

1.10 Lectura por teclado

1. La clase `System`
2. La clase `Scanner`
3. Buffer de datos

1. La clase `System`

Las aplicaciones pueden necesitar acceder a los recursos del sistema, como por ejemplo, a los dispositivos de entrada/salida estándar para recoger datos desde el 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. Dicha clase contiene el atributo **`in`** para el teclado y el atributo **`out`** para la pantalla.

Dichos atributos son estáticos por lo que se accede a ellos con el nombre de la clase `System`:

`System.in` Entrada estándar: teclado

`System.out` Salida estándar: pantalla

2. La clase `Scanner`

El kit de desarrollo de Java, a partir de su versión 1.5, incorpora la clase `Scanner` del paquete `java.util` la cual posee métodos para leer valores de entrada que pueden venir de varias fuentes, como por ejemplo, de datos introducidos por teclado o datos almacenados en un archivo. Para leer datos desde el teclado, tenemos que pasarle el `System.in` al constructor del `Scanner`:

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

Veamos algunos métodos de esta clase:

- `nextByte`, `nextShort`, `nextInt` y `nextLong`: para leer datos de tipo entero.
- `nextFloat` y `nextDouble`: para leer números decimales.
- `next`: lee una palabra, es decir, hasta que encuentre un espacio.
- `nextLine`: lee una línea completa, es decir, hasta que encuentre un salto de línea (`\n`).

Por ejemplo, para leer un dato de tipo `int` sería:

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

O bien esta otra instrucción para leer una línea completa:

```
String string = keyboard.nextLine();
```

He aquí un ejemplo de entrada de teclado con la clase `Scanner`:

```
package tema1_10_LecturaPorTeclado;
```

```

import java.util.Scanner;

public class ScannerClass {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        String string;
        int i;
        float f;
        boolean b;

        string = keyboard.nextLine();
        System.out.println(string);
        i = keyboard.nextInt();
        System.out.println(i);
        b = keyboard.nextBoolean();
        System.out.println(b);
        /*
         * El símbolo separador de decimales será la coma si nuestro idioma
por defecto
         * del sistema operativo está configurado en español:
         */
        f = keyboard.nextFloat();
        System.out.println(f);

    }

}

```

El símbolo separador de decimales será la coma si nuestro idioma por defecto del sistema operativo está configurado en español. Si deseamos que el separador decimal sea el punto en lugar de la coma, entonces tendremos que añadir `useLocale(Locale.US)` cuando creamos el Scanner:

```

package temal_10_LecturaPorTeclado;

import java.util.Locale;
import java.util.Scanner;

public class ScannerClassLocale {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in).useLocale(Locale.US);
        float f;
        // Debido al useLocale(Locale.US), el símbolo separador de decimales
será el .
        f = keyboard.nextFloat();
        System.out.println(f);

    }

}

```

Veamos un ejemplo de lectura de palabras con `next()`:

```

package temal_10_LecturaPorTeclado;

import java.util.Scanner;

```

```

public class Next1 {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        String string1, string2, string3;

        System.out.println("Introduzca 3 palabras separadas por espacio: ");
        string1 = keyboard.next();
        string2 = keyboard.next();
        string3 = keyboard.next();
        System.out.println("Primera palabra: " + string1);
        System.out.println("Segunda palabra: " + string2);
        System.out.println("Tercera palabra: " + string3);

    }

}

```

3. Buffer de datos

Un buffer de datos es un espacio de memoria donde se almacenan datos de manera temporal mientras son transferidos o procesados. Los sistemas de entrada de teclado poseen un buffer que almacena las teclas presionadas.

Cuando introducimos un dato por teclado, para indicar que hemos finalizado la introducción pulsamos la tecla Enter. Cuando se ejecuta cualquier *next* con la clase Scanner(excepto *nextLine*), se coge el dato correspondiente del buffer pero el Enter se queda guardado como un *\n* (salto de línea) en el buffer. Si a continuación se ejecuta un *nextLine*, lee hasta el salto de línea del buffer, es decir, lee cadena vacía y ya no le da opción al usuario de introducir la cadena. Una posible solución sería limpiar el buffer con un *nextLine* y luego realizar otro *nextLine* para pedirle la cadena al usuario.

```

package tema1_10_LecturaPorTeclado;

import java.util.Scanner;

public class NextLine {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        String string;
        int number;
        boolean b;

        // El nextLine toma el salto de línea dejado por cualquier otro next
        System.out.println("Introduzca un número entero: ");
        number = keyboard.nextInt();
        System.out.println(number);
        System.out.println("Introduzca una cadena: ");
        string = keyboard.nextLine();
        System.out.println(string);
        System.out.println("Introduzca un boolean: ");
        b = keyboard.nextBoolean();
        System.out.println(b);
        System.out.println("Introduzca una cadena: ");
        string = keyboard.nextLine();
    }

}

```

```

        System.out.println(string);
        System.out.println("Introduzca una cadena con next: ");
        string = keyboard.next();
        System.out.println(string);
        System.out.println("Introduzca una cadena con nextLine: ");
        string = keyboard.nextLine();
        System.out.println(string);
        // Solución: poner un nextLine que coja el salto de línea. A esto se le
        conoce
        // como limpiar el buffer.
        System.out.println("Vamos a solucionar el problema.\nIntroduzca un
        número entero: ");
        number = keyboard.nextInt();
        System.out.println(number);
        System.out.println("Introduzca una cadena: ");
        keyboard.nextLine(); // Limpieza del buffer
        string = keyboard.nextLine();
        System.out.println(string);

    }

}

```

Otra solución sería indicarle al objeto *scanner* que como delimitador use el `\n` ya que por defecto es el carácter espacio. De esta forma, cuando utilicemos *next()*, leerá todos los caracteres hasta el siguiente `\n`. El único inconveniente de esto es que no podremos leer palabra a palabra.

```

package temal_10_LecturaPorTeclado;

import java.util.Scanner;

public class Next2 {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in).useDelimiter("\\n");// Para
        Linux y Mac
        // Scanner keyboard = new
        Scanner(System.in).useDelimiter("\\r\\n");//Para Windows
        String string;
        int number;

        System.out.println("Introduzca un número entero: ");
        number = keyboard.nextInt();
        System.out.println(number);
        System.out.println("Introduzca una cadena: ");
        string = keyboard.next();
        System.out.println(string);

    }

}

```