Capítulo 2

Programación elemental

Motivaciones

En el capítulo anterior, aprendió a crear, compilar y ejecutar un programa Java. A partir de este capítulo, aprenderá a resolver problemas prácticos mediante programación. A través de estos problemas, aprenderá tipos de datos primitivos Java y temas relacionados, como variables, constantes, tipos de datos, operadores, expresiones y entradas y salidas.

Objetivos

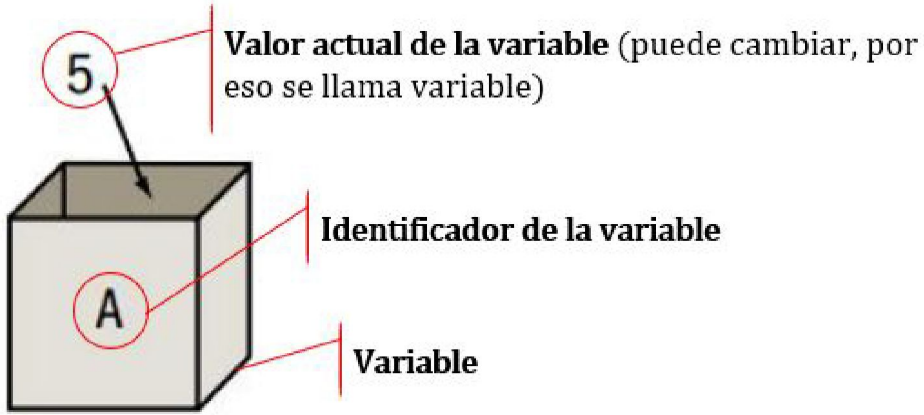
- Escribir programas Java para realizar cálculos simples.
- Obtener entradas desde la consola usando la clase Scanner.
- Utilizar identificadores para nombrar variables, constantes, métodos y clases.
- Utilizar variables para almacenar datos.
- Programar con instrucciones de asignación y expresiones de asignación.
- Utilizar constantes para almacenar datos permanentes.
- Nombrar clases, métodos, variables y constantes siguiendo sus convenciones de nomenclatura.
- Explorar los tipos de datos primitivos numéricos de Java: **byte, short, int, long, float** y **double**.
- Leer un valor de **byte, short, int, long, float o double** desde el teclado.

Objetivos

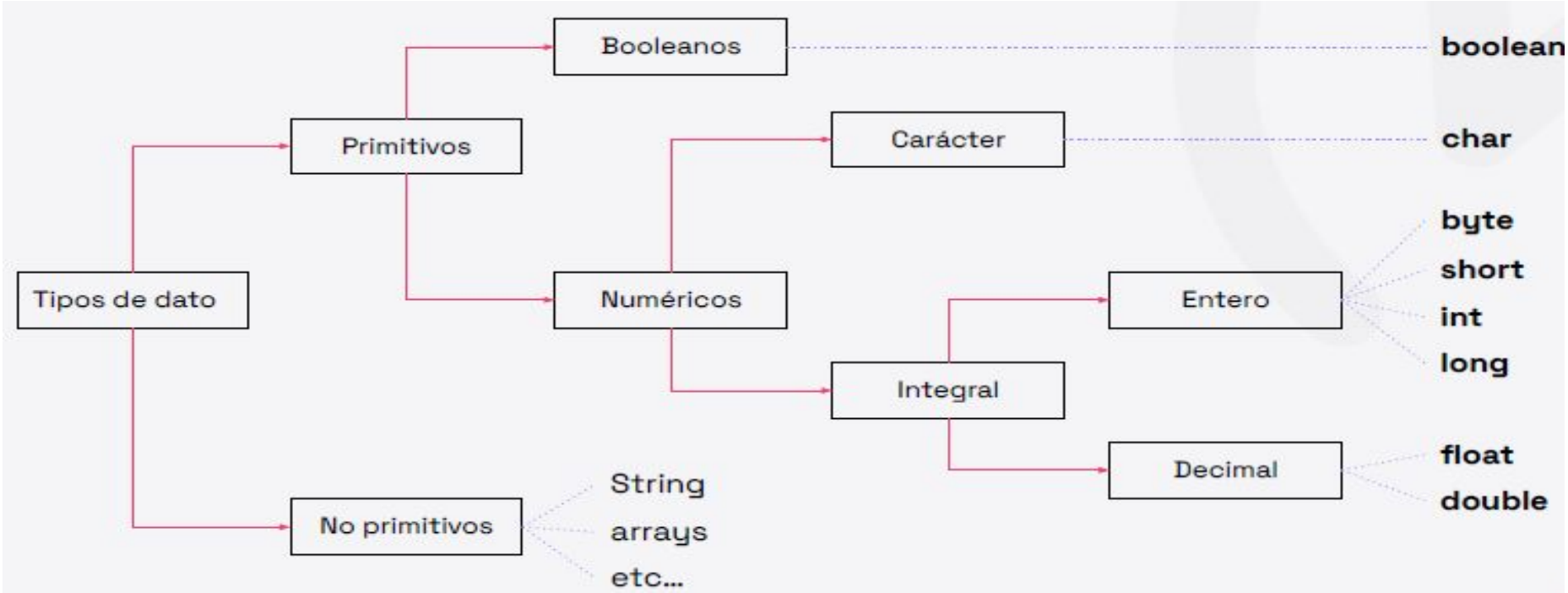
- Realizar operaciones usando los operadores +, -, *, / y % .
- Escribir literales enteros, literales de punto flotante y literales en notación científica.
- Escribir y evaluar expresiones numéricas.
- Utilizar operadores de asignación aumentada.
- Distinguir entre postincremento y preincremento y entre postdecremento y predecremento.
- Transmitir el valor de un tipo a otro tipo.
- Convertir explícitamente el valor de un tipo a otro tipo.
- Evitar errores comunes y trampas en la programación elemental.

VARIABLES

- Una **variable** es un **dato** que vamos a almacenar en **memoria**.
- Nos servirá para hacer algún tipo de **procesamiento** con ella.
- Por ejemplo: Para el cálculo del área de un triángulo necesitamos tres variables: base, altura y resultado.
- Una variable queda definida por su **identificador**, su **tipo de dato** y su **valor actual.**



Tipos de Datos



Tipos de datos primitivos numéricos

Tipo	Representación	Tamaño (bits)	Rango	Ejemplo
byte	N° entero con signo	8	-128,127	8
short	N° entero con signo	16	-32768, 32767	123
int	N° entero con signo	32	-2147483648, 2147483647	123456789
long	N° entero con signo	64	-9223372036854775808, 9223372036854775807	1234132412341324L
float	N° decimal (IEEE754)	32	$1.40239846 \times 10^{-45}$, $3.40282347 \times 10^{38}$	123.456f
double	N° decimal (IEEE754)	64	$4.9406564584124654 \times 10^{-324}$, $1.7976931348623157 \times 10^{308}$	12308941723412300000

Tipos Primitivos Booleano

- Posibles valores: verdadero, falso.

Tipo	Representación	Tamaño (bits)	Rango	Ejemplo
boolean	verdadero o falso	1	{ true, false }	true

- Ejemplo:

  boolean desempleado = false;

Tipo Primitivo Carácter

- Tipo primitivo **char**
  - Almacena un carácter
  - Valor interno numérico
  - Asignación usando comillas simples: ‘a’

Tipo	Representación	Tamaño (bits)	Rango	Ejemplo
char	Carácter Unicode (sin signo)	16	0 - 65535	a

- Ejemplo:

```
char letraInicio = ‘m’;
```

Tipo No Primitivo Cadena

- Tipo no primitivo **String**
  - Almacena una cadena de caracteres
  - Realmente es un objeto
  - Asignación entre comillas dobles: “Hola Mundo”
  - El tamaño en bits depende de la codificación usada; por defecto sería (nº de caracteres * 2) bytes

- Ejemplo:

  String msg = “Hola Mundo!”;

Identificadores de variables

- Un identificador es una secuencia de caracteres que constan de letras, dígitos, guiones bajos (_) y signos de dólar ($).
- Un identificador debe comenzar con una letra, un guión bajo (_) o un signo de dólar ($). No puede comenzar con un número.
- Un identificador no puede ser una palabra reservada.
- Un identificador no puede ser *true*, *false* o? *Null*.
- Un identificador puede ser de cualquier longitud.
- Deberían ser **autodescriptivos** y **adecuados** a su uso
- **Compromiso** entre **brevedad** y **expresividad**
- **Notación camelCase**: si son varias palabras
  - Sin espacios
  - Primera letra de primera palabra en minúscula
  - Resto de primeras letras en mayúsculas

Declaración de una variable

- **Declaración** de una variable
  - Operación por la cual decimos que una variable existe.
  - La podemos indicar en cualquier lugar del código.
  - No es obligatorio un valor inicial.

- Ejemplos:

```
String msg;
int x;
```

Asignación de valor a una variable

- **Asignación** de un valor a variable
  - Operación por la cual le modificamos un valor
  - Si es su **primer valor**, se llama **inicialización.**
  - La podemos hacer en cualquier lugar del código.

- Ejemplos:

```
String msg = “Hola Mundo”; // Declaración e inicialización
int x; // Declaración
x = 7; // Inicialización
msg = “Hasta luego”; // Asignación
```

Fuertemente Tipado

- Java es un lenguaje de programación **fuertemente tipado**.
  - Una variable no puede cambiar de tipo si no es a través de una conversión.
  - Ventajas: ayuda a prevenir errores
  - Desventajas: falta de flexibilidad
- Usualmente, asignamos el tipo de forma estática (al declarar la variable).

- Ejemplo:

```
int x;
```

Inferencia de Tipos

- Desde Java 10, el lenguaje permite que el tipo de dato se obtenga del contexto.
- Funciona siempre que sea una **variable local** (por ahora, para nosotros funciona siempre 😊)
- Se usa la palabra **var** para indicar que es una variable.
- Requiere de hacer **inicialización** a la vez de la declaración.

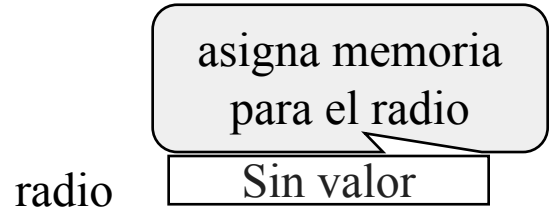
- Ejemplos:

  var msg = “Hola Mundo”; // Declaración e inicialización

  (no necesita que especifiquemos que msg es de tipo String)

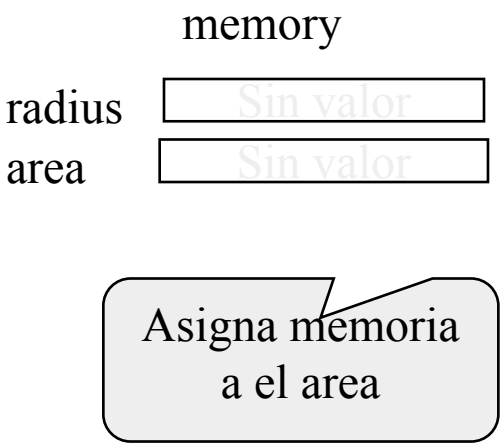
Seguimiento: Ej Calcular el área de un círculo

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radio;
    double area;

    // Asigna un valor a radio
    radio = 20;

    // Calcula el area
    area = radio * radio * 3.14159;

    // Muestra el resultado
    System.out.println("El area del círculo de radio " +
      radio + " es " + area);
  }
}
```



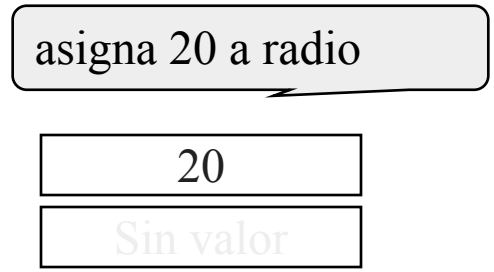
Seguimiento de la ejecución de un programa

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radio;
    double area;

    // Asigna un valor a radio
    radio = 20;

    // Calcula el area
    area = radio * radio * 3.14159;

    // Muestra el resultado
    System.out.println("El area del círculo de radio " +
      radio + " es " + area);
  }
}
```



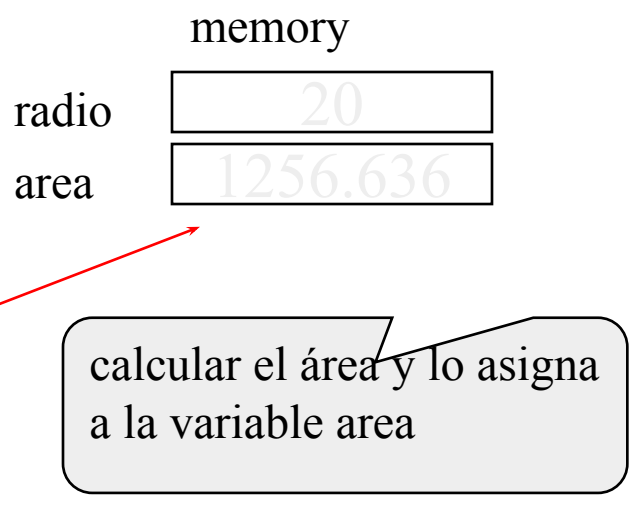
Seguimiento de la ejecución de un programa

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radio;
    double area;

    // Asigna un valor a radio
    radio = 20;

    // Calcula el area
    area = radio * radio * 3.14159;

    // Muestra el resultado
    System.out.println("El area del círculo de radio " +
      radio + " es " + area);
  }
}
```



Seguimiento de la ejecución de un programa

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radio;
    double area;

    // Asigna un valor a radio
    radio = 20;

    // Calcula el area
    area = radio * radio * 3.14159;

    // Muestra el resultado
    System.out.println("El area del círculo de radio " +
      radio + " es " + area);
  }
}
```



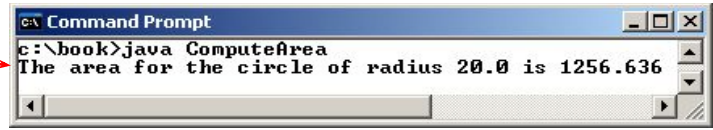
Seguimiento de la ejecución de un programa

```
public class ComputeArea {
  /** Main method */
  public static void main(String[] args) {
    double radio;
    double area;

    // Asigna un valor a radio
    radio = 20;

    // Calcula el area
    area = radio * radio * 3.14159;

    // Muestra el resultado
    System.out.println("El area del círculo de radio " +
      radio + " es " + area);
  }
}
```

memory

radio	20
area	1256.636

Muestra un mensaje por consola



CONSTATES

- Una variable que **no cambia su valor nunca** no debería llamarse variable. 🙊
- Este tipo de valores se llaman **constantes**.
- Se especifica añadiendo final delante del tipo de dato.

```
final tipodato NOMBRECONSTANTE = VALOR;
```

- Ejemplo:
  - final float PI = 3.141592f;
  - final double PI = 3.14159;
- Si intentamos **modificar** su valor se produce un **error de compilación**.
- Se pueden definir constantes de **cualquier tipo de dato**.
- Nomenclatura:
  - El identificador va en **mayúsculas**
  - Si son varias palabras, se separan con _ (guión bajo) (UPPER SNAKE CASE)

Entrada y Salida de Datos

Entrada de Datos

- Hasta ahora hemos trabajado con literales.
  - Valores escritos literalmente en el código.
- Los valores los debe introducir el usuario por teclado. Y para ello disponemos de la clase **Scanner**
  - Forma parte del paquete **java.util**
  - Sirve para **leer de teclado** (y de ficheros, y de conexiones de red)

```
var sc = new Scanner(System.in);
// Resto de código
sc.close();
```

Lectura desde el teclado

- Scanner tiene mecanismos para leer
  - Cadenas de caracteres: sc.nextLine();
  - Caracter: sc.next().charAt(0); //el 0 indica la posición primer carácter
  - Números: sc.nextInt(); (para byte, short, long, float y double)

Metodo	Descripción
`nextByte()`	lee un número de tipo `byte`.
`nextShort()`	lee un número de tipo `short`.
`nextInt()`	lee un número de tipo `int` .
`nextLong()`	lee un número de tipo `long` .
`nextFloat()`	lee un número de tipo `float`.
`nextDouble()`	lee un número de tipo `double`.

- Los usamos a la derecha de una asignación

  int numero = sc.nextInt();
- Podemos leer tantas veces como necesitemos

Ejemplo Entrada de dato

EjemploEntradaSalidaDatos.java

```java
package entradasalida;

import java.util.Scanner;

public class EjemploEntradaSalidaDatos {

    public static void main(String[] args) {

        var sc = new Scanner(System.in);

        System.out.print("Introduce un número: ");
        int numero = sc.nextInt();

        System.out.print("El número es: ");
        System.out.println(numero);


        sc.close();

    }

}
```

Problems Javadoc Declaration Console Git Staging

```
Introduce un número: 1234
El número es: 1234
```

```java
package prueba;

import java.util.Scanner;

public class MiPrueba {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Introduce un caracter:");
        char letra = sc.next().charAt(0);
        System.out.println("El caracter es " +letra);
    }

}
```

Console

<terminated> MiPrueba [Java Application] C:\Users\Moranua.GSTN024\.p2\pool\plugins\org.eclipse.justj.

```
Introduce un caracter:
t
El caracter es t
```

Importación Implícita y Explícita

```
java.util.* ; // Importación Implícita

java.util.Scanner; // Importación Explícita
```

No hay diferencia de rendimiento

OPERADOR

- Permite realizar un cálculo u operación con una o varias variables.
- **Operadores** según el número de **operandos:**
  - **Unarios**
  - **Binarios**
  - Ternarios

- Operadores **aritméticos**: trabajan con **operandos numéricos**
  - char también es un tipo numérico.

Operadores Aritméticos básicos

Nombre	Significado	Ejemplo	Resultado
+	Suma	34 + 1	35
-	Resta	34.0 - 0.1	33.9
*	Multiplicacion	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Resto	20 % 3	2

Ejemplos Operadores aritméticos

EjemploOperadoresAritmeticos.java

```java
package operadores;

public class EjemploOperadoresAritmeticos {

    public static void main(String[] args) {

        System.out.println("OPERADORES ARITMÉTICOS BÁSICOS");

        int i1 = 7;
        int i2 = 5;
        int i = i1 + i2;
        System.out.println(i);

        float f1 = 123.45f;
        float f2 = 456.34f;
        float f = f1 - f2;
        System.out.println(f);

        // No es los mismo división entera que con decimales

        int resultadoDivisionEntera = 3 / 2; // 1
        float resultadoDivisionDecimales = 3.0f / 2.0f;
        System.out.println("Resultado división entera -> " + resultadoDivisionEntera);
        System.out.println("Resultado división con decimales -> " + resultadoDivisionDecimales);

        int resto = i1 % i2;
        System.out.println(resto);
```

Problems Javadoc Declaration Console Git Staging

```
OPERADORES ARITMÉTICOS BÁSICOS
12
-332.89
Resultado división entera -> 1
Resultado división con decimales -> 1.5
2
```

Operadores de asignación combinado

Operador	Descripción	Ejemplo	Equivalente
+=	suma y asigna	i += 8	i = i + 8
-=	resta y asigna	i -= 8	i = i - 8
*=	multiplica y asigna	i *= 8	i = i * 8
/=	divide y asigna	i /= 8	i = i / 8
%=	resto y asigna	i %= 8	i = i % 8

Operadores de incremento y decremento

Operador	Nombre	Descripción	Ejemplo
++var	preincremento	incrementa var en 1 y usa el nuevo valor de var en la sentencia	**int j = ++i** //j es 2, i es 2
var++	posincremento	incrementa var en 1 pero usa el valor original en la sentencia	**int j = i++** //j es 1, i es 2
--var	predecremento	decrementa var en 1 y usa el nuevo valor de var en la sentencia	**int j = --i** //j es 0, i es 0
var--	posdecremento	decrementa var en 1 pero usa el valor original en la sentencia	**int j = i--** //j es 1, i es 0

Operadores de incremento y decremento

El uso de operadores de incremento y decremento hace que las expresiones sean cortas, pero también las hace complejas y difíciles de leer. Evite usar estos operadores en expresiones que modifiquen múltiples variables, o la misma variable varias veces como esta : int k = ++i + i.

```
int i = 10;
int newNum = 10 * i++;
```

El mismo efecto que

```
int newNum = 10 * i;
i = i + 1;
```

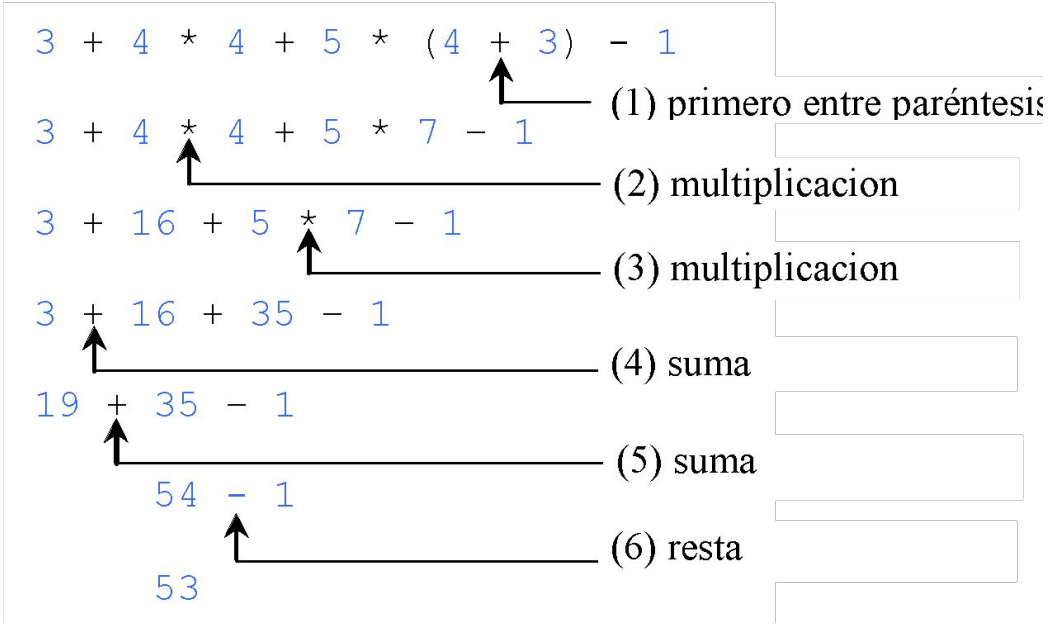
```
int i = 10;
int newNum = 10 * (++i);
```

El mismo efecto que

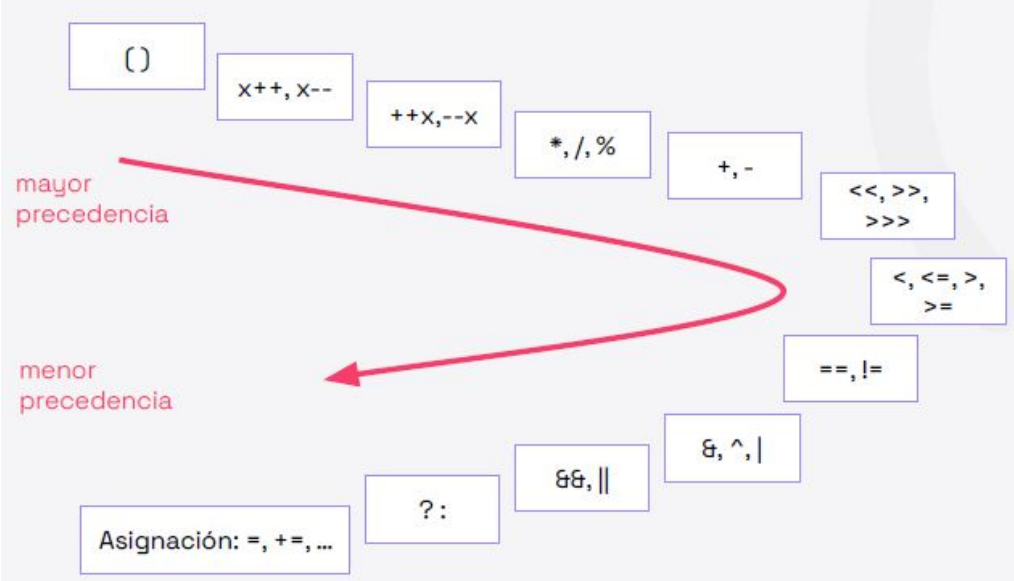
```
i = i + 1;
int newNum = 10 * i;
```

Cómo evaluar una expresión

Aunque Java tiene su propia manera de evaluar una expresión detrás de la escena, el resultado de una expresión Java y su correspondiente expresión aritmética son lo mismo. Por lo tanto, puede aplicar con seguridad la regla aritmética para evaluar una expresión Java.



Precedencia de Operadores



- **Si estamos en duda, usar paréntesis para agrupar y así pautar la precedencia nosotros.**

  Ejemplo: int r1 = 1 + 3 * 4; // 13

  Pero: int r2 = (1 + 3) * 4 // 16

Operadores de Comparación

- Similares a las matemáticas
  - Muy útiles para el control de flujo

Operador	Operación	Ejemplo
>	Mayor	int a = 7; int b = 6; a > b
>=	Mayor o igual	int a = 7; int b = 7; a >= b
<	Menor	int a = 3; int b = 5; a < b
<=	Menor o igual	int a = 3; int b = 3; a <= b
==	Igualdad	int a = 3; int b = 3; a == b
!=	Desigualdad	int a = 3; int b = 5; a != b

Operadores Lógicos

- Los operandos son valores booleanos.
- Sirven para hacer condiciones compuestas, más complejas.
  - Muy útiles para el control de flujo (próxima sección)
- Se rigen mediante una tabla de verdad.

A	B	A && B
F	F	F
V	F	F
F	V	F
V	V	V

A	B	A || B
F	F	F
V	F	V
F	V	V
V	V	V

A	!A
F	V
V	F

Operador	Operación	Ejemplo
&&	AND lógico	int x = 5; x > 0 && x < 10; // true
||	OR lógico	int x = 5; x <= 5 || x > 100; // true
!	Negación lógica	boolean v = true; !v; // false

Operador Ternario

- Único operador que recibe **tres operandos** cond ? v1 : v2
- Sirve para **evaluar una condición.**
- Si cond es true, se devuelve v1.
- Si cond es false, se devuelve v2.

- Ejemplo

```
var nota = (x >= 5) ? “APROBADO” : “SUSPENSO”;
```

Conversión de Tipos

- En ocasiones nos puede interesar convertir el tipo de dato de una variable en otro.
- También se le conoce como **casting.**
- Dos tipos:
  - Conversión **automática** o implícita.
  - Conversión manual o **explícita.**

Conversión automática

- Java es capaz de realizar una conversión automática entre tipos compatibles, **conversiones primitivas de ampliación**
  - byte a short, int, long, float, o double
  - short a int, long, float, o double
  - char a int, long, float, o double
  - int a long, float, o double
  - long a float o double
  - float a double
- Por ejemplo: Podemos almacenar un valor int en una variable long sin pérdida de información.

```
int i = 1234567;
long l = i; // No hay pérdida de información
```

¡OJO! Casos especiales con int o long hacia float o double.

Conversión automática

- Si convertimos int → float, long → float o long → double podemos **perder precisión**, y OJO! obtendremos un **número redondeado**.

Ejemplo:

```
long l2 = 123_456_789_123_456l;
System.out.println(l2);
float f2 = l2;
System.out.printf("%.2f",f2);
System.out.println();
```

Así podemos manejar la cantidad de decimales a mostrar

Ejemplos Conversión automática

EjemplosConversion.java

```
1  package variables;
2
3  public class EjemplosConversion {
4
5      public static void main(String[] args) {
6
7
8          // Conversión sin pérdida de información
9
10         int i = 1234567;
11         long l = i;
12
13         System.out.println("Conversión de int -> long");
14         System.out.println(i);
15         System.out.println(l);
16
17         // Con pérdida
18
19         long l2 = 123_456_789_123_456l;
20         System.out.println("Conversión de long -> float");
21         System.out.println(l2);
22         float f2 = l2;
23         System.out.printf("%.2f", f2);
24         System.out.println("");
25
26
27     }
```

Problems Javadoc Declaration Console Git Staging

```
<terminated> EjemplosConversion [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java (1 may 2023 20:27:52 – 20:27:52) [pid: 84227]
Conversión de int -> long
1234567
1234567
Conversión de long -> float
123456789123456
123456788103168,00
```

Reglas de conversión

Al realizar una operación binaria con dos operandos de diferentes tipos, Java convierte automáticamente los operandos utilizando las siguientes reglas:

1. Si uno de los operandos es double, el otro se convierte en double.
2. De lo contrario, si uno de los operandos es float, el otro se convierte en float.
3. De lo contrario, si uno de los operandos es long, el otro se convierte en long.
4. De lo contrario, ambos operandos se convierten en int.

Conversión Explícita

- Se realiza indicando el tipo al que queremos convertir entre **paréntesis**.
- Se indica a la **izquierda del valor** que queremos transformar.

- Ejemplo: de int a short.

```
int i3 = 1234;
short s3 = (short) i3;
```

- Ejemplo:

```
int i = (int)3.0; (estrechamiento de tipo)
int i = (int)3.9; (La parte de la fracción se trunca)
```

Errores comunes y trampas

- Error común 1: Variables no declaradas / no inicializadas y variables no utilizadas

  Ejemplo:

  ```
  double tasaInteres = 0.05;
  double interes = tasainteres * 45;
  ```

- Error común 2: Desbordamiento de entero

  Ejemplo:

  ```
  int valor = 2147483647 + 1;
  // valor será realmente -2147483648
  ```

- Error común 3: Errores de redondeo
- Error común 4: División de números enteros no deseados

Errores comunes y trampas

- Error común 3: Errores de redondeo

  Ejemplo:
  System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
  System.out.println(1.0 - 0.9);

- Error común 4: División de números enteros no deseados

```
int numero1 = 1;
int numero2 = 2;
double media = (numero1 + numero2) / 2;
System.out.println(media);
```

(a)

```
int numero1 = 1;
int numero2 = 2;
double media = (numero1 + numero2) / 2.0;
System.out.println(media);
```

(b)