

3. Lectura de datos desde teclado

Hemos visto en los capítulos anteriores cómo mostrar información por pantalla y cómo usar variables. Veremos ahora algo muy importante que te permitirá realizar programas algo más funcionales, que tengan una utilidad real; aprenderás a leer la información que introduce un usuario mediante el teclado.

El funcionamiento de casi todos los programas se podría resumir en los siguientes puntos:

1. Entrada de datos desde teclado (o desde cualquier otro dispositivo de entrada)
2. Procesamiento de los datos de entrada para producir un resultado
3. Visualización de los resultados por pantalla

De momento, hemos visto cómo realizar los puntos 2 y 3. Vamos a ver a continuación cómo se recoge la información desde el teclado.

3.1 Lectura de texto

Para recoger datos por teclado usamos `System.console().readLine()`. Cuando llegamos a esta sentencia, el programa se detiene y espera que el usuario introduzca información mediante el teclado. La introducción de datos termina con la pulsación de la tecla INTRO. Una vez que el usuario presiona INTRO, todo lo que se ha tecleado se almacena en una variable, en el siguiente ejemplo esa variable es `nombre`.

```
/**
 * Lectura de datos desde teclado
 *
 * @author Luis J. Sanchez
 */
public class DimeTuNombre {
    public static void main(String[] args) {
        String nombre;
        System.out.print("Por favor, dime cómo te llamas: ");
        nombre = System.console().readLine();
        System.out.println("Hola " + nombre + ", ¡encantado de conocerte!");
    }
}
```



Mediante `System.console().readLine()` se recoge una línea de texto introducida por teclado.

Cuando se piden datos por teclado es importante que el programa especifique claramente cuál es exactamente la información que requiere por parte del usuario. Date cuenta que este programa muestra el mensaje “Por favor, dime cómo te llamas:”. Imagina que se omite este mensaje, en la pantalla aparecería únicamente un cursor parpadeando y el usuario no sabría qué tiene que hacer o qué dato introducir.

También es importante reseñar que los datos introducidos por teclado se recogen como una cadena de caracteres (un `String`).

3.2 Lectura de números

Si en lugar de texto necesitamos datos numéricos, deberemos convertir la cadena introducida en un número con el método adecuado. Como se muestra en el ejemplo, `Integer.parseInt()` convierte el texto introducido por teclado en un dato numérico, concretamente en un número entero.

```
/**
 * Lectura de datos desde teclado
 *
 * @author Luis J. Sánchez
 */
public class LeeNumeros {
    public static void main(String[] args) {

        String linea;

        System.out.print("Por favor, introduce un número: ");
        linea = System.console().readLine();
        int primerNumero;
        primerNumero = Integer.parseInt( linea );

        System.out.print("introduce otro, por favor: ");
        linea = System.console().readLine();
        int segundoNumero;
        segundoNumero = Integer.parseInt( linea );

        int total;
        total = (2 * primerNumero) + segundoNumero;

        System.out.print("El primer número introducido es " + primerNumero);
        System.out.println(" y el segundo es " + segundoNumero);
    }
}
```

```
        System.out.print("El doble del primer número más el segundo es ");
        System.out.print(total);
    }
}
```

Este último programa se podría acortar un poco. Por ejemplo, estas dos líneas

```
int total;
total = (2 * primerNumero) + segundoNumero;
```

se podrían quedar en una sola línea

```
int total = (2 * primerNumero) + segundoNumero;
```

De igual modo, estas tres líneas

```
linea = System.console().readLine();
int primerNumero;
primerNumero = Integer.parseInt( linea );
```

también se podrían reducir a una sola tal que así

```
int primerNumero = Integer.parseInt( System.console().readLine() );
```

Es muy importante que el código de nuestros programas sea limpio y legible. A veces, abreviando demasiado el código se hace más difícil de leer; es preferible tener unas líneas de más y que el código se entienda bien a tener un código muy compacto pero menos legible.

3.3 La clase Scanner

El método `System.console().readLine()` funciona bien en modo consola (en una ventana de terminal) pero puede provocar problemas cuando se trabaja con IDEs como **Eclipse**, **Netbeans**, **JavaEdit**, etc. Para evitar estos problemas puedes usar la clase `Scanner` cuando necesites recoger datos desde teclado. La clase `Scanner` funciona tanto en entornos integrados como en una ventana de terminal.

```
import java.util.Scanner;

/**
 * Lectura de datos desde teclado usando la clase Scanner
 *
 * @author Luis J. Sánchez
 */
public class LeeDatosScanner01 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.print("Introduce tu nombre: ");
        String nombre = s.nextLine();

        System.out.print("Introduce tu edad: ");
        int edad = Integer.parseInt(s.nextLine());

        System.out.println("Tu nombre es " + nombre + " y tu edad es " + edad);
    }
}
```

Fíjate que en el programa anterior la sentencia

```
s.nextLine()
```

sería el equivalente a

```
System.console().readLine()
```

Mediante el uso de la clase `Scanner` es posible leer varios datos en una misma línea. En el programa anterior se pedía un nombre y una edad, en total dos datos que había que introducir en líneas separadas. Observa cómo en el siguiente ejemplo se piden esos dos datos en una sola línea y separados por un espacio.

```
import java.util.Scanner;

/**
 * Lectura de datos desde teclado usando la clase Scanner
 *
 * @author Luis J. Sánchez
 */
public class LeeDatosScanner02 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
```

```

    System.out.print("Introduce tu nombre y tu edad separados por un espacio: ");
    String nombre = s.next();
    int edad = s.nextInt();

    System.out.println("Tu nombre es " + nombre + " y tu edad es " + edad);
}
}

```

Fíjate cómo se ha utilizado `s.next()` para leer una cadena de caracteres y `s.nextInt()` para leer un número entero, todo ello en la misma línea.

El siguiente programa de ejemplo calcula la media de tres números decimales. Para leer cada uno de los números en la misma línea se utiliza `s.nextDouble()`.

```

import java.util.Scanner;

/**
 * Lectura de datos desde teclado usando la clase Scanner
 *
 * @author Luis J. Sánchez
 */
public class LeeDatosScannerMedia {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.print("Introduce tres números (pueden contener decimales) separados por espacio\
s: ");
        double x1 = s.nextDouble();
        double x2 = s.nextDouble();
        double x3 = s.nextDouble();

        double media = (x1 + x2 + x3) / 3;

        System.out.println("La media de esos tres números es " + media);
    }
}

```



Recuerda que **¡a programar se aprende programando!** Es muy importante que pruebes los ejemplos - si los modificas y experimentas, todavía mejor - y que realices los ejercicios. Para aprender a programar no basta con leer el libro y entenderlo, es una condición necesaria pero no suficiente. Igual que para ser piloto hacen falta muchas horas de vuelo, para ser programador hacen falta muchas horas picando código, probando, corrigiendo errores... ¡Ánimo! ¡El esfuerzo merece la pena!