1. Variables e identificadores

El compilador necesita tener una tabla de símbolos o palabras clave (*keywords*), también llamadas palabras reservadas, que conozca y pueda identificar para comprobar que nuestro código está bien escrito y es correcto. Vamos a conocer cuáles son, sin olvidar que nosotros como programadores ampliaremos ese vocabulario creando palabras reservadas cuando lo necesitemos.

1.1. Variables

Según la definición que hemos dado, las variables vienen determinadas por:

- Un **nombre** o **identificador**, que permite al programa acceder al valor que contiene en la memoria. Debe ser un identificador válido. Esto lo explicaremos en el siguiente punto.
- Un **tipo de dato**, que especifica qué clase de información guarda la variable en esa zona de memoria. También vamos a indicaros los diferentes tipos existentes en Java.
- Un **rango** de valores que puede admitir dicha variable. El tipo de dato limita estos valores.

Las **variables pueden clasificarse** de diferentes formas atendiendo a sus características:

Según el tipo de información contenida, podrá tomar unos valores u otros, y se podrán definir sobre ella unas operaciones u otras.	**De tipos primitivos.** Tanto el valor como la variable están en la misma posición de la memoria.
	De referencia. Guarda una posición o dirección de memoria adonde lleva el contenido que se va a almacenar.
Dependiendo de si su valor puede cambiar o no durante la ejecución del programa.	**Variables dinámicas.** Sirven para almacenar los datos durante la ejecución del programa. Pueden estar formadas por cualquier tipo de dato primitivo o de referencia. Su valor puede cambiar varias veces a lo largo de todo el programa.
	Constantes o variables finales. Su valor no puede cambiar a lo largo de todo el programa. Así, por ejemplo, el valor de PI o de los valores MIN_VALUE y MAX_VALUE para el valor mínimo y máximo del tipo de dato que utilizamos en cada momento. Esto lo veremos en la tabla de los tipos de datos.
En función del lugar donde aparezcan en el programa, pueden ser de tipos primitivos o referencias, variables o constantes.	**Variables miembro**. Se crean dentro de una clase, fuera de cualquier método. En un lenguaje orientado a objetos —como es Java—, todo se basa en la utilización de objetos, los cuales se crean usando clases.
	Variables locales. Se crean y usan dentro de un método o, en general, dentro de cualquier bloque de código. La variable deja de existir cuando la ejecución del bloque de código o el método finaliza.

Tabla 2.1. Formas de clasificar las variables

1.2. Identificadores

Se llama **identificador** al nombre que damos a la variable. Cuanta mayor relación tenga ese nombre con lo que va a representar, mejor. Por ejemplo, un identificador adecuado podría ser **apellido1**. Rápidamente nos indica qué es y no necesitamos más información para saber de qué estamos hablando. Por tanto, los identificadores:

- Sirven para nombrar constantes, variables/atributos, métodos/funciones, clases e interfaces, objetos o paquetes. Iremos viendo qué es cada cosa en las próximas unidades. Debe ser elegido para indicar al programador la intención de su uso. No hay una longitud máxima establecida para el identificador. Si las variables o métodos tienen varias palabras para hacerlas más entendibles, se puede utilizar una de las siguientes 3 notaciones: **`UpperCamelCase`** o **`CamelCase`** (para nombres de Clases), **`lowerCamelCase`** o **`CamelCase`** (para variables/atributos y métodos/funciones) y **`SCREAMING_SNAKE_CASE`** o **`SNAKE_CASE`** (para constantes). Existe otra denominada **`kebab-case`** que no se puede utilizar en Java. Cuanto más entendible por sí solo, mejor identificador será. Por ejemplo, **mayoriaDeEdadLegal**, si no es una constante. No puede contener espacios en blanco, debe ser un único token. Tampoco se recomienda poner acentos, diéresis ni ningún tipo de símbolo de acentuación. Pero en las constantes podemos separar las palabras con guiones bajos; por ejemplo: **MAYORIA_DE_EDAD_LEGAL**.
- Son **sensibles a las mayúsculas** (en inglés, *case sensitive*), lo que significa que se distinguen las mayúsculas de las minúsculas. No es lo mismo HolaMundo que holamundo u HOLAMUNDO. Serían tres identificadores distintos para el compilador.
- Deben comenzar con una letra. Dependiendo de lo que identifique, esta irá en mayúscula (constantes, clases e interfaces) o en minúscula (variables, métodos, objetos, paquetes). Los siguientes caracteres pueden ser letras o dígitos. No debería comenzar con un guion bajo ('_') o por caracteres especiales, como, por ejemplo, un signo de dólar '$'.
- Aunque parezca obvio, tampoco podemos usar identificadores repetidos. Aunque sí que podremos tener varios métodos sobrecargados con el mismo nombre, gracias al polimorfismo, como veremos en unidades posteriores.

IMPORTANTE

Existe una guía de estilo de Java (versión original en inglés de 1997) donde se nos indican unas recomendaciones, que os iremos explicando, de cuándo interesa hacer uso de unas u otras.

1.3. Palabras clave

En el lenguaje de programación Java se puede hacer uso de las palabras clave (*keywords*), también llamadas palabras reservadas, que escribiremos siempre en minúsculas. Dichas palabras no pueden ser utilizadas como identificadores por los programadores para definir variables, constantes, etcétera. Ya tienen una función asignada que iremos conociendo en las próximas unidades. En la siguiente tabla, puedes ver las 50 keywords que existen ordenadas alfabéticamente por columnas:

abstract	continue	for	new	switch
assert	default	~~goto~~	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
~~const~~	float	native	super	while

Tabla 2.2 Listado ordenado alfabéticamente por columnas de las palabras clave en Java

IMPORTANTE

Debes saber que, de las 50 palabras clave iniciales, las marcadas en color verde ya las hemos utilizado y explicado. Las resaltadas en amarillo son los tipos de datos primitivos que explicaremos en el siguiente apartado. Y las marcadas en rojo, const y goto, ya están obsoletas, luego se han dejado de utilizar. A esta lista deberíamos añadir algunas nuevas palabras clave: true, false y null, que sirve para inicializar los objetos a nulo, como veremos en próximas unidades. También se reservaron estas otras palabras reservadas: cast, uture, generic, inner, operator, outer, rest y var. Como hoy en día aún no tienen asignada ninguna funcionalidad, hemos preferido no añadirlas a la tabla 2.1.

En las próximas unidades iremos aprendiendo para qué sirve cada una de estas palabras clave. En esta unidad aprenderás ocho más que se utilizan para los tipos de datos primitivos: boolean, byte, char, double float, int, long y short, sin olvidar enum para los tipos de datos enumerados que vamos a ver a continuación.