**Escribe tu primer Hola Mundo en Java**

Los archivos de Java usan la extensión **.java**. Por lo tanto, para crear nuestro primer “Hola, mundo” podemos hacerlo desde un archivo HolaMundo.java.

El método **main** es el punto de entrada de una aplicación en diferentes lenguajes como Java, [Kotlin](https://platzi.com/clases/kotlin/" \t "_blank) y [C++](https://platzi.com/clases/c-plus-plus/). Sin este método nuestra aplicación no se ejecutará y mostrará un error.

En Java definimos este método de la siguiente manera:

**public** **static** **void** **main** (String[] args) {

// acciones

}

Por lo tanto, este será el código de nuestro HolaMundo.java y podremos ejecutarlo con Click derecho > Run 'HolaMundo.main()':

**public** **class** **HolaMundo** {

**public** **static** **void** **main** (String[] args) {

System.out.println("Hola, mundo!");

}

}

Recuerda que nuestro IDE nos proporciona algunos atajos. Por ejemplo, con solo escribir la palabra sout podremos autocompletar la sentencia System.out.println();.

¡No olvides dejar todas tus dudas y comentarios en la sección de discusiones! Así podemos ayudarte a resolver todos tus inconvenientes.

# Variables en Java

Una variable es un espacio de memoria (RAM) que contiene un dato de tipo numérico, booleano, de texto u otros tipos de datos un poco más complejos.

Las variables en Java se componen de un nombre único y un valor que puede cambiar a lo largo de la ejecución del programa. Al declarar las variables debemos definir el tipo de dato que vamos a usar y un punto y coma al final:

// Variables.java

**public** **class** **Variables** {

**public** **static** **void** **main**(String[] args) {

// Declarar la variable speed de tipo int (números enteros)

**int** spped;

// Actualizar el contenido de la variable speed:

spped = 10; // si ya habías declarado la variable

// Declarar una variable y asignarle un valor al mismo tiempo:

**int** salary = 1000;

// Crear una variable de tipo String:

String eployeeName = "Anahí Salgado";

}

}

# Actualizando variables

Java nos permite actualizar nuestras variables reutilizando los valores que tenían anteriormente, de esta forma evitamos errores o inconsistencias en nuestro código:

**public** **class** **UpdatingVariable** {

**public** **static** **void** **main**(String[] args) {

// Actualizar datos numéricos:

**int** salary = 1000;

salary = salary + 200;

System.out.println(salary); // 1200

salary += 300;

System.out.println(salary); // 1500

// Actualizar variables de tipo String:

String employeeName = "Anahí";

employeeName = employeeName + "Salgado";

System.out.println(salary); // Anahí Salgado

employeeName += "Díaz de la Vega";

System.out.println(employeeName); // Anahí Salgado Díaz de la Vega

employeeName = "Irene" + employeeName;

System.out.println(employeeName); // Irene Anahí Salgado Díaz de la Vega

}

}

# Convención de Nombres en Java

Una convención de nombres es un patrón que deben seguir los nombres de las variables para que el código esté organizado, entendible y sin repetidos.

* Java es sensible a mayúsculas y minúsculas, este punto es clave al seguir una convención.
* Las variables siempre deben comenzar con un simbolo de letra, $ o \_.
* No puedes usar el simbolo - en ninguna parte de la variable.

Las variables constantes son variables cuyo valor nunca va a cambiar, por lo que se deben escribir completamente en mayúsculas y usando el caracter \_.

# Técnica de Naming: Camel Case

Camel Case es una convención muy popular para nombrar nuestras variables. Podemos usarlo en modo Upper Camel Case o Lower Camel Case, la diferencia es si comenzamos el nombre de la variable con mayúscula o minúscula.

// Upper Camel Case:

**class** **SoyUnaClase** {};

// Lower Camel Case

**int** soyUnNumeroInt = 10;

Debemos usar Upper Camel Case en los nombres de las clases y archivos. Y Lower Camel Case en los nombres de las variables o métodos.

# Tipos de datos numéricos

Tipos de datos para números enteros (sin decimales):

* **byte**: Ocupa 1 byte de memoria y su rango es de -128 hasta 127.
* **short**: Ocupa 2 bytes de memoria y su rango es de -32,768 hasta 32,727.
* **int**: Ocupa 4 bytes de memoria y su rango es de -2,147,483,648 hasta 2,147,483,647. Es muy cómodo de usar, ya que no es tan pequeño para que no quepan nuestros números ni tan grande como para desperdiciar mucha memoria. Puede almacenar hasta 10 dígitos.
* **long**: Ocupa 8 bytes de memoria y su rango es de -9,223,372,036,854,775,808 hasta 9,223,372,036,854,775,807. Para diferenciarlo de un tipo de dato long debemos terminar el número con la letra L.

Por ejemplo:

// Int:

**int** n = 1234567890;

// Long:

**long** nL = 123456789012345L;

Tipos de datos para números flotantes (con decimales):

* **float**: Ocupan 4 bytes de memoria y su rango es de 1.40129846432481707e-45 hasta 3.40282346638528860e+38. Así como long, debemos colocar una letra F al final.
* **double**: Ocupan 8 bytes de memoria y su rango es de 4.94065645841246544e-324d hasta 1.79769313486231570e+308d.

Por ejemplo:

// Double:

**double** nD = 123.456123456;

// Float

**float** nF = 123.456F;

# Tipos de datos char y boolean

* **char**: Ocupa 2 bytes y solo puede almacenar 1 dígito, debemos usar comillas simples en vez de comillas dobles.
* **boolean**: Son un tipo de dato lógico, solo aceptan los valores true y false. También ocupa 2 bytes y almacena únicamente 1 dígito.

Seguro te diste cuenta que siempre debemos escribir el tipo de dato de nuestras variables antes de definir su nombre y valor. Pero esto cambia a partir de Java 10: solo debemos escribir la palabra reservada var y Java definirá el tipo de dato de nuestras variables automáticamente:

var salary = 1000; // INT

var pension = salary \* 0.03; // DOUBLE

var totalSalary = salary - pension; // DOUBLE

Recuerda que esto solo funciona con versiones superiores a Java 10.