JAVA

MÓDULO Nro.3

1000 PROGRAMADORES

Ministerio de Economía y Servicios Públicos
Gobierno de Salta

Ministerio de Educación Cultura, Ciencia y Tecnología
Gobierno de Salta

Universidad Nacional de Salta

Temas:

Clase nro. 1

- **Entrada y Salida por consola**
- Variables de texto. Leer/escribir textos.
- Pilas.
- Colas.
- Listas.
- Hash Map
- Recursión

1000 PROGRAMADORES

MÓDULO 3.

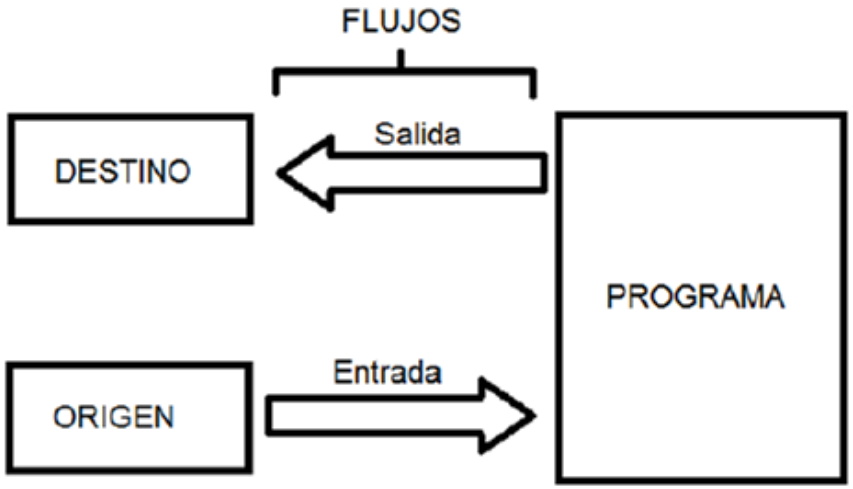
Soria Juan Pablo

Entradas y Salidas:

Se realizan mediante **FLUJOS (*stream*)**, para tratar la comunicación de información entre el programa y el exterior.

- **Flujos de entradas**: Se abre el flujo de entrada, para leer la información del flujo hasta el final y por último cerrar el flujo.

- **Flujos de Salidas**: Se abre el flujo de salida y a continuación se escribe en él toda la información que se desee, por último, se cierra el flujo.



Este esquema de entradas y salidas basadas en un flujo permite que:

- Las entradas sean independientes de la fuente de datos

- Las salidas sean independiente del destino de los datos.

Entradas y Salidas:

FLUJOS Estándar

En Java se accede a la E/S estándar a través de campos estáticos de la clase ***java.lang.System***

- ***System.in*** implementa la entrada estándar
- ***System.out*** implementa la salida estándar
- ***System.err*** implementa la salida de error

Entradas y Salidas:

1000 PROGRAMADORES

FLUJOS Estándar

System.in	
Instancia de: Clase InputStream	
Flujo de entrada: Bytes	
Metodos	**descripción**
read()	permite leer un byte de la entrada como entero
skip()	ignora n bytes de la entrada
available()	número de bytes disponibles para leer en la entrada

Entradas y Salidas:

1000 PRO GRA MA DORES

FLUJOS Estándar

System.out	
Instancia de: Clase PrintStream	
Flujo de Salida: Bytes	
Metodos	**descripción**
print()	muestra caracteres por patanlla
println()	muestra caracteres por patanlla pero hace un salto de línea al final
flush()	vacía el buffer de salida escribiendo su contenido

Entradas y Salidas:

Ejemplo de Flujos estándar: Se lee caracteres hasta encontrar el fin de línea (“\n”), posteriormente se muestra el total de bytes ingresados (se tiene en cuenta el salto de línea “\n”).

App.java

src > App.java > LecturaDeLinea

```java
class LecturaDeLinea {
    Run | Debug
    public static void main(String[] args) throws Exception {
        int c;
        int contador = 0;
        System.out.print( s: "se lee hasta encontrar el fin de línea: ");
        while( (c = System.in.read() ) != '\n' )
        {
            contador++;
            System.out.println("lo que interpreta el compilador: "+  c );
            System.out.println("lo que ve la persona: "+ (char) c );
        }
        System.out.println(); // Se escribe el fin de línea
        System.err.println( "Contados "+ contador +" bytes en total." );
    }
}
```

- El programa en este caso lee caracter por caracter.
- **NO LEE CADENA DE TEXTO.**
- **MUESTRA CARACTER POR CARACTER**
- Ejecutar para ver el resultado

System.err es similar a System.out (se utiliza para enviar msj de error)

Soria Juan Pablo

Entradas y Salidas:

Clasificación de FLUJOS

En Java se pueden representar de las siguientes forma los flujos de E/S:

- Flujos de bytes**:** clases **InputStream y OutputStream** (flujos estándar)

- Flujos de caracteres: clases **Reader** y **Writer**
    - Se puede pasar de un flujo de bytes a uno de caracteres con **InputStreamReader y OutputStreamWrite**

Propósito

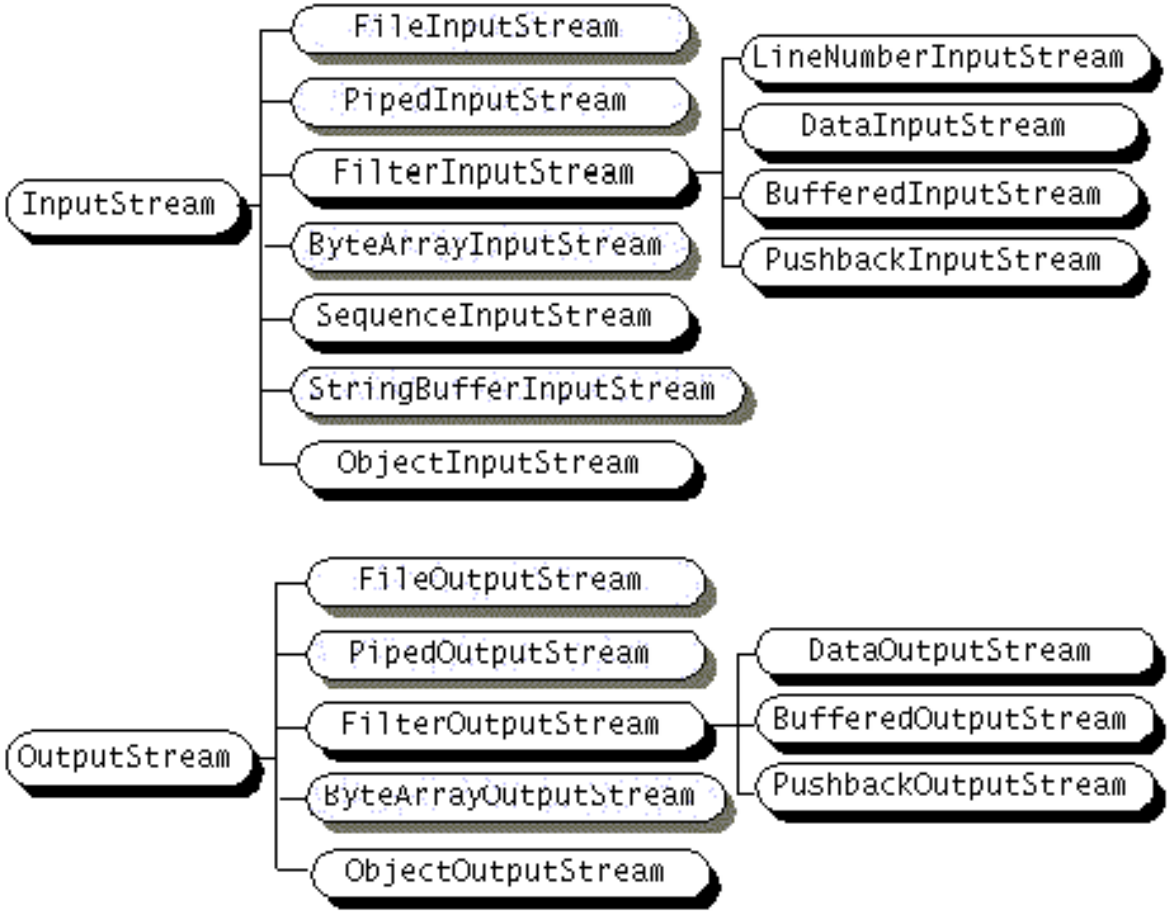
- Entrada: **InputStream, Reader**
- Salida: **OutputStream, Write**
- Lectura/Escritura: **RandomAccessFile**
- Transformación de los datos
    - Realizan algún tipo de procesamiento sobre los datos (p.e. buffering, conversiones, filtrados): **BuffuredReader**, **BufferedWriter**

Entradas y Salidas:

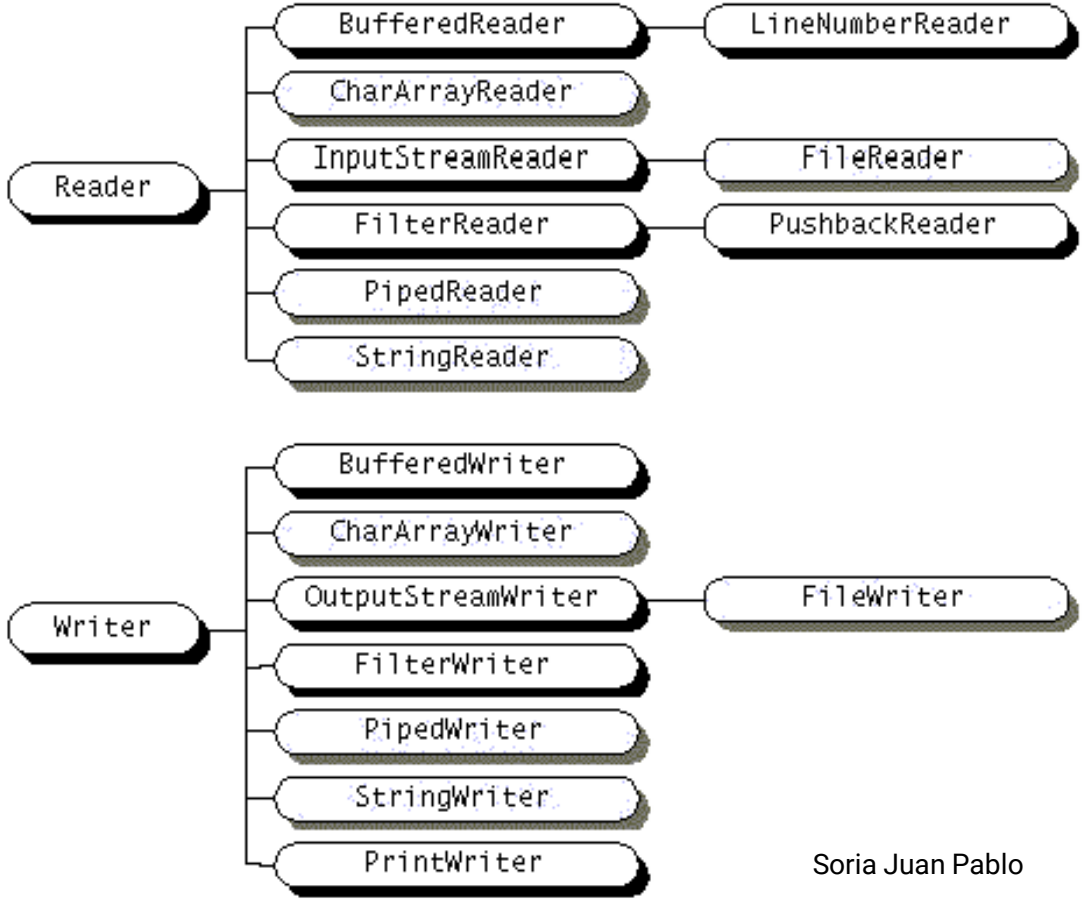
1000 PRO GRA MA DORES

Jerarquía de las clases de los flujos

flujos de E/S de bytes (8 bits)



flujos de E/S de caracteres(char unicode 16 bits)



Soria Juan Pablo

Entradas y Salidas:

1000 PRO GRA MA DORES

FLUJOS de Caracteres

- **InputStreamReader**
  - Lee Bytes (8 bits) de un flujo de InputStream y los convierte en caracteres unicode (16 bits)
  - Métodos de utilidad:
    - read() lee un único caracter
    - ready() indica cuando está listo el flujo de lectura
- **BufferedRead**
  - Entrada mediante búfer, mejora el rendimiento
  - Método de utilidad
    - readLine() lectura de una línea como cadena

Soria Juan Pablo

Entradas y Salidas:

Ejemplo de Flujos de caracteres: Se lee caracteres hasta encontrar el fin de línea (enter), posteriormente se muestra el total de bytes ingresados.

App.java

src > App.java > LecturaDeLinea > main(String[])

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

class LecturaDeLinea {
    Run | Debug
    public static void main(String[] args) throws Exception {
        String c;
        System.out.print( s: "se lee hasta encontrar el fin de línea: ");
        InputStreamReader entrada = new InputStreamReader(System.in);
        BufferedReader teclado = new BufferedReader(entrada);
        c = teclado.readLine();
        System.err.println( "Contados "+ c.length() +" bytes en total." );
    }
}
```

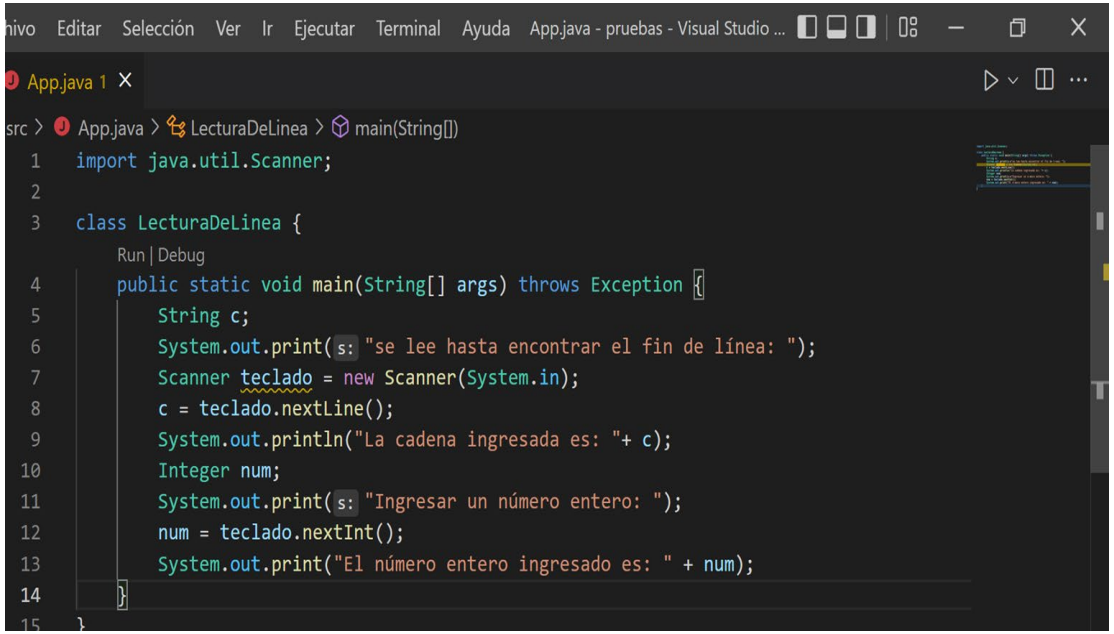
- El programa en este caso lee los ccaracter y los almacena en un búffer.
- **SE ALMACENA EN UNA CADENA DE TEXTO.**
- Ejecutar para ver el resultado

Soria Juan Pablo

Entradas y Salidas:

FLUJOS de Caracteres - Clase SCANNER

- Es la forma más fácil de leer datos
- No es muy eficiente para programación competitiva (paralelismo)
- Funciones asociadas:
    - **nextLine()**: Lee cadenas (String)
    - **nextInt() / nextDouble()** : Lee números enteros / número flotante
    - **next().charAt(0)**: Lee un solo carcater



Soria Juan Pablo

Temas:

Clase nro. 1

- Entrada y Salida por consola.
- **Variables de texto. Leer/escribir textos.**
- Pilas.
- Colas.
- Listas.
- Hash Map
- Recursión

MÓDULO 3.

1000 PROGRAMADORES

Soria Juan Pablo

Variables de texto. Leer/escribir textos:

1000 PROGRAMADORES

- **Clase File**
  - La clase File se usa para obtener información sobre **archivos y directorios**. Además permite crear y eliminar archivos y directorios.
  - Constructores:
    - File(String ruta)
    - File(String ruta, String nombre)
    - File(File directorio, String nombre)
  - Algunos Métodos (consultar: https://docs.oracle.com/javase/8/docs/api/java/io/File.html)
    - canRead(): comprueba si el fichero se puede leer
    - canWrite(): comprueba si el fichero se puede escribir
    - delete(): borra dicho fichero
    - getPath(): devuelve la ruta del fichero
    - mkdir(): crea un directorio con la ruta del objeto que lo recibe
    - isDirectory(): comprueba si dicho fichero es un directorio

Variables de texto. Leer/escribir textos:

1000 PROGRAMADORES

- **Clase File**
  - Constructores de otras clases
    - FileReader(File fichero)
    - FileWriter(File fichero)

Leer datos de un archivo

- Las clases **FileReader y BufferedReader**
  - la primera crea el flujo principal y la segunda el flujo intermedio.

FileReader se usa para leer un archivo desde una unidad de disco, mientras que **BufferedReader** no está limitado solo a leer archivos. Se puede utilizar para leer datos de cualquier secuencia de caracteres.

Soria Juan Pablo

Variables de texto. Leer/escribir textos:

1000 PROGRAMADORES

- **Diferencias entre BufferedReader y FileReader**

BASE	BUFFEREDREADER	FILEREADER
Usar	Se utiliza para leer caracteres de cualquier tipo de flujo de entrada de caracteres (string, archivos , etc.)	Solo se puede usar para leer archivos.
Buffer	Utiliza Buffer internamente para leer caracteres.	No usa Buffer. Lee directamente del archivo accediendo al disco duro.
Velocidad	Más rápido	Más lento
Eficiencia	Mucho más eficiente para leer archivos	Menos eficiente
Líneas de lectura	BufferedReader se puede utilizar para leer un solo carácter a la vez, así como una línea a la vez.	Puede leer solo un carácter a la vez, no puede leer líneas

Variables de texto. Leer/escribir textos:

1000 PROGRAMADORES

Escribir datos en un archivo

- Las clases **FileWriter y BufferedWriter**
  - la primera crea el flujo principal y la segunda el flujo intermedio.

FileReader nos permite realizar escrituras de caracteres de texto sobre un fichero, mientras que **BufferedWriter** nos permite escribir texto en un Outputstream, utilizando un buffer para proporcionar una escritura eficiente de caracteres, arrays y strings.

Las diferencias entre una y otra es similares a su contraparte de lectura.

Ver: código Fichero

Gracias.

WEB: http://milprogramadores.unsa.edu.ar/

CANAL TELEGRAM: https://t.me/milprogramadoressaltenios

CENTRO DE AYUDA: http://ayudamilprogramadores.com/

1000 PROGRAMADORES