
  - System.out.println(“Escribe algo:”);
  - InputStreamReader isr = new InputStreamReader(System.in);
  - BufferedReader br = new BufferedReader(isr);
  - String s = br.readLine();
  - System.out.println(“Escribiste: “ + s);

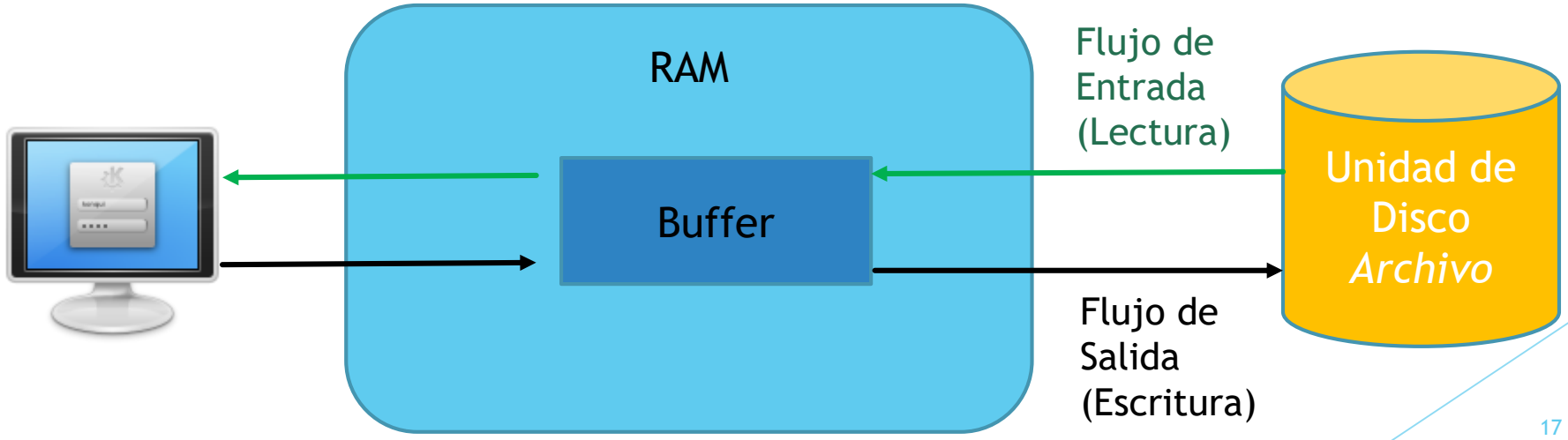
- En caso de leer un número y desea convertirlo en un entero utilizar:
  - int n = Integer.parseInt(s);

Lectura de una Cadena (2/2)

- Leyendo una línea de texto:
  - System.out.println(“Escribe una línea:”);
  - Scanner sc = new Scanner(System.in);
  - String s = sc.nextLine();
  - System.out.println(“Escribiste: “ + s);

Lectura/Escritura

- Las operaciones relacionadas al flujo de entrada son de **lectura (read, r)**
- Las operaciones relacionadas al flujo de salida son de **escritura (write, w)**



Operaciones Básicas de Archivos

- Creación
- Apertura
- Lectura
- Escritura
- Recorrido
- Cierre

Actividad 8 – Flujos de Entrada/Salida

- Elaborar un reporte abordando los siguientes aspectos
- Explicar las diferencias entre archivos de texto y archivos binarios
- Investigar la entrada estándar en Java, implementa un ejemplo donde solicites dos números y realices una operación aritmética
- Investigar el concepto ***casting*** en Java
- Entrega: en clases mediante un escrito