ENTRADA Y SALIDA ESTÁNDAR

1. Escritura de datos en la consola del sistema

2. Lectura de datos a través de la consola

2.1. Sistema de lectura de datos en la consola del sistema; clases InputStreamReader y BufferedReader

2.2. Sistema de lectura de datos en la consola del sistema; clase Scanner

3. Lectura de datos a través de consola con sistema de validación

1. Escritura de datos en la consola del sistema

Ya hemos usado en todos nuestros programas la salida de datos por la pantalla mediante **System.out** (representa la consola de salida del sistema) que pertenece a la clase **PrintStream**. Esta clase cuenta, entre otros métodos, con **print (), println ()** que permiten enviar cadenas de caracteres a la consola del sistema.

La clase **PrintStream**, está siendo sustituida por la clase **PrintWrite** que gestiona adecuadamente los caracteres Unicode. Sin embargo, los métodos anteriores se siguen usando sin que el compilador muestre el mensaje de método obsoleto (*deprecation*).

Desde la versión J2SE5, se incorpora a estos métodos de escritura en la consola del sistema, el método **printf,** que admite mascarillas para dar formato a los datos de salida, de forma similar a la función printf del lenguaje C. Podéis ver este método y sus dos versiones en la clase PrintStream.

2. Lectura de datos a través de la consola

En general para leer una fuente de datos, bien sea un fichero, el teclado u otro periférico, es necesario construir tres objetos que denominaremos *fuente, flujo y filtro,* siendo el objeto *fuente* el que representa al dispositivo origen de los datos, *flujo* el medio para transportar los datos, y *filtro* el dispositivo de interpretación de los datos para cada tipo y uso específico.

En el siguiente apartado, se recoge la forma clásica de leer datos del teclado a través de la consola del sistema en versiones previas a la J2SE5 (todavía usada pero en proceso de desaparición). Después, veremos la forma que probablemente ya se usa con más frecuencia a partir de la versión anterior de Java, que está basada en la clase **Scanner**.

2.1. Sistema de lectura de datos en la consola del sistema; clases InputStreamReader y BufferedReader

El canal identifica la fuente de datos, en el caso de la lectura de datos a través de la consola del sistema, existe el objeto **System.in** (objeto in de la clase System) que identifica la consola como periférico de entrada. El flujo es el objeto encargado de transferir los datos a través de un flujo de bytes indiferenciados. El filtro es el objeto que proporciona los métodos que posibilitan la interpretación del flujo de bytes y los convierte en primitivas o cadenas de caracteres.

El siguiente ejemplo, muestra cómo leer una cadena de caracteres de la consola del sistema.

La primera sentencia es:

import java.io.*;

Esta sentencia permite usar las clases de la librería java.io (package) que contiene entre otras las utilidades de entrada y salida de datos en el programa.

NOTA: A partir de ahora, cada vez que tengamos que usar algún método de la API de java, tendremos que importar el paquete correspondiente donde se encuentredicho método mediante sentencias de este tipo. Es posible importar el paquete entero o una única clase del paquete.

La sentencia: ***throwsException***

Permite compilar el programa, ya que las sentencias de entrada de datos requieren la técnica de control de excepciones (errores) que se verá en el tema 8. De momento, para poder usar las sentencias de leer datos desde el teclado se usa esta sentencia, que establece si se produce una excepción, es decir, un error en el programa, se genere un objeto especial de la clase **Exception.** Este proceso, tal como se usa aquí, provoca que el programa termine de forma "traumática". Más adelante mostraremos cómo controlar esta terminación anormal.

```
import java.io.*;
public class DameTuNombre {
    public static void main (String arg[]) throws Exception{
        String nombre;
        //se declaran las variables de referencia
        //de nombre flujo de la clase InputStreamReader
        //y teclado de la clase BufferedReader
        InputStreamReader flujo;
        BufferedReader teclado;

        //se construyen los objetos flujo y teclado
        flujo = new InputStreamReader(System.in);
        teclado = new BufferedReader(flujo);
        System.out.print("Escribe tu nombre= ");
        nombre = teclado.readLine();
```

```
        System.out.println("Tu nombre es: "+ nombre);
    }
}
```

El constructor toma como argumento el objeto que identifica la fuente de datos, en este caso, **System.in.** Se usa la palabra clave **new** para solicitar la ejecución del constructor de la clase y que proceda a la instanciación de un objeto.

Como ya hemos visto, salvo en objetos especiales, **como los de la clase String, el objeto System.in o System.out**, es necesario utilizar el operador new para la construcción de objetos. Con la referencia del objeto flujo como argumento, se construye el objeto **filtro**, que en este caso se denomina **teclado:**

```
teclado = new BufferedReader (flujo);
```

Con el objeto teclado se lee la cadena de caracteres escrita por el usuario, a través de la consola del sistema, con la siguiente sentencia:

```
nombre = teclado.readLine ( );
```

Como ya hemos visto en otras ocasiones, es frecuente en Java declarar un objeto en la misma sentencia que se construye, así las sentencias de construir el flujo y el canal quedan de la siguiente manera:

```
InputStreamReaderflujo = new InputStreamReader (System.in);
BufferedReaderteclado = new BufferedReader (flujo);
```

Incluso es frecuente juntar las dos sentencias en una sola, creando el objeto de InputStreanReader en el propio argumento del constructor de BufferedReader, quedando las sentencias con el siguiente formato:

NOTA: Esta forma es la que hemos usado nosotros, en una sola línea.

```
BufferedReaderteclado = new BufferedReader (new InputStreamReader (System.in));
```

NOTA: Para leer números desde el teclado de esta manera podemos ver el siguiente ejemplo:

```
// SumarDosNumeros
// Sumando dos números tecleados por el usuario
import java.io.*;
public class SumarDosNumeros {
    public static void main(String[] args) {
        try {

            BufferedReaderentrada=newBufferedReader(newInputStreamReader(System.in));


            System.out.print ("¿Primer número a sumar? ");
            String num1 = entrada.readLine();
            System.out.print("¿Segundo número a sumar? ");
            String num2 = entrada.readLine();


            System.out.println ("Su sumaes: ");
            System.out.println(Double.parseDouble(num1) +Double.parseDouble(num2));
    }
    catch(Exception e) {
            System.err.println("No se pudo acceder al teclado");
    }
   }
}
```

NOTA: Double.parseDouble (num1) sirve para pasar la cadena num1 al tipo Double como objeto.

Si queremos convertir un String a número entero (sin decimales), lo haríamos con "Integer.parseInt(texto)".

2.2. Sistema de lectura de datos en la consola del sistema; clase Scanner

En la versión de Java anterior, se ha incluido la clase **Scanner** que permite leer datos en un flujo de entrada tal como el teclado del sistema, un fichero, o incluso un texto en una cadena de caracteres en memoria.

El uso de los métodos de esta clase, no generan la necesidad de capturar las eventuales excepciones que se puedan producir, y además su sintaxis es muy sencilla.

El siguiente ejemplo usa un objeto de la clase Scanner para la lectura de datos a través del teclado. El método nextLine ( ) lee una cadena de caracteres en la entrada del teclado.

```
import java.util.*;
public class DameTusDatos{
    public static void main(String args[]){

        Scanner sc = new Scanner(System.in);//Creamos el objeto de la clase Scanner
        System.out.print ("Introduce tu nombre: ");
        String nombre = sc.nextLine();//El objeto sc llama al método nextLine para leer la línea de la consola y se guarda en la variable tipo String llamada nombre
        System.out.print ("Introduce tu edad: ");
        String aux = sc.nextLine ( );
        int edad = Integer.parseInt (aux);
        System.out.print ("Introduce tu estatura: ");
        aux = sc.nextLine();
        double estatura = Double.parseDouble(aux);
        System.out.println ("Tus datos son:");
        System.out.printf ("Nombre:%s\nEdad:%d\nEstatura: %5.2f",nombre,edad,estatura);
    }
}
```

La sentencia:

```
int edad = Integer.parseInt (aux);
```

Declara la variable de tipo int y el método **parseInt** perteneciente a la clase **Integer** convierte el dato de la edad, escrito por el usuario en formato de texto, en un valor de tipo entero (int). Igualmente hay que convertir el dato de la estatura en un valor numérico de tipo double.

NOTA: La clase Scanner cuenta con métodos para leer datos de tipo numérico, si bien resulta más seguro y claro leer siempre datos en forma de cadena de caracteres con el método **nextLine ()** y después convertirlos al tipo que se necesite.

3. Lectura de datos a través de consola con sistema de validación

Son muchas las ocasiones en las que se requiere que el usuario introduzca un dato y este dato tenga que cumplir unas determinadas condiciones específicas, por ejemplo, no se admitiría que un usuario introdujera una edad negativa. En otras ocasiones se tratará de un dato formado por una cadena de caracteres el que tiene que ser validado, por ejemplo, el programa solicita el nombre del usuario, este debe estar formado al menos por tres caracteres para que pueda considerarse un nombre válido.

En general el proceso de validación de datos en la consola del sistema se realiza envolviendo a la petición de datos dentro de una estructura bucle*do-while,* de tal forma que, si no se cumplen las condiciones de la validación, el programa repite la validación hasta que las condiciones se cumplan, y sólo termina la petición cuando el usuario introduce un dato válido.

El ejemplo siguiente muestra este tipo de estructuras de validación rechazando el dato del correo electrónico si no tiene el carácter @ y tiene al menos tres caracteres (aunque no es la mejor forma sirve de ejemplo).

```
/* El programa DameTuCorreo muestra cómo realizar la validación de una cadena de caracteres en la consola del sistema (no es el mejor método pero es un ejemplo académico)*/

import java.io.*;

public class DameTuCorreo{

	public static void main(Stringargs[]) throws IOException {

		String correoE;

		BufferedReader teclado = new BufferedReader(new
		InputStreamReader(System.in));

		do{

			System.out.println ("Introduce tu email");

			correoE=teclado.readLine();

			//Veremos el método indexOf con los envoltorios

			if(correoE.length()<3||correoE.indexOf('@')<0){

			System.out.println("Dirección no válida");

			}

		}while(correoE.length() < 3 ||correoE.indexOf('@') < 0);

		System.out.println("Tu email es: "+correoE);

	}

}
```