

Guía de aprendizaje

Problemas algorítmicos con expresiones aritméticas y lógicas

Exp 1 – Semana 3

Fundamentos de Programación (PRY2201)

**Facilitador disciplinar:** Josué Oteiza

**Asesor par:** Luis Videla

# Índice

[Introducción a la semana 3](#_Toc152772606)

[Resultado de aprendizaje 4](#_Toc152772607)

[Conceptos relevantes 4](#_Toc152772608)

[Preguntas activadoras 5](#_Toc152772609)

[Actividad 5](#_Toc152772610)

[Expresiones aritméticas y expresiones lógicas 6](#_Toc152772611)

[Componentes esenciales en la creación de un algoritmo 6](#_Toc152772612)

[Expresiones aritméticas 7](#_Toc152772613)

[Implementando Expresiones Lógicas 13](#_Toc152772614)

[Uso conjunto de expresiones aritméticas y expresiones lógicas 21](#_Toc152772615)

[Errores comunes y buenas prácticas 23](#_Toc152772616)

[Comentarios en Java 24](#_Toc152772617)

[Convenciones 25](#_Toc152772618)

[¿Dónde comenzar a programar en Java? 26](#_Toc152772619)

[JDK 26](#_Toc152772620)

[Apache NetBeans 27](#_Toc152772623)

[Ejercicios resueltos 31](#_Toc152772626)

[Cierre de la semana 34](#_Toc152772627)

[Referencias 35](#_Toc152772628)

[Lecturas de la semana 35](#_Toc152772629)

[Apuntes 36](#_Toc152772630)

# Introducción a la semana

En el desarrollo de algoritmos, la inclusión de expresiones aritméticas y lógicas marca un punto de inflexión donde la abstracción se encuentra con la precisión. Estos elementos no son meras formalidades, sino herramientas concretas que dan estructura y funcionalidad a nuestras creaciones. A medida que avanzamos en nuestro aprendizaje, nos sumergiremos en la integración de expresiones matemáticas y lógicas en nuestro código, entendiendo cómo estas operaciones fundamentales se convierten en los cimientos sobre los cuales se erige la lógica y la toma de decisiones en la programación. Así, exploraremos la utilidad pragmática de estas expresiones, aportando una dimensión más precisa y efectiva a nuestros algoritmos en desarrollo.

# 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.

**IL2.**  Aplica Pseudocódigo en la implementación de algoritmos, brindando una solución a las problemáticas planteadas.

**IL3.** Utiliza Variables, Expresiones Aritméticas y Lógicas en la Resolución de Problemas Algorítmicos, garantizando la exactitud y coherencia de sus soluciones.

# Conceptos relevantes

|  |  |  |
| --- | --- | --- |
| Operadores aritméticos | Operadores lógicos | Expresiones aritméticas |
| Comentarios | convenciones |  |

# Preguntas activadoras

* ¿Cómo influyen las variables y expresiones en la resolución de problemas algorítmicos, y cuáles podrían ser los desafíos comunes asociados con su manipulación?
* ¿Cuáles son las consideraciones clave al trabajar con variables y expresiones en algoritmos, y cómo estas pueden afectar la eficiencia y precisión de la solución algorítmica?
* ¿Puedes identificar ejemplos específicos de problemas algorítmicos donde la correcta manipulación de variables y expresiones es fundamental, y cómo abordarías esos desafíos de manera efectiva?

# Actividad

**Descripción de la actividad**

En esta tercera semana realizarás una actividad sumativa individual, llamada "Resolviendo Problemas algorítmicos con Expresiones aritméticas y lógicas”, donde deberás crear un tipo de algoritmo y representarlo en pseudocódigo para dar solución a la problemática planteada. Además, a tu solución se le deben aplicar variables y expresiones aritméticas y lógicas, garantizando exactitud y coherencia.

# Expresiones aritméticas y expresiones lógicas

## Componentes esenciales en la creación de un algoritmo

La creación de algoritmos efectivos en Java, un lenguaje de programación versátil y potente es una habilidad fundamental para cualquier desarrollador o desarrolladora. Un aspecto clave de la programación en Java es la correcta implementación de expresiones aritméticas y lógicas, las cuales forman la base de la toma de decisiones y el cálculo en cualquier programa. En otras palabras, permiten realizar cálculos y evaluar condiciones.

**Figura 1**

*Operación aritmética*

*Nota:* Ejemplo en Java de operadores aritméticos. *Tutorias.co. (s.f.). Programación en Java.* <https://tutorias.co/operadores-java-operacion-aritmetica-paso-a-paso/>

A continuación, conocerás la descripción y ejemplos de cada una:

## Expresiones aritméticas

Son operaciones o combinaciones de variables que involucran números y operadores aritméticos, es decir, que permiten realizar cálculos y evaluar condiciones. En lenguajes como Python y Java se realizaría de la siguiente manera:

Phyton: ‘resultado = a + b \* c);’

Java: ‘int resultado = a + (b \* c);’

En lo que Java corresponde, esta herramienta soporta **operadores aritméticos comunes,** que son:

**Tabla 1**

*Operadores aritméticos*

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Descripción | Ejemplo de expresión | Resultado del ejemplo |
| + | Suma | 2.5 + 7.1 | 9.6 |
| - | Resta | 235.6 – 103.5 | 132.1 |
| \* | Multiplicación o Producto | 1.2 \* 1.1 | 1.32 |
| / | División (tanto entera como real) | 0.050 / 0.2  7 / 2 | 0.25  3 |
| % | Módulo  Resto de la división entera | 20 % 7 | 6 |

Conozcamos a continuación algunos ejemplos sobre cada uno de ellos:

**Figura 2**

*Operador aritmético ‘Suma’*

*Nota.* Ejemplo de expresión aritmética: ‘Suma’. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Una expresión aritmética de suma en Java, como la mostrada arriba, combina dos operandos numéricos (‘numero1’ y ‘numero2’) mediante el operador ‘+’ para realizar la operación de suma. La variable ‘resultado’ almacenará el valor resultante de sumar ‘numero1’ y ‘numero2’. En este contexto, la expresión aritmética representa la **adición de dos valores numéricos.**

**Figura 3**

*Operador aritmético ‘Resta'*

*Nota.* Ejemplo de expresión aritmética: ‘Resta’. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este caso, la variable ‘resultado’ almacenará el valor resultante de restar ‘numero2’ de ‘numero1’.

**Figura 4**

*Operador aritmético ‘Multiplicación’*

*Nota.* Ejemplo de expresión aritmética: ‘Multiplicación’. The Apache Software Foundation. (2023)*.* *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, la expresión aritmética representa la operación de multiplicación entre dos variables numéricos.

**Figura 5**

*Operador aritmético ‘División’*

*Nota.* Ejemplo de expresión aritmética: ‘División’*.* The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Como muestra la imagen, mediante el operador ‘/’, se realiza la operación de división entre dos valores numéricos.

**Figura 6**

*Operador aritmético ‘Módulo’ (resto)*

*Nota.* Ejemplo de expresión aritmética: ‘Módulo’ (resto). The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En la imagen recién vista, el operador ‘%’ (módulo), representa el residuo de la división entre dos valores numéricos. Por ejemplo, si ‘numero1’ es 7 y ‘numero2’ es 3, entonces ‘resultado’ será 1, ya que 7 dividido por 3 tiene un residuo de 1.

Una buena demostración de la implementación de las operaciones recién vistas sería el usar la clase ‘CalculadorDeNotas’, como se muestra a continuación:

**Figura 7**

*Operaciones aritméticas*

*Nota.* Ejemplo de la utilización de operaciones aritméticas. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, se asume que hay dos exámenes y un proyecto, y se calcula la nota total, el promedio simple, el promedio ponderado según pesos dados y la diferencia respecto a la máxima nota posible.

Importante

Recuerda que todos los nombres de las clases con mayúscula y la primera letra de cada palabra interna en el nombre de la clase, debe ir en mayúscula.

**Ejemplo**: ClienteNacional.

Las clases representan “cosas” y no “acciones”, por tal motivo hay que evitar verbos como nombre de clase. El nombre de la clase debe estar en singular y ser un sustantivo.

## Implementando Expresiones Lógicas

En lo que respecta a las expresiones lógicas, estas son combinaciones de variables, constantes y operadores lógicos que se utilizan para evaluar condiciones y producir un resultado booleano (verdadero o falso). En lenguaje Java sería:

‘boolean condicion = (x > y) && (z == 10);’

Veamos los diferentes operadores de esta categoría a través de la siguiente tabla:

**Tabla 2**

*Operadores lógicos más comunes*

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Descripción | Ejemplo de expresión | Resultado del ejemplo |
| ! | Negociación – NOT (unario) | *! false*  *! (5==5)* | *true*  *false* |
| | | Suma lógica – OR (binario) | *true | false*  *(5==5) | (5<4)* | *true*  *true* |
| ^ | Suma lógica exclusiva – XOR (binario) | *true ^ false*  *(5==5) | (5<4)* | *true*  *true* |
| & | Producto lógico – AND (binario) | *true & false*  *(5==5) & (5<4)* | *false*  *false* |
| | | | Suma lógica con cortocircuito: si el primer operando | *true | | false*  *(5==5) & (5<4)* | *true*  *true* |
| && | Producto lógico con cortocircuito: si el primer operando es *false*, entonces el segundo se salta y el resultado es *false* | *false && true*  *(5==5) && (5<4)* | *false*  *false* |

En las expresiones lógicas se utilizan dos tipos de operadores: **operadores lógicos** y **operadores relacionales.** Conozcamos más sobre ellos:

**Tabla 3**

*Operadores lógicos*

|  |  |  |
| --- | --- | --- |
| Operadores lógicos  Actúan exclusivamente sobre valores de expresiones lógicas. | Tipo | Ejemplo |
| AND lógico (‘&&’)  Retorna a verdadero si ambas condiciones son verdaderas. | ‘(a > b) && (c == 0)’  Evalúa a verdadero si ‘a’ es mayor que ‘b’ y ‘c’ es igual a cero. |
| OR lógico (‘||’)  Retorna verdadero si al menos una de las condiciones es verdadera. | ‘(x < 0) || (y < 0)’  Evalúa a verdadero si ‘x’ es menor que cero o ‘y’ es menor que cero. |
| NOT lógico (‘!’)  Invierte el valor de la condición. Si la condición es verdadera, NOT la hace falsa, y viceversa. | ‘!(a == b)’  Evalúa a verdadero si ‘a’ no es igual a ‘b’. |

Estas expresiones retornan un valor booleano (`true` o `false`) y son comúnmente usadas en estructuras de control como if-else y bucles.

En el ejemplo que se presenta abajo, el programa solicita al usuario que ingrese dos números, realice operaciones aritméticas básicas (suma, resta, multiplicación, división y módulo) y luego muestre los resultados. Además, se incluyen comprobaciones para evitar la división por cero y el cálculo del módulo cuando el divisor es cero:

**Figura 8**

*Operadores lógicos*

*Nota.* Ejemplo de operadores lógicos. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

**Tabla 4**

*Operadores relacionales*

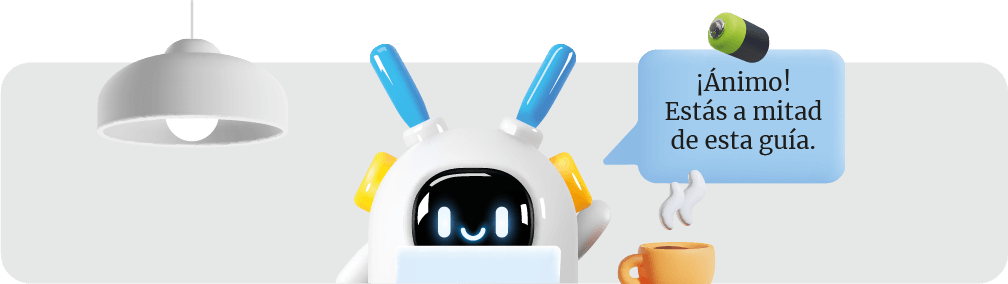
|  |  |  |
| --- | --- | --- |
| **Operadores relacionales**  Se utilizan para comparar los valores de dos expresiones. Deben ser del mismo tipo (aritméticas, lógicas, de carácter o de cadena). | Tipo | Ejemplo |
| Igual a (‘==’)  Compara si dos valores son iguales. | ‘a == b’ e  Evalúa a verdadero si ‘a’ es igual a ‘b’. |
| No igual a (‘!=’)  Compara si dos valores no son iguales. | ‘a != b’  Evalúa a verdadero si ‘a’ no es igual a ‘b’. |
| Mayor que (‘>’)  Compara si un valor es mayor que otro. | ‘a > b’  Evalúa a verdadero si ‘a’ es mayor que ‘b’. |
| Mayor o igual que (‘>=’)  Compara si un valor es mayor o igual a otro. | ‘a >= b’  Evalúa a verdadero si ‘a’ es mayor o igual a ‘b’. |
| Menor o igual que (‘<=’)  Compara si un valor es menor o igual a otro. | ‘a <= b’  Evalúa a verdadero si ‘a’ es menor o igual a ‘b’. |

En el siguiente ejemplo te mostramos, a partir de la clase ‘Estudiante’, la utilización de tres operadores relacionales para comparar las edades de dos estudiantes:

**Figura 9**

*Operadores relacionales en Java*

*Nota.* Ejemplo de operadores relacionales. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este ejemplo, la clase ‘Estudiante’ tiene dos atributos: ‘nombre’ y ‘edad’. Se crean dos instancias de la clase ‘Estudiante’ (estudiante1 y estudiante2), y luego se utilizan tres operadores relacionales para comparar las edades de los estudiantes. Estos son:

1. Operador de igualdad (‘==’):

‘boolean sonIguales = (estudiante1.edad == estudiante2.edad);’

Se utiliza para verificar si la edad de ‘estudiante1’ es igual a la edad de ‘estudiante2’.

1. **Operador mayor que (`>'):**

Se utiliza para verificar si la edad de ‘estudiante1’ es mayor que la edad de ‘estudiante2’.

‘boolean esMayorQue = (estudiante1.edad > estudiante2.edad);’

1. Operador menor o igual que (‘<=’):

Se utiliza para verificar si la edad de ‘estudiante1’ es menor o igual a la edad de ‘estudiante2’.

‘boolean esMenorOIgualQue = (estudiante1.edad <= estudiante2.edad);’

Estos operadores relacionales se aplican a los valores numéricos de las edades en este caso específico. Los resultados de estas comparaciones son almacenados en variables booleanas (‘sonIguales’, ‘esMayorQue’, y ‘esMenorOIgualQue’) y se imprimen en la consola para indicar el resultado de cada comparación.

Importante

Hasta ahora, hemos revisado las expresiones aritméticas y expresiones lógicas por separado, sin embargo, es relevante señalar que, en la programación real, a menudo necesitas combinarlas. Esto es especialmente común en bucles y condiciones donde se requiere evaluar y manipular datos numéricos.

# Uso conjunto de expresiones aritméticas y expresiones lógicas

Las expresiones aritméticas y lógicas pueden aplicarse a una clase para realizar cálculos y tomar decisiones basadas en condiciones lógicas, tal como se muestra a continuación:

**Figura 10**

*Expresiones aritméticas y lógicas en Java*

*Nota.* Ejemplo de combinación de expresiones aritméticas y lógicas. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Este ejemplo muestra cómo una expresión aritmética (`edad >= 18`) se combina con una estructura de control lógica (if-else) para ejecutar un fragmento de código basado en una condición.

De la siguiente forma, se puede observar la ejecución de un programa estructurado en Java:

**Figura 11**

*Expresiones aritméticas y lógicas en Java*

*Nota*. Ejemplo de ejecución de un programa estructurado en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En este otro ejemplo, se ve el uso de las expresiones aritméticas y lógicas. En un programa estructurado en Java son componentes fundamentales que permiten realizar cálculos y tomar decisiones dentro del código.

Los operadores aritméticos y lógicos son los componentes fundamentales (símbolos) que se utilizan dentro de las expresiones aritméticas y lógicas para realizar cálculos y evaluar condiciones, respectivamente. Las expresiones son combinaciones más complejas que incluyen estos operadores junto con variables y valores.

# Errores comunes y buenas prácticas

Es fácil cometer errores al trabajar con estas expresiones, especialmente cuando son complejas. Algunos errores comunes incluyen:

* La mala precedencia de operadores,
* La división por cero, y
* La confusión entre operadores de igualdad (`==`) y asignación (`=`).

Para evitarlos, es esencial comprender bien estos conceptos y practicar la escritura de código limpio y bien documentado. Veamos algunos ejemplos:

**Ejemplo:**

1. Mala precedencia de operadores:

int result = 5 + 2 \* 3;

// Resultado esperado: 11, pero el resultado real será 17 (se evalúa la multiplicación antes de la suma)

1. División por cero:

int numerator = 10;

int denominator = 0;

int result = numerator / denominator; // Esto lanzará una ArithmeticException en tiempo de ejecución

1. Confusión entre operadores de igualdad y asignación:

int x = 5;

if (x = 10) {

// Este bloque de código siempre se ejecutará incorrectamente, ya que se está utilizando el operador de asignación en lugar del operador de igualdad

Para evitar otros errores, te proponemos a continuación las siguientes buenas prácticas a realizar dentro de tu proyecto de software:

## Comentarios en Java

Los comentarios en Java es una práctica que se utiliza para dejar documentación asociada a cada línea de código, y tiene como función documentar los programas y mejorar su legibilidad. El compilador de Java ignora los comentarios cuando el programa se ejecuta. Existen tres tipos:

* **‘Javadoc’:** prepara la documentación de su programa, en formato HTML, es decir, todo lo que esté en este tipo de comentario aparecerá en la documentación. Los comentarios de 1 y varias líneas, son internas para el programador.
* **De 1 línea’:** es un cometario para una línea dentro de la programación.
* **De varias líneas’:** son cometarios para varias líneas de programación.

## Convenciones

Otra práctica recomendada -y establecida por la comunidad de desarrolladores- es utilizar convenciones de codificación en Java. Estas son pautas y reglas que las y los desarrolladores siguen al escribir código en este lenguaje de programación para mejorar la legibilidad y la consistencia del código. Algunas de las razones por las que se generan y siguen estas convenciones son:

* **Legibilidad del código:** las convenciones de codificación están diseñadas para hacer que el código sea más fácil de leer y entender. Esto es fundamental para la colaboración en equipos de desarrollo y facilita el mantenimiento del código a lo largo del tiempo.
* **Consistencia:** las convenciones promueven la consistencia en la forma en que se escribe el código. Cuando todos los miembros de un equipo siguen las mismas reglas, el código tiende a tener un estilo uniforme, lo que facilita la comprensión y el mantenimiento.
* **Facilita la revisión de código:** al seguir convenciones comunes, los revisores de código pueden identificar más fácilmente posibles errores, problemas de estilo o malentendidos. Esto agiliza el proceso de revisión y mejora la calidad del código.

# ¿Dónde comenzar a programar en Java?

Si pudiste darte cuenta, todo lo que has aprendido ha estado relacionado con crear tu proyecto en Java, sin embargo, hasta el momento solo hemos utilizado la herramienta PSeInt para la creación de algoritmos. Pues bien, ya es tiempo de que pongas manos a la obra y descargues las siguientes herramientas:

La primera a instalar será Java Development Kit o JDK.

## Logotipo, Java Development KitJDK

JDK es un entorno de Desarrollo para crear aplicaciones utilizando el lenguaje de programación Java y que incluye herramientas para desarrollar y probar programas escritos en lenguaje de programación Java.

# Links de Interés

Presiona sobre el siguiente enlace para obtener la descarga del Kit de Desarrollo Java (para Windows, la recomendación es descargar el x64 installer:

<https://www.oracle.com/cl/java/technologies/downloads/>

# Videos

Puedes revisar el siguiente video para seguir los pasos de instalación y configuración de JDK: <https://www.youtube.com/watch?v=81WQ7oSwIgU&t=33s>

Una vez instalado, ya podemos realizar la descarga del entorno de desarrollo llamado Apache NetBeans.

## Logotipo: Apache NetBeans IDEApache NetBeans

Este es uno de los programas más simples en Java y es a menudo el primer programa que los principiantes escriben al aprender un nuevo lenguaje de programación.

# Links de Interés

Puedes descargar Apache NetBeans de forma completamente gratuita a través del siguiente enlace, donde si bien tienes a disposición varias versiones, te recomendamos que descargues la versión 18, por ser la más estable hasta el momento:

<https://netbeans.apache.org/front/main/download/archive/>

# Videos

También puedes mirar el siguiente video para realizar su instalación:

<https://www.youtube.com/watch?v=dVaZg_ZTxHA>

Cuando ejecutes Apache NetBeans, verás la frase “Hola, mundo” impresa en la consola, tal como se muestra en la imagen:

**Figura 12**

*(“Hola, mundo”)*

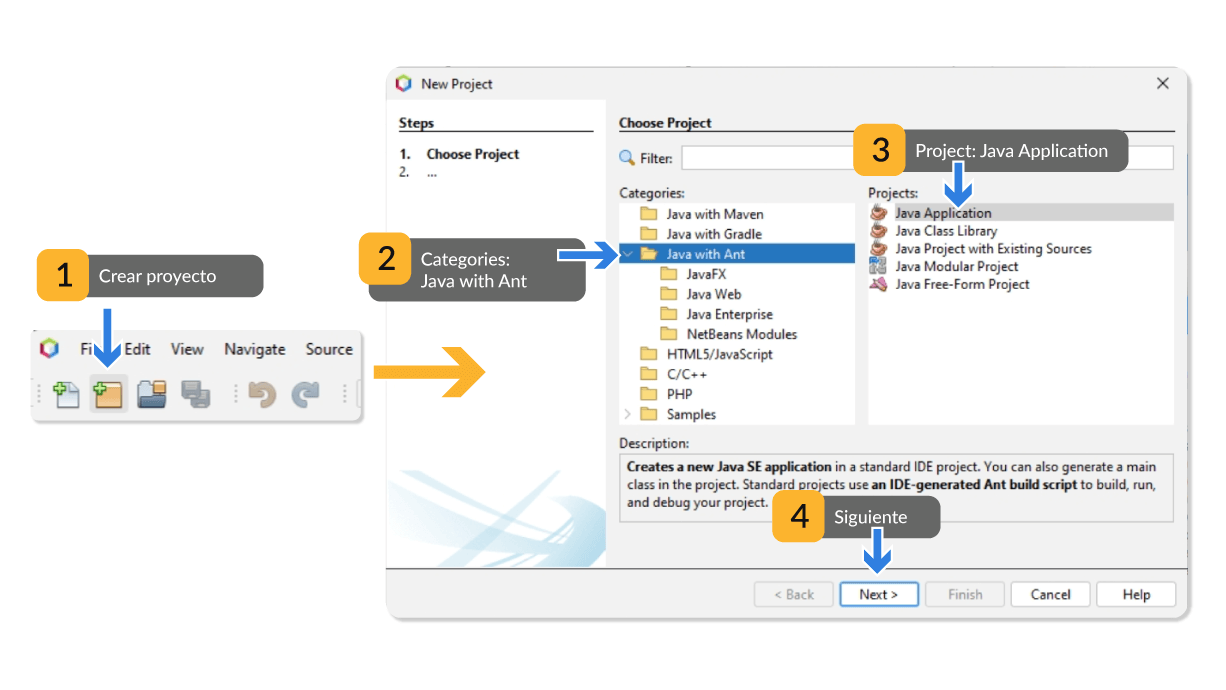
Nota. Proceso inicial en Java. The Apache Foundation. (2023). Apache NetBeans (19). [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

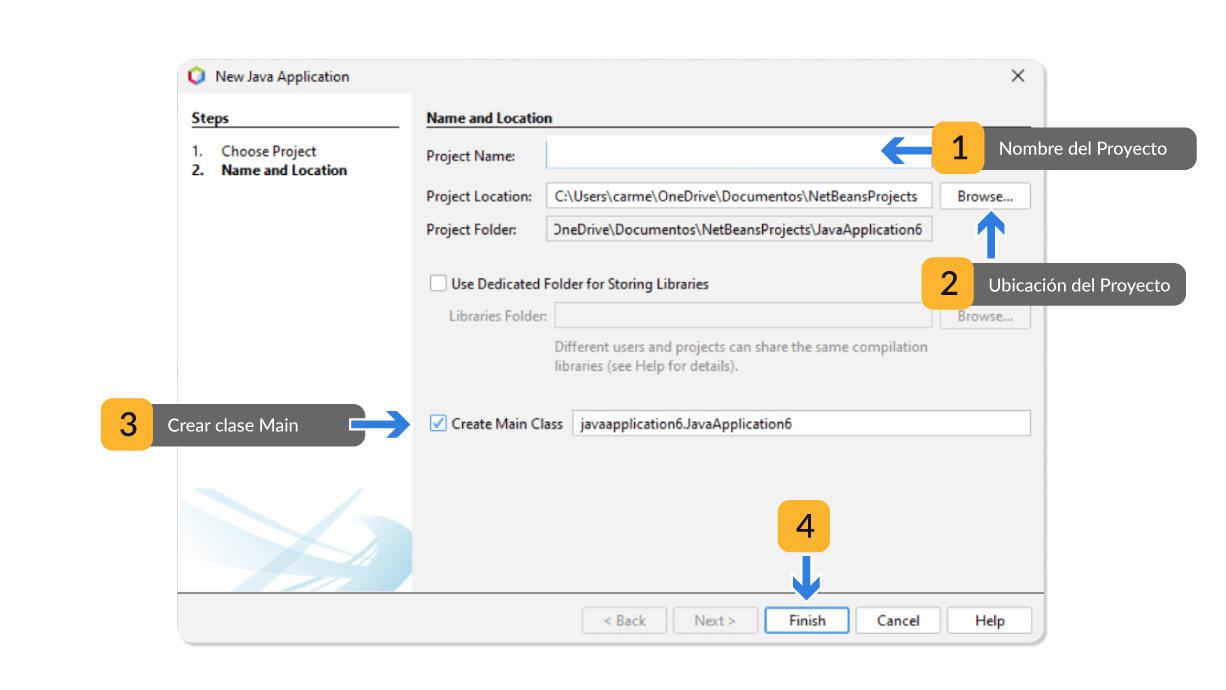
Este programa consta de una sola clase llamada ‘Saludo’ con un método principal llamado ‘main’. Dentro del método ‘main’. Utilizamos ‘System.out.println’ para imprimir el mensaje de la consola.

Importante

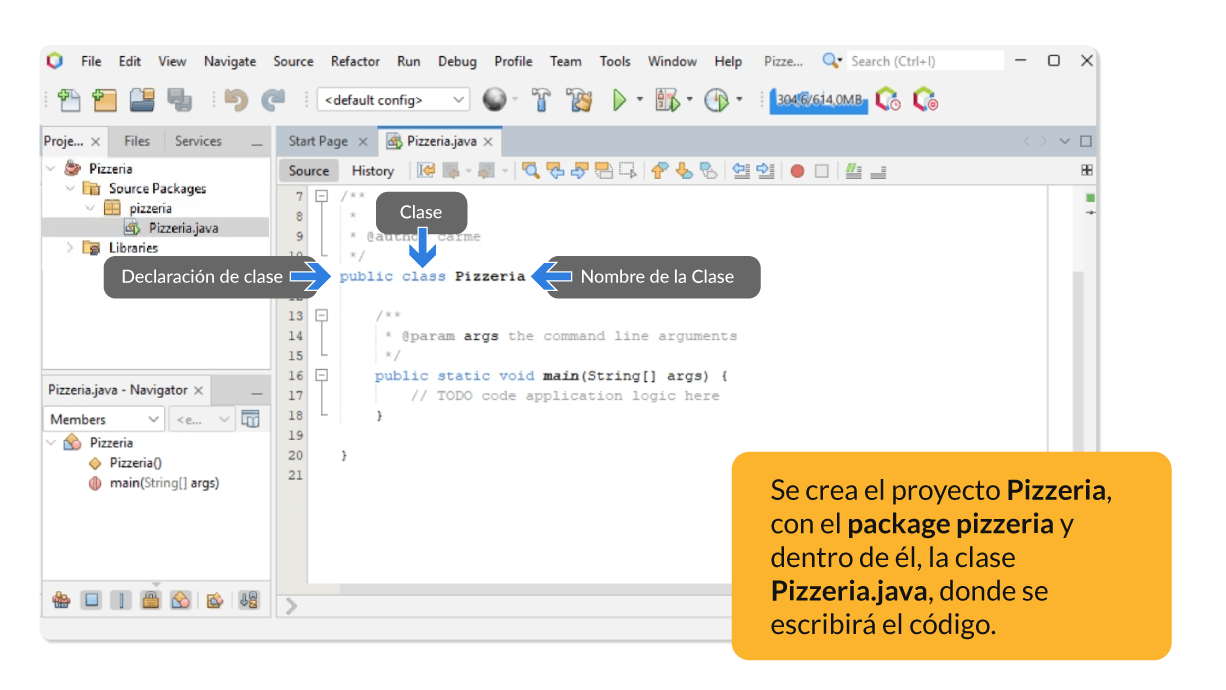
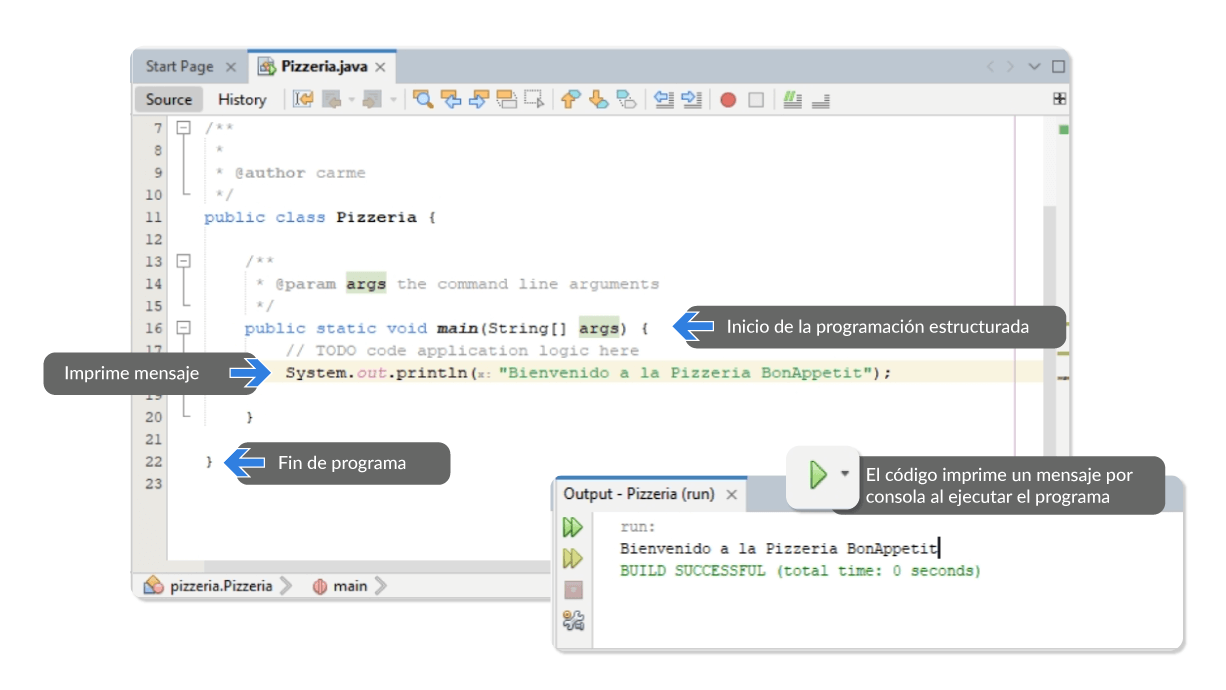
Puedes copiar este código en un archivo Java (asegúrate de que tenga la extensión .java, como por ejemplo, ‘Salud.java’), compilarlo y ejecutarlo para ver el resultado. Es un excelente punto de partida para familiarizarte con la estructura básica de un programa Java.

Ahora, veamos un paso a paso de cómo crear tu primer programa en Java NetBeans:

**Paso 1:** Abrir IDE NetBeans y proceder a crear el primer programa en Java.**Paso 2:** crear el proyecto agregando el nombre.



**Paso 3:** una vez que el proyecto está creado, aparecerá la siguiente pantalla:

**Paso 4:** ahora podrás escribir tu primer código. Para finalizar y afianzar tu comprensión, aquí hay algunos ejercicios resueltos donde se aplicaron expresiones aritméticas y lógicas.

# Ejercicios resueltos

1. Se muestra un programa simple que calcula el área de un círculo, dado su radio, utilizando la fórmula ‘área = π \* radio^2’.

**Figura 13**

*Cálculo de área de un círculo*

*Nota*. Ejemplo de cálculo del área de un círculo en Java. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

En el ejercicio proporcionado, se utilizaron las siguientes expresiones aritméticas:

1. **Cálculo del área:** la expresión aritmética utilizada para calcular el área del círculo es:

double area = Math.PI \* Math.pow(radio, 2);

* ‘Math.PI’: Representa el valor de π (pi).
* ‘Math.pow(radio, 2)’: eleva el radio al cuadrado.

La multiplicación de ‘Math.PI’ por el cuadrado del radio completa la fórmula del área del círculo fórmula ‘área = π \* radio^2.

1. **Entrada del usuario:** la expresión aritmética utilizada para leer el radio ingresado por el usuario es:

double radio = scanner.nextDouble();

‘scanner.nextDouble()’: lee el valor ingresado por el usuario como un número decimal de tipo ‘double’ y lo asigna a la variable ‘radio’.

1. Se desarrolla un programa que usa expresiones lógicas para verificar si un año es bisiesto.

**Figura 14**

*Expresiones lógicas para verificar si un año es bisiesto*

*Nota.* Ejemplo de programa que use expresiones lógicas para verificar si un año es bisiesto. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Para determinar si un año es bisiesto, se debe seguir una regla específica: un año es bisiesto si es divisible por 4, excepto en los casos en que también es divisible por 100, a menos que sea divisible por 400. En este programa:

1. Se utiliza la expresión lógica ‘(year % 4 == 0) && (year % 100 != 0 || year % 400 == 0)’ para verificar si el año es bisiesto.
   * ‘(year % 4 == 0)’: verifica si el año es divisible por 4.
   * ‘(year % 100 != 0 || year % 400 == 0)’: verifica si el año no es divisible por 100 a menos que también sea divisible por 400.
2. Dependiendo del resultado de la expresión lógica, se imprime en la consola si el año es bisiesto o no.

Para probar el programa, deberías simplemente compilarlo y ejecutarlo. Luego, se ingresa un año cuando se solicite y el programa determinará si es bisiesto o no.

# Cierre de la semana

En programación, las expresiones aritméticas son construcciones que involucran operaciones matemáticas básicas para realizar cálculos. Estas operaciones pueden incluir suma, resta, multiplicación, división, y otras más complejas. Las expresiones aritméticas son esenciales para realizar todo tipo de cálculos numéricos en programas informáticos.

Por otro lado, Las expresiones lógicas son construcciones que involucran operadores lógicos para realizar evaluaciones de verdad o falsedad. Estas expresiones son fundamentales para la toma de decisiones en programas, controlando el flujo del programa mediante la evaluación de condiciones. Operadores lógicos comunes incluyen AND (&&), OR (||), y NOT (!).

En resumen, las expresiones aritméticas se centran en realizar cálculos numéricos, mientras que las expresiones lógicas se centran en evaluar condiciones de verdad o falsedad para la toma de decisiones en programas informáticos. Ambas son herramientas esenciales en el desarrollo de software.

# Referencias

* Ceballos, Javier. (2011). Java 2 curso de programación. [Ebook]. Fuente del recurso <http://biblioteca.duoc.cl.webezproxy.duoc.cl/bdigital/elibros/15550-Java%202%20:%20curso%20de%20programacion/>
* Deitel, Harvey. (2004). Cómo programar en C/C++ y Java. [Ebook]. Fuente del recurso. <http://biblioteca.duoc.cl.webezproxy.duoc.cl/bdigital/elibros/23957-Como%20programar%20en%20C/>

# Lecturas de la semana

* Capítulo 5: Controlling Program Flow with Decision-Making.

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/106/> Páginas 97 - 134

# 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.