[1.Introducción 4](#_Toc171636199)

[**-Renuncia-** 4](#_Toc171636200)

[2.¡Hola mundo! 5](#_Toc171636201)

[2.1 Desafío 5](#_Toc171636202)

[3.La clase principal 6](#_Toc171636203)

[3.1 Desafío 6](#_Toc171636204)

[4.Enteros 8](#_Toc171636205)

[4.1 Desafío 9](#_Toc171636206)

[5.Floats 10](#_Toc171636207)

[5.1 Desafío 10](#_Toc171636208)

[6.Caracteres 11](#_Toc171636209)

[6.1 Desafío 11](#_Toc171636210)

[7.Booleans 12](#_Toc171636211)

[7.1 Desafío 12](#_Toc171636212)

[8.Strings 13](#_Toc171636213)

[8.1 Desafío 13](#_Toc171636214)

[9.Recapitulación Desafío #1 14](#_Toc171636215)

[9.1 Desafío 14](#_Toc171636216)

[10.Recapitulación Desafío #2 15](#_Toc171636217)

[10.1 Desafío 15](#_Toc171636218)

[11.Operadores aritméticos 16](#_Toc171636219)

[11.1 Desafío 16](#_Toc171636220)

[12.Incremento y decremento 17](#_Toc171636221)

[12.1 Desafío 17](#_Toc171636222)

[13.Operadores de asignación 18](#_Toc171636223)

[13.1 Desafío 18](#_Toc171636224)

[14.Operadores de comparación 20](#_Toc171636225)

[14.1 Desafío 20](#_Toc171636226)

[15.Operadores lógicos 22](#_Toc171636227)

[15.1 Desafío 23](#_Toc171636228)

[16.Recapitulación Desafío #2 25](#_Toc171636229)

[16.1 Desafío 25](#_Toc171636230)

[17.if 26](#_Toc171636231)

[17.1 Desafío 26](#_Toc171636232)

[18.Else if 28](#_Toc171636233)

[18.1 Desafío 29](#_Toc171636234)

[19.Recapitulación Desafío #1 30](#_Toc171636235)

[19.1 Desafío 30](#_Toc171636236)

[20.Recapitulación Desafío #2 31](#_Toc171636237)

[20.1 Desafío 31](#_Toc171636238)

[21.Output 32](#_Toc171636240)

[21.1 Desafío 32](#_Toc171636241)

[22.Input 33](#_Toc171636243)

[22.1 Desafío 33](#_Toc171636244)

[23.Recapitulación Desafío #1 35](#_Toc171636246)

[23.1 Desafío 35](#_Toc171636247)

[24.For 36](#_Toc171636249)

[24.1 Desafío 37](#_Toc171636250)

[25.While 38](#_Toc171636252)

[25.1 Desafío 38](#_Toc171636253)

[26.Break 40](#_Toc171636255)

[26.1 Desafío 40](#_Toc171636256)

[27.Continue 41](#_Toc171636258)

[27.1 Desafío 41](#_Toc171636259)

[28.Recapitulación Desafío #1 43](#_Toc171636261)

[28.1 Desafío 43](#_Toc171636262)

[29.Recapitulación Desafío #2 44](#_Toc171636264)

[29.1 Desafío 44](#_Toc171636265)

[30.Adivina el número 45](#_Toc171636267)

[30.1 Desafío 45](#_Toc171636268)

[31.Calculadora avanzada 46](#_Toc171636270)

[31.1 Desafiar 46](#_Toc171636271)

# 1.Introducción

**Java**es un lenguaje de programación versátil y ampliamente utilizado conocido por su independencia de plataforma, lo que significa que las aplicaciones Java pueden ejecutarse en varios dispositivos y sistemas operativos sin modificaciones. Desarrollado por Sun Microsystems (ahora propiedad de Oracle), Java es conocido por su simplicidad, legibilidad y fuerte apoyo de la comunidad.

En este curso, aprenderá los conceptos básicos de Java, obtendrá desafíos y escribirá código real.

**-Renuncia-**

Para tener éxito en este curso, tienes que **hacer activamente los desafíos**.

¡Empecemos!

# 2.¡Hola mundo!

**"¡Hola Mundo!"** programa es un código simple que se envía a la pantalla.Hello World!

A menudo se utiliza para presentar un nuevo lenguaje de programación a un principiante.

Veamos el programa **Java** "Hello World!",

public class Main {  
    public static void main(String []args) {  
        System.out.println("Hello World!");  
    }  
}

La tercera línea da como resultado. Hello World!

# 2.1 Desafío

Principiante

Ahora es tu turno, usa el editor de código para generar Hello World!

public class Main {  
    public static void main(String []args) {  
        System.out.println("Hello World!");  
    }  
}

# 3.La clase principal

Antes de sumergirnos, aclaremos algo que nos acompañará a lo largo de todo el curso.

**La clase y el método Main**

Un vistazo rápido al código de la última lección (programa Hello World!),

public class Main {  
    public static void main(String []args) {  
        System.out.println("Hello World!");  
    }  
}

Para ejecutar todos los programas Java, necesita el **método principal**,

public class Main {  
    public static void main(String []args) {  
          
    }  
}

Y su código irá después del símbolo y antes del cierre.{}

public class Main {  
    public static void main(String []args) {  
        // Write code here  
    }  
}

El símbolo marca un comentario, ¡todo lo que venga después no se ejecutará!//

# 3.1 Desafío

Principiante

En cada lección, se le dará el código predeterminado, como en esta lección, y se le pedirá que escriba algún código.

Para este reto, lo haremos paso a paso.

Escriba un programa que genere Estoy aprendiendo Java!

**Pasos:**

1. Suprímase la línea 3 (el comentario)
2. Inserte una nueva línea después de la línea 2 que contenga la siguiente línea:  
   System.out.println("Estoy aprendiendo Java!");
3. public class Main {  
       public static void main(String []args) {  
           // Write code here
4. }  
   }

# 4.Enteros

**Las variables** son contenedores de valores/datos, son los bloques de construcción del código.

Se utilizan para almacenar, manipular, presentar o calcular datos. En resumen, una variable es como una unidad de memoria a la que podemos acceder escribiendo el nombre de la variable.

Cada variable tiene un **tipo** y un **nombre.**El **tipo** es responsable de que el programa sepa cómo manejar los datos. Hay muchos **tipos**, como números, caracteres, etc.

Para declarar una variable, utilizamos la siguiente sintaxis:

tipo nombre;

Para declarar y almacenar, guardar o inicializar un valor dentro de una variable, escribimos:

tipo nombre = valor;

Analicemos nuestro primer tipo:

int - número entero, por ejemplo:  1-2

En el siguiente ejemplo, inicializamos una variable de tipo con el nombre y el valor .int a 3

int a = 3;

**Tenga en cuenta** que cada línea en Java termina con “;“ (hay algunas excepciones, a las que llegaremos).

# 4.1 Desafío

Principiante

Escriba un programa que inicialice una variable de tipo con el nombre y el valor .int var 3

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("var = " + var);

    }

}

*Aviso: ¡No debes cambiar las líneas debajo de la línea 5!*

# 5.Floats

float - número de coma flotante, por ejemplo: o 3.2 - 1.03

En el siguiente ejemplo, inicializamos una variable de tipo con el nombre y el valor . float pi 3.14

float pi = 3.14f;

*Fíjate en el final del número f*

# 5.1 Desafío

Principiante

Escriba un programa que inicialice una variable de tipo con el nombre y el valor . float golden 1.618

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("golden = " + golden);

    }

}

# 6.Caracteres

char - **un solo** carácter, por ejemplo: o 'a''?'

El carácter debe estar entre comillas simples. ''

En el siguiente ejemplo, inicializamos una variable de tipo con el nombre y el valor .char dollar '$'

char dollar = '$';

# 6.1 Desafío

Principiante

Escriba un programa que inicialice una variable de tipo con el nombre y el valor **y** una variable de tipo con el nombre y el valor .

char bi 'b'

char dot '.'

Para inicializar múltiples variables, simplemente comience desde una nueva línea y haga otra declaración, por ejemplo,

int a = 3;  
int b = 9;

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("bi = " + bi);

        System.out.println("dot = " + dot);

    }

}

# 7.Booleans

boolean - los valores son true o false

En el siguiente ejemplo, inicializamos una variable de tipo con el nombre y el valor .boolean isNight false

boolean isNight = false;

# 7.1 Desafío

Principiante

Escriba un programa que inicialice una variable de tipo con el nombre y el valor .boolean isJavaFun true

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("isJavaFun = " + isJavaFun);

    }

}

# 8.Strings

String - colección de caracteres, por ejemplo: "Hola mundo!" o "Me gusta Java"

*Un valor debe estar entre****comillas dobles****. String*

En el siguiente ejemplo, inicializamos una variable de tipo con el nombre y el valor .String password "verystrongpass"

String password = "verystrongpass";

# 8.1 Desafío

Principiante

Escriba un programa que inicialice una variable de tipo con el nombre y el valor .String s "Estoy aprendiendo Java"

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("s = " + s);

    }

}

# 9.Recapitulación Desafío #1

¡Recapitulemos las variables!

# 9.1 Desafío

Principiante

Escriba un programa que inicialice las siguientes variables:

* k de tipo y valor int 180
* half de tipo y valor float 0.5
* a de tipo y valor char'c'

Las variables se pueden inicializar en cualquier orden.

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("k = " + k);

        System.out.println("half = " + half);

        System.out.println("a = " + a);

    }

}

# 10.Recapitulación Desafío #2

# 10.1 Desafío

Principiante

Escriba un programa que inicialice las siguientes variables:

* isFun de tipo y valor boolean true
* dollar de tipo y valor char '$'
* sad de tipo y valor boolean false
* fullName de tipo y valor String "John Doe"

}class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("isFun = " + isFun);

        System.out.println("dollar = " + dollar);

        System.out.println("sad = " + sad);

        System.out.println("fullName = " + fullName);

    }

}

# 11.Operadores aritméticos

**Operadores** utilizados para realizar operaciones en valores.

Primero, discutiremos los **operadores aritméticos** más básicos, tal vez, son familiares de las clases de matemáticas.

| **Operador** | **Operación** | **Ejemplo** |
| --- | --- | --- |
| + | Adición | 3 + 2 = 5 |
| - | Sustracción | 3 - 2 = 1 |
| \* | Multiplicación | 3 \* 2 = 6 |
| / | División | 4 / 2 = 2 |
| % | Módulo (Resto después de la división) | 3 % 2 = 1 |

Veamos un ejemplo de uso,

int a = 2;  
int b = 4;  
int c = a + b;

# 11.1 Desafío

Principiante

Escriba un programa que inicialice 2 variables de tipo, con los valores y (respectivamente), y después de inicializar otra variable de tipo que contendrá el resultado de . int a 5 b 2 int c = a \* b

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("a = " + a);

        System.out.println("b = " + b);

        System.out.println("c = " + c);

    }

}

# 12.Incremento y decremento

Los operadores más utilizados son los operadores de incremento y decremento.

* ++ - incremente el valor del número en 1 int float
* -- - Disminuir el valor del número en 1

int a = 7;  
a++;

En el siguiente ejemplo, aumentamos en uno la variable a, mantiene el valor al final. a = 8

Aquí hay otro ejemplo de decremento, se mantendrá al final: b = 2.5

float b = 3.5f;  
b--;

¡Veremos el uso de estos operadores cuando aprendamos sobre **los bucles**!

# 12.1 Desafío

Principiante

Inicialice la variable de tipo y valor e incremente su valor **dos veces**. count int 0

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("count = " + count);

    }

}

# 13.Operadores de asignación

Ya hemos visto el operador de asignación básico al inicializar variables con : =

int num = 100; // use of assignment operator

Hay más tipos de operadores de asignación en Java para que las operaciones sean más rápidas:

| **Operador** | **Ejemplo** | **Equivalente a** |
| --- | --- | --- |
| = | a = b | a = b |
| += | a += b | A = A + B |
| -= | a -= b | a = a - b |
| \*= | a \*= b | a = a \* b |
| /= | a /= b | A = A / B |
| %= | a %= b | a = a % b |

# 13.1 Desafío

Principiante

Se **le da** un programa, ¡refactorícelo para que funcione más rápido!

En cualquier lugar donde pueda cambiar la operación aritmética con un operador de asignación más rápido, hágalo.

**Nota**: Después de la declaración de una variable, por ejemplo: , puede usarla sin volver a usarla .int a = ...;a = ...;

class Main {

    public static void main(String[] args) {

        int a = 3;

        int b = 4;

        int c = 5;

        a = a + 3;

        b = b - a;

        c = a \* c;

        // Don't change below this line

        System.out.println("a = " + a);

        System.out.println("b = " + b);

        System.out.println("c = " + c);

    }

}

# 14.Operadores de comparación

**Los operadores de comparación** se utilizan para comparar entre dos operandos.

A veces necesitamos comprobar si un operando es más grande/más pequeño/... que otro operando. En la tabla siguiente se muestran los posibles operadores para la comparación:

| **Operador** | **Significado** | **Ejemplo** |
| --- | --- | --- |
| == | Igual | 1 == 2 return false |
| != | No es igual | 1 != 2 devuelve true |
| > | Mayor que | 1 > 2 return false |
| < | Más bajo entonces | 1 < 2 devuelve true |
| >= | Mayor o igual | 1 >= 2 return false |
| <= | Menor o igual | 1 <= 2 return true |

El operador de comparación devuelve **un valor booleano**, si la relación se mantiene o no. True o false

int var1 = 13;  
int var2 = 12;  
boolean var3 = var1 != var2;

En este ejemplo, combinamos el operador de asignación con el operador de comparación, ¿qué valor se mantiene? !=var3

var3 Tiene.true

# 14.1 Desafío

Principiante

Escriba un programa que inicialice 2 variables y de tipo int con los valores correspondientes (n1 = 8 y n2 = 9) y después de eso, inicialice otra variable de tipo bool que contenga si es mayor n1 que n2.

class Main {

    public static void main(String[] args) {

        // Write code here

        // Don't change below this line

        System.out.println("n1 = " + n1);

        System.out.println("n2 = " + n2);

        System.out.println("b = " + b);

    }

}

# 15.Operadores lógicos

**Los operadores lógicos** se utilizan para comprobar combinaciones de comparaciones y devolver true o false.

Por ejemplo, ¿5 es mayor que 3 y menor que 6? Sí.

| **Operador** | **Significado** | **Ejemplo** |
| --- | --- | --- |
| **&&** | **Y - true si todos los operandos son verdaderos** | **A & B** |
| **||** | **O - true si algún operando es verdadero** | **A || b** |
| **!** | **No: verdadero si el operando es falso** | **!A** |

Veamos algunos ejemplos,

**5 es mayor que 3 y 1 es igual a 1:**

boolean b1 = (5 > 3) && (1 == 1); // holds true

**1 no es igual a 1 o falso:**

boolean b2 = (1 != 1) || false; // holds false

**no (3 más grande que 4):**

boolean b3 = !(3 > 4); // holds true

**no (5 mayor que 10 o 5 mayor que 1):**

boolean b4 = !(5 > 10 || 5 > 1); // holds false

En el próximo capítulo, veremos algunos de los usos reales de los operadores lógicos.

En las siguientes tablas, **0** significa falso y **1** verdadero

**&& operador** (significa **AND)** en detalle,

| **a** | **b** | **A & B** |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Por ejemplo, si es **0**(falso) y también es **0**(falso), entonces también es **0** (falso) a && b

**|| operador** (significa **OR)** en detalle,

| **a** | **b** | **A || b** |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**! operador (**significa **NOT)** en detalle,

| **a** | **!a** |
| --- | --- |
| 0 | 1 |
| 1 | 0 |

# 15.1 Desafío

Principiante

Se **le da** un programa, inserte los valores que faltan para las variables y así mantendrá b1, b2 y b3 = true

class Main {

    public static void main(String[] args) {

        boolean b1 = ?;

        boolean b2 = ?;

        boolean b3 = b1 && b2;

        // Don't change below this line

        System.out.println("b3 = " + b3);

    }

}

# 16.Recapitulación Desafío #2

# 16.1 Desafío

Principiante

Se le da un programa.

¡A las variables , y les faltan sus valores!abc

Su tarea es completar los valores correctos para que la variable en la línea 8 se mantenga d = true

**Nota:** el uso de paréntesis es el mismo que probablemente conozcas de las clases de matemáticas en la escuela.

class Main {

    public static void main(String[] args) {

        int a = ?;

        int b = ?;

        int c = ?;

        // Don't change below this line

        boolean d = ((a - b) > c) && (b == c);

        System.out.println("d = " + d);

    }

}

# 17.if

Una parte importante de la programación es **la toma de decisiones**, eligiendo entre un par de caminos.

Para tomar una decisión sencilla en Java utilizamos la siguiente sintaxis,if

if (condition) {  
    // body of condition  
}

Cuando se cumple la condición, el programa entra en el cuerpo de la instrucción entre el .true{ }

int num = 12;  
if (num > 10) {  
    System.out.println("num es más grande que 10!");  
}

En el ejemplo anterior, el programa entra en el cuerpo porque es mayor que 10, por lo que se cumple la condición. num

int num = 9;  
if (num > 10) {  
    System.out.println("num es más grande que 10!");  
}

En el ejemplo anterior, el programa **no entra en**el cuerpo de la instrucción. (9 no es mayor que 10) if num

**Nota**: ¡El punto y coma () no es obligatorio después del cierre!;}

# 17.1 Desafío

Principiante

Se le da un programa.

Las variables y los valores que faltan, rellénelos para que se ejecute el código dentro de la instrucción (línea 7). a b if

**Bonus**: ¡intenta encontrar más de una solución!

class Main {

    public static void main(String[] args) {

        int a = ?;

        int b = ?;

        // Don't change below this line

        if (a >= b) {

            System.out.println("You are here!");

        }

    }

}

# 18.Else if

La instrucción puede tener una cláusula **opcional** que se activa cuando no se cumple la condición.  
Úselo con la sintaxis de la siguiente manera:if else

if (condition) {  
    // executed if condition is true  
}  
else {  
    // executed if condition is false  
}

La declaración se utiliza cuando se toma una decisión entre dos alternativas. if...else

Sin embargo, a veces queremos alternar entre más opciones para poder usar la siguiente declaración:else if

if (condition1) {  
    // executed if condition1 is true  
}  
else if (condition2) {  
    // executed if condition2 is true  
}  
else {  
    // executed if condition1 and condition2 are false  
}

Es posible agregar tanto como sea necesario.else if

Ejemplo:

if (3 != 3) {  
    // code 1  
}  
else if (2 > 1) {  
    // code 2  
}  
else {  
    // code 3  
}

En este ejemplo, el código 2 se ejecutará ya que 2 es mayor que 1 y 3 es igual a 3.

# 18.1 Desafío

Principiante

Se le da un programa que obtiene información del usuario, su edad y la almacena en una variable llamada age.

**Nota**: Aprenderemos en las próximas lecciones cómo obtener información del usuario, actualmente, simplemente no toque el código por encima de la línea 6.

Su tarea es imprimir (**mostrar en pantalla**) "Por encima de 18" si es mayor que 18, de lo contrario imprimir "Por debajo o igual a 18".age 18

Como todavía no aprendimos a imprimir, aquí están los códigos:

imprimir "Por encima de 18",

System.out.println("Por encima de 18");

print "Por debajo o igual que 18",

System.out.println("Por debajo o igual que 18");

*Compruebe los casos de prueba para ver todas las entradas y las salidas esperadas*

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int age = scanner.nextInt();

        // Don't touch above this line

        // Write code here

    }

}

# 19.Recapitulación Desafío #1

# 19.1 Desafío

Principiante

Se le da un programa que recibe la entrada del usuario y la guarda en la variable num

Su tarea es agregar código que inicialice una nueva variable que contendrá si es **mayor que 5**, de lo contrario x 5 num 0

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int num = scanner.nextInt();

        // Don't change above this line

        // Write code here

        // Don't change below this line

        System.out.println("x = " + x);

    }

}

# 20.Recapitulación Desafío #2

¡Vamos a crear una calculadora!

# 20.1 Desafío

# Principiante

Se le da un programa que obtiene **tres** entradas del usuario, (char) op, (int) n1 y (int) n2.

Su tarea es agregar código que inicialice una nueva variable (int) res que contendrá lo siguiente:

si op es igual a '+', se mantendrá res = n1 + n2

de lo contrario, si op es igual a '-', se mantendrá  res = n1 - n2

de lo contrario, si op es igual a '\*', se mantendrá . res = n1 \* n2

de lo contrario, mantendrá  res = 0.

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        char op = scanner.next().charAt(0);

        int n1 = scanner.nextInt();

        int n2 = scanner.nextInt();

        // Don't change above this line

        // Write code here

        // Don't change below this line

        System.out.println(res);

    }

}

# 21.Output

Ya hemos visto cómo imprimir, ¡pero ahora vamos a sumergirnos!

*En programación, a menudo se llama "****impresión****" para generar algo.*

En Java, se usa para imprimir algo en la pantalla System.out.println()

Por ejemplo

System.out.println("Hello");

El ejemplo anterior se imprime en la pantalla."Hello"

**Tenga en cuenta**que debe envolver lo que desea imprimir (A String).""

System.out.println() imprime en pantalla con una nueva línea al final, si desea obtener una salida sin nada al final, use System.out.print()

System.out.print("Hello");  
System.out.print("World");

Las siguientes impresiones (¡sin línea nueva al final!)Hello World

# 21.1 Desafío

# Principiante

Escriba un programa que imprima ."Me gusta programar en Java"

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}

# 22.Input

Aprendamos cómo obtener información del usuario.

*Tenga en cuenta que obtener información del usuario en Java requiere conocimientos en temas que no se aprenden en este curso, como clases y bibliotecas, pero aún creemos que es importante aprender como principiante y comprender algunos de los temas más adelante*

Primero debe declarar al principio del archivo,

import java.util.Scanner;

Ahora, en el código principal (crear un objeto de escáner),

Scanner scanner = new Scanner(System.in);

Y ahora puede usar el objeto para obtener información scanner

Obtener un int

int num = scanner.nextInt();

Obtener un float

float num = scanner.nextFloat();

Obtener un boolean

boolean b = scanner.nextBoolean();

Obtener un String

String s = scanner.nextLine();

# 22.1 Desafío

# Principiante

Escriba un programa que obtenga dos números enteros como entrada y genere la suma de los dos números.

*Deberá almacenar los dos números enteros de la entrada en variables e imprimir el cálculo*

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}

# 23.Recapitulación Desafío #1

# 23.1 Desafío

# Principiante

Escribe un programa que:

1. Impresiones (con nueva línea)"Ingresa tu edad:"
2. Obtiene la entrada del usuario, un número entero (la edad del usuario)
3. Calcula la diferencia entre la edad del usuario y los 120 años. Por ejemplo: para los 32 años, se calculará 88.
4. Imprime el número calculado.

¡El objetivo del programa es calcular el número de años que quedan hasta los 120 años!

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}

# 24.For

A veces, a la hora de programar, es necesario realizar la misma o casi la misma operación un par de veces.

Para evitar escribir lo mismo una y otra vez, podemos usar **Loops**.

El bucle tiene la siguiente sintaxis:for

for (initialization; condition; update) {

// body of loop

}

Al principio, se ejecuta la **inicialización** y, a continuación, el cuerpo del bucle se ejecuta en un **bucle**.  
Después de cada iteración del **bucle**, la **condición** se comprueba de nuevo y, si no se cumple (devuelve - false), el bucle se detiene.  
Si se cumple la condición, se ejecuta la **actualización** y, a continuación, también se ejecuta el cuerpo del bucle.

Digamos que quieres calcular la suma de todos los números entre 1 y 100, podrías hacer algo como esto,

int sum = 0;  
sum += 1;  
sum += 2;  
.  
.  
.  
sum += 100;

Esto funcionará... pero es largo escribirlo.  
La misma solución pero con bucle se verá así,for

int sum = 0;  
for (int i = 1; i <= 100; i++) {  
    sum += i;  
}

Al dividir en partes:

* int i = 1; (**inicialización**) - inicializar la variable de tipo con el valor iint1
* i <= 100 (**condición**) - El bucle **continuará** hasta que no se cumpla la condición, por lo tanto, en nuestro caso, hasta que i > 100
* i++ (**actualización**) - aumentar después de cada operación (bucle)i += 1

**Notar:** En el ejemplo anterior, el cuerpo del bucle se ejecuta **exactamente 100 veces**.

# 24.1 Desafío

# Principiante

Escribe un programa que imprima (con una nueva línea) 10 veces. Hazlo usando un **bucle for**. "hello"

El cuerpo del bucle debe ejecutarse **exactamente 10 veces**. ¿Cómo lo haces?

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}

# 25.While

El siguiente tipo de bucle es el bucle, con la siguiente sintaxis,while

while (condition) {  
    // body of loop  
}

El bucle solo tiene una **condición** y, mientras se cumpla la condición, el bucle continuará.while

Cuando se trata de bucles, es importante no entrar en un bucle sin fin, por ejemplo:

while (true) {  
    // body of loop  
}

Este es un bucle sin fin porque la condición es siempre !true

El mismo problema de la última lección con el bucle se verá así,while

int sum = 0;  
int i = 1;  
while (i <= 100) {  
    sum += i  
    i++;  
}

¡Encuentra las diferencias entre los bucles y!for while

# 25.1 Desafío

# Principiante

¡Al igual que en la última lección!

Escribe un programa que imprima 10 veces, hazlo usando un **bucle while**. "hello"

¡Hay muchas maneras de hacerlo!

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}

# 26.Break

La instrucción detiene el bucle instantáneamente cuando se encuentra.  
Por ejemplo break

for (int i = 0; i < 10; ++i) {  
    if (i == 6) {  
        break;  
    }  
    System.out.println("1");  
}

En el siguiente ejemplo, el bucle continuará hasta que sea igual a y, a continuación, el programa entrará en la instrucción, ejecutará la instrucción y **saldrá** del bucle. i 6 if break

En total, se imprimirán seis a medida que vayamos subiendo y cuando salgamos.1 i = 0,1,2,3,4,5 i = 6

# 26.1 Desafío

# Principiante

Se le da un código.

El código imprime números entre 0 y 100 con saltos de 3. Su tarea es agregar código entre las líneas 6 y 9, lo que hace que el bucle salga **después** de que se imprima 21.

¡No cambies todo el código dado!

class Main {

    public static void main(String[] args) {

        for (int i = 0; i < 100; i += 3) {

            // Don't change above this line

            // Write code here

            // Don't change below this line

            System.out.println(i);

        }

    }

}

# 27.Continue

La instrucción, cuando se ejecuta, omite la iteración del bucle actual y **continúa** en la siguiente.continue

for (int i = 0; i < 10; i++) {  
    if (i % 2 == 0) continue;  
    System.out.println(i);  
}

En el siguiente ejemplo se imprimirá ya que todos los números pares (divisibles por 2) se **omiten** con .1 3 5 7 9 continue

**Nota al margen:** puedes hacer el mismo ejemplo solo con el bucle for como este,

for (int i = 1; i < 10; i+=2) {  
    System.out.println(i);  
}

¿Puedes notar las diferencias?

# 27.1 Desafío

# Principiante

Escriba un programa que obtenga una entrada entera del usuario, llamémoslo , luego imprime todos los números de a (**incluidos**) tales que **no son divisibles** por **o**con una nueva línea entre ellos.num1num37

**Por ejemplo:**

para la entrada debe imprimir,num = 22

1  
2  
4  
5  
8  
10  
11  
13  
16  
17  
19  
20  
22

use loop y statement para decidir cuándo omitir, después, intente resolver esto sin la instruccióncontinue

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Write code here

    }

}

# 28.Recapitulación Desafío #1

# 28.1 Desafío

# Principiante

Escriba un programa que obtenga información del usuario, un número entero, llamémoslo, el programa imprime la suma de todos los números desde hasta .num1num \* 3

Ejemplo

**Entrada**: 2

**Salida**: 21

**Explicación**: La suma de los números del 1 al 6 (2\*3) es 1+2+3+4+5+6=21

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}

# 29.Recapitulación Desafío #2

**La exponenciación** es una operación matemática, escrita como *bn*, que involucra dos números, la *base* *b* y el *exponente* o *potencia* *n.*

*bn=b\*...\*b (b para****n****veces)*

*por ejemplo*

23 = 2 \* 2 \* 2 = 8

34 = 3 \* 3 \* 3 \* 3 = 81

# 29.1 Desafío

# Principiante

Escriba un programa que obtenga dos números enteros del usuario: base y potencia (en este orden), **calcule la exponenciación**y la imprima.

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}

# 30.Adivina el número

¡Hagamos parte del juego **Adivina el número**!

# 30.1 Desafío

# Principiante

Se le da un número - (supongamos que es un número aleatorio :)), sus tareas son: randNum

1. Obtener el número de entrada del usuario (la suposición)

2. Si la conjetura es mayor que la salida del número dado Too high

3. Si la conjetura es menor que la salida del número dado Too low

4. Si es el número exacto de salida You are right!

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int randNum = scanner.nextInt();

        // Write code here

    }

}

# 31.Calculadora avanzada

# 31.1 Desafiar

Medio

Escribe un programa que actúe como una calculadora como en las lecciones anteriores, pero ahora recibirá información del usuario hasta que ingrese e

El cálculo se realizará en el formato, (separado por una nueva línea), donde se ,,,. las posibles operacionesoperationfloatfloat+-\*/

Cada salida (resultado del cálculo) está separada por una nueva línea.

Consulte los casos de prueba para ver ejemplos.

Para obtener la entrada para un uso de char,

char c = scanner.nextLine().charAt(0);

Es posible que deba omitir una línea para poder usar solo scanner.nextLine()

import java.util.Scanner;

class Main {

    public static void main(String[] args) {

        // Write code here

    }

}