ENTRADA Y SALIDA POR CONSOLA

Los programas a veces necesitan acceder a los recursos del sistema, como por ejemplo los dispositivos de entrada/salida estándar, para recoger datos de teclado o mostrar datos por pantalla.

En Java, la entrada por teclado y la salida de información por pantalla se hace mediante la clase `System` del paquete `java.lang` de la Biblioteca de Clases de Java.

Como cualquier otra clase, está compuesta de métodos y atributos. Los atributos de la clase System son tres objetos que se utilizan para la entrada y salida estándar. Estos objetos son los siguientes:

- `System.in`. Entrada estándar: teclado.
- `System.out`. Salida estándar: pantalla.
- `System.err`. Salida de error estándar, se produce también por pantalla, pero se implementa como un fichero distinto al anterior para distinguir la salida normal del programa de los mensajes de error. Se utiliza para mostrar mensajes de error.

No se pueden crear objetos a partir de la clase System, sino que se utiliza directamente llamando a cualquiera de sus métodos con el operador de manipulación de objetos, es decir, el operador punto (`.`):

```
System.out.println("Bienvenido a Java");
```

1. Conceptos sobre la clase System.

En *java* tenemos accesible el teclado desde *System.in*, que es un *InputStream* del que podemos leer *bytes*. Por ejemplo, podemos leer *bytes* del teclado de esta forma

```
// Lectura de un byte
int byte = System.in.read();

// Lectura de hasta 10 bytes
byte [] buffer = new byte[10];
System.in.read(byte);
```

El problema de leer *bytes*, es que luego debemos convertirlos a lo que necesitemos. Por ejemplo, si tecleamos una letra A mayúscula, el *byte* leído es el 65, correspodiente a la A mayúscula en código *ASCII*. Si tecleamos un 3 y un 2, es decir, un 32, leeremos dos bytes 51 y 52, correspondientes a los caracteres *ASCII* del 3 y del 2, NO leeremos un 32.

Con esta clase podemos utilizar por ejemplo el método `read()` que permite leer un byte de la entrada o `skip(long n)`, que salta n bytes de la entrada. Pero lo que realmente nos interesa es poder leer texto o números, no bytes, para hacernos más cómoda la entrada de datos. Para ello se utilizan las clases:

- `InputStreamReader`. Convierte los bytes leídos en caracteres. Particularmente, nos va a servir para convertir el objeto `System.in` en otro tipo de objeto que nos permita leer caracteres.
- `BufferedReader.` Lee hasta un fin de línea. Esta clase tiene un método readLine() que nos va a permitir leer caracteres hasta el final de línea.

La forma de instanciar estas clases para usarlas con System.in es la siguiente:

```
InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader (isr);
```

En el código anterior hemos creado un `InputStreamReader` a partir de `System.in` y pasamos dicho `InputStreamReader` al constructor de `BufferedReader`. El resultado es que las lecturas que hagamos con el objeto br son en realidad realizadas sobre `System.in`, pero con la ventaja de que podemos leer una línea completa. Así, por ejemplo, si escribimos una `A`, con:

```
String cadena = br.readLine();
```

Obtendremos en `cadena` una "`A`".

Sin embargo, seguimos necesitando hacer la conversión si queremos leer números. Por ejemplo, si escribimos un entero `32`, en cadena obtendremos "`32`". Si recordamos, para convertir cadenas de texto a enteros se utiliza el método estático `parseInt()` de la clase `Integer`, con lo cual la lectura la haríamos así:

```
int numero = Integer.parseInt (br.readLine());
```

2. Entrada por teclado. Clase System.

A continuación vamos a ver un ejemplo de cómo utilizar la clase `System` para la entrada de datos por teclado en Java.

```
import java.io.BufferedReader;

import java.io.InputStreamReader;

/*
 * Ejemplo de entrada por teclado con la clase System
 */
public class EntradaTecladoSystem {
    public static void main(String[] args) {
        try
        {
            InputStreamReader isr = new InputStreamReader(System.in);
            BufferedReader br = new BufferedReader(isr);

            System.out.print("Introduce el texto: ");
            String cad = br.readLine();

            //salida por pantalla del texto introducido
            System.out.println(cad);

            System.out.print("Introduce un numero: ");
            int num = Integer.parseInt(br.readLine());

            // salida por pantalla del numero introducido
            System.out.println(num);

        } catch (Exception e) {
           // System.out.println("Error al leer datos");
            e.printStackTrace();
        }
    }
}
```

3. Entrada por teclado. Clase Scanner.

La entrada por teclado que hemos visto en el apartado anterior tiene el inconveniente de que sólo podemos leer de manera fácil tipos de datos `String`. Si queremos leer otros tipos de datos deberemos convertir la cadena de texto leída en esos tipos de datos.

El kit de Desarrollo de Java, a partir de su versión 1.5, incorpora la clase `java.util.Scanner`, la cual permite leer tipos de datos `String`, `int`, `long`, etc., a través de la consola de la aplicación. Por ejemplo para leer un tipo de datos entero por teclado sería:

```
Scanner teclado = new Scanner (System.in);

int i = teclado.nextInt ();
```

O bien esta otra instrucción para leer una línea completa, incluido texto, números o lo que sea:

```
String cadena = teclado.nextLine();
```

En las instrucciones anteriores hemos creado un objeto de la clase Scanner llamado teclado utilizando el constructor de la clase, al cual le hemos pasado como parámetro la entrada básica del sistema System.in que por defecto está asociada al teclado.

```
import java.util.Scanner;

/*
 * Ejemplo de entrada de teclado con la clase Scanner
 */
public class EntradaTecladoScanner {

    public static void main(String[] args) {

        // Creamos objeto teclado
        Scanner teclado = new Scanner(System.in);

        // Declaramos variables a utilizar
        String nombre;
        int edad;
        boolean estudias;
        float salario;

        // Entrada de datos
        System.out.println("Nombre: ");
        nombre=teclado.next();
        System.out.println("Edad: ");
        edad=teclado.nextInt();
        System.out.println("Estudias: ");
        estudias=teclado.nextBoolean();
        System.out.println("Salario: ");
        salario=teclado.nextFloat();

        // Salida de datos
        System.out.println("Bienvenido: " + nombre);
        System.out.println("Tienes: " + edad +" años");
        System.out.println("Estudias: " + ((estudias)?”Si”:”No”));
        System.out.println("Tu salario es: " + salario +" euros");

        teclado.close();
    }

}
```

Esta clase es bastante potente, por las siguientes características:

- Tiene varios constructores que admiten, además de System.in, cosas como secuencias de bytes o ficheros. Esto nos permite leer, por ejemplo, ficheros de forma más cómoda.
- Los métodos nextInt() admiten un entero. Por ejemplo, si tecleamos FF y hacemos la lectura con 16, obtendríamos un 255.

```
// Lectura de un número en hexadecimal.
int entero = sc.nextInt(16);
```

- Admite Expresiones Regulares en Java como patrones de búsqueda, por lo que podemos leer trozos de línea directamente usando los separadores que queramos o buscando expresiones concretas. Por ejemplo, si introducimos 11:33:44, usando el siguiente código obtendremos los número 11, 33 y 44

```
Scanner sc = new Scanner(System.in);

// Usamos como delimitador los dos puntos, o bien cualquier espacio (el \\s)
sc.useDelimiter("[:\\s]");

// Leemos los tres enteros
int a = sc.nextInt();
int b = sc.nextInt();
int c = sc.nextInt();

// Obtendremos 11-33-44 de salida.
System.out.println(a +"-"+ b +"-"+ c);
```

4. Salida por pantalla.

La salida por pantalla en Java se hace con el objeto `System.out`. Este objeto es una instancia de la clase `PrintStream` del paquete `java.lang`. Si miramos la API de `PrintStream` obtendremos la variedad de métodos para mostrar datos por pantalla, algunos de estos son:

- `void print(String s):` Escribe una cadena de texto.
- `void println(String x):` Escribe una cadena de texto y termina la línea.
- `void printf(String format, Object… args):` Escribe una cadena de texto utilizando formato.

En la orden print y println, cuando queramos escribir un mensaje y el valor de una variable debemos utilizar el operador de concatenación de cadenas (+), por ejemplo:

```
System.out.println(“Bienvenido, “ + nombre);
```

Escribe el mensaje de "Bienvenido, Carlos", si el valor de la variable nombre es Carlos.

Las órdenes `print` y `println,` todas las variables que escriben las consideran como cadenas de texto sin formato, por ejemplo, no sería posible indicar que escriba un número decimal con dos cifras decimales o redondear las cifras, o escribir los puntos de los miles, por ejemplo. Para ello se utiliza la orden `printf()`.

La orden `printf()` utiliza unos códigos de conversión para indicar del contenido a mostrar de qué tipo es. Estos códigos se caracterizan porque llevan delante el símbolo `%`, algunos de ellos son:

- `%c:` Escribe un carácter.
- `%s:` Escribe una cadena de texto.
- `%d:` Escribe un entero.
- `%f:` Escribe un número decimal.

- `%e:` Escribe un número en punto flotante en notación científica.

Estas órdenes pueden utilizar las secuencias de escape que vimos en unidades anteriores, como `"\n"` para crear un salto de línea, `"\t"` para introducir un salto de tabulación en el texto, etc.

Esto mismo sirve para darle formato a una cadena con el método estático *format* de la clase String. Ejemplo:

```
String resultado;
resultado=String.format("Nombre: %s Edad: %d Sueldo: %f", "Juan",20,1896.23);
System.out.println(resultado);
```

Si queremos mostrar el número 12.3698 de tipo double con dos decimales:

System.out.printf("%.2f %n", 12.3698);

El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.
.2 indica el número de decimales.
La f indica que el número es de tipo float o double.
%n indica un salto de línea. Equivale a \n. Con printf podemos usar ambos para hacer un salto de línea.

La salida por pantalla es:
12,37

Comprobamos que printf realiza un redondeo para mostrar los decimales indicados.
Lo más común será que tengamos el valor en una variable, en ese caso si queremos escribir el valor de n con tres decimales:

double n = 1.25036;
System.out.printf("%.3f %n", n);
Salida:
1,250

Si el número a mostrar es un entero se utiliza el caracter d:
int x = 10;
System.out.printf("%d %n", x);
Salida:
10

Para mostrarlo con signo:
int x = 10;
System.out.printf("%+d %n", x);
Salida:
+10

Si el número fuera negativo habría salido -10.

Para mostrar varias variables pondremos tantos % como valores vamos a mostrar. Las variables se escriben a continuación de las comillas separadas por comas:

```
double n = 1.25036;
int x = 10;
System.out.printf("n = %.2f x = %d %n", n, x);
```
Salida:
n = 1,25 x = 10

Cuando hay varias variables podemos indicar de cual de ellas es el valor a mostrar escribiendo 1$, 2$, 3$, ... indicando que el valor a mostrar es el de la primera variable que aparece a continuación de las comillas, de la segunda, etc.
La instrucción anterior la podemos escribir así:
```
System.out.printf("n = %1$.2f x = %2$d %n", n, x);
```

Este número es opcional, si no aparece se entenderá que el primer valor proviene de la primera variable, el segundo de la segunda, etc.

Si queremos mostrar el número 123.4567 y su cuadrado ambos con dos decimales debemos escribir:

```
double n = 123.4567;

System.out.printf("El cuadrado de %.2f es %.2f\n", n, n*n);
```

Salida:

El cuadrado de 123,46 es 15241,56

printf permite mostrar valores con un ancho de campo determinado. Por ejemplo, si queremos mostrar el contenido de n en un ancho de campo de 10 caracteres escribimos:

```
double n = 1.25036;
System.out.printf("%+10.2f %n", n);
```
Salida:
bbbbb+1.25
Donde cada *b* indica un espacio en blanco.
El 10 indica el tamaño en caracteres que ocupará el número en pantalla. Se cuentan además de las cifras del número el punto decimal y el signo si lo lleva. En este caso el número ocupa un espacio de 5 caracteres (3 cifras, un punto y el signo) por lo tanto se añaden 5 espacios en blanco al principio para completar el tamaño de 10.

Si queremos que en lugar de espacios en blancos nos muestre el número completando el ancho con ceros escribimos:
```
System.out.printf("%+010.2f %n", n);
```
Salida:
+000001.25

Más ejemplos de printf:

Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.

```
double precio = 1.22;

System.out.printf("%10.2f", precio);
```

Salida:

*bbbbbb*1.22
(el carácter b indica un espacio en blanco)

El número ocupa un espacio total de 10 caracteres incluyendo el punto y los dos decimales.

Mostrar la cadena "Total:" con un ancho de 10 caracteres y alineada a la izquierda:

System.out.printf("%-10s", "Total:");

Salida:

Total:*bbbb*

El caracter s indica que se va a mostrar una cadena de caracteres.
El signo - indica alineación a la izquierda.

Mostrar la cadena "Total:" con un ancho de 10 caracteres y alineada a la derecha:

System.out.printf("%10s", "Total:");

Salida:

*bbbb*Total: