

Java desde 0

¿Qué es JAVA?

- Es un lenguaje de programación de alto nivel desarrollado por Sun Microsystems.
- Lanzado el año de 1995, es uno de los tres lenguajes de programación más demandados y más utilizados.
- Java no genera código nativo, genera código que se puede ejecutar en una máquina virtual, sus programas son portables.

Código Nativo	VS	Código No Nativo
Funciona en una máquina específica		Es un lenguaje interpretado y cualquier máquina lo puede interpretar sin problemas

¿Qué es un IDE?

Integrated development Environment o entorno de desarrollo integrado, un IDE es un programa que permite desarrollar y ejecutar todo tipo de software, posee algunas herramientas o extensiones que facilitan la tarea del programador, un ejemplo de esto es cuando al digitar código algunas palabras o funciones se autocompleta. Tiene características como:

-
- Editor de código: Es el editor de texto para trabajar con código fuente.
- Compilador: Traduce las instrucciones que creamos con el editor.
- Depurador: Nos permite probar y buscar errores en el programa.
- Linker: Herramienta para combinar archivos de código.
- Refactorización de código: Para actualización o reformato en el código

¿Qué es JRE?

Java Runtime Environment (Entorno de ejecución Java)

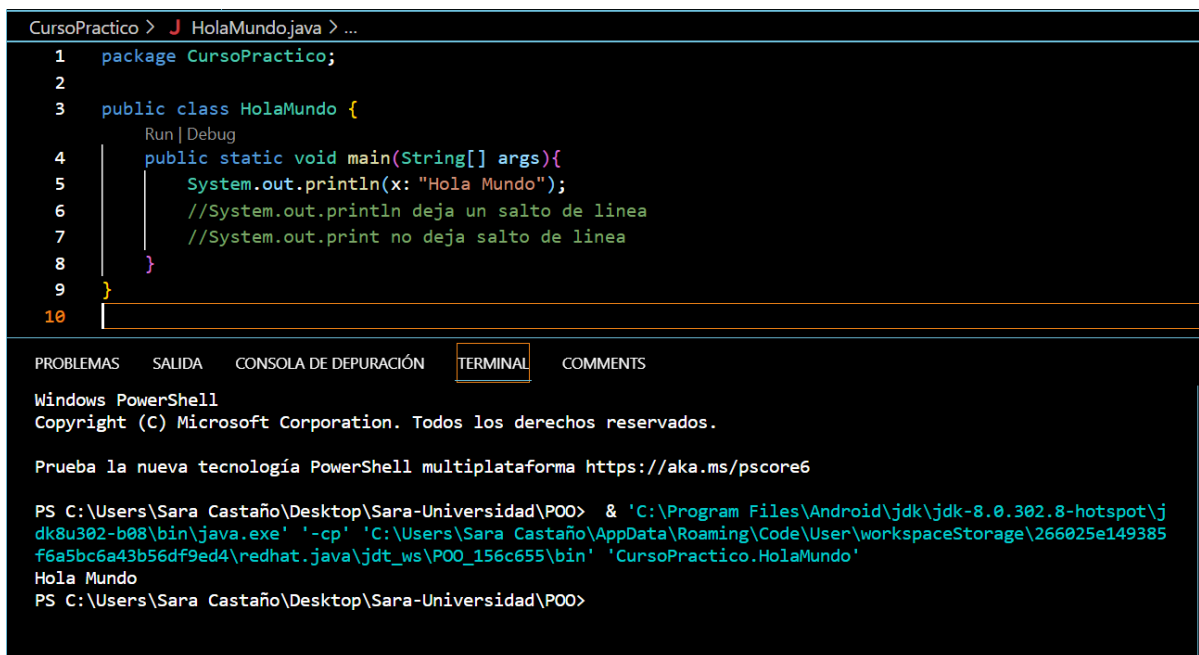
También denominado en sus inicios como JVM = Java Virtual Machine (Máquina Virtual Java), es necesario instalarlo porque es un lenguaje multiplataforma.

¿Cómo iniciar?

Para iniciar en este camino de la programación en Java hice uso de una serie de cursos y recursos, de los que vamos a estar realizando ejercicios y tomando apuntes, todo esto quedará escrito a continuación.

Utilidad de Jvaas en los distintos

Hola Mundo



```
CursoPractico > J HolaMundo.java > ...
1 package CursoPractico;
2
3 public class HolaMundo {
4     Run | Debug
5     public static void main(String[] args){
6         System.out.println(x: "Hola Mundo");
7         //System.out.println deja un salto de linea
8         //System.out.print no deja salto de linea
9     }
10 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL** COMMENTS

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\P00> & 'C:\Program Files\Android\jdk\jdk-8.0.302.8-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385f6a5bc6a43b56df9ed4\redhat.java\jdt_ws\P00_156c655\bin' 'CursoPractico.HolaMundo'

Hola Mundo

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\P00>

La estructura base de un programa se puede evidenciar en el ejemplo anterior, cabe resaltar que el `System.out.println("El texto a ingresar");` es la instrucción para que lo esté entre las comillas se muestre en pantalla.

Comentarios

//Esta estructura funciona para los comentarios de una sola línea

/*

Esta estructura funciona para los comentarios de más de una línea

*/

Identificadores

Tipo de Identificador	Convencion	Ejemplo
Nombre de una clase	Comienza por mayúscula	String, Practica, Rectangulo
Nombre de función	Comienza por minúscula	calcularEdad, getValue, setColor
Nombre de variable	Comienza por minúscula	edad, color, appletSize
Nombre de constante	Todo en mayúsculas	PI, MAX_ANCHO, MIN_ANCHO

Reglas para los identificadores:

- Se debe comenzar con una letra, una barra baja, el símbolo de dólar. Los demás caracteres pueden ser letras, dígitos, barra baja o símbolos de dólar.
Ejemplo: pruebalzyAcademy, \$izyacademy, prueba_izyacademy.
- No se pueden usar palabras reservadas del lenguaje Java.
- Java distingue las mayúsculas y minúsculas, por lo tanto, los identificadores serán totalmente diferentes si se escribe identificadoruno a si se escribe identificadorUno

Variables y tipos de variables

¿Que es una variable?

Es un espacio de memoria donde puedes almacenar un valor

TIPOS DE VARIABLES ENTERAS		
Nombre	Tamaño	Rango
Byte	8 Byte	-128 a 127
Short	16 Byte	-32,768 a 32,767
Int	32 Byte	-2,147,483,648 a 2,147,483,648
Long	64 Byte	-9,223,372,036,854,775,808 a 9,223,372,854,775,807
TIPOS DE VARIABLES DECIMALES O REALES		
Nombre	Tamaño	Rango
Double (numeros muy extensos)	64 Byte	4,9e-324 a 1.8e+308
Float (f después de escribir el número)	32 Byte	1.4e-045 a 3.4e+038
TIPOS DE VARIABLES DE CARACTERES		
Boolean	1 Byte	True,False
Char	16 Byte	Unicode:sistema de codificación de caracteres utilizado para el almacenamiento y el intercambio de datos en formato de texto, este tipo comprende caracteres alfanuméricos.' ' Comillas simples

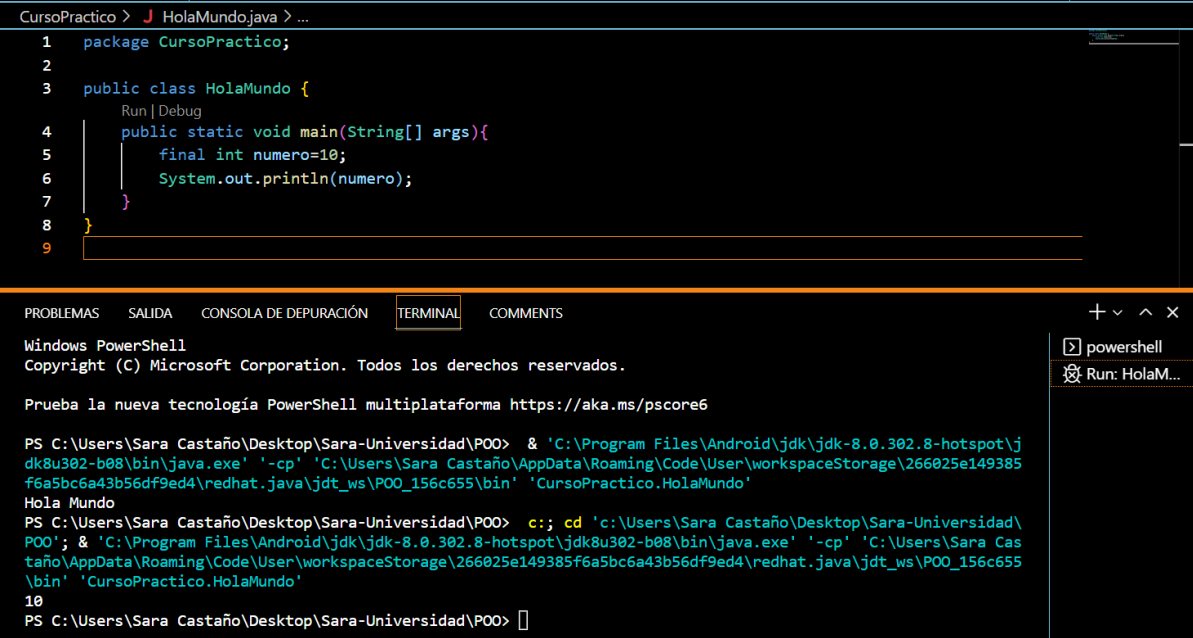
TIPOS DE VARIABLES NO PRIMITIVOS		
Nombre	Tamaño	Rango
Integer	64	-9223372036854775808 a +9223372036854775807 y contempla el valor null
Espacio en blanco	1 Byte	Corresponde a un char
Salto de línea	1 Byte	Corresponde a un byte
String	2 Bytes por cada char	" " Comillas dobles

Consideraciones:

- Un objeto es una cosa distinta a un tipo primitivo, aunque "porten" la misma información.
- Tener siempre presente que los objetos en Java tienen un tipo de tratamiento y los tipos primitivos otro. Si en algún momento tienen la misma información

Constante:

Puedes almacenar en un tipo de dato, un valor fijo que no va a cambiar durante el desarrollo del programa. Para declararla tiene que poner la palabra final antes del tipo de dato, de la manera como se muestra a continuación.



The screenshot shows an IDE with a Java file named `HolaMundo.java`. The code defines a package `CursoPractico` and a class `HolaMundo` with a `main` method. Inside the `main` method, a constant `numero` is declared with the value 10, and it is printed to the console. Below the code editor, there is a terminal window. The terminal shows the execution of the Java program using `java` and `cd` commands. The output of the program is `Hola Mundo`.

```
CursoPractico > J HolaMundo.java > ...
1 package CursoPractico;
2
3 public class HolaMundo {
4     public static void main(String[] args){
5         final int numero=10;
6         System.out.println(numero);
7     }
8 }
9

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL COMMENTS
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> & 'C:\Program Files\Android\jdk\jdk-8.0.302.8-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385f6a5bc6a43b56df9ed4\redhat.java\jdt_ws\POO_156c655\bin' 'CursoPractico.HolaMundo'
Hola Mundo
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> c:: cd 'c:\Users\Sara Castaño\Desktop\Sara-Universidad\POO'; & 'C:\Program Files\Android\jdk\jdk-8.0.302.8-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385f6a5bc6a43b56df9ed4\redhat.java\jdt_ws\POO_156c655\bin' 'CursoPractico.HolaMundo'
10
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO>
```

ENTRADA Y SALIDA DE DATOS:

Para leer un nuevo dato que nos ingrese el usuario usamos la función `Scanner`, también debemos importar la clase `import java.util.Scanner;` para que no nos genere un error, todo esto se muestra a continuación, aparte comprobamos que el dato que ingresa el usuario sea el mismo que se guarda en la variable y que posteriormente se muestra en pantalla con la instrucción `System.out.printl("")`

También debemos tener en cuenta de que tipo es la variable declarada, en la línea 10 estamos asignando el valor que ingreso el usuario a numero:

EJEMPLOS DE FORMAS DE ASIGNACIÓN	
<code>.nextInt();</code>	Siempre y cuando trabajemos con un entero
<code>.nextFloat();</code>	Siempre y cuando trabajemos con un float, en algunas computadores no acepta el punto sino la coma
<code>.nextDouble();</code>	Siempre y cuando trabajemos con un double
<code>.next();</code>	Siempre y cuando trabajemos con una cadena, este <code>nextLine</code> , permite guardar y mostrar solo la primera parte de la cadena. Si ingresamos "Hola que tal", se imprime "Hola"
<code>.nextLine();</code>	Siempre y cuando trabajemos con una cadena, este <code>nextLine</code> , permite guardar y mostrar la cadena completa. Si ingresamos "Hola que tal", se imprime "Hola que tal"
<code>.next().charAt(0);</code>	Siempre y cuando trabajemos con un char, al ponerle (0) solo se guardará el primer dígito o primer carácter que encuentre, si ingresamos "Hola", nos la guarda e imprime la "H"

Ejemplo con `.nextInt();`

```
CursoPractico > J EntradaySalida.java > EntradaySalida > main(String[])
1 package CursoPractico;
2
3 import java.util.Scanner; //Importemos la clase que permite leer las variables
4
5 public class EntradaySalida {
6     public static void main(String[] args){
7         Scanner entrada = new Scanner(System.in); //System.in viene de entrada de datos
8         int numero;
9         System.out.println(x: "Digite un numero ");
10        numero = entrada.nextInt(); //Con .next.Int guardamos en la variable lo que el usuario nos digita
11        System.out.println("El numero que digitado fue "+numero);
12    }
13 }
14
15
```

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL COMMENTS

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> & 'C:\Program Files\Android\jdk\jdk-8.0.302.8-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385f6a5bc6a43b56df9ed4\redhat.java\jdt_ws\POO_156c655\bin' 'CursoPractico.EntradaySalida'

Digite un numero
12
El numero que digitado fue 12
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO>

Ejemplo con .nextLine();

```
CursoPractico > J EntradaySalida.java > EntradaySalida > main(String[])
1 package CursoPractico;
2
3 import java.util.Scanner; //Importemos la clase que permite leer las variables
4
5 public class EntradaySalida {
6     public static void main(String[] args){
7         try (Scanner entrada = new Scanner(System.in)) {
8             String cadena;
9             System.out.println(x: "Digite una cadena");
10            cadena = entrada.nextLine(); //Con .next.Int guardamos en la variable lo que el usuario nos digita
11            System.out.println("La cadena es: "+cadena);
12        }
13    }
14 }
15
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL COMMENTS

Run: EntradaySalida

Abrir archivo en el editor (ctrl + clic) PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> & 'C:\Program Files\Android\jdk\jdk-8.0.302.8-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385f6a5bc6a43b56df9ed4\redhat.java\jdt_ws\POO_156c655\bin' 'CursoPractico.EntradaySalida'

Digite una cadena
Hola Sara, espero estes bien
La cadena es: Hola Sara, espero estes bien
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO>

ENTRADA Y SALIDA DE DATOS CON JOptionPane

En este capítulo del curso estamos capturando datos para mostrarlos a través de un Cuadro de Diálogo, para capturar datos seguimos la estructura de la línea 14,15,16,17 y para mostrarlo seguimos la estructura de la línea 19,20,21,22

Cuando tenemos un número entero, puede que nos genere un error, entonces lo convertimos entonces usamos `Integer.parseInt` y dentro del paréntesis ponemos lo que se muestra a continuación:

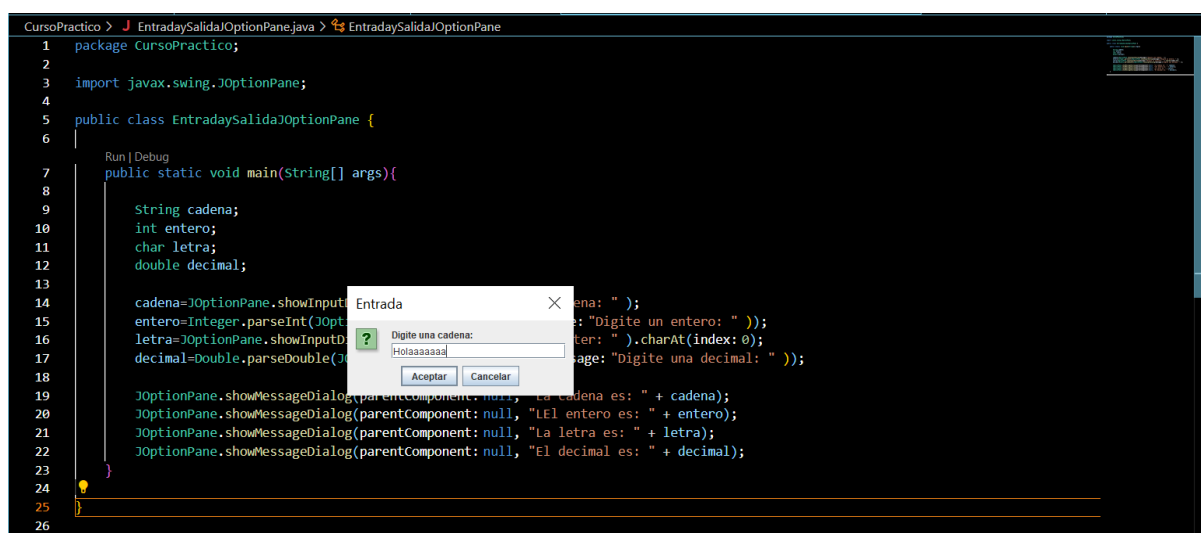
```
entero=Integer.parseInt(JOptionPane.showInputDialog("Digite un entero: " ));
```

Si lo que debemos convertir es de tipo `double`, se realiza como se mostrara en la siguiente línea:

```
decimal=Double.parseDouble(JOptionPane.showInputDialog("Digite una decimal: " ));
```



```
1 package CursoPractico;
2
3 import javax.swing.JOptionPane;
4
5 public class EntradaySalidaJOptionPane {
6
7     public static void main(String[] args){
8
9         String cadena;
10        int entero;
11        char letra;
12        double decimal;
13
14        cadena=JOptionPane.showInputDialog(message: "Digite una cadena: " );
15        entero=Integer.parseInt(JOptionPane.showInputDialog(message: "Digite un entero: " ));
16        letra=JOptionPane.showInputDialog(message: "Digite un caracter: " ).charAt(index: 0);
17        decimal=Double.parseDouble(JOptionPane.showInputDialog(message: "Digite una decimal: " ));
18
19        JOptionPane.showMessageDialog(parentComponent: null, "La cadena es: " + cadena);
20        JOptionPane.showMessageDialog(parentComponent: null, "El entero es: " + entero);
21        JOptionPane.showMessageDialog(parentComponent: null, "La letra es: " + letra);
22        JOptionPane.showMessageDialog(parentComponent: null, "El decimal es: " + decimal);
23    }
24
25
26 }
```



```
1 package CursoPractico;
2
3 import javax.swing.JOptionPane;
4
5 public class EntradaySalidaJOptionPane {
6
7     public static void main(String[] args){
8
9         String cadena;
10        int entero;
11        char letra;
12        double decimal;
13
14        cadena=JOptionPane.showInputDialog(message: "Digite una cadena: " );
15        entero=Integer.parseInt(JOptionPane.showInputDialog(message: "Digite un entero: " ));
16        letra=JOptionPane.showInputDialog(message: "Digite un caracter: " ).charAt(index: 0);
17        decimal=Double.parseDouble(JOptionPane.showInputDialog(message: "Digite una decimal: " ));
18
19        JOptionPane.showMessageDialog(parentComponent: null, "La cadena es: " + cadena);
20        JOptionPane.showMessageDialog(parentComponent: null, "El entero es: " + entero);
21        JOptionPane.showMessageDialog(parentComponent: null, "La letra es: " + letra);
22        JOptionPane.showMessageDialog(parentComponent: null, "El decimal es: " + decimal);
23    }
24
25
26 }
```

```
1 package CursoPractico;
2
3 import javax.swing.JOptionPane;
4
5 public class EntradaySalidaOptionPane {
6
7     public static void main(String[] args){
8
9         String cadena;
10        int entero;
11        char letra;
12        double decimal;
13
14        cadena=JOptionPane.showInputDialog("Digite un entero: ");
15        entero=Integer.parseInt(JOptionPane.showInputDialog("Digite un entero: "));
16        letra=JOptionPane.showInputDialog("Digite una letra: ");
17        decimal=Double.parseDouble(JOptionPane.showInputDialog("Digite una decimal: "));
18
19        JOptionPane.showMessageDialog(parentComponent: null, "La cadena es: " + cadena);
20        JOptionPane.showMessageDialog(parentComponent: null, "El entero es: " + entero);
21        JOptionPane.showMessageDialog(parentComponent: null, "La letra es: " + letra);
22        JOptionPane.showMessageDialog(parentComponent: null, "El decimal es: " + decimal);
23    }
24 }
25
26
```

Operadores y tipos de operadores

¿Que es un operador?

Los operadores son caracteres especiales dentro del lenguaje Java que permiten manipular y realizar operaciones con los datos de tipo primitivo. Los hay unarios, binarios y ternarios.

TIPOS DE OPERADORES DE ASIGNACIÓN		
Nombre	Simbolo	Descripción
Operador de asignación	=	Es usado para asignar un valor a un tipo de variable, se puede mezclar para hacer una declaración más amplia; !=
TIPOS DE OPERADORES ARITMÉTICOS		
Suma	+	Suma dos valores numéricos o concatena dos cadenas de texto.
Resta	-	Resta dos valores (del valor de la derecha al valor de la izquierda).
Multiplicacion	*	Multiplica dos números.
Division	/	Divide el valor de la izquierda por el valor de la derecha.

Modulo	%	Divide el valor de la izquierda por el valor de la derecha y devuelve el residuo.
Incremento	++	Incrementa el valor en 1.
Decremento	—	Decrementa el valor en 1
TIPOS DE OPERADORES RELACIONALES		
Mayor Que	>	El valor del operador de la izquierda es mayor que el valor del operador de la derecha.
Menor que	<	El valor del operador de la izquierda es menor que el valor del operador de la derecha
Mayor o igual	>=	El valor del operador de la izquierda es mayor o igual que el valor del operador de la derecha.
Menor o igual	<=	El valor del operador de la izquierda es menor o igual que el valor del operador de la derecha.
Igual	==	Los valores de los dos operadores son iguales.
Diferente	!=	Los valores de los dos operadores son diferentes.
TIPOS DE OPERADORES LÓGICOS		
AND	&&	Este retorna true si la salida de ambos operadores es verdadera.
OR		Este retorna true si la salida de alguno de los operadores es verdadera.
NOT	!	Este operador invierte el estado de la condición.
TIPOS DE OPERADORES CLASE MATH		

Math.sqrt(9)	sqrt=Raíz	Este operador hace uso de la clase Math y sirve para sacar una raíz cuadrada
Math.pow(base, exponente)	pow=Potencia	Este operador hace uso de la clase Math y sirve para sacar elevar a un exponente
Math.round(doble a long, float a int)	round=redondear	Este operador hace uso de la clase Math y sirve para redondear un número, hay que hacer unas conversiones.
Math.random()	random=azar	Este operador hace uso de la clase Math y sirve para que se me muestre en consola un número al azar

Condicionales

If: Se le indica una condición, si esta es verdadera se ejecuta, de lo contrario las instrucciones que estén dentro del mismo no se ejecutan. Se suele traducir como “Si se cumple esta condición haz esto”

```
CursoPractico > J SentenciaIf.java > ...
1 package CursoPractico;
2
3 public class SentenciaIf {
4     public static void main(String[] args) {
5
6         int precio=300;
7
8         if (precio>100){
9             System.out.println(x: "El precio es mayor que 100");
10        }
11    }
12
13 }
14
```

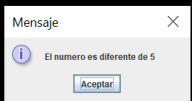
if – else: es como el anterior solo que después de cerrarse la llave de if, se añade else sin indicarle ninguna condición. Esto sirve para que si la condición del if no es verdadera, ejecute otras instrucciones que estarán dentro de else. Se suele traducir como “Si se cumple esta condición haz esto y sino haz esto”

```
CursoPractico > J SentenciaIf.java > SentenciaIf > main(String[])
1 package CursoPractico;
2
3 import javax.swing.JOptionPane;
4
5 public class SentenciaIf {
6     public static void main(String[] args) {
7
8         int numero,dato=5;
9         numero=Integer.parseInt(JOptionPane.showInputDialog(message: "Digite un numero"));
10
11         if (numero==dato){
12             JOptionPane.showMessageDialog(parentComponent: null,message: "El numero es 5");
13         }else{
14             System.out.println(x: "El numero es diferente de 5");
15         }
16     }
17
18 }
19
```



En este ejercicio pedimos un número a través de un cuadro de diálogo, comparamos ese número con lo que tenemos guardado en dato, entramos al if, si es igual a dato se muestra que el número es 5, de lo contrario, entramos al else y decimos que el número es distinto de 5, los datos se capturan a través de cuadros de diálogo

```
CursoPractico > J SentenciaIf.java > SentenciaIf > main(String[])
1 package CursoPractico;
2
3 import javax.swing.JOptionPane;
4
5 public class SentenciaIf {
6     public static void main(String[] args) {
7
8         int numero,dato=5;
9         numero=Integer.parseInt(JOptionPane.showInputDialog(message: "Digite un numero"));
10
11         if (numero==dato){
12             JOptionPane.showMessageDialog(parentComponent: null,message: "El numero es 5");
13         }else{
14             JOptionPane.showMessageDialog(parentComponent: null,message: "El numero es diferente de 5");
15         }
16     }
17
18 }
19
```



if -elseif: esta estructura es como una mezcla de los anteriores, esto nos permite, que si no se cumple la condición podamos indicar otra condición para hacerlo aún mas específico. Se suele traducir como “Si se cumple esta condición haz esto y sino si cumple esta condición haz esto”.

```
CursoPractico > J SentenciaIf.java > SentenciaIf
1 package CursoPractico;
2
3 public class SentenciaIf {
4     public static void main(String[] args) {
5
6         int precio=50;
7
8         if (precio>100){
9             System.out.println(x: "El precio es mayor que 100");
10        }else {
11            if(precio>80){
12                System.out.println(x: "El precio es mayor que 80");
13            }else{
14                System.out.println(x: "El precio es menor que 80");
15            }
16        }
17    }
18 }
```

Switch: Esta estructura condicional es como si fuera algun tipo de selección múltiple, le damos un valor (puedes ser una variable) y una lista de casos, si el valor cumple alguna condición de algún caso determinado, se realizan las instrucciones asociadas a ese caso, si no cumple con ninguna condición, el default le indica al usuario un mensaje, generalmente de error. A continuación su sintaxis y un ejemplo:

```
CursoPractico > J SentenciaSwitch.java > ...
1 package CursoPractico;
2
3 public class SentenciaSwitch {
4     public static void main(String[] args){
5
6         switch (valor){
7             case caso1:
8                 Instrucciones
9                 break;
10            case caso2:
11                Instrucciones
12                break;
13            case caso N:
14                Instrucciones
15                break;
16            default:
17                Instrucciones
18        }
19    }
20
21 }
22 }
```

En el siguiente ejercicio queremos evaluar cuál de los casos coincide con el día Lunes, hacemos un recorrido por todos los días de la semana, el caso 1 cumple con la condición de día="Lunes", entonces como vemos en consola se imprime "Hoy es Lunes", en caso de que no hubiera coincidido con ningún caso, se imprime el mensaje que está en default "No has introducido un día correcto".

```
CursoPractico > J SentenciaSwitch.java > ?a SentenciaSwitch > @ main(String[])
1 package CursoPractico;
2
3 public class SentenciaSwitch {
4     Run | Debug
5     public static void main(String[] args){
6         String dia="Lunes";
7
8         switch (dia){
9             case "Lunes":
10                 System.out.println("Hoy es "+dia);
11                 break;
12             case "Martes":
13                 System.out.println("Hoy es "+dia);
14                 break;
15             case "Miercoles":
16                 System.out.println("Hoy es "+dia);
17                 break;
18             case "Jueves":
19                 System.out.println("Hoy es "+dia);
20                 break;
21             case "Viernes":
22                 System.out.println("Hoy es "+dia);
23                 break;
24             case "Sabado":
25                 System.out.println("Hoy es "+dia);
26                 break;
27             case "Domingo":
28                 System.out.println("Hoy es "+dia);
29                 break;
30             default:
31                 System.out.println(x: "No has introducido un dia correcto");
32             }
33         }
34     }
35 }

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL COMENTARIOS
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/powershell
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> & 'C:\Program Files\Android\jdk\jdk-8.0.302-b08-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\26502
5e149385f6a50c6a4365d9ed4\vscode\java\jdt_ws_VPOO_156c655\bin' 'CursoPractico.SentenciaSwitch'
Hoy es Lunes
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO>
```

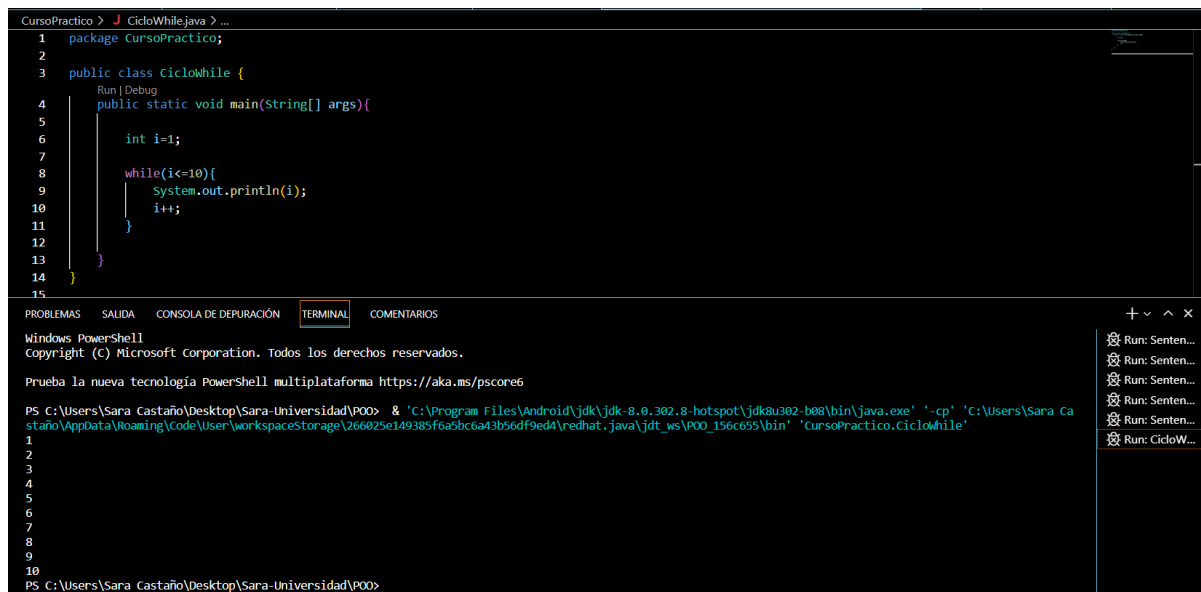
Ciclos o Bucles:

Ciclo while: El bucle while es tan sencillo como decir, que **mientras** se cumpla la condición se ejecuta el código que haya dentro del bucle, y en el momento que ya no se cumpla esa condición se sale del bucle. A continuación su sintaxis y un ejercicio.

```
CursoPractico > J CicloWhile.java > ...
1 package CursoPractico;
2
3 public class CicloWhile {
4     Run | Debug
5     public static void main(String[] args){
6         while(Condicion){
7             Instrucciones
8         }
9     }
10 }
11 }
12
13
14
15
```

En el ejercicio que desarrollamos a continuación, inicializamos la variable $i=1$, el ciclo while hara lo siguiente, Mientras (i sea menor o igual a 10), imprime i en pantalla, el $i++$, genera un aumento de 1 en 1, cuando se vuelva a entrar al bucle despues de la primer vez, i ya no es igual a 1, sino a 2, y asi sucesivamente, hasta que i vale 10 y se sale del ciclo while. En caso de que queramos que se escriba de forma descendente, cambiamos la condicion y el

i++, por un i-. En este bucle si la condición no se cumple no se ejecuta ni una vez, mientras que en el do-while si se debe ejecutar al menos una vez



```
CursoPractico > J CicloWhile.java > ...
1 package CursoPractico;
2
3 public class CicloWhile {
4     public static void main(String[] args){
5         int i=1;
6         while(i<=10){
7             System.out.println(i);
8             i++;
9         }
10    }
11
12
13
14
15
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL COMENTARIOS

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

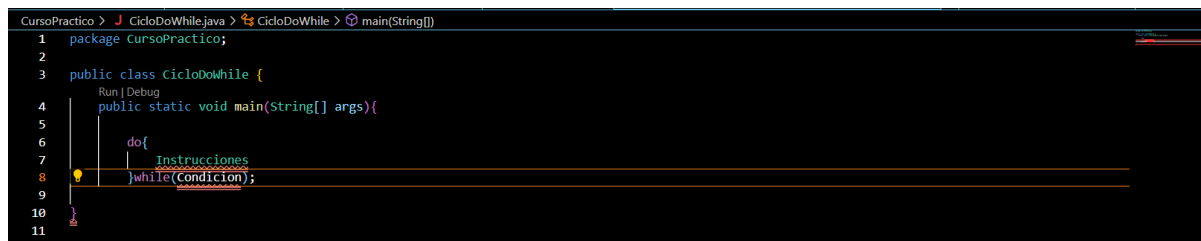
Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> & 'C:\Program Files\Android\jdk-8.0.302-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385f6a5bc6a43b56df9ed4\redhat.java\jdt_ws\POO_156c655\bin' 'CursoPractico.CicloWhile'

1
2
3
4
5
6
7
8
9
10
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO>

Run: Senten...
Run: Senten...
Run: Senten...
Run: Senten...
Run: Senten...
Run: CicloW...

Ciclo Do-while: El bucle do while es prácticamente igual al while, pero con la diferencia de que el código del bucle se ejecutará al menos una vez ya que la comprobación se hace después de cada iteración y no antes como en el caso del while. A continuación su sintaxis y un ejercicio.



```
CursoPractico > J CicloDoWhile.java > CicloDoWhile > main(String[])
1 package CursoPractico;
2
3 public class CicloDoWhile {
4     public static void main(String[] args){
5
6         do{
7             Instrucciones
8         }while(condicion);
9
10    }
11
```

En el siguiente ejercicio, haciendo uso del do-while queremos que se repitan los números siempre y cuando sea menor a 10, en el ejemplo anterior veíamos que se verifica si la condición era verdadera, para así ejecutar el ciclo, pero aquí, se ejecuta al menos una vez incluso si la condición no era verdadera, por eso en este caso se muestra el 11, y al evaluar si se pueden seguir imprimiendo más números el programa se da cuenta de que no, por que i no es menor que 10, entonces finaliza ahí el programa.


```
CursoPractico > J CicloDoWhile.java > CicloDoWhile > main(String[])
1 package CursoPractico;
2
3 public class CicloDoWhile {
4     Run | Debug
5     public static void main(String[] args){
6         int i=11;
7         do{
8             System.out.println(i);
9             i++;
10        }while(i<=10);
11    }
12
13
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL COMENTARIOS
Windows PowerShell
Copyright (c) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> & "C:\Program Files\Android\jdk\jdk-8.0.302.8-hotspot\jdk8u302-b08\bin\java.exe" "-cp" "C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385fea5bc6a43b56df9ed4\redhat.java\jdt_ws\POO_156c655\bin" "CursoPractico.CicloDoWhile"
11
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO>
```

Ciclo for: El bucle for sirve para ejecutar un código un número conocido de veces, por ejemplo recorrer un array o cualquier otro tipo de colección o simplemente ejecutar el código un número concreto de veces

En java hay dos posibles tipos de bucle for:

- El más habitual es en el que se establece el número desde el que va a empezar el bucle, hasta cuando se va a seguir ejecutando el bucle y finalmente el modo en el que se actualiza el contador del bucle, habitualmente lo que se hace es incrementar en una unidad el contador. $i + \text{paso}$ es donde se define cuánto va a ir aumentando la variable de control.

```
CursoPractico > J CicloFor.java > ...
1 package CursoPractico;
2
3 public class CicloFor {
4     Run | Debug
5     public static void main(String[] args){
6
7         for(int i = valor inicial; i <= valor final; i = i + paso){
8             Bloque de Instrucciones....
9         }
10    }
11
12
13
14
```

```
CursoPractico > J CicloFor.java > ...
1 package CursoPractico;
2
3 public class CicloFor {
4
5     Run | Debug
6     public static void main(String[] args){
7         for(int i=1; i<=10; i++){
8             System.out.println(i);
9         }
10    }
11 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL COMENTARIOS

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO> & 'C:\Program Files\Android\jdk-8.0.302-8-hotspot\jdk8u302-b08\bin\java.exe' '-cp' 'C:\Users\Sara Castaño\AppData\Roaming\Code\User\workspaceStorage\266025e149385f6a5b66a3b56df9eda\redhat.java\jdt_ws\POO_156c655\bin' 'CursoPractico.CicloFor'

1
2
3
4
5
6
7
8
9
10
11
PS C:\Users\Sara Castaño\Desktop\Sara-Universidad\POO>

- El otro formato de bucle for es el ideal para recorrer colecciones de objetos sean del tipo que sean (arrays, ArrayList, HashMap, ...) y en este caso hay que definir un iterador que nos devolverá un elemento de la colección en cada iteración y la colección que se quiere recorrer.

```
CursoPractico > J CicloFor.java > ...
1 package CursoPractico;
2
3 public class CicloFor {
4
5     Run | Debug
6     public static void main(String[] args){
7         for (iterador; coleccion) {
8             Bloque de instrucciones
9         }
10    }
11 }
12 }
13 }
```

```
CursoPractico > J CicloFor.java > ...
1 package CursoPractico;
2
3 public class CicloFor {
4
5     public void verDiasSemana() {
6         String[] dias = {"Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo"};
7
8         System.out.println(x: "Los dias de la semana son:");
9         for (String d: dias) {
10             System.out.println(d);
11         }
12    }
13 }
```

```
CursoPractico > J CicloFor.java > CicloFor > verDiasSemanaV2()
1 package CursoPractico;
2
3 public class CicloFor {
4
5     public void verDiasSemanaV2() {
6         String[] dias = {"Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo"};
7
8         System.out.println(x: "Los dias de la semana son:");
9         for (int i = 0; i < dias.length; i++) {
10             System.out.println(dias[i]);
11         }
12    }
13 }
```

Arrays:

Arrays: Un array o arreglo en java es una estructura de datos que nos permite almacenar un conjunto de datos de un mismo tipo. El tamaño de los Arrays se declara en un primer momento y no cambia durante la ejecución del programa.

Arrays Unidimensionales	
Tipo_de_variable[] Nombre_del_array = new Tipo_de_variable[];	
int[] edad= new int[4] long edad=new long[4] char[] sexo =new char[2] String[] nombre=new String[2]	
Arrays Multidimensionales	
Bidimensionales	Tridimensionales
Tipo_de_variable[][] Nombre_del_array= new Tipo_de_variable[][]	Tipo_de_variable[][][] Nombre_del_array= new Tipo_de_variable[][][]
int[][] edad= new int[4][4];	int[][][] edad= new int[3][3][3];

ArrayList: Es una clase que trabaja con listas genéricas a través de la lógica, esto quiere decir que se puede insertar o extraer cualquier elemento de la lista.

Es importante tener presente que existe una gran diferencia entre el ArrayList y el LinkedList, ya que esta se da principalmente en la implementación de los algoritmos;

- La clase LinkedList emplea una lista doblemente encadenada
- La clase ArrayList utiliza un arreglo que se redimensiona en forma automática según se efectúan inserciones y extracciones de datos.

Estos son algunos de los métodos que se pueden implementar con un ArrayList:

- add(): Método que nos permite añadir un elemento a un ArrayList.
- addAll(): Método que nos permite añadir una colección al final o en un punto en concreto de un ArrayList.
- clear(): Elimina todos los elementos de un ArrayList.
- iterator(): Devuelve un iterador sobre los elementos de la lista.
- listIterator(): Devuelve un iterador sobre los elementos de la lista en la posición indicada como parámetro.
- get(): Devuelve un elemento del ArrayList de la posición indicada como parámetro.

- `remove()`: Elimina un elemento dentro de un `ArrayList`; Bien sea indicando la posición en la que se encuentra el elemento, o bien por coincidencia con el objeto pasado como parámetro.
- `set()`: Método que permite sustituir un elemento por otro dentro de un `ArrayList`.
- `size()`: Devuelve el tamaño de un `ArrayList`.

Una de las principales ventajas de emplear la clase `ArrayList` es que el acceso a un elemento determinado de la lista es inmediato mediante el método `'get'`, en cambio, la implementación del método `'get'` en la clase `LinkedList` requiere recorrer en forma secuencial la lista hasta llegar a la posición a buscar.

Programación Orientada a Objetos

Tipos de aplicaciones en las que se usa Java:

1. Aplicaciones Cliente: es aquella que funciona sobre un propio sistema operativo de un dispositivo, es decir, que la aplicación se instala y corre de forma local en una máquina. Su principal característica es que **no hay que estar conectado a una red todo el tiempo para trabajar con ellas.**
2. Aplicaciones Cliente/Servidor: es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. **Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.**
3. Aplicaciones WEB: son programas Java que se ejecutan en un servidor de páginas web. Estas aplicaciones reciben "solicitudes" desde un ordenador y envían al navegador (Internet Explorer, Firefox, Safari, etc.) que actúa como su cliente páginas de respuesta en HTML.

Estos son algunos de los ejemplos de toda la potencia que tiene Java como lenguaje para su aprendizaje. Como te puedes dar cuenta Java es un lenguaje muy robusto con el que se pueden realizar muchas cosas, pero de igual forma se requiere de tiempo para conocerlo a fondo.

¿Es Java un Software Libre?

Es una tecnología semi-liberada, y podemos acceder a todo lo necesario para desarrollar de forma gratuita: compilador, maquina virtual, biblioteca de clases, etc. Java es un lenguaje multiplataforma, esto quiere decir que funciona en cualquier sistema operativo.

¿Que es una clase?


La clase se diseña para especificar las propiedades y el comportamiento que tiene un objeto. Las propiedades del objeto se implementan o definen a través de variables y el comportamiento por medio de métodos, las clases son como moldes a partir de las cuales se pueden producir varios objetos.

¿Que es un objeto?

Es una instancia de una clase. Todos los objetos comparten dos características su estado y su comportamiento. Se puede decir entonces que un objeto es una colección de métodos y atributos.

Ejemplo: Piense en un molde de chocolate como la clase, es decir, el que define la forma que tendrá la chocolatina. El proceso de vaciar el chocolate en el molde como un método "*Preparar chocolatina*" y al obtener una chocolatina como el objeto, pero se pueden obtener chocolatinas con diferentes sabores y colores, en donde estos serán los atributos.

Estructuras y Componentes de una clase



Estructura de una clase

```
package persona;

import java.util.Vector;

public class Persona {

    private Vector companieros;

    public Persona() {
        this.companieros = new Vector();
    }

    public Persona(Vector amigos) {
        this.companieros = amigos;
    }

    public void addCompaniero(Persona unaPersona){
        this.companieros.add(unaPersona);
    }

    public Persona getPrimerCompaniero(){
        return (Persona)this.companieros.remove(0);
    }

}
```

- Paquete 1
- Imports 2
- Declaración de la clase 3
- Variables 4
- Constructores 5
- Métodos 6

Introducción al lenguaje Java - Olyvia García Cárdena

Sentencia Package(Paquete):

Esta sentencia puede o no aparecer en una clase, sin embargo toda clase de Java pertenece a un package, dado el caso que no esté especificada es por que la clase pertenece al paquete de omisión (Default), si está especificada debe ser la primera sentencia y sólo debe haber una sentencia package en java, no más.

Sentencia Import:

Cuando las clases e interfaces Java que utilizará otra clase Java se encuentran en el mismo paquete no es necesario especificar el nombre del paquete antes del nombre de esas clases o interfaces, pero si no es así, es necesario escribir el nombre completo de la clase con todo y nombre del paquete al que pertenece dicha clase o interfaz utilizada.

Si la sentencia import es usada, debe aparecer después de la sentencia package, si no, la clase no podrá compilar. La sentencia import puede aparecer varias veces en la clase.

```
1 // Sin sentencia import
2 package escuela;
3 class Curso {
4     persona.Maestro maestro;
5 }
6
7 // Con sentencia import
8 package escuela;
9 import personal.Maestro;
10 class Curso {
11     Maestro maestro;
12 }
```

Comentarios:Java permite hacer o escribir comentarios en el código. Los comentarios pueden aparecer en cualquier parte de la clase.

```
1 // Comentario fin de linea antes de sentencia package
2 package escuela;
3 /*
4  * Este es un comentario multilinea
5  * continua el comentario
6  */
7 class Curso {
8
9     // Este es un comentario fin de linea
10
11     persona.Maestro maestro; //Otro comentario fin de linea
12
13     /* Otro comentario multilinea en una sola linea, es válido */
14
15 }
```

Declaracion de la Clase:La declaración class marca el inicio de la clase. Basta con especificar la palabra reservada class y enseguida el nombre de la clase. La declaración de una clase esta compuesta por varias partes:

//Ejemplo de la definición mínima obligatoria de una clase

```
class Punto {}
```

//Ejemplo de la definición completa de una clase

```
apublic bfinal cclass dPredio eextends Poligono eimplements Geometria f{ }
```

- a) Modificadores de acceso
- b) Modificadores de no acceso
- c) Nombre de la clase
- d) Nombre de la clase base si ésta extiende de otra clase
- e) Nombre de todas las interfaces que implemente, si está implementado algun interface
- f) Cuerpo de la clase (campos de clase [class fields], métodos, constructores), incluidos dentro de llaves de inicio y fin, {}

Nota: De estas partes o componentes sólo la palabra reservada class, el nombre de la clase y las llaves de apertura y cierre son obligatorias, el resto son opcionales.

Variables: En java podemos declara variables a nivel de la clase y a nivel del método:

Las variables declaradas a nivel de la clase son llamadas variables de instancia o atributos de instancia, y almacenan el estado de un objeto (también llamado instancia de una clase) .

Cada objeto o instancia de una clase tiene su propia copia de estas variables de instancia. Si los valores de estas variables cambian, no afectan a los valores de instancia de otros objetos. .

```
class NombreDeLaClase {  
  
    tipoDato variableDeInstancia1;  
    tipoDato variableDeInstancia2;  
    //Más variables de instancia...  
  
    tipoDato nombreDelMetodo1(argumentos){  
        //Cuerpo del método  
    }  
  
    tipoDato nombreDelMetodo1(argumentos){  
        //Cuerpo del método  
    }  
}
```

Métodos:Un método en Java es un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.

```

public class Aritmetica{

    int sumar(int a, int b){
        //realiza la suma y regresa el resultado como un entero
        return a + b;
    }
}

```

Constructor:

Un constructor de clase se usa para inicializar o crear los objetos de una clase. Una clase puede definir varios constructores que acepten diferentes conjuntos de parámetros de método.

Características de un constructor:

- Estos métodos sólo se puede ejecutar al momento de la creación de un objeto, no es posible utilizarlos después.
- No devuelven ningún valor.
- El nombre del constructor es idéntico al nombre de la clase
- Por defecto en Java, se crea un constructor sin argumentos, conocido como constructor vacío. Este constructor lo agrega en automático el compilador de Java a nuestra clase, sin embargo si nosotros definimos un constructor distinto al constructor vacío, es decir, con argumentos, entonces Java ya no agrega el constructor vacío y es nuestra responsabilidad agregar el constructor vacío a nuestra clase si fuera necesario

```

//Constructor sin argumentos
Aritmetica(){
    //Cuerpo del constructor
}

//Constructor con 2 argumentos
Aritmetica( int arg1 , int arg2){
    //Cuerpo del constructor
}

```

Destructor:

Un destructor es un método opuesto a un constructor, éste método en lugar de crear un objeto lo destruye liberando la memoria de nuestra computadora para que pueda ser utilizada por alguna otra variable u objeto.

PILARES POO

Encapsulamiento (Modificadores de Acceso)

Los modificadores de acceso en java, como su nombre indica, determina desde qué clases se puede acceder a un determinado elemento.

En Java tenemos 4 tipos: public , private , protected y el tipo por defecto, que no tiene ninguna palabra:

	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
Public	x	x	x	x
Protected	x	x	x	
Default	x	x		
Private	x			

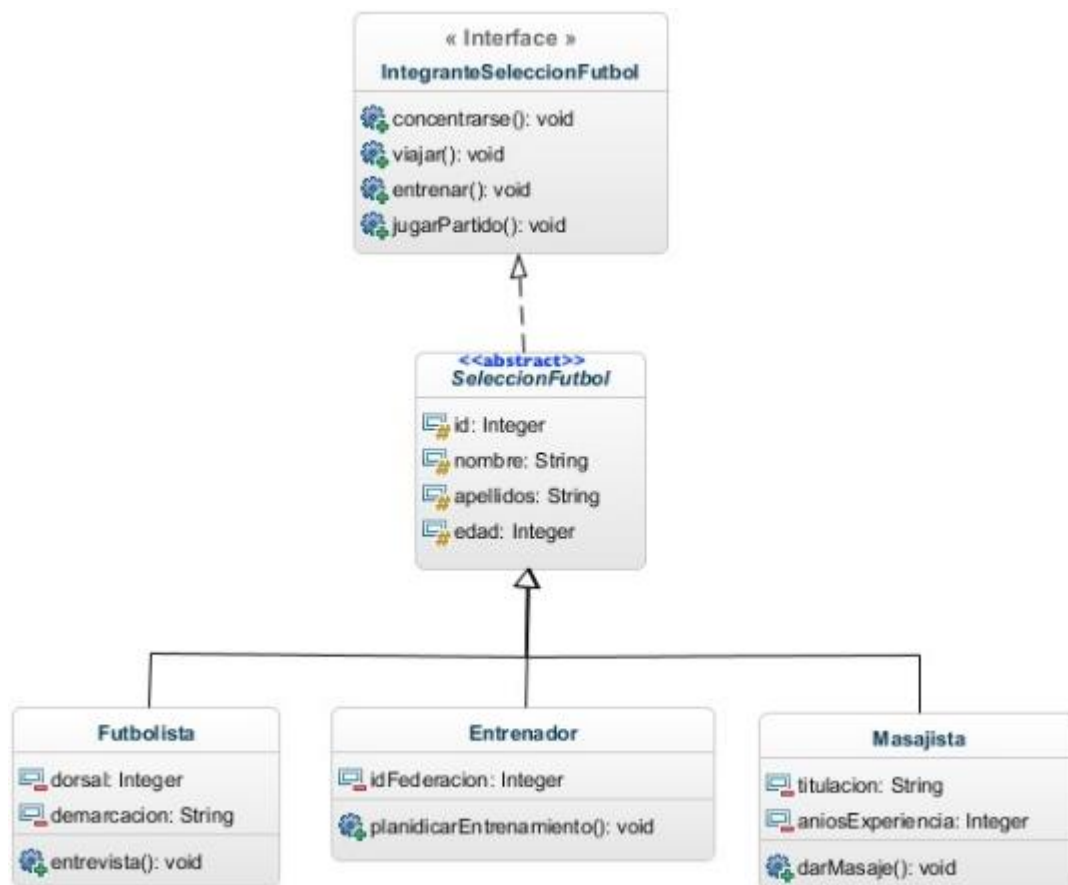
Para entender mejor los modificadores de acceso, acudimos al siguiente video y replicamos lo que allí se explica:

Video: [\(94\) 67. Programación en Java || POO || Modificadores de acceso - YouTube](#)

Interfaces

El concepto de Interface lleva un paso más allá el concepto de una clase abstracta en la que vimos que una clase abstracta es una clase que no se puede instanciar (crear un objeto de esa clase) pero sí se pueden definir atributos e implementar métodos en ella para que sus clases hijas los puedan utilizar. }

Una Interface es una clase abstracta pura en la que todos sus métodos son abstractos y por tanto no se pueden implementar en la clase Interface.



Lo primero que vemos es la clase "IntegranteSeleccionFutbol" que es una Interface en la que tiene definido cuatro métodos (es decir una clase abstracta pura). Como se ha dicho, sólo se definen los métodos pero no se implementan; así que a nivel de código la clase quedaría de la siguiente manera:

```

IntegranteSeleccionFutbol.java
1 package interfaces;
2
3 public interface IntegranteSeleccionFutbol {
4
5     void concentrarse();
6
7     void viajar();
8
9     void entrenar();
10
11     void jugarPartido();
12
13 }
  
```

Lo siguiente que vemos en el diagrama de clases es la clase "SeleccionFutbol" que es una clase abstracta que utiliza (implements) la Interface "IntegranteSeleccionFutbol". Al ser esta una clase abstracta no se puede instanciar y por tanto en ella no es necesario implementar los métodos de la "Interfece"; pero si será obligatorio implementarlo en sus clases hijas (Futbolista, Entrenador y Masajista).

