1.3 Variables y constantes

1. Introducción
2. Declaración de variables
3. Inicialización de variables
4. Ámbito de vida de las variables
5. Constantes

1. Introducción

Las variables son contenedores que sirven para almacenar los datos que utiliza un programa. Dicho más sencillamente, son nombres que asociamos a determinados datos. La realidad es que cada variable ocupa un espacio en la memoria RAM del ordenador para almacenar el dato al que se refiere. Es decir, cuando utilizamos el nombre de la variable realmente estamos haciendo referencia a un dato que está en memoria.

Las variables tienen un nombre (un **identificador**) que se escribe en minúscula, y si consta de varias palabras, se utiliza la notación lowerCamelCase. Ejemplo: myFirstVariable. Además, deben cumplir lo siguiente:

- No deben comenzar con los caracteres guion bajo (_) o el signo de dólar ($), aunque ambos se admiten.
- Se admiten los números pero no como primer carácter.
- Deben ser cortos pero significativos. La elección de un nombre de variable debe ser mnemónico, es decir, diseñado para indicar al observador casual la intención de su uso. Por ejemplo, si queremos usar una variable para almacenar una edad, la llamaremos *edad*.
- Se deben evitar los nombres de variables de un solo carácter excepto para las variables temporales "usar y tirar". Los nombres comunes de las variables temporales son i, j, k, m, y n para enteros; c, d, y e para los caracteres.

2. Declaración de variables

Antes de poder utilizar una variable, esta se debe declarar de la siguiente manera: `tipo nombrevariable;` Donde *tipo* es el tipo de datos que almacenará la variable (texto, números enteros,...) y *nombrevariable* es el identificador de la variable. Ejemplos:

```java
int days; // days es un número entero, sin decimales
boolean exit; //exit sólo puede ser verdadera o falsa
```

Java es un lenguaje muy estricto al utilizar tipos de datos. Variables de datos distintos son incompatibles. Algunos autores hablan de lenguaje **fuertemente tipado** o incluso lenguaje muy tipificado. Se debe a una traducción muy directa del inglés strongly typed referida a los lenguajes que, como Java, son muy rígidos en el uso de tipos. El caso contrario sería el lenguaje C en el que jamás se comprueban de manera estricta los tipos de datos. Parte de le seguridad y robustez de las que hace gala Java se deben a esta característica.

Por convención de código, todas las declaraciones de variables se ponen al principio.

3. Inicialización de variables

En Java se utiliza el operador asignación **=** para inicializar una variable, es decir, para darle un valor inicial.

La inicialización se puede realizar:

- En la misma línea de código que la declaración:

```
int x=7;
```

- En cualquier otro momento, pero siempre después de haberla declarado:

```
int x;
...
x=7;
```

También se puede utilizar una expresión para asignar un valor a una variable:

```
int x;
...
x=7;
...
x=x*2;//Utilización de una expresión en la asignación de la variable
```

Incluso se puede utilizar una expresión en la misma inicialización:

```
int a=13, b=18;
int c=a+b; //es válido, c vale 31
```

Se puede declarar más de una variable a la vez del mismo tipo en la misma línea si las separamos con comas:

```
int days, year, weeks;
```

Incluso se pueden también inicializar:

```
int days=365, year=2019, weeks;
```

4. Ámbito de vida de las variables

Toda variable tiene un ámbito de vida. Esto es la parte del código en la que una variable se puede utilizar, que es en el bloque donde se ha declarado. De hecho las variables tienen un ciclo de vida:

1. En la declaración se reserva el espacio necesario para que se puedan comenzar a utilizar (digamos que se avisa de su futura existencia)
2. Se la asigna su primer valor (la variable nace)
3. Se la utiliza en diversas sentencias. Cuando finaliza el bloque en el que fue declarada, la variable muere. Es decir, se libera el espacio que ocupa esa variable en memoria.
4. Una vez que la variable ha sido eliminada, no se puede utilizar. Dicho de otro modo, no se puede utilizar una variable más allá del bloque en el que ha sido definida. Ejemplo:

```
{//Se utiliza { para indicar el comienzo del bloque de código
    int x=9;
}//Se utiliza } para indicar el fin del bloque de código
int y=x; //error, ya no existe x
```

5. Constantes

Una constante es un valor que no puede ser modificado durante la ejecución de un programa, únicamente puede ser leído.

La forma de declarar constantes es la misma que la de las variables pero hay que anteponer la palabra **final** que es la que indica que estamos declarando una constante:

```
final double PI=3.141591;
PI=4; //Error, no podemos cambiar el valor de PI
```

Los nombres de las constantes se deben escribir en mayúsculas. Pueden contener también guiones bajos. Incluso pueden contener dígitos pero no como primer carácter.

Ejemplos:

```
final int MAX_PARTICIPANTS = 10;
final int _MIN1 = 1;
```

Cuando un mismo valor se utilice en varias partes del código, entonces hay que declararlo como una constante ya que si en algún momento de la vida de la aplicación, ese valor varía, solamente hay que cambiar el valor de la constante y no estar cambiándolo en todos los sitios del código donde aparezca.