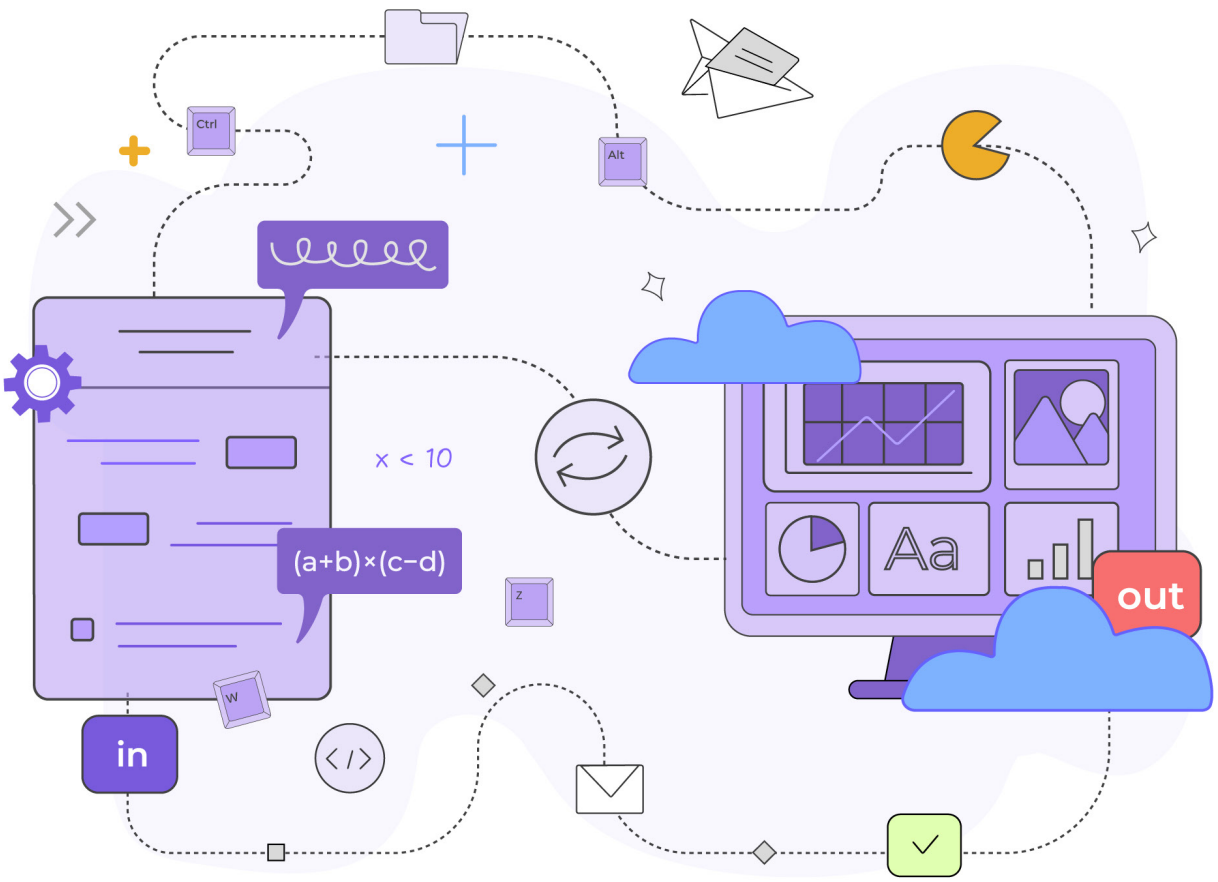
Capítulo 3

Entrada y salida de datos



Capítulo 3

Entrada y salida de datos

Objetivo

El capítulo tiene como objetivo desarrollar la interacción entre un programa y el usuario. Se explicarán las herramientas que el lenguaje de programación nos ofrece para leer los datos del usuario, procesarlos y finalmente mostrar resultados de forma clara y atractiva.

Al finalizar este capítulo, podrás crear programas que interactúen con el usuario de forma efectiva. Habrás adquirido una comprensión profunda de las diferentes técnicas de entrada y salida de datos que provee el lenguaje, así como la importancia de la interacción con el usuario. Además, estarás en condiciones de diseñar preliminarmente interfaces intuitivas que brinden una experiencia satisfactoria.

Variables, constantes y expresiones

Como mencionamos anteriormente, una variable es un espacio en la memoria que un programa utiliza para almacenar un valor y que además puede cambiar durante su ejecución. Es un contenedor de datos al que se le puede asignar un valor y modificar durante la ejecución del programa.

La variable debe ser identificada con un nombre único dentro del bloque que la alcance. Esta última parte, que mencionamos sobre el alcance, lo veremos en profundidad más adelante. Por ahora pensemos el alcance como algo que cubre todo el programa.

A continuación declaramos una variable de tipo numérica, o sea, una variable que nos permitirá almacenar valores enteros. También aprovechamos la declaración para asignarle un valor inicial:

```
//
int enteros = 10;
```

Esta asignación inicial es opcional, puede omitirse.

[-] constante

Una constante es un espacio de memoria que contiene un valor, al igual que las variables. Este valor, una vez que ha sido establecido, no puede cambiar durante toda la ejecución del programa. El lugar donde se establece este valor es en su declaración.

Las constantes son útiles para:

- Mejorar la legibilidad del código: usar nombres descriptivos para valores que no cambian hace que el código sea más fácil de entender.
- Evitar errores: si un valor no cambia, no hay riesgo de que se modifique accidentalmente y cause un error.
- Compartir valores: las constantes se pueden usar para compartir valores entre diferentes partes del programa.

La forma de declarar una constante en Java es mediante el uso de la palabra reservada "**final**". Indica que el espacio de la memoria que contiene el valor no se puede modificar durante toda la ejecución del programa.

Por convención suelen utilizarse identificadores descriptivos y en mayúsculas.

Veamos cómo se declara una constante:

```
//una constante
final int DIAS_LABORALES = 5;

//una constante
final String MENSAJE_BIENVENIDA = "Buenos días...";
```

A diferencia de la variable, la posibilidad de asignar un valor en la declaración deja de ser una opción, es obligatorio.

[-] expresión

Una expresión es una combinación de valores, variables, operadores y funciones que se evalúan para obtener un resultado único. Es como una fórmula matemática que se puede calcular.

Las expresiones son útiles para:

- Realizar cálculos: se pueden usar para sumar, restar, multiplicar, dividir y realizar otras operaciones matemáticas y lógicas.
- Comparar valores: se pueden usar para comparar si dos valores son iguales, diferentes, mayores o menores.
- Obtener información: se pueden usar para obtener información de variables o funciones.

La evaluación de expresiones en Java se realiza siguiendo un orden de precedencia y asociatividad específica. Este orden determina cómo se interpretan los operadores y en qué orden se realizan las operaciones.

El orden de precedencia define qué operadores se evalúan primero. Los operadores con mayor precedencia se evalúan antes que los de menor precedencia.

El siguiente es el orden de precedencia de los operadores en Java, de mayor a menor:

- Operadores unarios: +, -, ~, !
- Operadores de multiplicación y división: *, /, %
- Operadores de suma y resta: +, -
- Operadores relacionales: <, <=, >, >=, ==, !=
- Operadores lógicos: &&, ||
- Operador de asignación: =

La asociatividad define en qué orden se evalúan los operadores que tienen la misma precedencia. En Java, la mayoría de los operadores son asociativos de izquierda a derecha, lo que significa que se evalúan de izquierda a derecha. Los operadores de asignación son asociativos de derecha a izquierda.

```
3 + 5 * 4   ← retorna 3 más la multiplicación de 5 por 4

x = y = 5  ← asigna 5 a la variable “y” y luego a x
```