

ISSD

—Asc—  
Analista de  
Sistemas

CONCEPTOS  
INICIALES

# AED1

## Algoritmos y Estructuras de Datos 1

Prof. Ing. Eduardo Mansilla  
Módulo didáctico - 2019



*Unidad 1*



*Clase 1*



Introducción a la lógica de la  
programación



Al finalizar esta primera clase ya  
te habrás puesto en contacto con:

- | La lógica de la programación,
- | Algoritmos,
- | Diagramas de flujo,
- | Variables,
- | Operadores aritméticos,
- | Instalación de *Ready to Program*,
- | y finalmente, con la Codificación.



# *¡Qué buen programa!*

¡Bienvenidos a la primera clase!

El objetivo fundamental de la materia "Algoritmos y Estructuras de Datos 1" es que puedas resolver problemas de distinta índole (matemáticos, administrativos, gráficos, contables, etc.) empleando como herramienta la computadora.

Tené en cuenta que para llegar a ser programador debés recorrer un largo camino en donde cada tema es fundamental para conceptos futuros. Por eso es importante que no dejes temas sin entender y relacionar. La programación, requiere un estudio metódico y ordenado y en AED1, se sientan las bases para todas las otras materias orientadas a la programación.

Como la programación es una actividad nueva para vos -ya que no hay en los estudios secundarios una materia parecida- te recomendamos tener paciencia cuando no puedas resolver los ejercicios por completo.

Sin embargo, es de fundamental importancia que dediques tiempo al análisis individual de los problemas y que ante la presencia de dudas consultes con tu tutor.

Este material de estudio no es una fuente de información completa o definitiva, sino sólo una guía general. Un estudio profundo de esta asignatura, debería ser complementado con libros y manuales de consulta, además de una muy abundante ejercitación en la resolución de problemas y trabajo sobre la computadora.

En esta primera clase intentaremos, por un lado, que comprendas lo que significa programar una computadora, diferenciando el **diseño del algoritmo** de la **codificación** propiamente dicha.

Además trabajaremos para que puedas valorar la importancia de analizar a fondo el **problema a resolver** e **identificar sus elementos centrales** antes de proceder a programar su solución.





Por otro lado, trataremos que entiendas cómo un **diagrama de flujo** puede simplificar la construcción del **algoritmo** de resolución de un problema y también cómo utilizar **Java** como lenguaje de programación.

Para organizar estos contenidos los hemos dividido en cinco temas:

| Primero vas a estudiar **¿qué significa programar?**

| Luego analizaremos **¿cómo se programa una computadora?**

| Posteriormente responderemos a la pregunta **¿por qué vamos a aprender Java?** y a través de esto estudiaremos **las lógicas de programación, ¿qué es un algoritmo?, los diagramas de flujo y los componentes más utilizados para confeccionarlo.**

| En cuarto lugar, trabajaremos sobre unos **primeros ejercicios de**

**ejemplo** a partir de los cuales estudiaremos las **variables** (sus **nombres válidos**, los **tipos de variables**) y los **operadores aritméticos**.

| Finalmente, veremos la **instalación de “Ready to Program”** y realizaremos **nuestro primer programa en Java**.

En esta materia en particular, vas a sentar las bases que te permitirán ser un Analista de Sistemas, Programador o Desarrollador de Software de alto nivel. Cuando abordes cada nuevo tema, tengas que resolver una situación problemática propuesta o profundices en los conceptos centrales de la lógica y el lenguaje de programación, no pierdas de vista que tu esfuerzo vale la pena, y que aquí estamos para ayudarte y acompañarte.

*¡Bienvenidos al maravilloso mundo de la Programación!*

*Ing. Eduardo Mansilla*



Antes de comenzar con la primera clase, leé estas dos noticias publicadas recientemente en medios de prensa.

## **Cada año, la Argentina tiene un déficit de 15 mil profesionales de la informática**

*Pese a que es uno de los rubros con menor desempleo y remuneraciones mayores al promedio laboral, las carreras vinculadas con los sistemas tienen una alta tasa de deserción.*

Año tras año, el mundo de la informática se extiende a nuevos dominios: de las PC a los smartphones, de los autos inteligentes a los electrodomésticos en red y de los videojuegos a tener wi-fi en todas partes. Este positivo avance digital necesita apoyarse en expertos en tecnologías de la información (TIC) capaces de producir software e instalar y mantener redes. Y allí es donde la Argentina se enfrenta con un grave problema.

*"La industria tech demanda mucho personal. Cada año, solamente las empresas desarrolladoras de software tienen una demanda de más de 5 mil profesionales. Sin embargo, de las carreras específicas egresan apenas 3.600 expertos", le detalló a PERFIL Santiago Ceria, director ejecutivo de la Fundación Sadosky, una organización cuyo objetivo*

*es facilitar la articulación entre el sistema científico y la estructura productiva.*

Pero el déficit es todavía más grave: José María Louzao Andrade, presidente de la Cámara Argentina de Software y Servicios Informáticos, le comentó a este diario que *"la demanda insatisfecha total de profesionales de tecnología, en el mercado, ronda los 15 mil expertos cada año"*.

Para el doctor Marcelo De Vincenzi, decano de la Facultad de Tecnología Informática de la UAI, los *"cálculos indican que cada año el déficit laboral de puestos TIC es cercano al 30%. Y muchas empresas comentan que deben postergar proyectos informáticos porque no pueden encontrar el personal necesario, pese a que las remuneraciones son mejores que el promedio"*.

(...)

*Diario Perfil – 14/3/2015*



## ***Hoy, el coding es un idioma universal***

La del software es una industria con *sequía* de talento, al menos en la Argentina. La Cámara de Empresas de Software y Servicios Informáticos (Cessi), que nuclea a las empresas del sector, calcula que hay más de 5.000 puestos de trabajo que no pueden ser cubiertos cada año en un área de la economía que no para de crecer, y que para 2016 planea avanzar un 8,9% en empleos y así buscar 7.000 nuevos profesionales.

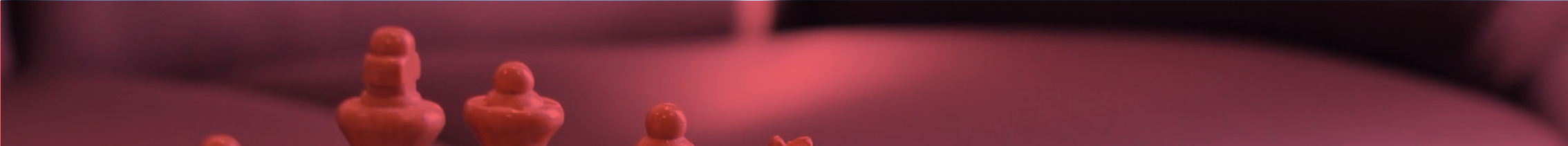
Nelson Duboscq es uno de los socios de una escuela de programación que se propuso hacer un aporte para reducir aquella escasez de capital humano y hacer del coding un lenguaje más accesible. *"Hay dos idiomas universales: el inglés y el coding (la programación, el <lenguaje de la tecnología>)"*, resaltó Duboscq. El problema es que, al menos en la Argentina, no existen iniciativas para enseñar el segundo en las escuelas. El emprendedor resaltó que, cuando idearon el proyecto, tenían a Mercado Libre, Despegar.com y Globant en mente, empresas a las que describió como "abanderadas" del país.

La idea de que se creen más de esas compañías y de que la Argentina incremente sus exportaciones de talento los incentivó a trabajar

para ofrecer capacitaciones a la fuerza laboral nacional. *"Hoy, la Argentina representa el 1% del PBI mundial, y aquellas empresas están demostrando que podemos tener marcas globales. Sueño con que en 10 o 15 años baje en un aeropuerto extranjero y encuentre nuevas marcas de origen local que aporten tecnología"*, dijo.

En una época de estancamiento de la creación del empleo privado, en 2015 el 86% de las empresas de software y servicios de tecnologías de la información de la Argentina buscaron desarrolladores de aplicaciones para su plantilla. Los sueldos promedio, según la Cessi, son de \$ 13.100 para la categoría de programadores sin experiencia previa y de \$ 23.500 para los senior. Empresas como Belatrix contratarán 1.200 profesionales de aquí a cuatro años, y Softtek, unos 200 hasta fin de año. Compañías como FlechaBus, Peñaflor y Carrefour también buscan programadores.

En 2015, las ventas superaron los US\$ 3.479 millones y las exportaciones, los US\$ 1.000 millones. Para 2016 se proyecta que esos valores crezcan 32,6% y 18,9%, respectivamente. Los principales socios comerciales durante el año pasado fueron los Estados Unidos



(que acaparó la mitad de los desarrollos que salen del país), Uruguay y México.

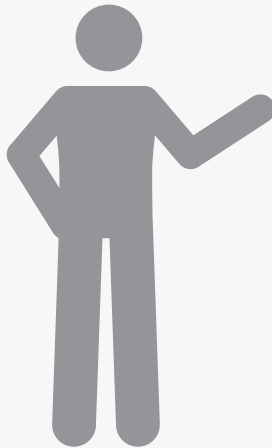
*"Trabajar en el conocimiento es nuevo. Recién estamos aprendiendo de productividad. Sabemos que podemos hacer traslación de ingresos y recibir subsidios, pero a largo plazo, si no somos competitivos, no vamos a tener una economía sustentable", expresó Duboscq. Luego, añadió: "Hoy, lo que queremos decir es que tiene sentido formarse en nuevas tecnologías, estudiar carreras duras y que el Estado dé mayor conectividad. Es una oportunidad que tenemos".*

Duboscq explicó que la transformación digital hizo que los programadores ya no estén en un área aislada dentro de las empresas, sino que hoy están en todos los sectores de las compañías donde se necesite desarrollar aplicaciones y páginas Web. Asimismo, destacó un "cambio en el cliente", que llevó a que este año las ventas por el Hot Sale crecieran en un 100% con respecto a 2015.

*La Nación – 20/6/2016*



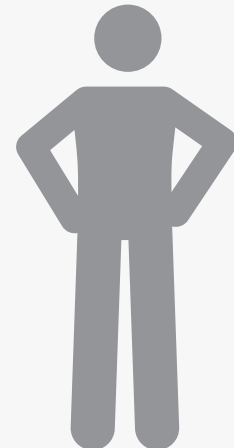
Como habrás notado, en Argentina -y en gran parte del mundo- se repite una paradoja: el área de la Programación y los Sistemas es la que requiere cada vez más profesionales capacitados y la que paga mejores salarios, pero a su vez los estudiantes eligen cada vez menos estudiar estas carreras.



Las proyecciones a futuro son claras y contundentes: has elegido una carrera clave para tu desempeño y crecimiento profesional, con demanda asegurada y altas posibilidades de inserción laboral de calidad.



Además, la Programación te permitirá trabajar en proyectos creativos, en relación constante con personas conformando equipos diversos –muchas veces transnacionales-, en ambientes llenos de desafíos y que demandan una actualización permanente.





# ¿Qué significa programar?

Muchos aparatos de uso cotidiano, contienen algún tipo de programa: desde el lavarropas automático, hasta el microondas o la heladera. Estos dispositivos, siguen una secuencia de pasos que alguien determinó para que logren cumplir con su función.

Por ejemplo: un lavarropas automático al ser encendido, en primer lugar trabará la puerta activando una cerradura magnética. Luego, abrirá la válvula para cargar el agua de la red domiciliaria. Una vez detectado el nivel adecuado, la cerrará y encenderá la resistencia para calentarla. Alcanzada la temperatura especificada, apagará la resistencia y encenderá el motor, generando un ciclo de movimientos, alternando el sentido

de giro. Pasado cierto tiempo, se abrirá la bomba de expulsión de agua, y se desalojará el tambor. Una vez vacío, el motor generará durante algunos minutos, un giro enérgico a muchas revoluciones por minuto, con el fin de centrifugar la ropa. Por último, destrabará la cerradura magnética para que se pueda quitar la ropa ya lavada.

Este proceso puede ser ajustado por el usuario mediante los controles del dispositivo, normalmente ubicados en su panel frontal.

Por ejemplo: un lavarropas permitirá elegir entre un programa para ropa delicada, con una duración reducida y una temperatura baja, y otros para lavados más prolongados





y efectivos, pero que desgastan más las prendas. También podrá seleccionar manualmente la temperatura del agua, la velocidad del centrifugado, etc.

Este proceso en realidad no es tan simple, ya que deberá además automáticamente contemplar las posibles situaciones excepcionales: ¿qué pasará si no hay agua, o si el usuario olvidó abrir la canilla?, ¿y si se produce un fallo en la bomba y el agua no sale del receptáculo como es debido?

Quien haya realizado la programación, deberá prever todo el proceso cuidadosamente, teniendo en cuenta las variables que intervienen en el mismo.

Resumiendo: un programa es un conjunto de instrucciones detalladas que le dirán al dispositivo qué hacer exactamente, paso a paso.





## Desempeños

### Desempeño 1

Escribí en una hoja, cómo sería el programa de otro dispositivo de uso cotidiano, como el microondas o la heladera.



# ¿Cómo se programa una computadora?

A diferencia de un lavarropas o un microondas (aparatos concebidos para cumplir una sola función), una computadora es un dispositivo totalmente flexible, capaz de ejecutar programas muy complejos y de naturaleza extremadamente variada. Con el mismo dispositivo, podemos en un momento procesar millones de datos, buscar información en Internet, escuchar música, ver una película o jugar... e incluso ¡todas estas actividades al mismo tiempo!

Como programadores, seremos capaces de escribir las instrucciones que necesitemos, de modo tal que la computadora lleve a cabo una secuencia de acciones, con el fin de resolver problemas de diferentes ámbitos. En otras palabras, la programación de computadoras

es el arte de lograr que una máquina haga lo que nosotros querramos.

Un programa de computación es simplemente un conjunto de instrucciones que le dicen a la computadora cómo realizar una tarea en particular.

Un programa es parecido a una receta: un grupo de instrucciones que le indican al cocinero cómo preparar un determinado plato. Describe los ingredientes (los datos) y la secuencia de pasos (el proceso) necesarios





para convertir los ingredientes (entrada) en una torta (salida).

Una computadora no *entiende* cualquier instrucción que le demos. Para poder comunicarnos con ella e indicarle con precisión una tarea, tendremos que utilizar un determinado *lenguaje*, algo así como un *idioma*.

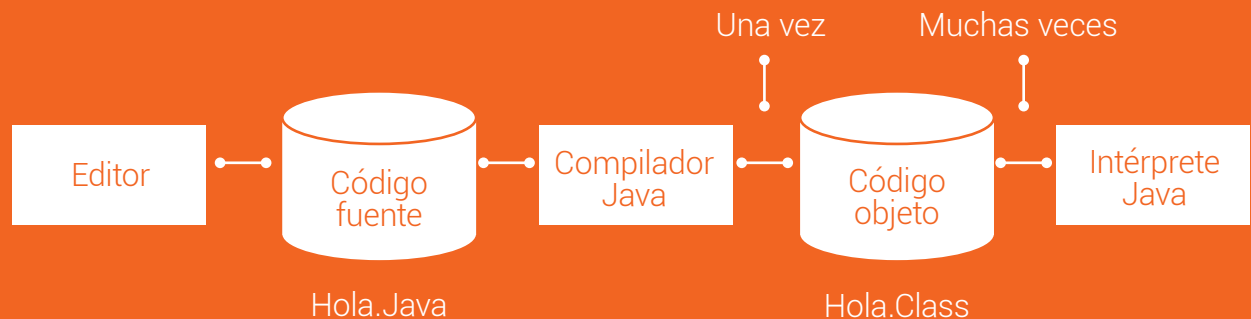
Como todo lenguaje, los lenguajes de programación poseen ciertas reglas, palabras claves, órdenes permitidas, sintaxis correctas y posibles errores a tener en cuenta.


Existen diferentes lenguajes de programación, algunos más adecuados que otros de acuerdo al tipo de aplicación que vayamos a desarrollar.

Un programador escribe instrucciones utilizando un lenguaje de programación denominado “de alto nivel”, líneas de texto que se asemejan al lenguaje con el que nos comunicamos diariamente –pero en inglés–, formalizado de modo tal que no presente ambigüedades.

A estos programas se los denomina código fuente.

Pero para que la computadora pueda entender estas órdenes, debemos traducirlas antes a un formato particular conocido como **código objeto** mediante aplicaciones denominadas **compiladores**, **intérpretes** o **ensambladores**. Una vez traducidas, las instrucciones estarán listas para ser ejecutadas por la computadora.





## → Desempeño 2

Elegí una comida sencilla que sepas cocinar (puede ser incluso preparar un sándwich o hacer un huevo frito), y escribí en una hoja paso a paso cómo realizarla.

Es muy importante que tengas en cuenta que la persona que va a seguir tu receta, no tiene ninguna experiencia, por lo tanto, no ahorres detalles al describir el proceso.





# ¿Por qué vamos a aprender Java?

Un ejemplo de lenguaje de programación es Java, el que aprenderemos durante este semestre y los que siguen.

Java es la base para prácticamente todos los tipos de aplicaciones de red, además del estándar mundial para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en la web y software para empresas. Se calcula que actualmente hay más de 9 millones de desarrolladores en todo el mundo.

En los semestres que siguen en tu carrera, también conocerás y utilizarás otros lenguajes y entornos, como por ejemplo C# o PHP.

**En pocas palabras: en esta materia vamos a dar juntos los primeros pasos**

**en Java, de modo tal que puedas utilizar este lenguaje para crear programas que luego ejecutará la computadora.**

## La lógica de programación

Toda persona que pretenda construir un programa que dé solución a un determinado problema, se enfrentará con dos grandes tareas:

la primera es definir el **qué**, es decir, la secuencia de acciones que debe realizar para poder resolver el problema. Esta etapa se puede cumplir con papel y lápiz, como parte

del trabajo de mesa previo a toda actividad de programación.

La segunda tarea es el **cómo**, o sea, de qué instrucciones nos vamos a valer, para escribir el código que realice las acciones determinadas en el **qué**, las que estarán determinadas por el lenguaje de programación seleccionado.

No es difícil apreciar que el **qué** es lo más importante, ya que si a la hora de construir un programa no contamos con un algoritmo adecuado de solución, podemos perder horas frente a la máquina escribiendo un programa que corre el riesgo de no conseguir la resolución del problema.

Es importante por lo tanto, que pongas énfasis en el diseño previo, y es aquí donde entra a jugar su papel la **Lógica de Programación**.



# ¿Qué es un algoritmo?

Los **pasos secuenciales**, son los que deben ser ejecutados uno después de otro, y los **pasos ordenados** son los que deben llevar un orden obligatorio.

Como puede notarse, lo que permite un algoritmo es lograr un objetivo.

***Nuestra herramienta mental más importante para competir con la complejidad es la abstracción. Por tanto, un problema no deberá considerarse inmediatamente en términos de instrucciones de un lenguaje, sino de elementos naturales del problema mismo, abstraídos de alguna manera.***

Niklaus Wirth

Creador del Lenguaje Pascal

Volviendo el ejemplo del lavarropas automático, la secuencia de pasos ordenados que describimos constituiría un algoritmo, cuya finalidad será lavar la ropa.

Debido a la dificultad inherente a la construcción de un algoritmo informático,

muchas personas optan por construir diagramas de flujo. Estos, pueden ayudar a resolver cualquier algoritmo, antes de la codificación real en un lenguaje de programación específico. A continuación vamos a aprender cómo aplicar esta herramienta gráfica.

## Diagramas de flujo

Los diagramas de flujo son modelos tecnológicos utilizados para comprender la programación de procesos.

Se les llama *diagramas de flujo*, porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación.

La respuesta obtenida con el desarrollo de un diagrama, no es única, sino que es una de las tantas que se pueden obtener.

Cada persona tiene una forma de razonar distinta a los demás, por lo tanto, distintas personas pueden llegar a la solución de un mismo problema de diversas maneras, es

Un Algoritmo es un conjunto de pasos secuenciales y ordenados que permiten lograr un objetivo.

Un diagrama de flujo es la forma más tradicional de especificar los detalles algorítmicos de un proceso. Estos diagramas utilizan una serie de símbolos con significados especiales. Son la representación gráfica de los pasos de un proceso que se realizan para entenderlo mejor.

decir, que puede haber varias soluciones para un determinado problema.

Las ventajas de usar diagramas de flujo son:  
| Favorecen la comprensión del proceso a través de mostrarlo como un dibujo. Un buen diagrama de flujo reemplaza varias páginas de texto.

| Permiten distinguir los problemas y las oportunidades de mejora del proceso. Se identifican los pasos redundantes, los conflictos, las responsabilidades y los puntos de decisión.

## Componentes más utilizados para confeccionar diagramas de flujo

**Inicio/Final:** Se utiliza para indicar el inicio y el final de un diagrama; del Inicio sólo puede salir una línea de flujo y al Final sólo debe llegar una línea.

**Entrada General:** Entrada de datos en general.

**Rectángulo.** Se usa para representar un evento o proceso determinado.

**Rombo.** Se utiliza para representar una condición. Normalmente el flujo de información entra por arriba y sale por un lado si la condición se cumple o sale por el lado opuesto si la condición no se cumple.

**Salida Impresa:** Indica la presentación de uno o varios resultados.

**Flecha.** Indica el sentido y trayectoria del proceso de información o tarea.

**Círculo.** Representa un punto de conexión entre procesos. Se utiliza cuando es necesario dividir un diagrama de flujo en varias partes, por ejemplo: por razones de espacio o simplicidad. Una referencia debe darse dentro para distinguirlo de otros, por ejemplo: un número.

Inicio / fin

Entrada

Procesos

Decisión

Display

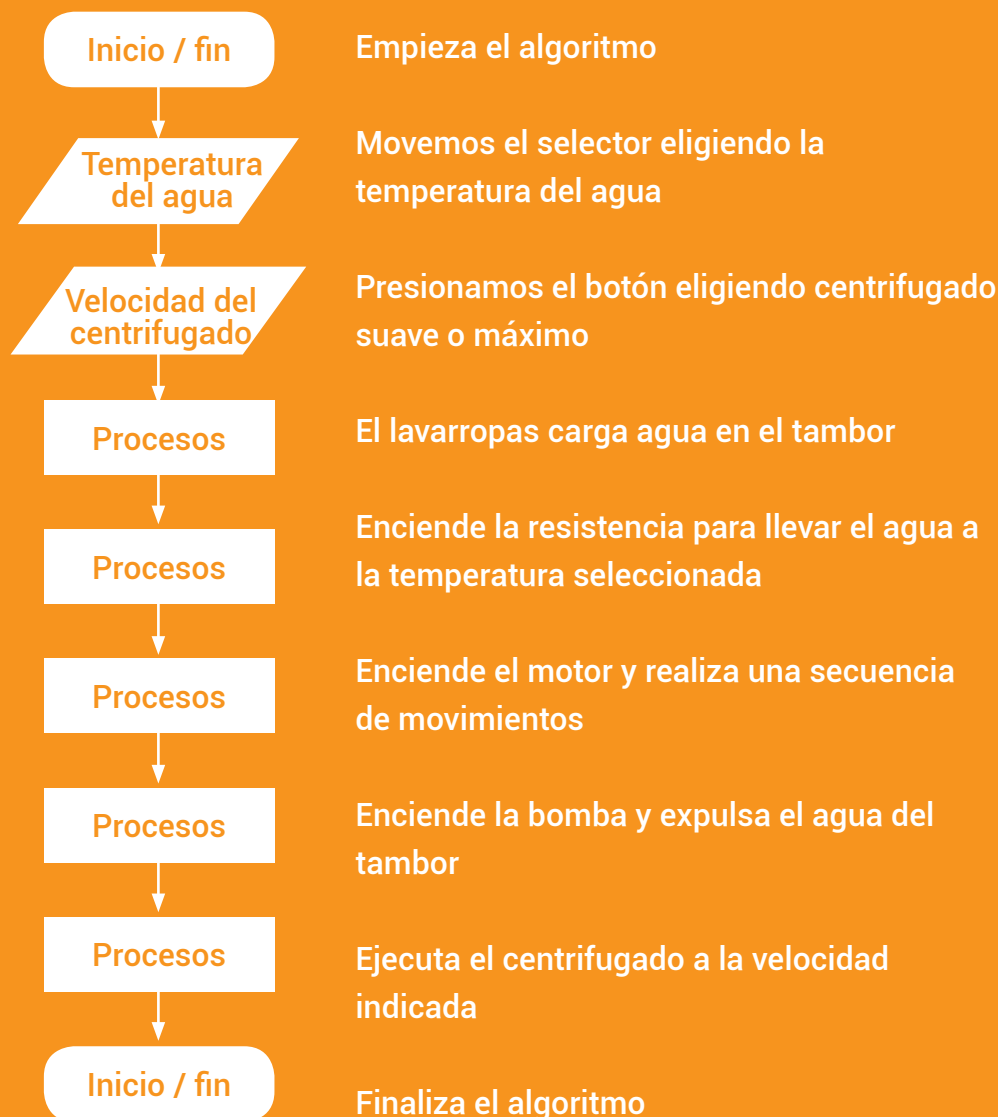


Volvamos al ejemplo del lavarropas con el que iniciamos esta clase: ¿cómo quedaría graficado el algoritmo mediante un diagrama de flujo?

¡En el diagrama del lado derecho te mostramos el resultado!



Como podés observar, hemos simplificado el algoritmo a los fines de que la explicación resulte más sencilla. Podríamos haber agregado muchos pasos básicos más, como por ejemplo, trabar y destrabar la puerta, encender las luces del panel, girar el tambor en un sentido y luego en otro, elegir el tiempo de lavado, incluir un ciclo de prelavado, etc.





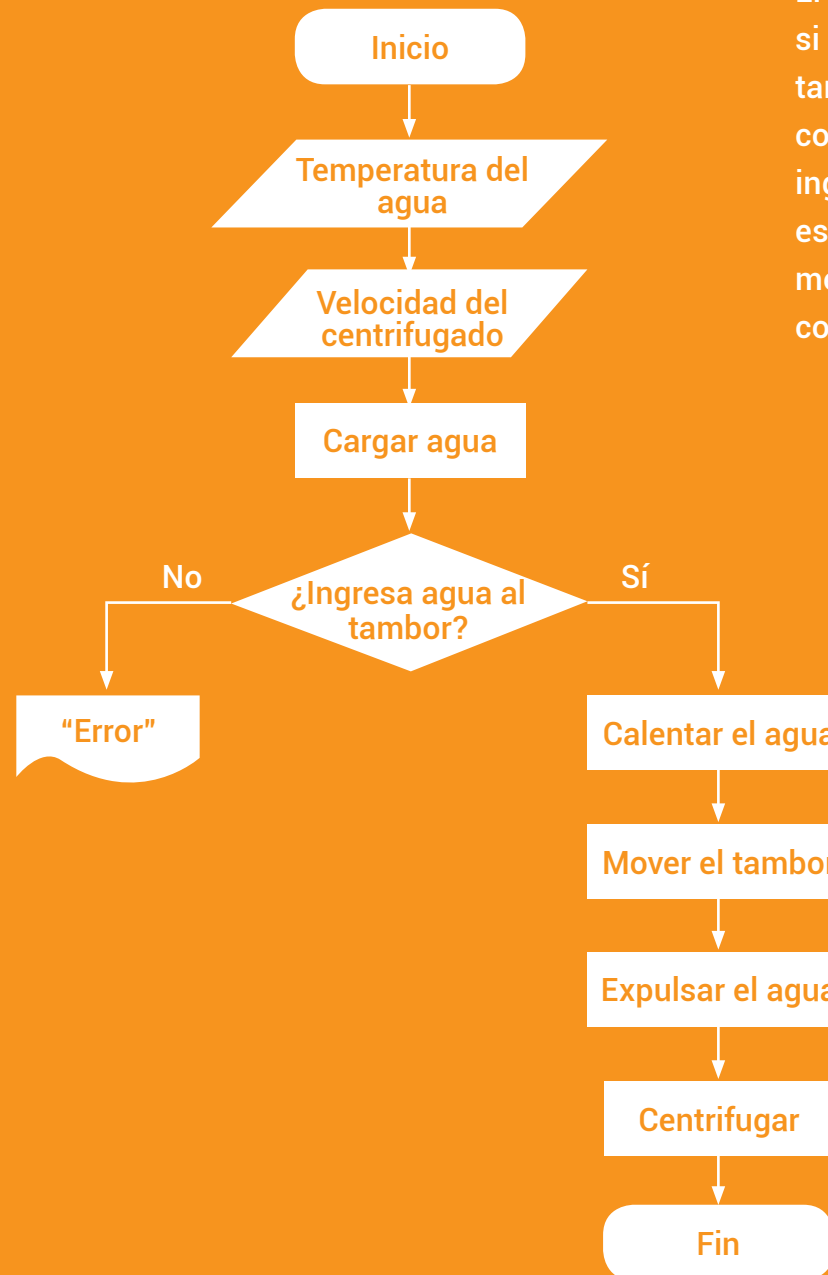
Hay algunos símbolos que no utilizamos en el ejemplo anterior, como el rombo (de decisión). Para aplicarlo, vamos a complejizar el problema, suponiendo que debemos contemplar dentro del algoritmo, si el agua realmente está ingresando al lavarropas, o si el usuario se olvidó de abrir la canilla. Si esto ocurre, supongamos que mostraremos un error en el panel del dispositivo. Ahora, el diagrama quedaría como figura en la imagen de la derecha.

En definitiva, un rombo permite que el flujo del algoritmo se divida en dos **caminos alternativos**, de acuerdo a que una **condición** se cumpla o no.

En nuestro ejemplo del lavarropas, los caminos posibles serán:

- a |** seguir el proceso de lavado, y
- b |** interrumpir el proceso y mostrar un mensaje de error.

La condición será que el agua entre o no al tambor, situación que determinará cuál camino se tomará en cada caso.



El rombo indica una decisión: si está ingresando agua en el tambor, entonces el proceso continúa. Pero si el agua no ingresa (por ejemplo, la canilla está cerrada), se muestra un mensaje de error y el programa no continúa.



## Desempeño 3

Modificá el diagrama anterior para agregar una nueva condición: si está la puerta cerrada, el lavarropas empezará el proceso de lavado, pero si la puerta se encuentra abierta, deberá mostrar un mensaje de error, ¿te animás?



# Primeros ejercicios de ejemplo

¿Te quedó claro de qué se trata un diagrama de flujo? Vamos ahora a aplicarlo en la resolución de un problema simple: nuestro primer ejercicio de ejemplo.



Es muy importante que de aquí en más, no te limites a leer los ejemplos, sino que los pruebes en tu computadora.



Practicar, probar y encontrar los errores de todos los ejemplos y de todos los ejercicios propuestos, es la única forma de aprender a programar. Y no te olvides de recurrir al tutor en cuanto veas que algo no funciona.





# Ejemplo 1

## Cálculo del precio total de una compra

Conociendo el precio unitario de un artículo y la cantidad comprada, calcular el precio total a pagar.

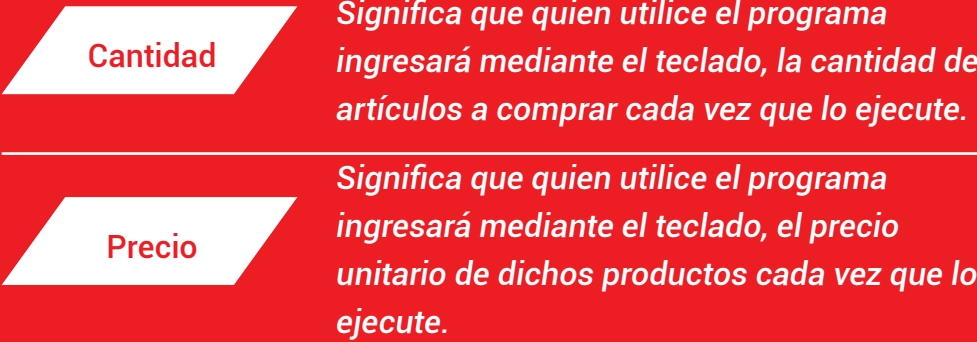
Si compramos 5 DVD y el precio unitario es de \$15, el precio total de la compra se obtendrá multiplicando la cantidad por el precio, es decir  $5 * 15$  (en la computadora, el símbolo de multiplicación que se utiliza es el asterisco (\*) y el de división es la barra (/)).

Si tuviéramos que escribir un algoritmo que resuelva este tipo de problemas (no solo el planteado, sino para cualquier cantidad de productos y de cualquier precio), según lo que aprendimos, el diagrama resultante sería el de la imagen de la derecha.

Como estarás suponiendo, estos valores (precio y cantidad), serán diferentes cada vez que el programa sea usado, es decir, que serán **variables**. La palabra "precio" y la palabra "cantidad" nos servirán como **identificadores** de dichos valores, para que luego podamos



Un par de consideraciones importantes para entender este algoritmo que acabamos de diseñar:



concretar el cálculo:  $\text{cantidad} * \text{precio}$ .  
Al programar, frecuentemente necesitaremos estos "contenedores" del valor que el usuario ingrese o del resultado que se obtenga.  
Formalmente, los llamaremos **variables**

## Las variables

En el ejemplo 1, supongamos que hacemos funcionar el programa e ingresamos el valor **5** para la cantidad y **15** para el precio. Las variables quedarán cargadas como lo muestra la imagen (1).

En nuestro programa necesitaremos además otra variable, a la que llamaremos *importe*, para poder almacenar en ella el resultado cuando realicemos el cálculo. (2)

La operación correspondiente al cálculo del importe total, la expresamos en el diagrama dentro del rectángulo, lo que debe leerse de la siguiente manera:



**Importe = cantidad \* precio**

*Multiplique el valor que contiene la variable "cantidad" por el valor que contiene la variable "precio" y guárdelo en la variable "importe".*



Es decir que el signo **igual** que figura en la operación implica una **asignación** del resultado a dicha variable: primero se multiplica la **cantidad** por el **precio** (lado derecho del signo igual), y luego al resultado se lo coloca en la variable **importe** (lado izquierdo del signo igual):

$$\text{importe} = \text{cantidad} * \text{precio}$$

### Analizando el problema

Volvamos a revisar el diagrama que acabamos de resolver. Podemos identificar en dicho problema, tres elementos fundamentales:

algunos datos de entrada, un proceso a realizar y datos de salida o resultado. (1)

Si analizamos de esta forma cualquier problema que necesitemos resolver, se nos facilitará su comprensión y solución. Vamos con otro ejemplo.

### Ejemplo 2 Cálculo del sueldo mensual de un operario

Calcular el sueldo mensual de un operario, conociendo la cantidad de horas trabajadas y el pago por hora.

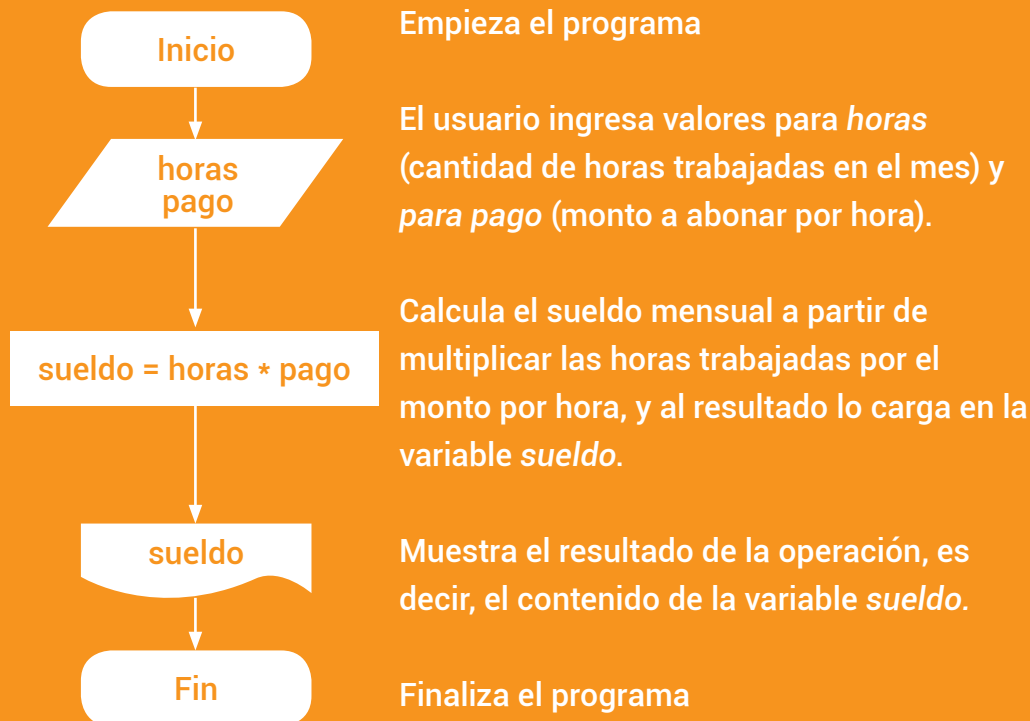
**Datos de Entrada:** Horas Trabajadas en el mes, Pago por hora.

**Datos de Salida:** Sueldo Mensual

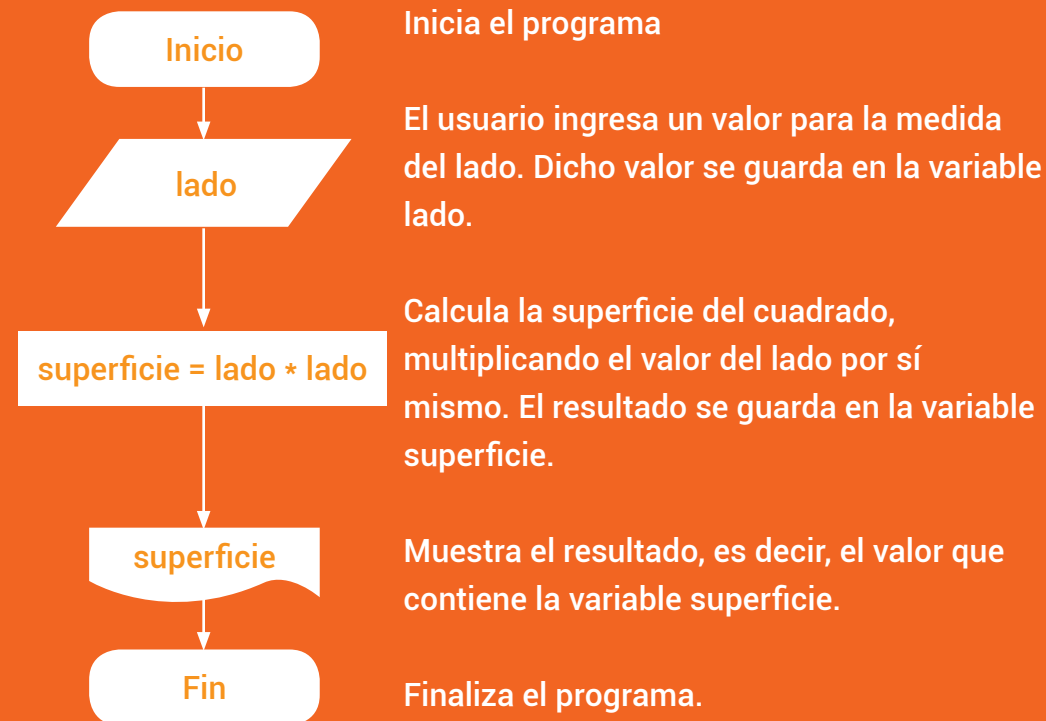
**Proceso:** Cálculo del Sueldo Mensual, que se obtiene de multiplicar la cantidad de Horas Trabajadas en el mes por el Pago por Hora.

Entonces, necesitaremos 3 variables: una para guardar las **Horas Trabajadas**, otra para guardar el **Pago por Hora**, y una tercera para almacenar el resultado, es decir, el **Sueldo Mensual**.

1



2



Para empezar a acostumbrarnos al formato correcto que luego usaremos al programar en Java, el nombre de una variable no puede contener espacios intermedios. Es buena costumbre colocarle a las variables nombres que nos recuerden su contenido. Por lo tanto, sería apropiado en este caso usar: horas, pago y sueldo.

El diagrama quedaría como el de la imagen (1).

### Ejemplo 3 Cálculo de la superficie de un cuadrado

**Obtener la superficie de un cuadrado a partir de conocer el valor de sus lados**

**Datos de Entrada:** Medida del Lado.

**Datos de Salida:** Superficie del Cuadrado.

**Proceso:** Cálculo de la Superficie del Cuadrado, que se obtiene de multiplicar Lado por Lado.

El diagrama quedaría como el de la imagen (2).

## Nombres válidos de variables

Como ya vimos, podemos definir a una **variable**, como un espacio físico que sirve para albergar un dato.

Toda variable posee un **contenido** (el que se ingresará o calculará cada vez que se utilice el programa), un **identificador** (el nombre que elegimos para la variable, en los ejemplos



anteriores *hora, sueldo, lado*, etc.) y un **tipo** (de qué clase es el contenido que va a almacenar dicha variable, por ejemplo: numérico entero, numérico decimal, texto, etc.).

Al nombre **identificador** de una variable, lo decide el programador y es necesario seguir ciertas reglas para que sea válido en Java.

### ¿Cuáles son esas reglas?

| El nombre identificador debe comenzar con una letra en minúscula. Los demás caracteres pueden ser otras letras, dígitos o el carácter subrayado.

Ejemplo de nombres identificadores válidos: **precio, cantidad, dni, folio67, dia\_laborable**, etc.

Ejemplo de nombres identificadores no válidos: **código, año, día, 5pesos**.

Estos últimos ejemplos no son válidos porque los caracteres acentuados y la letra ñ, son especiales, y no están permitidos en Java.

El cuarto identificador, no es válido porque comienza con un número.

| No debe haber espacios en blanco entre carácter y carácter.

| Cuando un nombre se forma como fusión de dos o más palabras, se comienza siempre con minúsculas y la primera letra de cada palabra posterior se coloca en mayúsculas.

Ejemplo: **diaDeLaSemana, sueldoDePersonal, precioUnitario**, etc.

Podemos definir a una variable, como un espacio físico que sirve para albergar un dato.

Toda variable posee un contenido, un identificador y un tipo. Al nombre identificador de una variable lo decide el programador y es necesario seguir ciertas reglas para que sea válido en Java.



## Desempeño 4

Indicá si los los siguientes identificadores de variables son válidos para Java o no, y en caso de que no lo sean indicá el motivo, y cómo podrías reformularlo para que sea un identificador válido (el punto **a**) ya está resuelto como ejemplo):

a | cumpleaños: *no válido, porque contiene un carácter prohibido (ñ).*

*Podría ser cumpleaños.*

b | díaDeLaSemana

c | UnidadDeMedida

d | fecha\_Inicio

e | cantidad total

f | solución

g | dia\_Nacimiento



# Tipos de Variables

Los datos que puede contener una variable son de diversos tipos. En Java se debe definir una variable de un determinado tipo, de acuerdo al dato que almacenará.

Veamos ahora cuáles son estos tipos de variables.

## Variable Tipo char

**Char** viene de la palabra **carácter**, es decir, **carácter**. Una variable definida de tipo char, puede almacenar un único carácter, o sea, puede almacenar cualquier símbolo de los indicados en el teclado.

Para definir una variable, se debe colocar primero el tipo y luego el nombre identificador de la misma. Si se necesita definir varias variables del mismo tipo, se pueden colocar todas separadas por comas. Al final de cada definición, se debe colocar un punto y coma, que es la forma en la que Java identifica que termina una instrucción.

Por ejemplo:

```
char letra, digito, sexo; } Definición de variables tipo char
```

Para asignar un dato a una variable definida de tipo char, se lo debe colocar entre apóstrofes (comillas simples).

```
letra = 'L' ;  
digito = '7' ;  
sexo = 'M' ;
```

} Asignación de valores a las variables

```
letra = 'L'
```

**Asignar** un dato a una variable, significa cargarle dicho contenido.

Por ejemplo, después de ejecutar las instrucciones anteriores, la variable de tipo *char* identificada como **letra**, pasa a contener la letra **L**; la variable identificada como **digito** pasa a contener el carácter **7**.



¡Cuidado! porque en este caso el **7** se considera un carácter, no un número. Por lo tanto, si la variable es de tipo *char*, por más que contenga un carácter numérico, no podrá luego ser utilizada para realizar una operación matemática (1).

## Variable Tipo int

**Int** viene de la palabra **integer**, es decir, **entero**. Una variable definida de este tipo, puede almacenar un número entero en el rango de  $[-2147483648 \text{ al } 2147483647]$ .

Para asignar un dato a una variable definida de tipo *int*, no es necesario usar apóstrofes. Por ejemplo:

```
int legajo, cantidad,
codigo, dia
```

} Definición de variables tipo *int*

```
legajo = 5832 ;
cantidad = 16 ;
codigo = 24 ;
dia = 12 ;
```

} Asignación de valores a las variables

## Variable Tipo float

**Float** significa **flotante**. Una variable definida de tipo *float*, puede almacenar

un número con punto decimal flotante en el rango de:  $[-3,4 \times 10^{38} \text{ al } 3,4 \times 10^{38}]$ .

Por ejemplo:

```
float sueldo, precio, altura,
distancia, ancho ;
```

} Definición de variables tipo *float*

```
sueldo = 5436.44f ;
precio = 35.50f ;
altura = 1.67f ;
```

} Asignación de valores a las variables



Es obligatorio colocar la letra **f** al final de cada valor, para indicarle a Java que el dato asignado a la variable es de tipo *float*.



# Variable Tipo String

**String** significa tira o hilera de **caracteres**. Una variable definida de tipo String, permite almacenar una palabra, frase o texto.

Por ejemplo:

String apellido, nombre,  
direccion, frase;

} Definición de  
variables tipo String

apellido = "Rodríguez" ;  
nombre = "Alberto" ;  
direccion = "9 de Julio 1653" ;  
frase = "Hoy es un día soleado" ;

} Definición  
de  
variables  
tipo String

Observen que para asignar un dato a una variable de tipo *String*, se lo hace entre comillas.



Existen muchos más tipos de variables, pero los estudiaremos más adelante.

# Operadores Aritméticos

En los problemas de ejemplo que vimos hasta ahora, realizamos algunos cálculos como multiplicar lado por lado, o el valor de cada hora por las horas trabajadas.  
¿Qué otros operadores aritméticos se pueden utilizar en una asignación?  
Java reconoce los siguientes:

Operación Aritmética	Operador
Suma	+
Resta	-
Multiplicación	*
División	/
Resto de División	%





## Desempeño 5

| Para cada uno de los siguientes datos, indicá el tipo de variable a utilizar, inventá un nombre válido en Java y escribí cómo sería su definición y asignación (el punto **a**) ya está resuelto como ejemplo):

**a** | 75.3: tipo float – *float porcentajeDeAciertos; – porcentajeDeAciertos = 75.3f;*

**b** | 10312

**c** | J

**d** | 0.18

**e** | Carlos Domínguez

**f** | Gral. Alvear 270

**g** | n



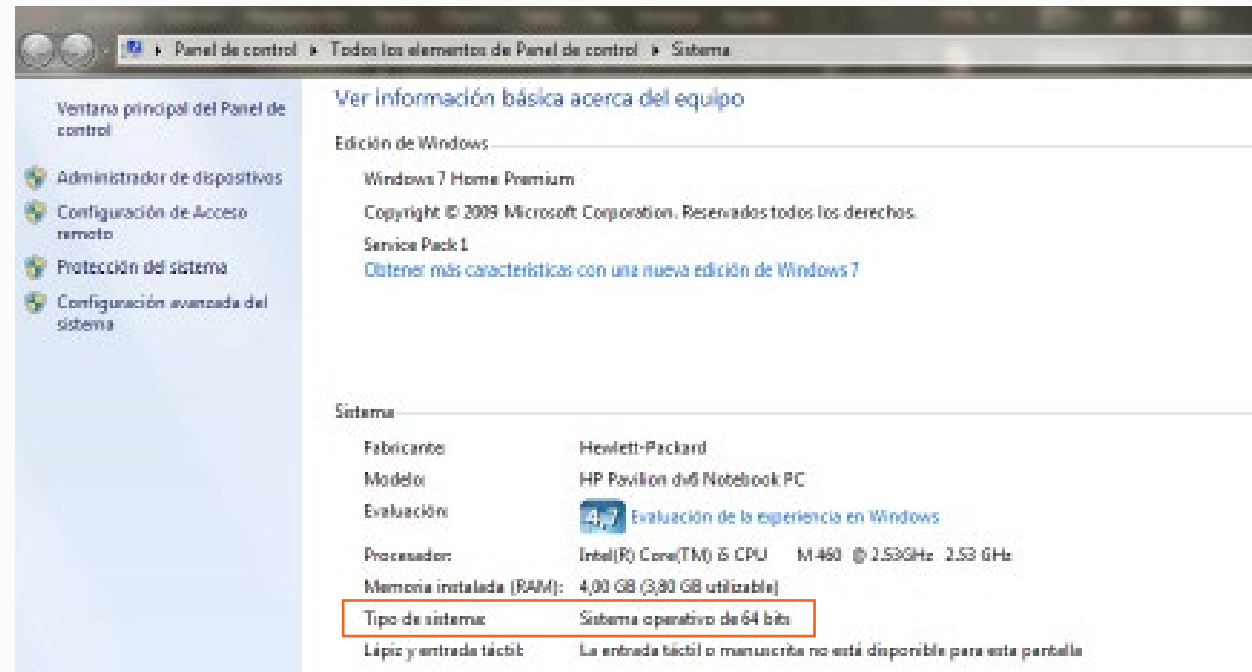
# Instalación de “Ready to Program”

Ya estamos casi listos para empezar a programar en Java. Para poder hacerlo, necesitamos instalar primero el entorno de programación que usaremos en esta materia, llamado *Ready to Program*.

Hay dos versiones de este entorno de programación según sea el sistema operativo que tengas en tu computadora: **32 bits** y **64 bits**. Si no estás seguro de cuál tenés, ingresá al Panel de Control de Windows, Sistema y allí podrás comprobarlo. En la imagen de la derecha te mostramos la pantalla para que te orientes.

|Para instalar la versión de 32 bits, **hacé clic aquí**.

|Para instalar la versión de 64 bits, **hacé clic aquí**.

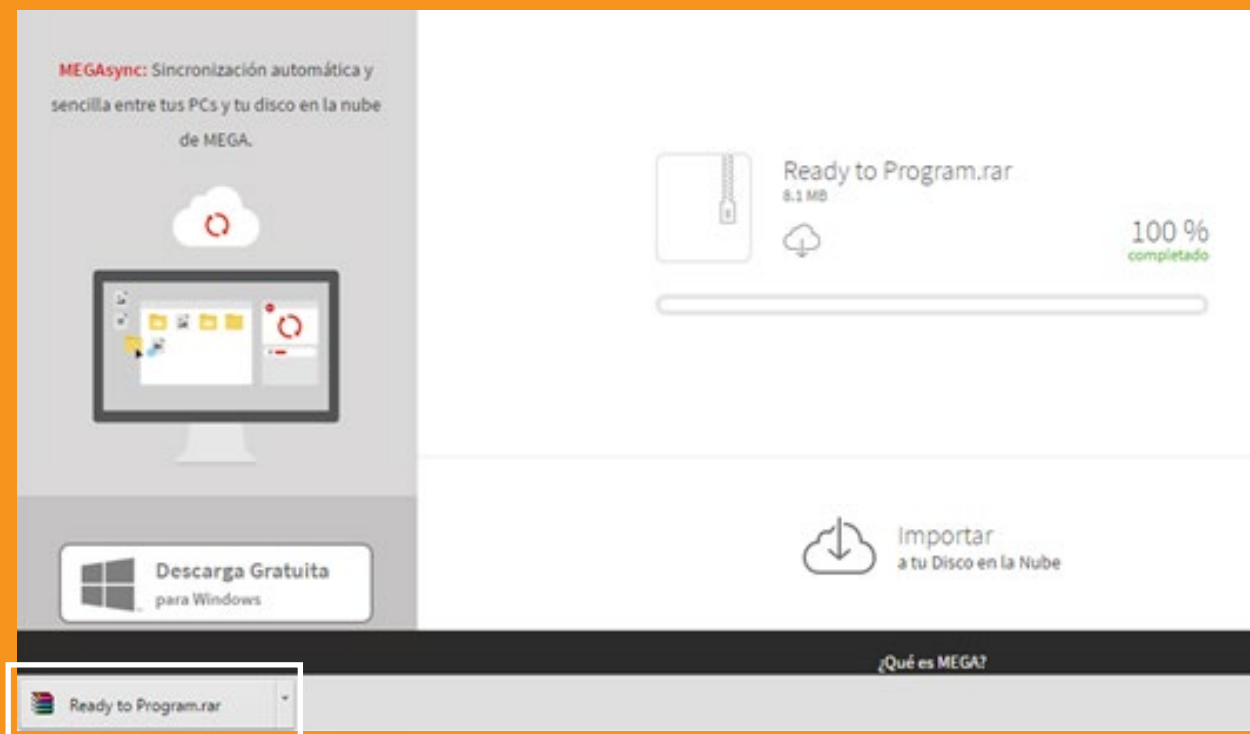
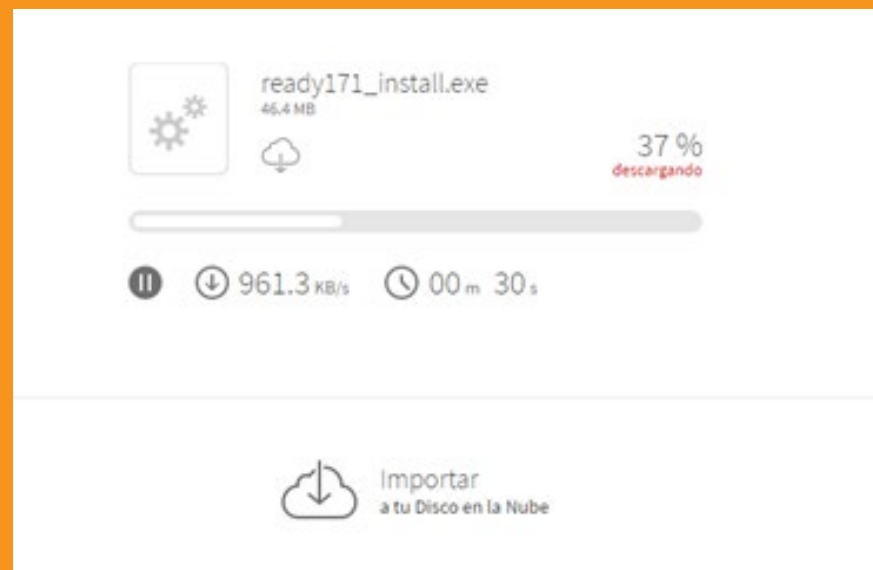


Elegí **Descargar con el navegador** y esperá hasta que se complete el 100%.

Luego, hacé click en la parte de abajo de la pantalla.

Si es la versión de 32 bits, deberás descomprimir el archivo en una carpeta de tu computadora y luego hacer click sobre *Setup* para que empiece a instalarse.

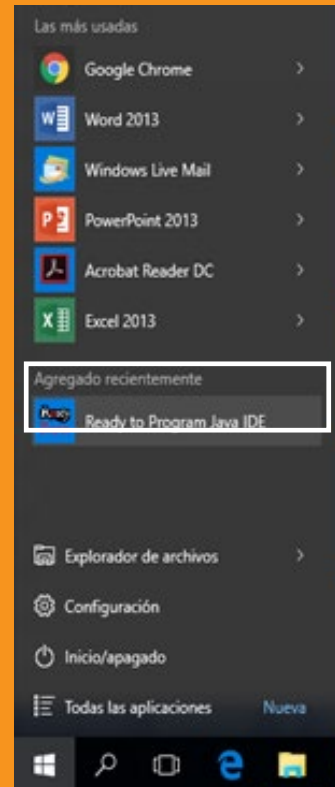
Si es la versión de 64 bits, directamente empezará a instalarse (imagen 1 y 2).



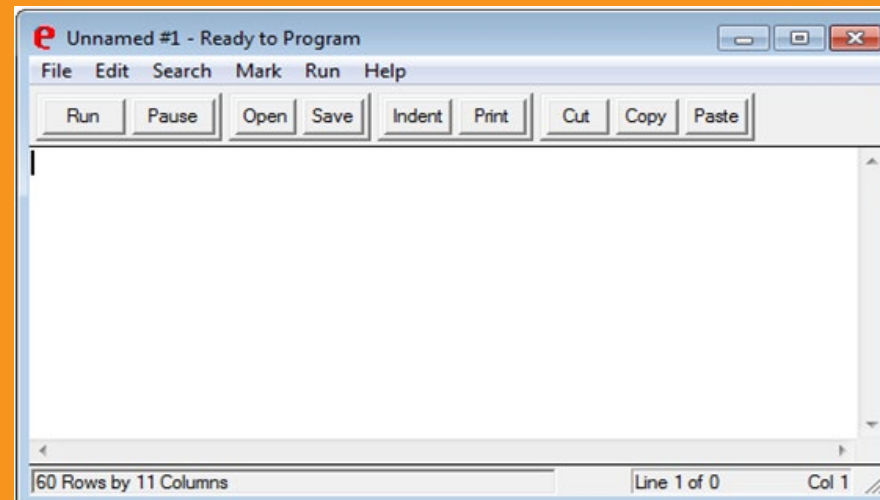


Para que empiece a funcionar, busqué en el botón Inicio la aplicación (1). Entonces, aparecerá la pantalla de trabajo (2)

1



2

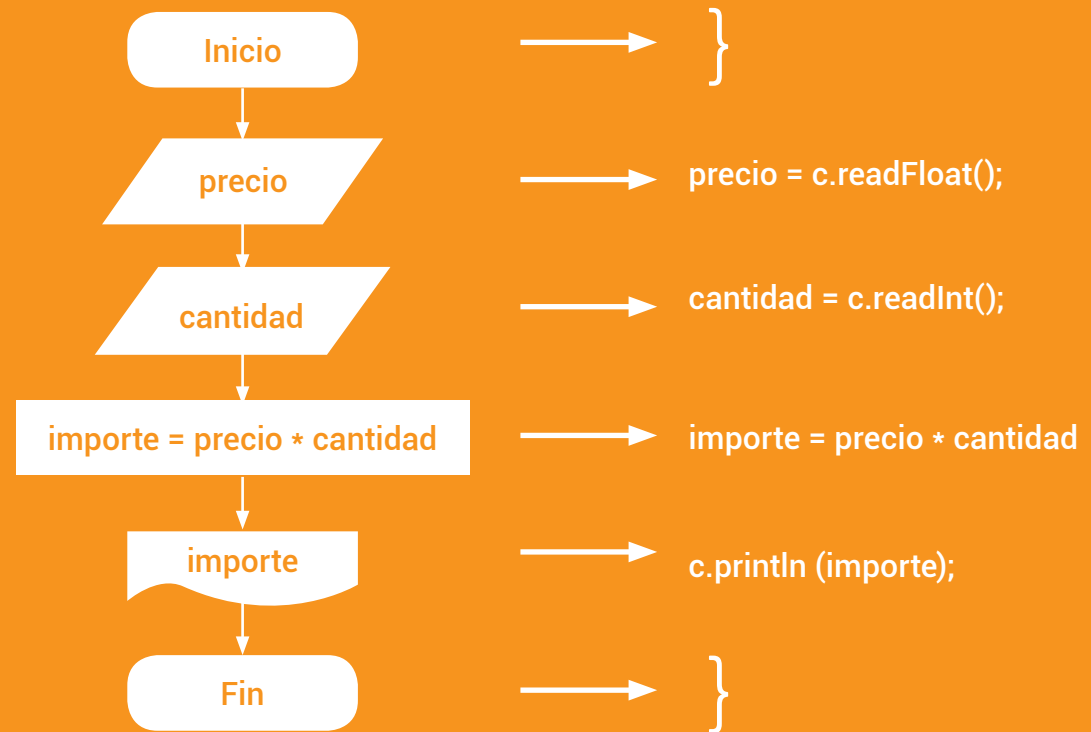


# Nuestro primer programa en Java

¡Ya estamos en condiciones de comenzar a escribir nuestro primer programa en Java! Lo haremos a partir del diagrama de flujo que armamos con anterioridad en el **Ejemplo 1** y que te volvemos a presentar en la imagen de la derecha.

El programa debe comenzar con una llave (**{**), que indica el inicio de la actividad.

Luego se debe pedir el **precio**, para ello se utiliza la instrucción **c.readFloat()**, que significa: "leer un valor con punto decimal flotante y almacenarlo en la variable precio". Posteriormente, se ingresa la **cantidad** con la instrucción **c.readInt()**, que significa: "leer un valor entero, y almacenarlo en la variable cantidad".



```

import hsa.Console;
class Ejemplo1
{
    static Console c;
    public static void main(String arg[])
    {
        float precio, importe;
        int cantidad;
        c = new Console();
        c.print("Ingreso Precio: ");
        precio = c.readFloat();
        c.print("Ingreso Cantidad: ");
        cantidad = c.readInt();
        importe = precio * cantidad;
        c.println();
        c.println("El importe a abonar es: " + importe);
    }
}

```

La asignación del cálculo del **importe**, se coloca en el programa exactamente igual como está en el diagrama.

Por último, para desplegar en la pantalla el resultado obtenido, se utiliza la instrucción **c.println()**, con la variable que se quiere mostrar dentro de los paréntesis.

Para finalizar el programa, se coloca otra llave (**)**), pero observen que ésta es de cierre, mientras que la inicial era de apertura.

¡Muy importante! A continuación, vamos

a agregar a nuestro programa una serie de elementos que por el momento **no deberás entender del todo**, pero que son indispensables para que funcione. En clases posteriores veremos en detalle para qué sirven.

Para comenzar a tipear el programa, se debe previamente indicar a Java con qué paquete de librerías vamos a trabajar, lo que se consigue con la instrucción:

`import hsa.Console;`  
que le dice a Java que importe la librería de clases **hsa.Console**. Esta librería contiene las

funciones y métodos apropiados de Entrada y Salida.

Se debe tener en cuenta además que en Java, todo programa es una clase y por lo tanto, debe llevar un nombre que debe comenzar en Mayúsculas y es precisamente con ese nombre como debe ser grabado en el disco.

El programa una vez tipeado y grabado, quedará tal como lo muestra la imagen **1**

En este caso, a la clase se le dió el nombre de **Ejemplo1**.

La instrucción: `static Console c ;` se coloca para poder utilizar los métodos de Entrada y Salida (`readInt`, `readFloat`, `println`) con el objeto `c`, mas adelante, se explicará en mayor medida esta línea.

La línea: `public static void main(String arg[])` es la cabecera del programa principal. Todos los programas que realicemos por el momento comenzarán de la misma forma.

El inicio del programa empieza definiendo las variables a utilizar, en este caso, las variables **precio** e **importe** son definidas de tipo *float* y **cantidad** de tipo *int*. Se define la variable como un objeto de la *Console*, por lo tanto, estará capacitada para acceder a los métodos de entrada y salida.

La línea: `c.print ("ingrese Precio: ");` no figura en el diagrama, pero es necesario colocarla para que la computadora, antes de detenerse pidiendo un dato con la línea:  
`precio = c.readFloat();`

muestre la leyenda correspondiente para ingresar el precio, de modo que el usuario tenga una guía acerca de qué dato está esperando el programa. Las demás líneas, continúan secuencialmente, de acuerdo con el diagrama de flujo.

La instrucción: `println();` se utiliza para dejar un renglón en blanco entre los datos introducidos y el resultado que nos entrega la computadora.

Observen que para realizar la impresión del resultado, se ha colocado una leyenda y al lado, la variable correspondiente separadas por un signo mas (+). Este signo, al utilizarlo con constantes o variables de tipo *String* realiza la concatenación de los elementos, es decir, que en este caso va a producir la unión entre la leyenda *El importe a abonar es:* con el valor que le corresponde a la variable **importe**.

Una vez codificado el programa, se lo debe ejecutar. Para ello se accede al botón [Run (F1)], que realiza la compilación del programa y posteriormente la ejecución. Compilar el programa, significa traducir cada una de las líneas escritas en Java, a un lenguaje entendible por la computadora. Si la traducción

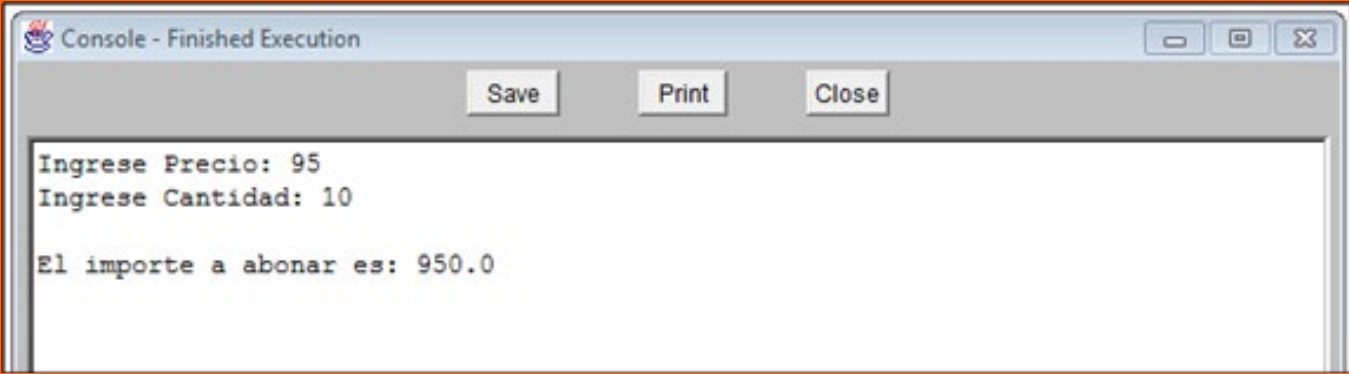
¿Qué significa Compilar un programa?

Significa traducir cada una de las líneas escritas en Java, a un lenguaje entendible por la computadora.

no registra ningún error, se observará una pantalla de salida del programa, como se muestra en la imagen 1.

En caso de haber errores en las líneas tipeadas, el programa no se ejecutará, y se mostrará reasaltada la línea en la cual se detectó el error, indicándose el tipo de error en la parte inferior de la ventana.

## Seguimos codificando



¡Hay varias cosas que no entiendo!

\* ! #

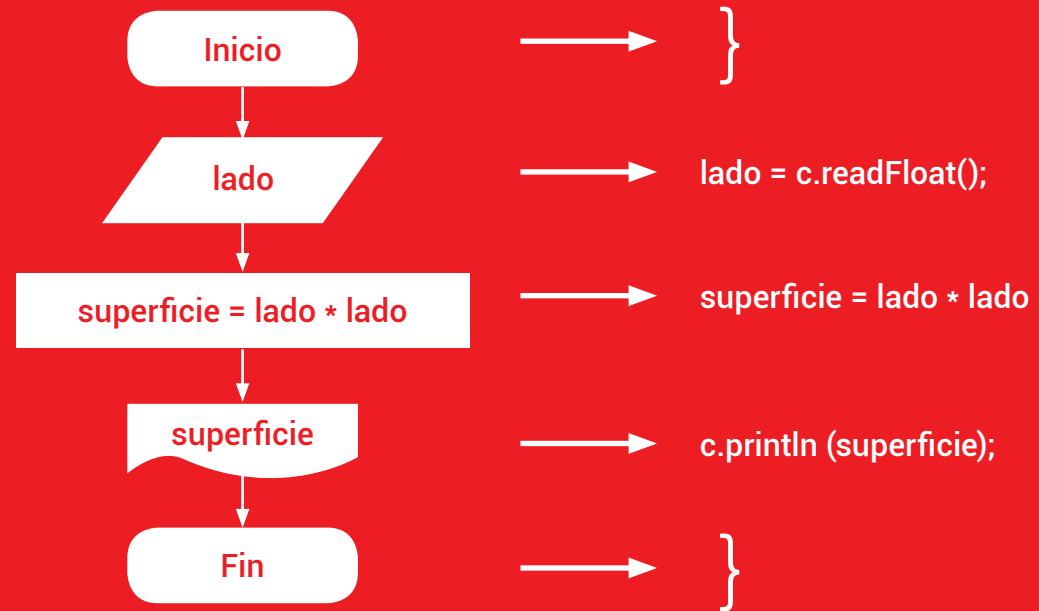


Lo que pasa es que le agregamos a nuestro programa una serie de elementos que por el momento son complicados de entender, pero que son indispensables para que funcione. En clases que vienen veremos en detalle para qué sirven y ya vas a poder comprender mejor...





Ahora haremos lo mismo con el problema que nos pedía hallar la superficie de un cuadrado conociendo el valor de uno de sus lados (2)



```
import hsa.Console;
class Ejemplo2
{
    static Console c;
    public static void main(String arg[])
    {
        float lado, superficie;
        c = new Console();
        c.print("Ingrese el valor de un Lado: ");
        lado = c.readFloat();
        superficie = lado * lado;
        c.println();
        c.println("El superficie del cuadrado es: " + superficie);
    }
}
```

The screenshot shows a Java IDE window with the title 'Ejemplo2.java\* - Ready to Program'. The code in the editor matches the code shown in the diagram above. The status bar at the bottom indicates 'Line 16 of 16' and 'Col 6'.

## Un ejercicio completo más

Para terminar con esta primera clase, hagamos un ejercicio nuevo completo.

### Ejemplo 4 Cálculo de la altura promedio de tres personas

**Ingresar tres nombres con sus respectivas alturas y determinar e imprimir, la altura promedio. (1)**

El código resultante, será el siguiente:

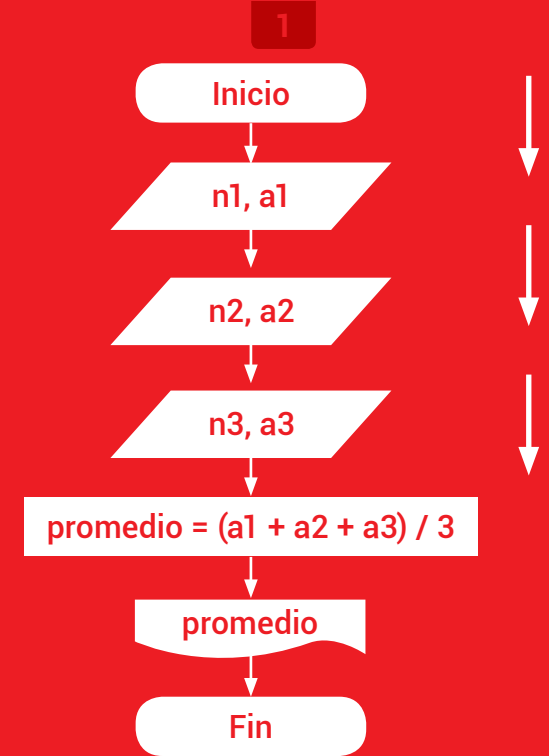
```
import hsa.Console;
class Ejemplo4
{
    static Console c;
    public static void main(String arg[])
    {
        String n1, n2, n3;
        float a1, a2, a3;
        float promedio;
        c = new Console();
```

```
        c.print("Ingrese primer nombre: ");
        n1 = c.readLine();
        c.print("Ingrese su altura: ");
        a1 = c.readFloat();
```

```
        c.print("Ingrese segundo nombre: ");
        n2 = c.readLine();
        c.print("Ingrese su altura: ");
        a2 = c.readFloat();
```

```
        c.print("Ingrese tercer nombre: ");
        n3 = c.readLine();
        c.print("Ingrese su altura: ");
        a3 = c.readFloat();
```

```
        promedio = (a1 + a2 + a3)/3;
        c.println();
        c.println("La altura promedio es: " +
                promedio);
    }
}
```



En el símbolo de introducción de datos, se puede colocar más de una variable, pero se debe tener en cuenta que por cada variable, se colocará una instrucción *read* en el programa.

En este caso será:

```
n1 = c.readLine();
a1 = c.readInt();
```

➡ Para pedir in dato de tipo *String*, se utiliza la instrucción `c.readLine()`, que significa *leer una línea*.



## → Desempeño 6

| Durante esta semana, deberás realizar los diagramas de flujo y codificación de los siguientes ejercicios. Es muy importante que los hagas a todos, y que consultes mediante el foro, en el caso de necesitar ayuda.

1 | Realizar el ingreso de dos números por teclado e imprimir su suma y su producto.

2 | Ingresar la base y la altura de un triángulo, imprimir posteriormente la superficie del mismo. (Recordar que la superficie de un triángulo es base por altura sobre 2). La fórmula sería:  $\text{superficie} = b * h / 2$  ;

3 | Ingresar el nombre de un artículo con su respectivo precio. Imprimir el importe a pagar teniendo en cuenta que se realizó un descuento del 12 % por abonar de contado.



# Bibliografía


## Imágenes

[www.pexels.com](http://www.pexels.com)

[www.pixabay.com](http://www.pixabay.com)

[www.flickr.com](http://www.flickr.com)





Esta semana hemos aprendido muchísimo: desde cómo analizar un problema y traducir su solución a un diagrama de flujo, hasta codificar nuestros primeros programas en Java.

La clave para seguir correctamente la materia es que lleves a cabo todos los ejercicios en la computadora, tanto los que son ejemplos como los desempeños, y que no dudes en consultar ante cualquier inconveniente.

*¡Nos vemos la próxima clase!*