CURSO
BACKEND 1

Introducción a Java

Introducción a Java

OBJETIVOS DE LA GUÍA

En esta guía aprenderemos a:

- Utilizar el nuevo IDE.
- Comprender la estructura básica de Java
- Comprender los tipos de datos en Java
- Definir y operar variables.
- Utilizar los métodos de escritura y salida
- Utilizar las clases de utilidad

egg | Argentina programa 4.0

Introducción a Java

Hasta el momento hemos aprendido los diferentes tipos de estructuras de control comunes a todos los lenguajes de programación, dentro del paradigma de programación imperativa, haciendo uso del pseudo intérprete PSeInt. A partir de esta guía comenzaremos a introducir cada uno de los conceptos vistos hasta el momento, pero haciendo uso de un lenguaje de programación de propósito general como lo es Java.

Java

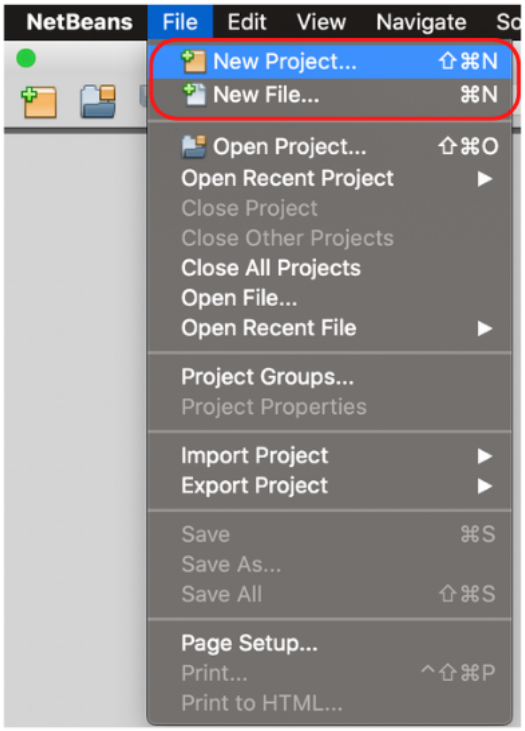
Java es un tipo de lenguaje de programación y una plataforma informática, creada y comercializada por Sun Microsystems en el año 1995 y desde entonces se ha vuelto muy popular, gracias a su fácil portabilidad a todos los sistemas operativos existentes.

Java es un lenguaje de programación de alto nivel, estos permiten escribir código mediante idiomas que conocemos (inglés, español, etc.) y luego, para ser ejecutados, se traduce al lenguaje de máquina mediante traductores o compiladores. Java es un lenguaje de alto nivel donde sus palabras reservadas están en **inglés.**

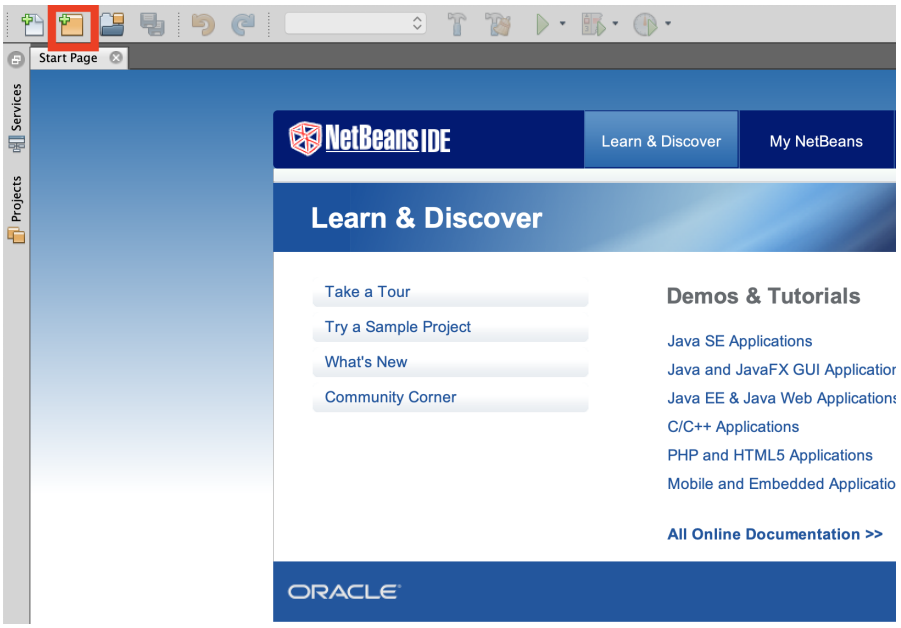
Estructura de un programa Java

Vamos a crear un programa desde cero. Estos pasos los repetirás cada vez que realices un nuevo ejercicio, de esta manera tendrás todos tus proyectos ordenados y podrás recurrir a ellos fácilmente cuando lo necesites.

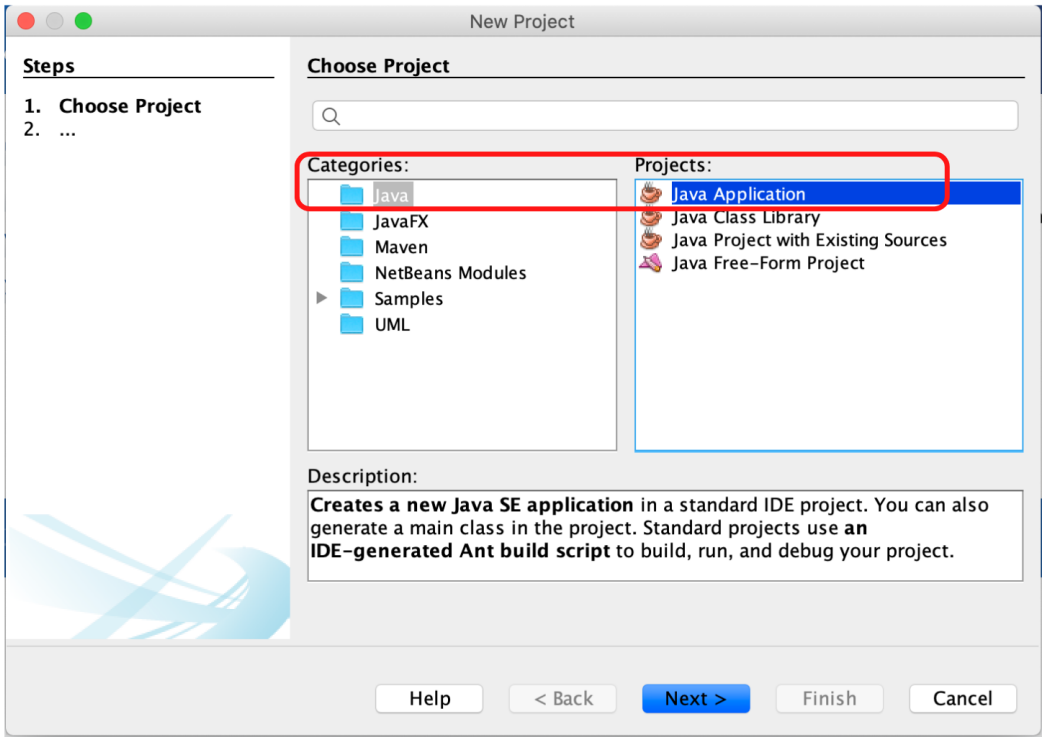
Primero deberemos ir a File → New Project



O podemos hacer click en la siguiente opción:

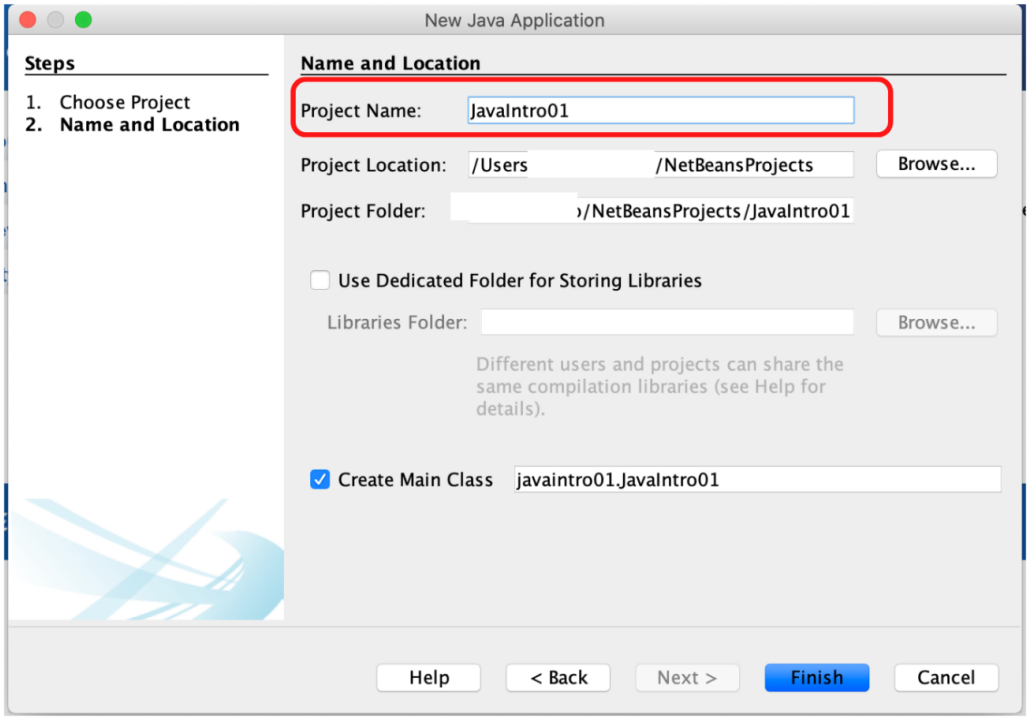


Ahora deberemos elegir el tipo de proyecto que queremos crear, vamos a elegir un proyecto de Java

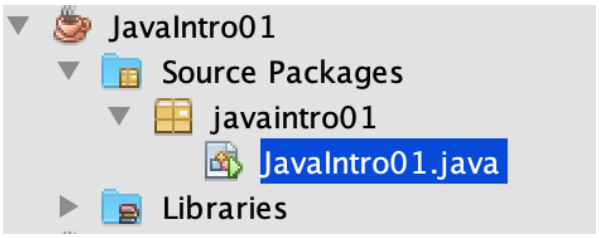


Por último deberemos ponerle un nombre a nuestro proyecto

Aquí colocamos el nombre del proyecto de la siguiente manera:



Nos quedaría así:



¿Cómo se ve un programa en java?

Un programa describe como un ordenador debe interpretar las órdenes del programador para que ejecute y realice las instrucciones dadas tal como están escritas. Un programador utiliza los elementos que ofrece un lenguaje de programación para diseñar programas que resuelvan problemas concretos o realicen acciones bien definidas.

El siguiente programa Java muestra un mensaje en la consola con el texto “Hola Mundo”.

```
/*
Este programa escribe el texto "Hola Mundo" en la consola,
utilizando el método System.out.println
 */

package primerprograma;

public class HolaMundo {

      public static void main(String[] args) {

      //La línea 12 muestra el mensaje por pantalla al ejecutar el
programa

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

    }
}
```

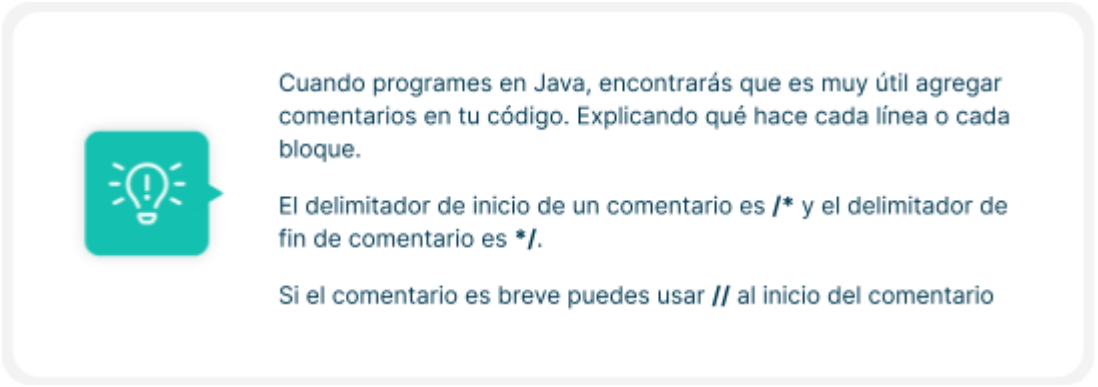
En este programa se pueden identificar los siguientes elementos del lenguaje Java: comentarios, paquete, definiciones de clase, definiciones de método y sentencias. Veamos qué es y qué hace cada uno:

¿Cuáles son los comentarios?

El texto del primer comentario de este ejemplo sería: *'Este programa escribe el texto "Hola Mundo" en la consola utilizando el método System.out.println()'.*

El segundo comentario es *// La línea 12 muestra el mensaje por pantalla al ejecutar el programa*

Los comentarios son ignorados por el compilador y solo son útiles para el programador. Los comentarios ayudan a explicar aspectos relevantes de un programa y lo hacen más legible. En un comentario se puede escribir todo lo que se desee, el texto puede ser de una o más líneas.



¿Cuál es el paquete?

Después del comentario viene escrito el nombre del paquete. Los paquetes son contenedores de clases y su función es la de organizar la distribución de las clases. Los paquetes y las clases son análogos a las carpetas y archivos utilizados por el sistema operativo, respectivamente. Esto quiere decir, cada paquete es una carpeta que contiene archivos, que son las clases.

El lenguaje de programación de la tecnología Java le provee la sentencia package como la forma de agrupar clases relacionadas. La sentencia package tiene la siguiente forma:

```
package <nombre_paq_sup>[.<nombre_sub_paq>]*;
```

La declaración package, en caso de existir, debe estar al principio del archivo fuente y sólo la declaración de un paquete está permitida. Los nombres de los paquetes los pondrá el programador al crear el programa y son jerárquicos (al igual que una organización de directorios en disco) además, están separados por puntos. Es usual que sean escritos completamente en minúscula.

¿Cuál es la clase?

La primera línea del programa, después del package. Define una clase que se llama HolaMundo. En el mundo de orientación a objetos, todos los programas se definen en términos de objetos y sus relaciones. Las clases sirven para modelar los objetos que serán utilizados por nuestros programas. Los objetos, las clases y los paquetes **son conceptos que serán abordados con profundidad más adelante en el curso.**

Una clase está formada por una parte correspondiente a la declaración de la clase, y otra correspondiente al cuerpo de la misma:

```
Declaración de clase {

Cuerpo de clase

}
```

En la plantilla de ejemplo se ha simplificado el aspecto de la Declaración de clase, pero sí que puede asegurarse que la misma contendrá, como mínimo, la palabra reservada class y el nombre que recibe la clase. La definición de la clase o cuerpo de las comienza con una llave abierta (**{**) y termina con una llave cerrada (**}**). El nombre de la clase lo define el programador.

¿Cuál es el Método?

Después de la definición de clase se escribe la definición del método main(). Pero que es un método?. Dentro del cuerpo de la clase se declaran los atributos y los métodos de la clase. Un método es una secuencia de sentencias ejecutables. Las sentencias de un método quedan delimitadas por los caracteres { y } que indican el inicio y el fin del método, respectivamente. Si bien es un tema sobre el que se profundizará más adelante en el curso, los métodos son de vital importancia para los objetos y las clases. En un principio, para dar los primeros pasos en Java nos alcanza con esta definición.

Método main()

Ahora sabemos lo que es un método, pero en el ejemplo podemos ver el método main(). El main() sirve para que un programa se pueda ejecutar, este método, vendría a representar el Algoritmo / FinAlgoritmo de PseInt y tiene la siguiente declaración:

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

A continuación, describiremos cada uno de los modificadores y componentes que se utilizan *siempre* en la declaración del método *main():*

public: es un tipo de acceso que indica que el método *main()* es público y, por tanto, puede ser llamado desde otras clases. Todo método *main()* debe ser público para poder ejecutarse desde el intérprete Java (JVM).

static: es un modificador el cual indica que la clase no necesita ser instanciada para poder utilizar el método. También indica que el método es el mismo para todas las instancias que pudieran crearse.

void: indica que la función o método *main()* no devuelve ningún valor.

El método *main()* debe aceptar siempre, como parámetro, un vector de strings, que contendrá los posibles argumentos que se le pasen al programa en la línea de comandos, aunque como es nuestro caso, no se utilice.

Luego, al indicarle a la máquina virtual que ejecute una aplicación el primer método que ejecutará es el método *main()*. Si indicamos a la máquina virtual que corra una clase que no contiene este método, se lanzará un mensaje advirtiendo que la clase que se quiere ejecutar no contiene un método *main()*, es decir que dicha clase no es ejecutable.

Si no se han comprendido hasta el momento muy bien todos estos conceptos, los mismos se irán comprendiendo a lo largo del curso.

Sentencia

Son las unidades ejecutables más pequeñas de un programa, en otras palabras, una línea de código escrita es una sentencia. Especifican y controlan el flujo y orden de ejecución del programa. Una sentencia consta de palabras clave o reservadas como expresiones, declaraciones de variables, o llamadas a funciones.

En nuestro ejemplo, del método main() se incluye una sentencia para mostrar un texto por la consola. Los textos siempre se escriben entre comillas dobles para diferenciarlos de otros elementos del lenguaje. **Todas las sentencias de un programa Java deben terminar con el símbolo punto y coma**. Este símbolo indica al compilador que ha finalizado una sentencia.

Una vez que el programa se ha editado, es necesario compilarlo y ejecutarlo para comprobar si es correcto. Al finalizar el proceso de compilación, el compilador indica si hay errores en el código Java, donde se encuentran y el tipo de error que ha detectado: léxico, sintáctico o semántico.

Puede que tantos conceptos nuevos, instalar e utilizar un nuevo IDE nos haga sentir un poco confundidos. Habla con tu equipo, charlen de lo leído hasta aquí e intercambien opiniones. Intenten recordar cómo se veía esto en PseInt y analicen las diferencias.

Revisemos lo aprendido hasta aquí

- Escribir comentarios en java
- Diferenciar el paquete, la clase y el método main.
- Comprender dónde deben escribirse las sentencias

¿Cuáles son los elementos de un programa?

Los conceptos vistos previamente, son la estructura de un programa, pero también existen los elementos de un programa. Estos son, básicamente, los componentes que van a conformar las sentencias que podamos escribir en nuestro programa. Recordemos que toda sentencia en nuestro programa debe terminar con el símbolo **punto y coma** Nos van a ayudar para crear nuestro programa y resolver sus problemas. Estos elementos siempre estarán dentro de un programa/algoritmo.

Los elementos de un programa son: **identificadores, variables, constantes, operadores, palabras reservadas.**

¿Qué son las palabras reservadas?

Palabras que dentro del lenguaje significan la ejecución de una instrucción determinada, por lo que no pueden ser utilizadas con otro fin. En Java, al ser un lenguaje que está creado en inglés, todas nuestras palabras reservadas van a estar en ese idioma.

¿Qué son los Identificadores?

Los identificadores son los nombres que se usan para identificar cada uno de los elementos del lenguaje, como ser, los nombres de las variables, nombres de las clases, interfaces, atributos y métodos de un programa. Si bien Java permite nombres de identificadores tan largos como se desee, es aconsejable crearlos de forma que tengan sentido y faciliten su interpretación. El nombre ideal para un identificador es aquel que no se exceda en longitud (lo más corto posible) y que califique claramente el concepto que intenta representar en el contexto del problema que se está resolviendo.

¿Recuerdas cómo utilizar correctamente el CamelCase?

Para identificar correctamente a las variables, clases, atributos, etc, al nombrarlas hazte la siguiente pregunta:

¿Refleja este nombre la información que aloja o la función que cumple?

Variables y Constantes

Recordemos que en Pseint dijimos que los programas de computadora necesitan **información** para la resolución de problemas. Está información puede ser un número, un nombre, etc. Para utilizar la información, vamos a guardarla en algo llamado, **variables y constantes**. Las variables y constantes vendrían a ser como pequeñas cajas, que guardan algo en su interior, en este caso información. Estas, van a contar como previamente habíamos mencionado, con un identificador, un nombre que facilitara distinguir unas de otras y nos ayudará a saber que variable o constante es la contiene la información que necesitamos.

Dentro de toda la información que vamos a manejar, a veces, necesitaremos información que no cambie. Tales valores son las **constantes**. De igual forma, existen otros valores que necesitamos que cambien durante la ejecución del programa; esas van a ser nuestras **variables**.

Declaración de variables en Java

Normalmente los identificadores de las variables y de las constantes con nombre deben ser declaradas en los programas antes de ser utilizadas. La sintaxis de la declaración de una variable en java suele ser:

```
<tipo_de_dato> <nombre_variable>;
```

Tipos de dato en Java

Java es un lenguaje de tipado estático, esto significa que todas las variables deben ser declaradas antes que ellas puedan ser utilizadas y que no podemos cambiar el tipo de dato de una variable, a menos que sea a través de una conversión.

Tipos de Datos Primitivos

Primitivos: Son predefinidos por el lenguaje. La biblioteca Java proporciona clases asociadas a estos tipos que proporcionan métodos que facilitan su manejo.

byte	Es un entero con signo de 8 bits, el mínimo valor que se puede almacenar es -128 y el máximo valor es de 127 (inclusive).
short	Es un entero con signo de 16 bits. El valor mínimo es -32,768 y el valor máximo 32,767 (inclusive).
int	Es un entero con signo de 32 bits. El valor mínimo es -2,147,483,648 y el valor máximo es 2,147,483,64 (inclusive). Generalmente es la opción por defecto.
long	Es un entero con signo de 64 bits, el valor mínimo que puede almacenar este tipo de dato es -9,223,372,036,854,775,808 y el máximo valor es 9,223,372,036,854,775,807 (inclusive).
float	Es un número decimal de precisión simple de 32 bits (IEEE 754 Punto Flotante).
double	Es un número decimal de precisión doble de 64 bits (IEEE 754 Punto Flotante).

boolean	Este tipo de dato sólo soporta dos posibles valores: verdadero o falso y el dato es representado con tan solo un bit de información.
char	El tipo de dato carácter es un simple carácter unicode de 16 bits. Su valor mínimo es de '\u0000' (En entero es 0) y su valor máximo es de '\uffff' (En entero es 65,535). Nota: un dato de tipo carácter se puede escribir entre comillas simples, por ejemplo 'a', o también indicando su valor Unicode, por ejemplo '\u0061'.
String	Además de los tipos de datos primitivos el lenguaje de programación Java provee también un soporte especial para cadena de caracteres a través de la clase String. Encerrando la cadena de caracteres con comillas dobles se creará de manera automática una nueva instancia de un objeto tipo String. String cadena = “Sebastián”; Los objetos String son inmutables, esto significa que una vez creados, sus valores no pueden ser cambiados. Si bien esta clase no es técnicamente un tipo de dato primitivo, el lenguaje le da un soporte especial y hace parecer como si lo fuera.

¿Cómo se ve en Java?

```
public class HolaMundo {
     //Este es el método main
      public static void main(String[] args) {

      String nombre;
      int numero;
      double decimales;

      }
}
```

Ejercicio 1

Crear un proyecto de Java y definir al menos 6 variables en tu IDE de distintos tipos de datos.

Detección de errores

¿Puedes corregir las siguientes declaraciones de variables?

```
public static void main(String[] args) {
      string nombre
      bolean bandera
      char char;
  }
}
```

¿Ya viste lo importante que es el ; (punto y coma) después de cada sentencia?

Revisemos lo aprendido hasta aquí

- Definir variables de todos los tipos primitivos.
- Nombrar correctamente a las variables.

Instrucciones primitivas

Dentro de las instrucciones previamente vistas, existe una subdivisión que son las instrucciones primitivas, las instrucciones primitivas van a ser las instrucciones de asignación, lectura y escritura.

Asignación

La instrucción de asignación permite almacenar un valor en una variable (previamente definida). Esta es nuestra manera de guardar información en una variable, para utilizar ese valor en otro momento.

```
<variable> = <expresión>
```

En Java, podemos definir una variable y al mismo tiempo podemos asignarle un valor a diferencia de Pseint:

```
<tipo_de_dato> <nombre_variable> = expresion;
```

Al ejecutarse la asignación, primero se evalúa la expresión de la derecha y luego se asigna el resultado a la variable de la izquierda. El tipo de la variable y el de la expresión deben coincidir.

```
public static void main(String[] args) {
      String nombre = "Mariano";
      int numero = 10;
      double decimales = 40.5;
      }
```

Valores por defecto

En Java no siempre es necesario asignar valores cuando nuevos atributos son declarados. Cuando los atributos son declarados, pero no inicializados, el compilador les asignará un valor por defecto. A grandes rasgos el valor por defecto será cero o null dependiendo del tipo de dato. La siguiente tabla resume los valores por defecto dependiendo del tipo de dato.

short	**0**
int	**0**
long	**0**
double	**0.0**
boolean	**False**
char	**'\u0000'**
String	**Null**
Objetos	**Null**

Las variables locales son ligeramente diferentes; el compilador no asigna un valor predeterminado a una variable local no inicializada. Las variables locales son aquellas que se declaran dentro de un método. Si una variable local no se inicializa al momento de declararla, se debe asignar un valor antes de intentar usarla. El acceso a una variable local no inicializada dará lugar a un error en tiempo de compilación.

Ejercicio 2

¿Recuerdas las variables que creaste en el ejercicio anterior? Ahora deberás asignarles un valor.

Detección de errores

```
public static void main(String[] args) {
   int numero = "48";
   double decimales = 3,55;
   bolean bandera -> "false";
 }
}
```

Operadores

Los operadores son símbolos especiales de la plataforma que permiten especificar operaciones en uno, dos o tres operandos y retornar un resultado. También aprenderemos qué operadores poseen mayor orden de precedencia. Los operadores con mayor orden de precedencia se evalúan siempre primero.

Primeramente, proceden los operadores unarios, luego los aritméticos, después los de bits, posteriormente los relacionales, detrás vienen los booleanos y por último el operador de asignación. La regla de precedencia establece que los operadores de mayor nivel se ejecuten primero. Cuando dos operadores poseen el mismo nivel de prioridad los mismos se evalúan de izquierda a derecha.

Operadores Aritméticos	
+	Operador de Suma
-	Operador de Resta
*	Operador de Multiplicación
/	Operador de División
%	Operador de Módulo
Operadores Unarios	
+	Operador Unario de Suma; indica que el valor es positivo.
-	Operador Unario de Resta; indica que el valor es negativo.
++	Operador de Incremento.
--	Operador de Decremento.

Operadores de Igualdad y Relación	
==	Igual
!=	Distinto
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

¿Cómo se ve en Java?

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

  int num1 = 5;
  int num2 = 5;

  int suma = num1 + num2;

  double division = num1 / num2;

  boolean logico = num2 < num1;

  num1++;

  }
```

¡MANOS A LA OBRA!

Ejercicio 3

Define variables donde puedas alojar los resultados y prueba usar dos operadores de cada tipo.

Puede que tantos conceptos nuevos, instalar e utilizar un nuevo IDE nos haga sentir un poco confundidos. Habla con tu equipo, charlen de lo leído hasta aquí e intercambien opiniones. Intenten recordar cómo se veía esto en PseInt y analicen las diferencias.

Revisemos lo aprendido hasta aquí

- Operar con las variables
- Comprender las operaciones de acuerdo con los tipos de variables.

Tipos de Instrucciones

Además de los elementos de un programa/algoritmo, tenemos las instrucciones que pueden componer un programa. Las instrucciones —acciones— básicas que se pueden implementar de modo general en un algoritmo y que esencialmente soportan todos los lenguajes son las siguientes:

- ✓ **Instrucciones de inicio/fin**, son utilizadas para delimitar bloques de código.
- ✓ **Instrucciones de asignación**, se utilizan para asignar el resultado de la evaluación de una expresión a una variable. El valor (dato) que se obtiene al evaluar la expresión es almacenado en la variable que se indique.
- ✓ **Instrucciones de lectura**, se utilizan para leer datos de un dispositivo de entrada y se asignan a las variables.
- ✓ **Instrucciones de escritura**, se utilizan para escribir o mostrar mensajes o contenidos de las variables en un dispositivo de salida.
- ✓ **Instrucciones de bifurcación**, mediante estas instrucciones el desarrollo lineal de un programa se interrumpe. Las bifurcaciones o al flujo de un programa puede ser según el punto del programa en el que se ejecuta la instrucción hacia adelante o hacia atrás.

Entrada y Salida de Información

Los cálculos que realizan las computadoras requieren, para ser útiles la entrada de los datos necesarios para ejecutar las operaciones que posteriormente se convertirán en resultados, es decir, salida.

Las operaciones de entrada permiten leer determinados valores y asignarlos a determinadas variables y las operaciones de salida permiten escribir o mostrar resultados de determinadas variables y las operaciones, o simplemente mostrar mensajes.

Escritura en Java

En nuestro ejemplo de código al principio de la guía, usábamos la instrucción **System.out.println()** para mostrar el mensaje Hola Mundo. Esta instrucción permite mostrar valores en el **Output**, que es la interfaz gráfica de Java. Todo lo que quisiéramos mostrar en nuestra interfaz gráfica, deberá ir entre comillas dobles y dentro del paréntesis.

```
System.out.println("Hola Mundo");
```

Si quisiéramos concatenar un mensaje y la impresión de una variable deberíamos usar el símbolo más para poder lograrlo.

```
System.out.println("La variable tiene el valor de: " + variable);
```

Si quisiéramos escribir sin saltos de línea, deberíamos quitarle el ln a nuestro System.out.println.

```
System.out.print("Hola");
System.out.print("Mundo");
```


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

    int num = 10;

    System.out.println("La variable tiene el valor de: " + num);

    System.out.print("Hola");
    System.out.print("Mundo");

    }
```

Ejercicio 4

Define una variable que aloje tu nombre y otra que guarde tu edad. Imprime ambas variables por pantalla.

Recomendamos que hagan el siguiente experimento: tipear en minúsculas la palabra sout y apenas terminamos de escribirla tocar el botón tab o mejor dicho tabular.

Esto nos va a generar un System.out.println() para poder escribir lo que queramos.

Clases de utilidad

Se acuerdan que en PseInt vimos una serie de funciones y dijimos que las funciones, son herramientas que nos proporciona PSeInt y cumplen el propósito de ayudarnos a resolver ciertos problemas. Bueno, en Java existe algo muy parecido que se llama **Clases de utilidad.**

¿Qué son las clases de utilidad?

Las clases de utilidad son clases que definen un conjunto de métodos/funciones que realizan operaciones, normalmente muy utilizadas o necesarias. Lo que va a facilitar las clases es no tener que escribir dicha operación nosotros, por ejemplo, supongamos que tenemos que calcular la raíz cuadrada de un número, Java no va a dar un método/función, que pasándole un número, nos devuelve el resultado de su raíz cuadrada.

Entre las clases de utilidad de Java más utilizadas y conocidas están las siguientes: Arrays, String, Integer, Math, Date, Calendar y GregorianCalendar.

En esta guía solo vamos a ver a **Math** y **String** para hacer algunos ejercicios y después veremos el resto en mayor profundidad. Estas nos van a ayudar junto con Java, a lograr resolver problemas de manera más sencilla.

Clase String

Método	Descripción.
charAt(int index)	Retorna el carácter especificado en la posición index
equals(String str)	Sirve para comparar si dos cadenas son iguales. Devuelve true si son iguales y false si no.
equalsIgnoreCase(String str)	Sirve para comparar si dos cadenas son iguales, ignorando la grafía de la palabra. Devuelve true si son iguales y false si no.

compareTo(String otraCadena)	Compara dos cadenas de caracteres alfabéticamente. Retorna 0 si son iguales, entero negativo si la primera es menor o entero positivo si la primera es mayor.
concat(String str)	Concatena la cadena del parámetro al final de la primera cadena.
contains(CharSequence s)	Retorna true si la cadena contiene la secuencia tipo char del parámetro.
endsWith(String suffix)	Retorna verdadero si la cadena es igual al objeto del parámetro
indexOf(String str)	Retorna el índice de la primera ocurrencia de la cadena del parámetro
isEmpty()	Retorna verdadero si la longitud de la cadena es 0
length()	Retorna la longitud de la cadena
replace(char oldChar, char newChar)	Retorna una nueva cadena reemplazando los caracteres del primer parámetro con el carácter del segundo parámetro
split(String regex)	Retorna un arreglo de cadenas separadas por la cadena del parámetro

startsWith(String prefix)	Retorna verdadero si el comienzo de la cadena es igual al prefijo del parámetro.
substring(int beginIndex)	Retorna la sub cadena desde el carácter del parámetro
substring(int beginIndex, int endIndex)	Retorna la sub cadena desde el carácter del primer parámetro hasta el carácter del segundo parámetro
toCharArray()	Retorna el conjunto de caracteres de la cadena
toLowerCase()	Retorna la cadena en minúsculas
toUpperCase()	Retorna la cadena en mayúsculas

Java al ser un lenguaje de tipado estático, requiere que para pasar una variable de un tipo de dato a otro necesitemos usar un conversor. Por lo que, para convertir cualquier tipo de dato a un String, utilicemos la función valueOf(n).

Ejemplo:

```
int numEntero = 4;
String numCadena = String.valueOf(numEntero);
```

Si quisiéramos hacerlo al revés, de String a int se usa el método de la clase Integer, parseInt().

Ejemplo:

```
String numCadena = "1";
int numEntero = Integer.parseInt(numCadena);
```

Clase Math

En ocasiones nos vemos en la necesidad de incluir **cálculos, operaciones, matemáticas, estadísticas**, etc en nuestros programas Java.

Es cierto que muchos cálculos se pueden hacer simplemente utilizando los operadores aritméticos que **Java** pone a nuestra disposición, pero existe una opción mucho más sencilla de utilizar, sobre todo para ***cálculos complicados***. Esta opción es la **clase Math** del paquete **java.lang**.

La clase Math nos ofrece numerosos y valiosos métodos y constantes estáticos, que podemos utilizar tan sólo anteponiendo el nombre de la clase.

Método	Descripción.
abs(double a)	Devuelve el valor absoluto de un valor double introducido como parámetro.
abs(int a)	Devuelve el valor absoluto de un valor Entero introducido como parámetro.
abs(long a)	Devuelve el valor absoluto de un valor long introducido como parámetro.
max(double a, double b)	Devuelve el mayor de dos valores double
max(int a, int b)	Devuelve el mayor de dos valores Enteros.
max(long a, long b)	Devuelve el mayor de dos valores long.

min(double a, double b)	Devuelve el menor de dos valores double.
min(int a, int b)	Devuelve el menor de dos valores enteros.
min(long a, long b)	Devuelve el menor de dos valores long.
pow(double a, double b)	Devuelve el valor del primer argumento elevado a la potencia del segundo argumento.
random()	Devuelve un double con un signo positivo, mayor o igual que 0.0 y menor que 1.0.
round(double a)	Devuelve el long redondeado más cercano al double introducido.
sqrt(double a)	Devuelve la raíz cuadrada positiva correctamente redondeada de un double.
floor(double a)	Devuelve el entero más cercano por debajo.

Método random() de la clase Math

El **método random** podemos utilizarlo para generar **números al azar**. El rango o margen con el que trabaja el método random oscila entre 0.0 y 1.0 (Este último no incluido)

Por lo tanto, para generar un número entero entre 0 y 9, hay que escribir la siguiente sentencia:

```
int numero = (int) (Math.random() * 10);
```

Pueden encontrar un ejemplo de las funciones de la clase String y Math en tu Aula Virtual.