

Guía de aprendizaje

Mi primer algoritmo

Exp 1 – Semana 1

Fundamentos de programación (PRY2201)

**Facilitador disciplinar:** Josué Oteiza

**Asesor par:** Luis Videla

# Índice

[Introducción a la semana 3](#_Toc158622481)

[Resultado de aprendizaje 4](#_Toc158622482)

[Conceptos relevantes 4](#_Toc158622483)

[Preguntas activadoras 5](#_Toc158622484)

[Actividad 5](#_Toc158622485)

[Mi primer algoritmo 6](#_Toc158622486)

[¿Qué es un algoritmo? 6](#_Toc158622487)

[Pensamiento lógico 7](#_Toc158622488)

[Mi primer algoritmo 10](#_Toc158622489)

[Variables en Java 11](#_Toc158622493)

[Tipos de datos en Java 15](#_Toc158622495)

[Estructuras de control 18](#_Toc158622496)

[Estructura Input-Process-Output 22](#_Toc158622497)

[Links de Interés 23](#_Toc158622498)

[Cierre de la semana 24](#_Toc158622499)

[Referencias 25](#_Toc158622500)

[Lecturas de la semana 25](#_Toc158622501)

[Apuntes 26](#_Toc158622502)

# Introducción a la semana

Te damos la bienvenida a la Semana 1 de la asignatura Fundamentos de Programación. Aquí, te sumergirás en la esencia de la programación al aprender a crear tu primer algoritmo, conociendo un IDE de programación asociada al lenguaje Java. Dentro de estos primeros pasos, está el desarrollo de la habilidad para crear algoritmos efectivos, componente fundamental de tu éxito en esta experiencia de aprendizaje.

En esta guía de aprendizaje, exploraremos dos conceptos fundamentales:

* La identificación de estrategias de abstracción, y
* Los diferentes tipos de algoritmos aplicables en la construcción de soluciones.

La programación no es solo un conjunto de instrucciones escritas en un lenguaje de programación; es un proceso creativo que implica la resolución de problemas de manera sistemática. Por ello, es esencial comprender cómo simplificar problemas complejos y seleccionar los algoritmos adecuados para resolverlos. Este proceso implica la abstracción, que es la capacidad de separar lo esencial de lo no esencial, y la elección de algoritmos, que son las herramientas que te permitirán encontrar soluciones eficientes.

Esta guía de aprendizaje te ayudará a comprender y aplicar estas estrategias de abstracción y tipos de algoritmos en situaciones reales. ¡Prepárate para sumergirte en el emocionante mundo de la programación! ¡Comencemos!

# Resultado de aprendizaje

### El estudiante será capaz de:

**RA1.** Utiliza estrategias de abstracción para la construcción de algoritmo, aplicando pseudocódigo con el objetivo de dar solución a problemáticas planteadas.

### Indicador de logro:

**IL1.** Identifica estrategias de abstracción y tipos de algoritmos aplicables en la construcción de algoritmos.

# Conceptos relevantes

| Algoritmo | Variables | Constantes |
| --- | --- | --- |
| Tipo de datos | Entrada de datos | Salida de datos |
| Estructura secuencial | Estructuras condicionales | Estructuras Iterativas |

# Preguntas activadoras

* ¿Qué es la programación estructurada y cómo se diferencia de otros paradigmas de programación?
* ¿Cuáles son los principios básicos de la programación estructurada y por qué son importantes en el desarrollo de software en Java?
* ¿Qué es una variable y cómo se declara en Java? ¿Por qué son esenciales en la programación estructurada?
* ¿Cuál es la importancia de las estructuras de control en Java, como los bucles y las estructuras condicionales, en la programación estructurada?
* ¿Cómo se definen y utilizan las funciones o métodos en Java para modularizar el código en la programación estructurada?
* ¿Cuál es la diferencia entre programación estructurada y programación orientada a objetos? ¿En qué casos es más apropiada la programación estructurada?

# Actividad

### Descripción de la actividad

En esta primera semana realizarás una actividad formativa en pareja con encargo, llamada "Creando mi primer algoritmo", donde deberás seleccionar una temática con la cual podrás practicar lo aprendido sobre estrategias de abstracción y proponer una solución a través de un algoritmo (entrada – proceso – salida).

# Mi primer algoritmo

Conociendo Java

Java es uno de los lenguajes de programación más populares y versátiles del mundo, utilizado en una amplia gama de aplicaciones, desde desarrollo web hasta aplicaciones móviles y sistemas empresariales. Es ampliamente elogiado por su portabilidad, lo que significa que las aplicaciones desarrolladas en Java pueden ejecutarse en una variedad de plataformas sin necesidad de modificaciones significativas.

Aquí podemos mencionar la estructura de datos en Java, el sistema mediante el cual se organizan los datos en la memoria de la aplicación que se está programando. Dicho de otra manera, es un conjunto de algoritmos que podemos usar en cualquier lenguaje de programación para estructurar los datos en la memoria. ¿Te parece si antes de continuar hablando de Java, conocemos lo que es un algoritmo?

## ¿Qué es un algoritmo?

En el ámbito de la computación, los algoritmos son una herramienta que permite describir claramente un conjunto finito de instrucciones ordenadas secuencialmente y libres de ambigüedad, que debe llevar a cabo un computador para lograr un resultado previsible. (López García, 2009)

Esto nos lleva a entender que, cuando realizamos una actividad en nuestro día, adoptamos una secuencia lógica de pasos que nos lleva al resultado final.

Supongamos que queremos hacer un pastel, donde debes seguir la siguiente secuencia de pasos:

1. Añadir harina según la medida;
2. Seleccionar los huevos;
3. Añadir mantequilla y azúcar a gusto;
4. Añadir la leche;
5. Mezclar todos los ingredientes en la taza;
6. Verter la masa en el molde;
7. Llevar al horno y esperar 40 minutos. Luego, retirar del horno.

Si bien puedes quitar o añadir pasos a la receta del pastel, la secuencia de pasos es la lógica que aplicamos para obtener un buen resultado y, en este caso, un rico pastel. Sin embargo, cuando de programar se trata, el definir los pasos es super importante, puesto que, al desarrollar un programa para ser ejecutado por la computadora, es vital dejar clara la secuencia que se debe seguir. Esta cadena lógica es llamada **Lógica de programación o Pensamiento lógico.**

## Pensamiento lógico

Como ya entendimos previamente, el pensamiento lógico se puede definir como la habilidad para razonar y resolver problemas de manera sistemática, siguiendo una secuencia de pasos o reglas bien definidas. Esta capacidad implica descomponer un problema en partes más pequeñas y abordar cada parte de manera lógica y coherente, con el objetivo de encontrar una solución eficiente y efectiva.

Una definición ampliamente reconocida de pensamiento lógico en el contexto de los algoritmos proviene de Donald E. Knuth, un renombrado informático y matemático. En su libro "The Art of Computer Programming" (El Arte de la Programación de Computadoras), publicada inicialmente en 1968, Knuth explora profundamente los fundamentos matemáticos y algorítmicos de la programación. Aquí, el autor hace hincapié en la importancia de la lógica en la creación de algoritmos eficientes y presenta una perspectiva integral del pensamiento lógico en la programación.

El pensamiento lógico, como lo describe Knuth, implica la capacidad de diseñar algoritmos de manera sistemática y lógica, teniendo en cuenta las restricciones del problema y la eficiencia computacional. Esto incluye las siguientes habilidades:

* Descomposición: romper grandes problemas en pasos más pequeños y fáciles de manejar.
* Abstracción: identificar partes de un problema que hace que una solución funcione para múltiples problemas.
* Patrones: analizar y buscar una secuencia repetitiva.
* Algoritmos: instrucciones paso a paso sobre cómo hacer algo.

**Figura 1**

*Habilidades del pensamiento lógico*

*Nota*. Infografía que refleja las 4 habilidades del pensamiento lógico: Descomposición, Abstracción, Patrones y Algoritmos. Figueroa, P. (s.f.) *¿Qué es el pensamiento computacional?* Red de Matemáticas <https://matematicas.cl/que-es-el-pensamiento-computacional/>

De estas cuatro habilidades, nos centraremos momentáneamente en la abstracción, puesto que existen ciertas estrategias que permiten simplificar la complejidad de un sistema al ocultar detalles innecesarios y centrarse en los aspectos más relevantes. A continuación, te describiremos cada una de ellas:

**1. Abstracción de datos:** se refiere a la capacidad de representar la información de manera simplificada, ocultando los detalles innecesarios y mostrando solo lo esencial para el usuario o programador. Este enfoque ayuda a manejar la complejidad de los programas al dividirlos en módulos o funciones que realizan tareas específicas.

En programación estructurada, la abstracción de datos se logra mediante el uso de Tipos de Datos Abstractos (TDA) y estructuras de control (las cuales veremos más adelante).

**2. Abstracción de control:** se refiere a la simplificación de la lógica de control o flujo de un programa al dividirlo en partes más pequeñas y manejables, como funciones o módulos, para facilitar la comprensión y el mantenimiento.

**Ejemplo:** la creación de funciones en lenguajes de programación, como Python o C++, es una forma común de aplicar la abstracción de control. Cada función representa una tarea específica y puede ser invocada desde diferentes partes del programa, lo que facilita la organización y el mantenimiento del código.

**3. Abstracción de representación de datos**: se trata de definir una estructura de datos que oculta los detalles internos de cómo se almacenan y manipulan los datos, permitiendo interactuar con ellos de manera abstracta.

**Ejemplo:** el uso de una lista en un lenguaje de programación. Los programadores o programadoras pueden agregar, eliminar y acceder a elementos de la lista sin necesidad de conocer cómo se almacenan internamente los datos, ya que la lista proporciona una interfaz abstracta para realizar estas operaciones.

Dicho esto, ya tienes las bases conceptuales básicas para poder comenzar con tu primer algoritmo.

## Mi primer algoritmo

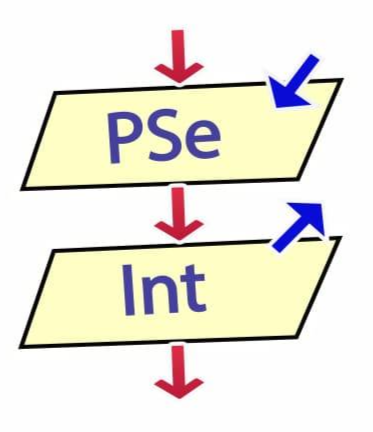
Antes de comenzar en el proceso de la construcción de tu primer algoritmo, queremos que visualices el siguiente caso, donde te mostraremos la creación de un algoritmo en una problemática planteada:

# Video

### Sistema informático para el Teatro Moro:

<https://ava.duoc.cl/bbcswebdav/xid-3025545_1>

Ahora que viste el **caso del Teatro Moro**, ¿qué crees que necesitarías para poner en marcha tu primer proyecto utilizando lenguaje Java? Para comenzar a practicar, lo que haremos será descargar la herramienta PSeInt.

PSeInt es una herramienta gratuita y de código abierto con la que puedes descubrir qué es la lógica de la programación y además, aprender conceptos básicos como las variables o las estructuras de control, fácilmente. Es un excelente punto de partida al momento de escribir algoritmos necesarios para crear programas o aplicaciones. Cuando domines PSeInt, serás capaz de trabajar con lenguajes más complejos (como en nuestro caso, Java).

# Links de Interés

A través del siguiente enlace, podrás ingresar e iniciar su descarga:

<https://pseint.sourceforge.net/index.php?page=descargas.php>

# Videos

Y a través de los siguientes videos, podrás ver cómo realizar la descarga y cómo utilizar la herramienta:

* Descargar PSeInt: <https://youtu.be/8gpgkbpdIcs>
* Utilizar PSeInt: <https://youtu.be/9Mqg1T4lPSg>

Ahora, conozcamos sobre las variables en Java.

## Variables en Java

Las variables en Java son contenedores que se utilizan para almacenar y manipular datos en un programa. Una variable tiene un tipo de dato que define qué tipo de valor puede contener y un nombre que se utiliza para hacer referencia a ella en el código:

**Figura 2**

*Declaración de una variable en Java*

*Nota*. Tipo de dato, nombre y valor de una variable. González, B. (2021). *Variables y tipos de datos en Java: qué son y cómo funcionan.* <https://profile.es/blog/variables-tipos-datos-java/>

En otras palabras, una variable en Java es un elemento fundamental en la programación que se utiliza para almacenar y manipular datos. Básicamente, es un contenedor con un nombre que se usa para representar un valor y un objeto o una referencia a una ubicación en la memoria. Estas variables son esenciales para cualquier programa Java, ya que permiten a los programadores guardar información temporalmente y realizar operaciones en ella.

Cada variable tiene tres componentes clave:

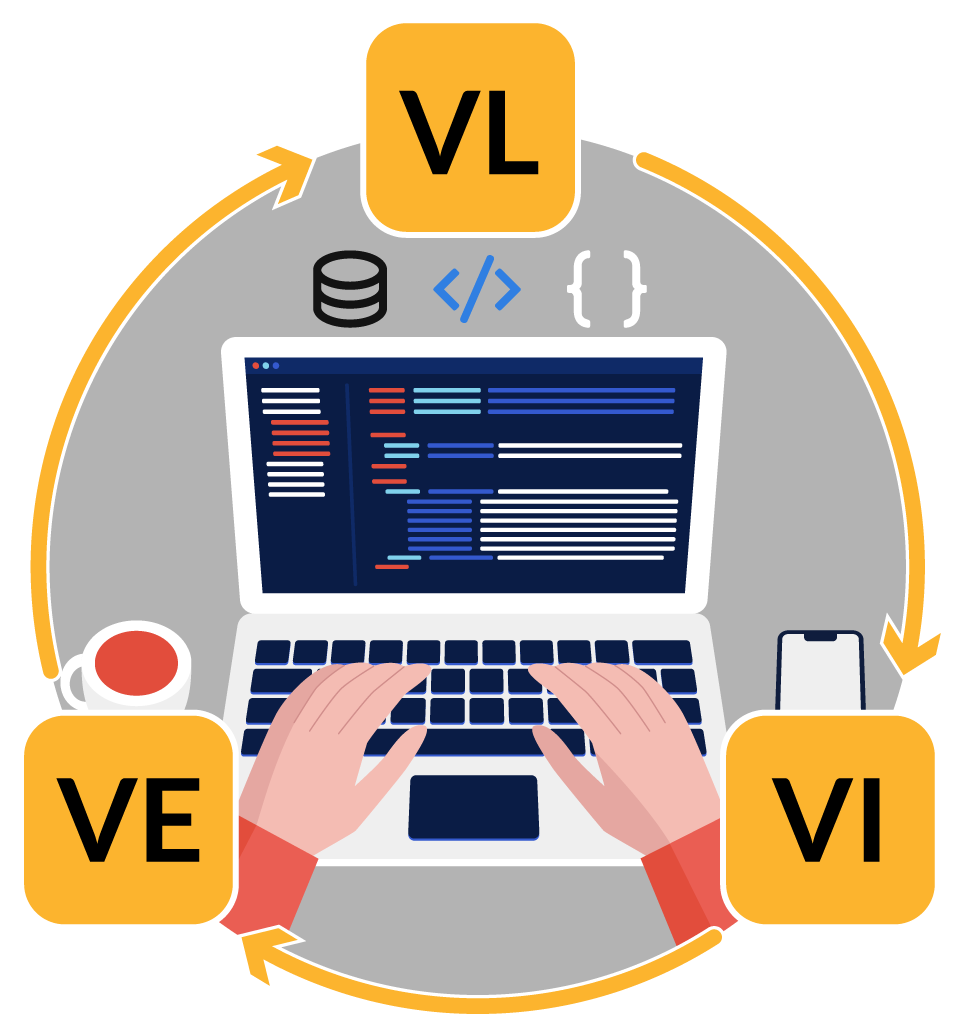
* Tipo de dato: define qué tipo de datos puede almacenar, como números enteros, decimales, texto, objetos, etc.
* Nombre de la variable: es una etiqueta que el programador asigna para referirse a ella en el código.
* Valor de la variable: es la información concreta que contiene y puede cambiar a lo largo de la ejecución del programa.

Estos tres componentes se reflejan de la siguiente manera:

**Figura 3**

*Asignación de valor de una variable*

*Nota.* Tipo de dato, nombre y valor de una variable. González, B. (2021). *Variables y tipos de datos en Java: qué son y cómo funcionan.* <https://profile.es/blog/variables-tipos-datos-java/>

Así también, existen diferentes tipos de variables:

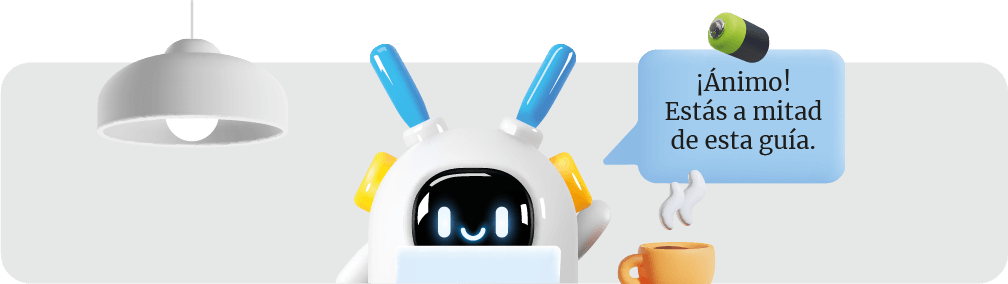
|  |
| --- |
| **Variable Local:** esta es una variable que se declara dentro del cuerpo de un método. |
| **Variable de Instancia:** esta variable de Java se define sin la palabra clave STATIC, si no como fuera de una declaración de método. Son variables específicas de objetos, por eso se conocen con este nombre. |
| **Variable Estática:** esta variable se inicia una sola vez, justo cuando se inicia la ejecución del programa. Es la variable que debe iniciarse primero, especialmente antes de iniciar una variable de instancia. |

Observemos a continuación, cómo se aplica el uso de tipos de variables en Java:

**Figura 4**

*Variables en Java*

*Nota.* Aplicando el uso de los tipos de variables en Java*. PSeInt. (s.f.).* PSeInt. <https://pseint.sourceforge.net/>



Ahora que ya hemos entendido los tipos de variables y definido que una variable almacena un valor, debes saber que una vez que has elegido un tipo de variable, se le necesitará dar un nombre que se mantendrá fijo, independiente de que el valor pueda cambiar a medida que el programa realice una secuencia de operaciones.

Las variables conllevan unas reglas, las cuales son:

**Tabla 1**

*Reglas de variables*

| Identificador | Convención | Ejemplo |
| --- | --- | --- |
| Nombre de variable | Comienza por letra minúscula, y si tienen más de una palabra se colocan juntas y el resto comenzando por mayúsculas | numAlumnos, suma |
| Nombre de constante | En letras mayúsculas, separando las palabras con el guion bajo, por convenio el guion bajo no se utiliza en ningún otro sitio | TAM\_MAX, PI |
| Nombre de una clase | Comienza por letra mayúscula | String, MiTipo |
| Nombre de función | Comienza con letra minúscula | modifica\_Valor, obtiene\_Valor |

# Links de Interés

En el siguiente enlace podrás conocer más sobre las variables. También podrás descargar un manual tutorial Java: <https://www.manualweb.net/java/variables-java/>

## Tipos de datos en Java

Los tipos de datos en Java son fundamentales para la programación, ya que determinan qué tipo de información puede ser almacenada en una variable y cómo se maneja. Java es un lenguaje de programación fuertemente tipado, lo que significa que debes especificar el tipo de dato al declarar una variable, y este tipo no puede cambiar una vez establecido. Aquí, describiremos los tipos de datos más comunes en Java:

### 1. Tipos de datos primitivos:

**Enteros:** incluye **byte, short, int** y **long,** y nos permiten trabajar con números enteros, tanto positivos como negativos (comúnmente se utilizará **int** para números positivos):

**Tabla 2**

*Datos primitivos*

| Nombre | Bytes | Rango |
| --- | --- | --- |
| Long | 8 | –9,223,372,036,854,775,808 a 9,223,372,036,854,775,807 |
| Int | 4 | –2,147,483,648 a 2,147,483,647 |
| Short | 2 | –32,768 to 32,767 |
| byte | 1 | –128 to 127 |

*Ejemplos*:

byte diasMes = 31; //Aproximadamente  
short diasLustro = (12 \* 31) \* 5;  
int velocidadLuz = 299792458;  
long añoLuz = velocidadLuz \* 365;

* **Decimales o Flotantes:** Java proporciona `**float**` y `**double`** para representar números decimales. **`float`** tiene menos precisión que `**double.**

**Tabla 3**

*Datos decimales*

|  |  |  |
| --- | --- | --- |
| Nombre | Bytes | Rango aproximado |
| double | 8 | 4.9e–324 to 1.8e+308 |
| float | 4 | 1.4e–045 to 3.4e+038 |

*Ejemplos*:

float pi = 3.1415926535f;  
double e = 2.718281828459045235360

* **Caracteres:** El tipo de dato `char` almacena un único carácter, como una letra o un símbolo.

*Ejemplos*:

char letraA = 'a';  
char letraANumerico = 61;

* **Boleanos:** el tipo `boolean` tiene solo dos valores posibles: `true` o `false`. Se utiliza para representar estados lógicos.

*Ejemplos*:

boolean verdadero = true;  
boolean falso = false;

### 2. Tipos de datos de referencia:

* **Clases:** los programadores pueden definir sus propias clases para crear tipos de datos personalizados. Estas clases pueden contener atributos y métodos que operan en los objetos creados a partir de ellas.
* **Arreglos:** los arreglos son estructuras que almacenan múltiples elementos del mismo tipo de dato. Por ejemplo, un `int[]` almacena una lista de números enteros.

### 3. Tipos de datos especiales:

* **Tipo de dato `String`:** aunque se parece a una cadena de caracteres, en realidad es una instancia de la clase `String` proporcionada por Java. Se utiliza para almacenar texto.

Ejemplo: `String nombre = "Juan";`

* **Tipos nulos (`null`):** en Java, cualquier referencia a un objeto puede ser nula, lo que indica que la variable no apunta a ningún objeto en particular. Se utiliza para representar la ausencia de valor.

Importante

La elección del tipo de dato adecuado es crucial, ya que afecta el uso de la memoria, el rendimiento y la precisión de los cálculos en tu programa. Utilizar el tipo de dato incorrecto puede llevar a errores de compilación o a resultados inesperados en tiempo de ejecución.

# Estructuras de control

Las estructuras de control en Java son elementos fundamentales que permiten a los programadores dirigir el flujo de ejecución de un programa. Estas estructuras determinan qué instrucciones se ejecutan, cuántas veces y en qué condiciones.

Java proporciona tres tipos principales de estructuras de control:

1. Estructuras de control condicional
2. Estructuras de control repetitiva, y
3. Estructuras de control de instrucciones de salto.

Conozcámoslas a continuación:

### 1. Estructuras de control condicional:

* If-else: permite tomar decisiones basadas en una condición. Si la condición es verdadera, se ejecuta un bloque de código; si es falsa, se ejecuta otro.
* Switch: permite tomar decisiones múltiples basadas en el valor de una expresión. Es útil cuando se necesita seleccionar una opción de entre varias posibles.

### 2. Estructuras de control repetitiva:

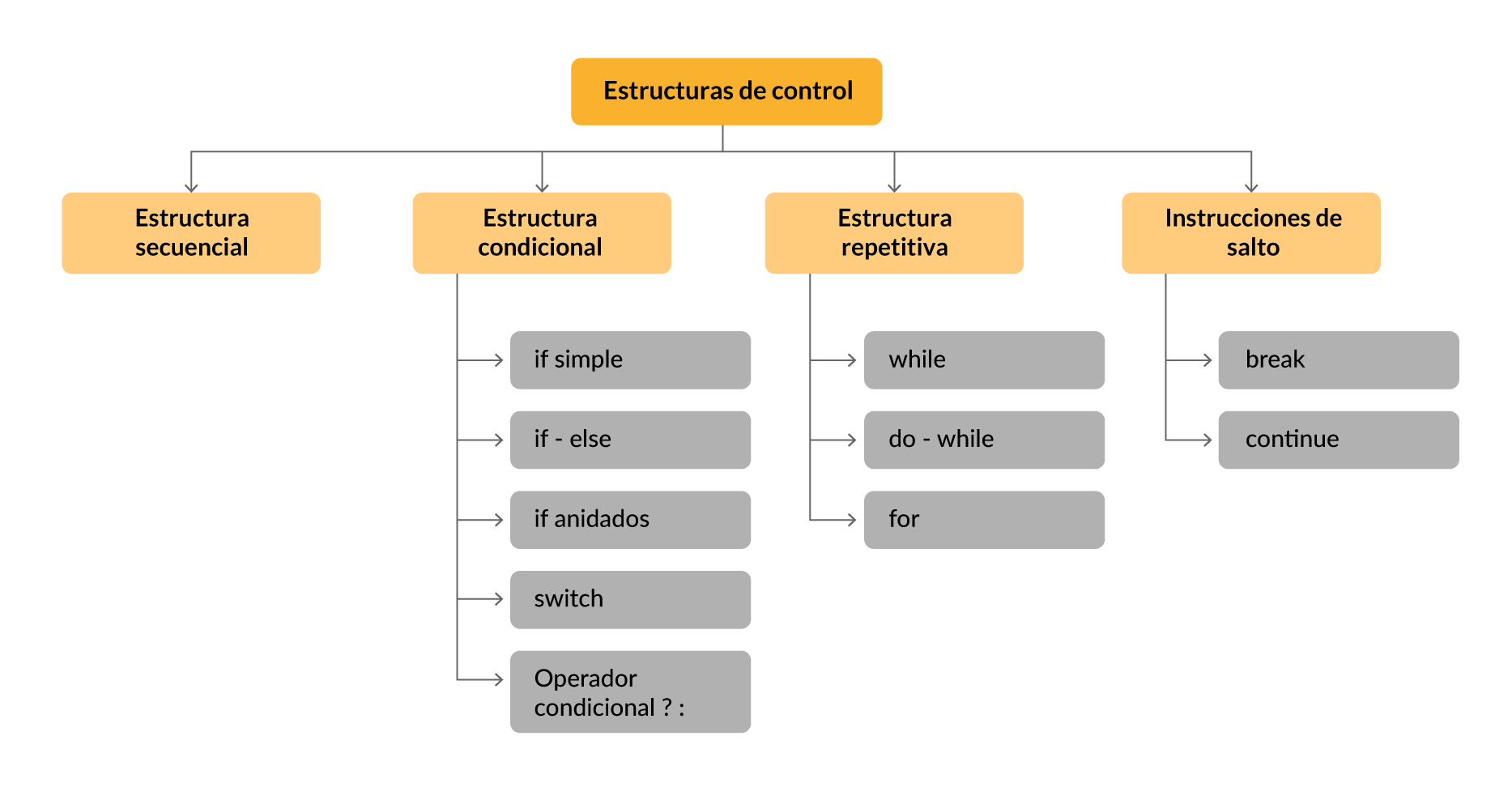
* For: bucle que permite ejecutar un bloque de código un número específico de veces. Se utiliza cuando se conoce de antemano el número de iteraciones.
* While: bucle que se utiliza cuando no se conoce el número exacto de iteraciones y se ejecuta mientras una condición sea verdadera.
* Do-while: similar al while, pero garantiza que el bloque de código se ejecute al menos una vez antes de verificar la condición.

### 3. Estructuras de control de instrucciones de salto:

* Break: sentencia que se utiliza para salir de un bucle (for, while o do-while) antes de que se complete su número normal de iteraciones.
* Continue: sentencia que se usa para saltar una iteración específica en un bucle y continuar con la siguiente iteración.

Veamos, a través de la siguiente imagen, todo lo que acabamos de mencionar referente a las estructuras de control del lenguaje Java:

**Figura 5**

*Estructuras de control en Java*

*Nota.* Todas las estructuras de control del lenguaje Java. García, E. (2020). *Estructuras de control en Java.* Programación Java*.* <https://puntocomnoesunlenguaje.blogspot.com/2012/04/estructuras-de-control.html>

Como te habrás dado cuenta, nos faltó detallar lo que es la estructura secuencial. Veámosla a continuación:

### Estructura Secuencial

En la programación, las instrucciones de un programa se ejecutan de manera predeterminada en una secuencia ordenada. En otras palabras, las instrucciones se llevan a cabo una tras otra, siguiendo el orden en que están escritas dentro del programa.

### Ejemplo de programa Java con estructura secuencial:

Un programa que solicita al usuario ingresar dos números mediante el teclado y luego los muestra en la pantalla. En este ejemplo, todas las instrucciones se encuentran dentro de un único bloque delimitado por las llaves de apertura y cierre del método (main). La ejecución de las instrucciones sigue el orden en el que están escritas:

**Figura 6**

*Programa Java con estructura secuencial*

*Nota.* Ejemplo de estructura secuencial. García, E. (2020). *Estructuras de control en Java.* Programación Java. <https://puntocomnoesunlenguaje.blogspot.com/2012/04/estructuras-de-control.html>

Ahora que revisamos cada elemento que nos permitirá crear un algoritmo propio, es importante señalar lo siguiente sobre en término “solución”, a través de:

## Estructura Input-Process-Output

En el contexto de la programación en Java y en muchos otros lenguajes de programación, el término "solución" se refiere a la resolución de un problema o tarea específica mediante un programa informático. Esta resolución suele seguir una estructura común conocida como Input-Process-Output (en español, Entrada-Proceso-Salida), que se utiliza para describir cómo se aborda un problema de programación. Veamos cada uno de ellos:

|  |  |
| --- | --- |
|  | Input (Entrada): se refiere a los datos o información que el programa recibe como entrada para resolver un problema. Estos datos pueden provenir de diversas fuentes, como el teclado, un archivo, una base de datos o cualquier otro dispositivo o fuente de datos. En Java, puedes utilizar clases como `Scanner` o `BufferedReader` para obtener datos de entrada del usuario o de un archivo. |
| Process (Proceso): se refiere a la lógica y las operaciones que el programa realiza sobre los datos de entrada para resolver el problema o realizar una tarea específica. Esta etapa implica la manipulación, el cálculo o cualquier otro tipo de procesamiento necesario para llegar a una solución. En Java, este proceso se implementa mediante el código que escribes dentro de tus clases y métodos. |
| Output (Salida): se refiere a los resultados o la información generada por el programa después de procesar los datos de entrada. Puede ser un resultado calculado, un mensaje para el usuario, un archivo generado o cualquier otro tipo de información que el programa debe proporcionar como resultado. En Java, puedes utilizar `System.out.println()` u otras operaciones de salida para mostrar información en la consola o escribir en archivos. |

# Links de Interés

Finalmente, para adentrarte mucho más en detalle en las estructuras de control en Java, te recomendamos visitar el siguiente enlace:

<https://puntocomnoesunlenguaje.blogspot.com/2012/04/estructuras-de-control.html>

Si, por otro lado, te interesa obtener alguna certificación en Java, te recomendamos el siguiente enlace, donde podrás ver las diferentes vías de certificación que ofrece Oracle University:

<https://education.oracle.com/oracle-certification-path/pFamily_48>

# Cierre de la semana

Java, un lenguaje de programación ampliamente utilizado, destaca por su versatilidad y portabilidad. Las aplicaciones desarrolladas en Java pueden ejecutarse en diversas plataformas sin modificaciones significativas. La estructura de datos en Java organiza la memoria de la aplicación mediante algoritmos, esenciales para cualquier programador.

El pensamiento lógico, crucial en programación, implica razonar sistemáticamente para resolver problemas. Donald E. Knuth destaca su importancia en la creación de algoritmos eficientes.

Los tipos de datos en Java incluyen primitivos (enteros) y de referencia (clases, arreglos). Las estructuras de control dirigen el flujo del programa, incluyendo condicionales (if-else, switch) y bucles (for, while, do-while). Java ofrece instrucciones de salto como break y continue.

En resumen, Java destaca por su versatilidad y portabilidad, y su uso implica comprender algoritmos, pensamiento lógico, variables, tipos de datos y estructuras de control.

# Referencias

López García, J. (2009). *Algoritmos y programación: guía para docentes.* [ebook]. <https://libros.metabiblioteca.org/server/api/core/bitstreams/a567dd25-1e96-4c0f-9b6a-7a844d0eb577/content>

# Lecturas de la semana

* Capítulo 1: All about Java
* Capítulo 2: All about Software

Burd, B. (2022). Java for Dummies. New Jersey: John Wiley & Sons. <https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50163-Java_fordummies/18/> Páginas 7 a la 32

* Capítulo 1: Introducción a Java

Vegas Gertrudix, J. M. (2022). Java 17: Fundamentos prácticos de programación. Bogotá: Ediciones de la U. <https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50229-Java_17/11/> Páginas 32 a la 40

# Apuntes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.