Entrada y Salida de datos

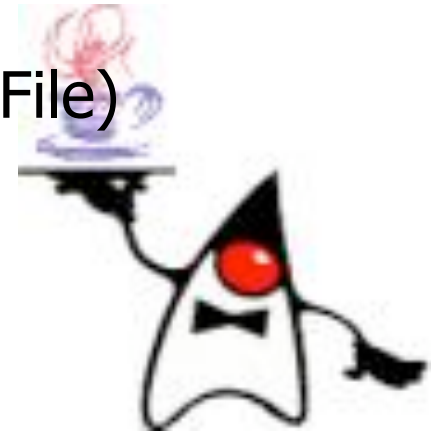
- El intercambio de datos entre el programa y el exterior da lugar a la diversidad de dispositivos y de formas de comunicación (modo de acceso, si es binaria, etc.).

- Los flujos estándar que maneja Java en el paquete *java.lang.System* son:
  - System.out (salida a pantalla)
  - System.err (salida de error a pantalla)
  - System.in (entrada desde teclado)

- Los Streams (flujos) son objetos de I/O con operaciones para aceptar y mandar bytes. Además se usan para leer y escribir datos independientes de la plataforma.

Clasificación de flujos

- Por la forma de representación de la información:
  - ✓ Flujos de bytes 8 bits (InputStream,OutputStream).
  - ✓ Flujos de caracteres 16 bits (Reader, Writer).
- Por su propósito:
  - ✓ Entrada (InputStream,Reader).
  - ✓ Salida(OutputStream,Writer).
  - ✓ Entrada/Salida (RandomAccessFile)
- Por el tipo de acceso:
  - ✓ Secuencial
  - ✓ Directo (RandomAccessFile)



Paquete java.io (bytes)

- InputStream

  Es una clase abstracta que es la superclase que representan los flujos de **entrada de bytes**.

  - ✓ AudioInputStream
  - ✓ ...
  - ✓ FileInputStream
  - ✓ FilterInputStream
    - – BufferedInputStream
    - – ...
    - – DataInputStream
  - ✓ ObjectInputStream
    - – DataInputStream
    - – ...
    - – ObjectInputStream
  - ✓ StringBufferInputStream@

Paquete java.io (bytes)

- InputStream
- Métodos más comunes:
  - public int read() throws IOException
  - public int read(byte []buffer) throws IOException
  - public int read(byte [] buffer,int offset, int length)
    throws IOException
  - public long skip(long n) throws IOException
  - public void close() throws IOException

Paquete java.io (bytes)

- OutputStream

  Es una clase abstracta que es la superclase que representan los flujos de **salida de bytes**.

  - ✓ ByteArrayOutputStream
  - ✓ ...
  - ✓ FileOutputStream
  - ✓ FilterOutputStream
    - – BufferedOutputStream
    - – ...
    - – DataOutputStream
  - ✓ ObjectOutputStream
    - – DataOutputStream
    - – ...
    - – ObjectOutputStream
  - ✓ PipedOutputStream

Paquete java.io (bytes)

- OutputStream
- Métodos más comunes
  - public void write(int b) throws IOException

    //Escribe sólo un Byte, los 24 Bits restantes son ignorados.
  - public void write(byte[] b) throws IOException
  - public void write(byte[] b, int off, int len) throws IOException
  - public void flush() throws IOException
  - public void close() throws IOException

Paquete java.io (caracteres)

- Reader

  Es una clase abstracta para la **lectura de** streams de **caracteres**.

  - ✓ BufferedReader
    - ✓ LineNumberReader
  - ✓ ...
  - ✓ FilterReader
    - – PushBackReader
  - ✓ InputSreamReader
    - – FileReader
  - ✓ StringReader

Paquete java.io (caracteres)

- Reader
- Métodos más comunes:
  - public int read() throws IOException
  - public int read(char[] cbuf) throws IOException
  - public int read(char[] cbuf, int off, int len) throws IOException
  - public long skip(long n) throws IOException
  - public void close() throws IOException

Paquete java.io (caracteres)

- Writer

  Es una clase abstracta para la **salida de** streams de **caracteres**.

  - ✓ BufferedWriter
  - ✓ ...
  - ✓ FilterReader
    - – PushBackReader
  - ✓ OutputSreamReader
    - – FileWriter
  - ✓ PrintWriter
  - ✓ StringWriter

Paquete java.io (caracteres)

- Writer
- Métodos más comunes:
  - public void write(int c) throws IOException
  - public void write(char[] cbuf)throws IOException
  - public void write(char[] cbuf,int off,int len)
    throws IOException
  - public void write(String str) throws IOException
  - public void write(String str,int off,int len) throws IOException
  - public void flush() throws IOException
  - public void close() throws IOException

System.out

- Instancia de la clase PrintStream.

- Métodos mas usado son:
  - ✓ print (escribe en el buffer)
  - ✓ println (escribe en el buffer y flush.Deja el cursor en la siguiente línea)
  - ✓ flush (vacía el buffer de salida escribiendo su contenido)

- System.err
  - » Su funcionamiento es similar que err, es usado para salida de errores.

System.in

- Instancia de la clase PrintStream.

- Métodos mas usado son:
  - ✓read (permite leer un byte de la entrada como entero)
  - ✓skip(*n*) (ignora *n* bytes de la entrada)
  - ✓available (número de bytes disponibles para leer en la entrada)

Ejemplo1 de System (.in,.out,.err)

- Ejemplo1:

```
import java.io.*;
class LecturaLinea{
    public static void main (String []arg) throws
    IOException{
       int teclado;
       int c=0;
      while ((teclado=System.in.read())!= '\n'){
            c++;
            System.out.print( (char) teclado);
       }
      System.out.println();
      System.err.println("Fueron: "+c+"bytes");
   }
}
```

Resumen de Clases Streams en Java

Type	Character Streams	Byte Streams
File	`FileReader` `FileWriter`	`FileInputStream` `FileOutputStream`
Buffering	`BufferedReader` `BufferedWriter`	`BufferedInputStream` `BufferedOutputStream`
Converting between bytes and character	`InputStreamReader` `OutputStreamWriter`	
Performing object serialization		`ObjectInputStream` `ObjectOutputStream`
Performing data conversion		`DataInputStream` `DataOutputStream`
Printing	`PrintWriter`	`PrintStream`

Ejemplos (Combinación de flujos)

- Ejemplo1

```
1  import java.io.*;
2
3 public class KeyboardInput {
4 public static void main (String args[]) {
5 String s;
6 // Create a buffered reader to read
7 // each line from the keyboard.
8 InputStreamReader ir
9 = new InputStreamReader(System.in);
10 BufferedReader in = new BufferedReader(ir);
11
12 System.out.println("Unix: Type ctrl-d to exit." +
13 ”\nWindows: Type ctrl-z to exit");
14 try {
15 // Read each input line and echo it to the screen.
16 s = in.readLine();
```

Ejemplos (Combinación de flujos)

- Ejemplo1 (continuación)

```
16 s = in.readLine();
17 while ( s != null ) {
18 System.out.println("Read: " + s);
19 s = in.readLine();
20 }
21
22 // Close the buffered reader.
23 in.close();
24 } catch (IOException e) {  // Catch any IO
     exceptions.
25 e.printStackTrace();
26 }
27 }
 28 }
```

Ejemplos (Combinación de flujos)

- Ejemplo2:

```
import java.io.*;

class EcoLinea{
    public static void main (String []arg) throws
    IOException{
      BufferedReader teclado =new
    BufferedReader(new
    InputStreamReader(System.in);
   String mensaje;
   System.out.println("Introducir una linea de texto");
   mensaje=teclado.readLine();
   System.out.println("Se introdujo: "+ mensaje);
   }
}
```

Ejemplos (flujo de texto)

- Ejemplo1

```
import java.io.*;
class BufferedConsola{
    public static void main (String []arg) throws
     IOException{
       BufferedReader br =new  BufferedReader(new
     InputStreamReader(System.in));
       BufferedWriter bw =new  BufferedWriter(new
     OutputStreamWriter(System.out));
      System.out.println("Introducir una serie de
     caracteres, EXIT para salir");
      String s=br.readLine();
      while (!s.equals("EXIT"){
                bw.write("Linea: " + s);
                bw.newLine();
                s=br.readLine();
    }
    bw.close();
    br.close();
}
}
```

Ejemplos (archivos de texto)

- Ejemplo2

```
import java.io.*;
class ArchivoTexto{
    public static void main (String []arg) {
    try{
       PrintWriter salida =new  PrintWriter(new
     FileWriter("Ejemplo.txt"));
      salida.println("Estamos realizando una prueba con
     las clases stream");
      salida.println("con manejo de caracteres");
      salida.close();
 //entrada
 BufferedReader entrada =new  BufferedReader(new
     FileReader("Ejemplo.txt"));

      String s1,s2=new String();
      while ((s1= entrada.readLine())!=null)
              s2= s2 + s1 +"\n";
    System.out.println("Texto leido desde archivo: "+
     s2);
    entrada.close();
    } catch(IOException e){ }
}
}
```