Programación

Entrada – Salida básica

Introducción

- Java permite realizar operaciones de entrada / salida de consola a través de:
  - System.out: Salida estándar.
  - System.in: entrada estándar.
  - System.err: salida de estándar para mostrar errores.

Salida de datos

- En un principio, la salida de datos se realizará vía terminal.
- Para ello, utilizaremos los mecanismos que veremos a continuación.

print / println

- print( ) ➔ imprime el argumento recibido
- println ( ) ➔ Imprime un argumento y añade un carácter de salto de línea (\n)

```
public static void main(String[] args) {
    String nombre;
    String apellido;
    int edad;
    int salarioAnual;

    nombre = "Pepito";
    apellido = "Grillo";
    edad = 25;
    salarioAnual = 16000;

    System.out.println("#########################################");
    System.out.println("Datos del trabajador");
    System.out.print("Nombre: " + nombre);
    System.out.print (", Apellido: " + apellido);
    System.out.println(", Edad: " + edad );
    System.out.println("Salario Anual: " + salarioAnual );
    System.out.print("#########################################");
}
```

```
##########################################################
Datos del trabajador
Nombre: Pepito, Apellido: Grillo, Edad: 25
Salario Anual: 16000
##########################################################
```

printf

- Permite igualmente la impresión por consola pero, en este caso permite introducir valores de variables dentro de las propias comillas del texto, evitando concatenar.
- Simplemente hay que identificar , a través de un código de formato de que tipo será el valor en cuestión o si es un carácter especial en caso de que proceda. Posteriormente, y en orden de aparición se indica el nombre de la variable que contiene cada valor que ha aparecido separadas coma.

printf – Códigos de formato

%s formatea el argumento como String %d %o %x formatea el argumento como decimal, octal o hexadecimal respectivamente.

%n salto de línea (equivalente a \n)

%f formatea en coma flotante.

Se puede precisar cuantos decimales se quieren con. Por ejemplo, si se desean dos decimales pondríamos %.2f

%c formatea el argumento como carácter

%g formatea con notación científica

%% si queremos añadir el carácter %

```java
public static void main(String[] args) {
    String nombre;
    String apellido;
    int edad;
    double salarioAnual;


    nombre = "Pepito";
    apellido = "Grillo";
    edad = 25;
    salarioAnual = 16000.45;


    System.out.printf("############################################## %n");
    System.out.printf("Datos del trabajador %nNombre: %s, Apellido: %s, Edad: %d %nSalario Anual: %.2f %n",
            nombre, apellido, edad, salarioAnual);
    System.out.printf("##############################################");
}
```

```
##############################################
Datos del trabajador
Nombre: Pepito, Apellido: Grillo, Edad: 25
Salario Anual: 16000,45
##############################################
```

Entrada de datos

¿Cómo leemos datos por teclado?

- Utilizaremos la clase Scanner.
- Tendremos que declarar un objeto de la clase Scanner. Por ejemplo, a continuación declaramos un objeto de la clase Scanner al que llamaremos teclado:

  Scanner teclado = new Scanner (System.in)

- Una vez creado el objeto para leer valores de un determinado Tipo lo haremos de la siguiente manera:

  teclado.next*Mitipo*()

  - Donde *Mitipo* hace referencia al tipo en cuestión. Por ejemplo:
    - Si quisiéramos leer un dato de tipo double ➔ **teclado.nextDouble()**
    - Si quisiéramos leer un dato de tipo int ➔ **teclado.nextInt()**
    - Si quisiéramos leer un dato de tipo Long ➔ **teclado.nextLong()**

      etc..

- Para utilizar objetos de la clase Scanner será necesario importarla:

  import java.util.Scanner

Programación

- ¿Qué ocurre si intentamos recoger un dato de un tipo diferente al introducido por el usuario? ➔ Obtendremos un error.
- La forma de realizar programación “defensiva” anticipándonos a ese error es mediante el método **hasNext*Mitipo*( )** de la clase Scanner.
  - Donde *Mitipo* hace referencia al tipo en cuestión. Por ejemplo:
    - Si quisiéramos leer un dato de tipo double ➔ **teclado.hasNextDouble()**
    - Si quisiéramos leer un dato de tipo int ➔ **teclado.hasNextInt()**
    - Si quisiéramos leer un dato de tipo Long ➔ **teclado.hasNextLong()**

    etc..
  - Lo veremos sobre ejemplos cuando hagamos los correspondientes estructuras condicionales. Así, podremos preguntar si hay un valor del tipo deseado antes de recogerlo y en caso que no sea así podremos realizar la acción correspondiente.

Métodos básicos para la entrada de datos

- nextXxx( )
  - Devuelve el siguiente token como un tipo básico. Xxx será el tipo en cuestión. Por ejemplo, nextInt para leer un entero, nextDouble para leer un double, etc.
- next( )
  - Devuelve el siguiente token como un String.
- nextLine( )
  - Devuelve la línea entera como un String. Eliminará el carácter de fin de línea \n del buffer.
- hasNext( )
  - Devuelve un boolean. Indica si existe o no un siguiente token para leer.ç
- hasNextXxx ( )
  - Devuelve un boolean. Indica si existe o no un siguiente token del tipo especificado en Xxx. Por ejemplo, hasNextDouble( )

- Nota: Con token nos referimos al dato introducido por el usuario.

Recoger un carácter introducido por teclado

- Inicialmente, facilitaremos la recepción de un carácter introducido por el usuario de la siguiente manera:
  - next().charAt(0)

Limpiar el buffer

- Cuando el usuario introduce un valor cuyo tipo no es un String (no se recibe con nextLine()) y posteriormente se espera que se introduzca un texto es necesario limpiar el buffer, ya que si no se realiza dicha acción no se le proporcionará al usuario la oportunidad de introducir el valor de tipo String o cadena de caracteres porque automáticamente se cogerá como tal el carácter de fin de línea introducido con el dato anterior.
- Por ello es necesario limpiar el buffer, es decir se necesita un nextLine( ) previo al nextLine( ) necesario para que el usuario introduzca la cadena de texto en cuestión y poder recogerlo adecuadamente.

```java
import java.util.Scanner;

public class IoEj {

    public static void main(String[] args) {
        String nombre;
        String apellido;
        int edad;
        String puesto;
        double salarioAnual;
        Scanner teclado = new Scanner (System.in);

        System.out.println("############ LECTURA DE DATOS ############");
        System.out.println("Introduce nombre del trabajador");
        nombre = teclado.nextLine();
        System.out.println("Introduce apellido del trabajador");
        apellido = teclado.nextLine();
        System.out.println("Introduce edad del trabajador");
        edad = teclado.nextInt();
        System.out.println("Introduce puesto del trabajador");
        teclado.nextLine(); // limpiamos buffer
        puesto = teclado.nextLine();
        System.out.println("Introduce salario del trabajador");
        salarioAnual = teclado.nextDouble();
```

```
        System.out.println("############ LECTURA DE DATOS ############");
        System.out.println("Introduce nombre del trabajador");
        nombre = teclado.nextLine();
        System.out.println("Introduce apellido del trabajador");
        apellido = teclado.nextLine();
        System.out.println("Introduce edad del trabajador");
        edad = teclado.nextInt();
        System.out.println("Introduce puesto del trabajador");
        teclado.nextLine(); // limpiamos buffer
        puesto = teclado.nextLine();
        System.out.println("Introduce salario del trabajador");
        salarioAnual = teclado.nextDouble();

        System.out.println();
        System.out.println("############ IMPRESIÓN DE DATOS ############");
        System.out.println("Datos del trabajador");
        System.out.print("Nombre: " + nombre);
        System.out.print (", Apellido: " + apellido);
        System.out.println(", Edad: " + edad );
        System.out.println("Introduce puesto del trabajador");
        System.out.println("Puesto: " + puesto );
        System.out.println("Salario Anual: " + salarioAnual );
        System.out.print("##########################################");

        teclado.close();

    }
}
```

```
############ LECTURA DE DATOS ############
Introduce nombre del trabajador
Pepito
Introduce apellido del trabajador
Grillo
Introduce edad del trabajador
34
Introduce puesto del trabajador
Desarrollador
Introduce salario del trabajador
25890,65

############ IMPRESIÓN DE DATOS ############
Datos del trabajador
Nombre: Pepito, Apellido: Grillo, Edad: 34
Introduce puesto del trabajador
Puesto: Desarrollador
Salario Anual: 25890.65
##########################################
```