



El futuro digital
es de todos

MinTIC

ESTRUCTURAS DE CONTROL EN LENGUAJE JAVA



Universidad
Industrial de
Santander



Mision
TIC2022

1.1. Introducción

Java es un lenguaje de programación y a la vez una plataforma informática que fue distribuida por primera vez en 1995 por la empresa Sun Microsystems. Hay muchas aplicaciones y sitios web que dejarían de funcionar si no se tiene java instalado. Java es considerada una plataforma rápida, segura y fiable.

En el presente libro, se observará la estructura general y los comandos básicos que permiten la generación de códigos para la solución de problemas cotidianos. Realmente, tiene la misma esencia de otros lenguajes, es decir, también se declararán variables, se harán operaciones matemáticas y booleanas, se tomarán decisiones, se realizarán ciclos repetitivos o bucles, tal y como se hizo en la primera etapa del curso con el lenguaje Python.

El lenguaje Java se creó principalmente para:

1

Poder realizar una programación orientada a objetos.

2

Ejecutar un mismo programa en múltiples sistemas operativos.

3

Incluir soporte para trabajo en red.

4

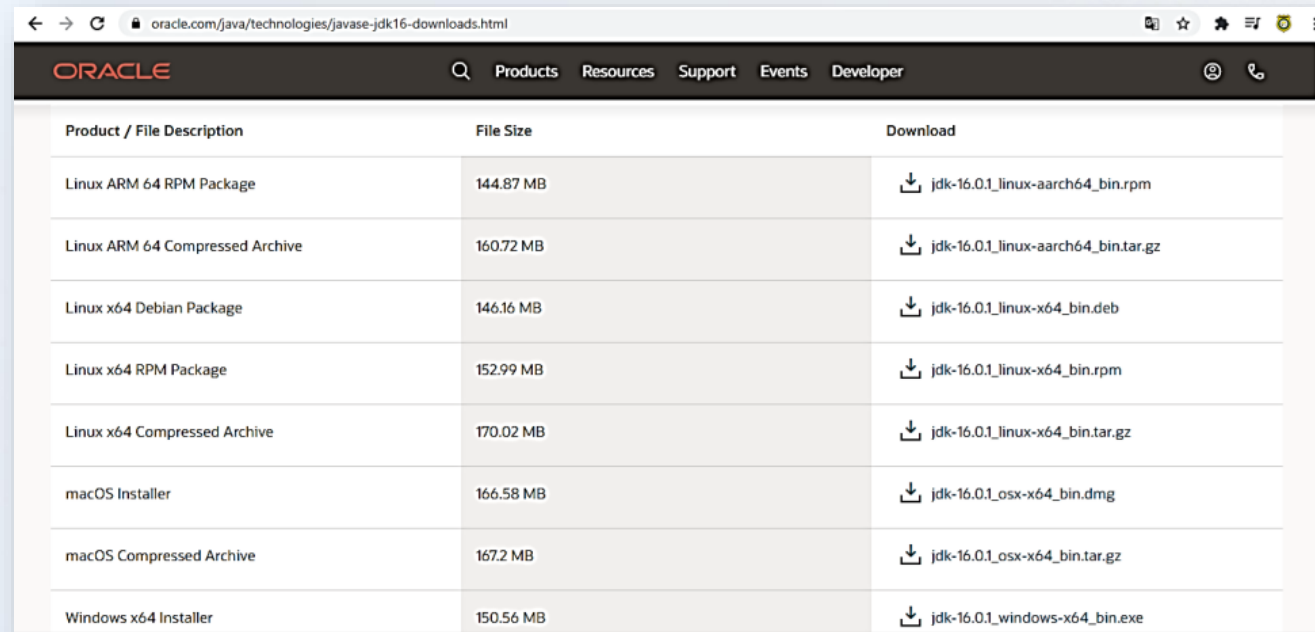
Tomar lo mejor de otros lenguajes orientados a objetos, como C++.

1.2. IDEs. (Netbeans. Instalación recomendada en versión 12.3)

Como primer paso se instala el java.

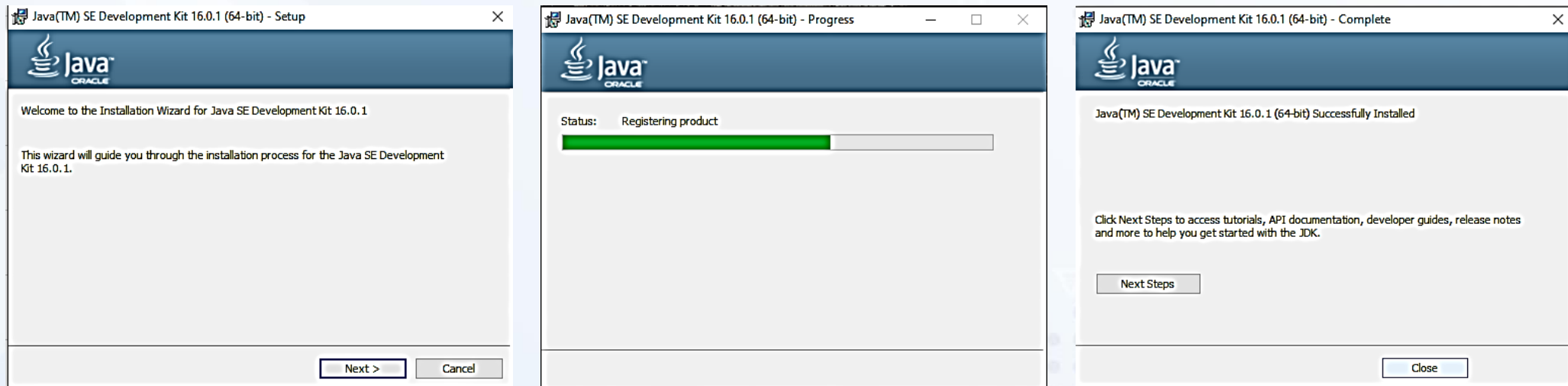
1. Se busca el instalador JDK 16 directamente de la página de Oracle, allí se selecciona la versión apropiada de acorde a su sistema operativo Windows, Linux o macOS

<https://www.oracle.com/co/java/technologies/javase-downloads.html>

A screenshot of a web browser displaying the Oracle JDK 16 download page. The browser's address bar shows the URL 'oracle.com/java/technologies/javase-jdk16-downloads.html'. The page features the Oracle logo and a navigation menu with links for Products, Resources, Support, Events, and Developer. Below the navigation bar is a table listing various download options for JDK 16.0.1 across different operating systems and architectures. The table has three columns: 'Product / File Description', 'File Size', and 'Download'. The download links are provided as direct file URLs.

Product / File Description	File Size	Download
Linux ARM 64 RPM Package	144.87 MB	jdk-16.0.1_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	160.72 MB	jdk-16.0.1_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	146.16 MB	jdk-16.0.1_linux-x64_bin.deb
Linux x64 RPM Package	152.99 MB	jdk-16.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive	170.02 MB	jdk-16.0.1_linux-x64_bin.tar.gz
macOS Installer	166.58 MB	jdk-16.0.1_osx-x64_bin.dmg
macOS Compressed Archive	167.2 MB	jdk-16.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	150.56 MB	jdk-16.0.1_windows-x64_bin.exe

2. Después de realizar la descarga, se lleva a cabo el proceso de instalación aceptando las condiciones sugeridas.



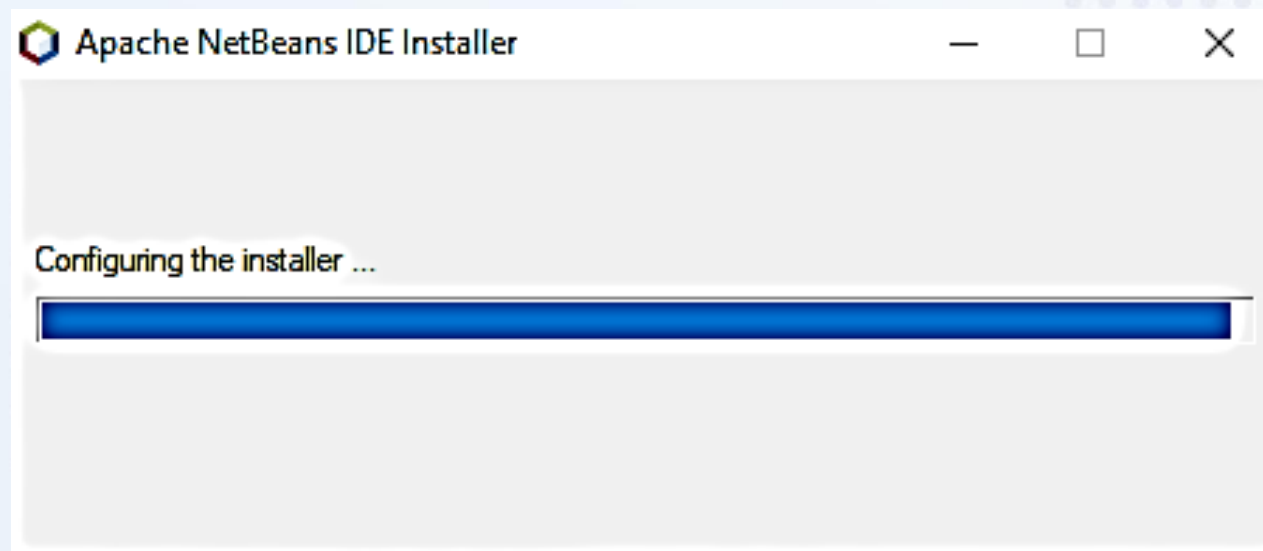
Puede saltar los tutoriales cerrando la pantalla (close) cuando finalice la instalación.

3. Una vez finalizada la instalación de Java, descarga la aplicación Netbeans 12.3 directamente de la página de apache

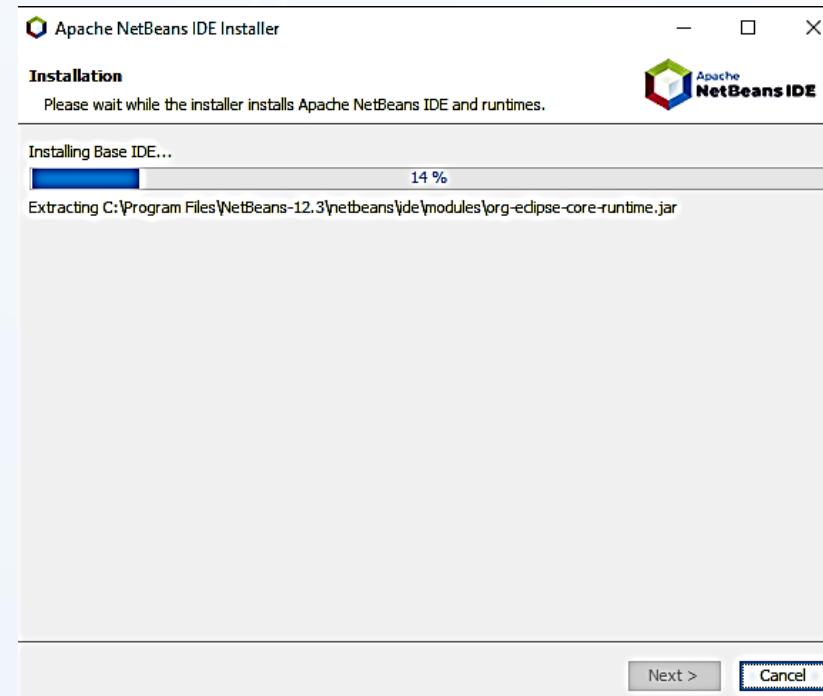
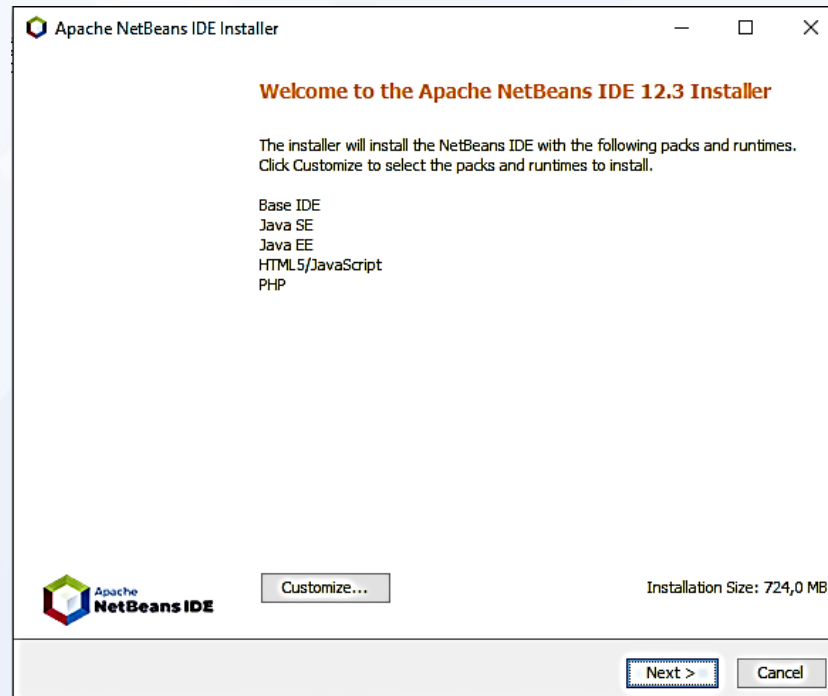
- [Apache-NetBeans-12.3-bin-windows-x64.exe](#) (SHA-512, PGP ASC)
- [Apache-NetBeans-12.3-bin-linux-x64.sh](#) (SHA-512, PGP ASC)
- [Apache-NetBeans-12.3-bin-macosx.dmg](#) (SHA-512, PGP ASC)

(Para macOS se requiere instalar primero [Swift 5 Runtime](#)).

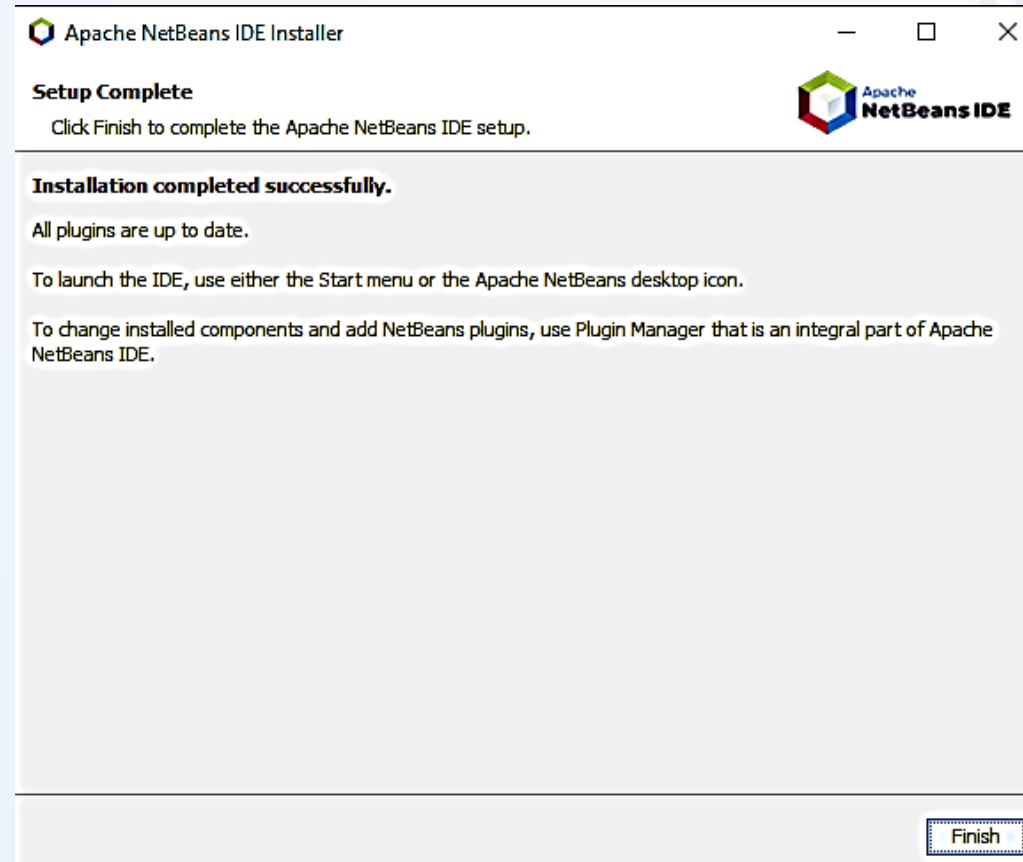
4. Ahora damos clic en el archivo descargado. En caso de que te pida permisos de administrador clicamos a "Sí".
5. Damos clic a "Instalar" y esperamos que inicie la instalación.



6. Durante la instalación se debe aceptar todas las condiciones sugeridas en el proceso de instalación



7. Seleccionamos para finalizar la opción Finish



1.3. Estructura básica de proyectos en Java

Para el caso de Java, se requiere de una estructura básica, la cual se debe colocar en todos los códigos que se deseen construir. Esta estructura consta de una CLASE PRINCIPAL y de un MÉTODO MAIN.

CLASE PRINCIPAL Y MÉTODO MAIN

Un programa puede construirse empleando múltiples *clases*. Sin embargo, como mínimo se debe tener declarada la clase principal, la cual contiene el programa, rutina o método principal: `main()`, en el cual se incluyen las sentencias del programa principal. Es muy importante separar las sentencias entre sí a través de un punto y coma.

La estructura de un programa simple en Java es la siguiente:

```
public class ClasePrincipal {  
    public static void main(String[] args) {  
        sentencia_1;  
        sentencia_2;  
        // ...  
        sentencia_N;  
    }  
}
```

1.4. Datos primitivos, cadenas, constantes

Los tipos de datos primitivos en Java son:

TIPO.	DESCRIPCIÓN.	TAMAÑO EN Bits.	RANGO DE VALORES
byte	entero corto	8 bits	- 128 a 127
short	entero	16 bits	-32,768 a 32,767
int	entero	32 bits	-2,147,483,648 a 2,147,483,647
long	largo	64 bits	-9,223,372,036,854,774,808 a
float	real	32 bits	-10 ³² a 10 ³²
double	Real (doble precisión)	64 bits	-10 ³⁰⁰ a 10 ³⁰⁰
boolean	lógico	1 bit	True o false

la declaración de variables en java, se realiza de la siguiente manera:

Tipo_de_varaible nombre_de_la_variable;

El *nombre de variable* debe especificarse atendiendo a las siguientes reglas:

Java diferencia las mayúsculas de las minúsculas.

Por ejemplo la variable «Nombre» es diferente a la variable «nombre».

Es importante tener en cuenta que el nombre de una variable no debe empezar por un número o carácter especial, tiene que empezar con una letra, un subrayado (_) o el signo de dólar (\$). después, es decir, a partir del segundo carácter, sí podrán incluirse números y otros caracteres.

Los nombres utilizados para las variables no tienen un tamaño máximo determinado; sin embargo, se recomienda que sean pequeños y de fácil entendimiento.

Lo más recomendado es utilizar minúsculas a menos que sean varias palabras en una sola. Por ejemplo, la variable «cuotaAnual» ó «balanceDelAño».

Las <strings> son un tipo de variables en **cadena**. Se declaran igual que una variable y representan la inserción de texto dentro de dicha variable. A diferencia de las otras variables, no permite operaciones aritméticas entre ellas.

Una **constante** se utiliza cuando un valor es repetitivo durante un código de programación. En java se declara anteponiendo la palabra clave **final**, la cual puede ir acompañada de comando **static** si desea hacer que la constante sea estática. La constante se declara con la siguiente estructura:

Para constantes:

```
final Tipo IDENTIFICADOR = valor;
```

```
// En el ámbito de una clase, o es miembro de la clase.
```

Para constantes estáticas.

```
static final Tipo IDENTIFICADOR = valor;
```

```
// En el ámbito de una clase, o es miembro de la clase.
```


1.5. Entrada y salida de datos

ENTRADA DE DATOS

Para la entrada de datos se tienen varias posibilidades, la primera de ellas lo hace a través del teclado y las otras por medio del uso de clases.

1. Mediante el teclado
2. Utilizando la Clase Buffered Reader
3. Utilizando la Clase Scanner
4. Utilizando la Clase Console

Veamos ejemplos de aplicaciones de cada una de las diferentes opciones posibles:

Mediante el teclado se usa `System.console().readLine()`

Ejemplo:

```
class EntradaTexto {  
    public static void main(String[] args) {  
        String nombre;  
        System.out.print("Por favor, dime tu nombre: ");  
        nombre = System.console().readLine();  
        System.out.println("Hola " + nombre + ", ¡bienvenido a Java desde Cero!");  
    }  
}
```



Utilizando la Clase Buffered Reader

Ejemplo:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class LecturaDatos
{
    public static void main(String[] args) throws IOException
    {
        //Ingresar los datos usando BufferedReader
        BufferedReader reader =
            new BufferedReader(new InputStreamReader(System.in));

        // Lee los datos usando readLine
        String name = reader.readLine();

        // Imprime la línea de lectura
        System.out.println(name);
    }
}
```

Utilizando la Clase Scanner

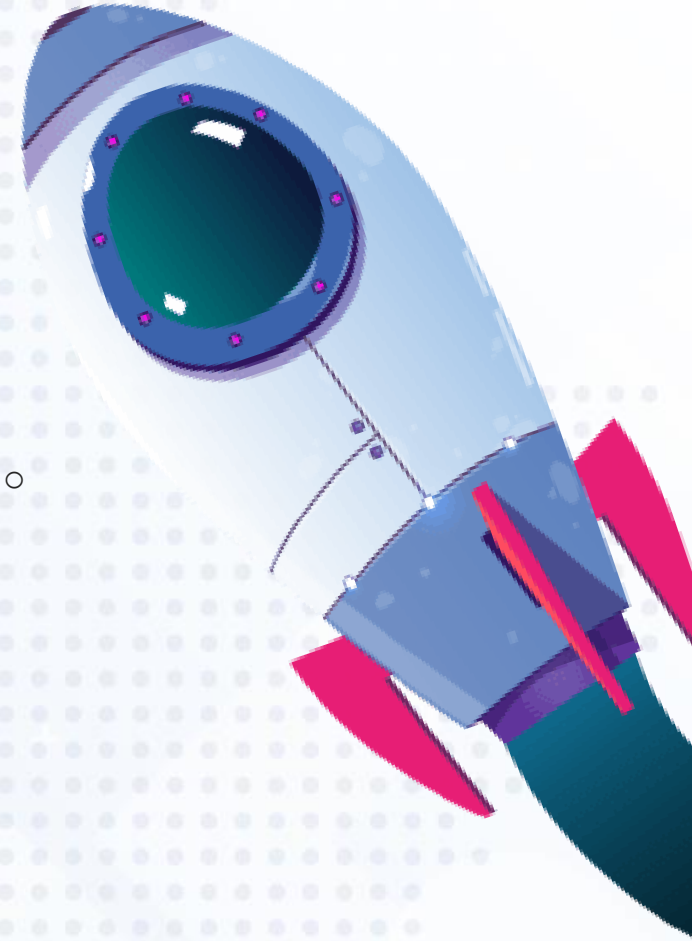
Ejemplo:

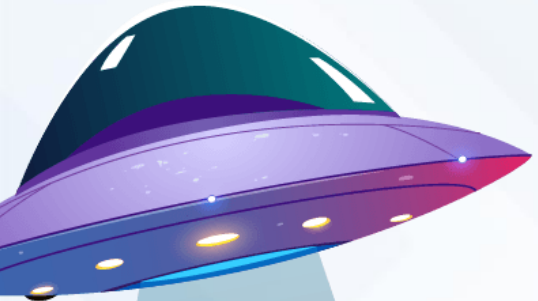
```
class ObtenerEntradaUsuario
{
    public static void main(String args[])
    {
        // Con el uso de Scanner se puede obtener información del usuario
        Scanner in = new Scanner(System.in);

        String s = in.nextLine();
        System.out.println("Usted ha ingresado la cadena: "+s);

        int a = in.nextInt();
        System.out.println("Usted ha ingresado un entero: "+a);

        float b = in.nextFloat();
        System.out.println("Usted ha ingresado un float: "+b);
    }
}
```





Utilizando la Clase Console

Ejemplo:

```
public class Sample
{
    public static void main(String[] args)
    {
        String name = System.console().readLine();
        System.out.println(name);
    }
}
```

SALIDA DE DATOS

Para la salida de datos se utiliza:

```
System.out.println();
```

Como ejemplo sencillo se tiene el siguiente programa denominado Hola:

```
/*  
La clase hola construye un programa que  
muestra un mensaje en pantalla  
*/  
public class Hola {  
    public static void main(String[] args) {  
        System.out.println("Hola, ");  
        System.out.println("Esta es una clase de MinTic20211");  
        System.out.println("Hasta luego");  
    }  
}
```

En el ejemplo anterior se puede observar que el comando `/*` permite agregar comentarios hasta que finalice con el comando `*/`. También se puede utilizar `//` para comentarios de una sola línea



1.6. Operadores aritméticos y lógicos

Los operadores aritméticos son similares a los manejados en Python

OPERADOR	USO
+	Suma.
-	Resta.
*	Multiplicación.
/	División.
%	Resto de la división o Módulo.
++	Incremento.
--	Decremento.



Como ejemplo del uso del comando ++ se tiene:

si se tiene:

```
int dato=3;  
dato ++;  
//ahora dato valdrá 4
```

Es decir, dato ++; equivale a colocar dato = dato + 1;

También se puede realizar lo siguiente

```
b= 2;  
a = b ++;
```

La variable a tomará el valor de 3

El operador — funciona igual al operador ++, pero decrementando.

A continuación, se muestran los operadores lógicos usados para generar comparaciones y generar resultados verdaderos o falsos.

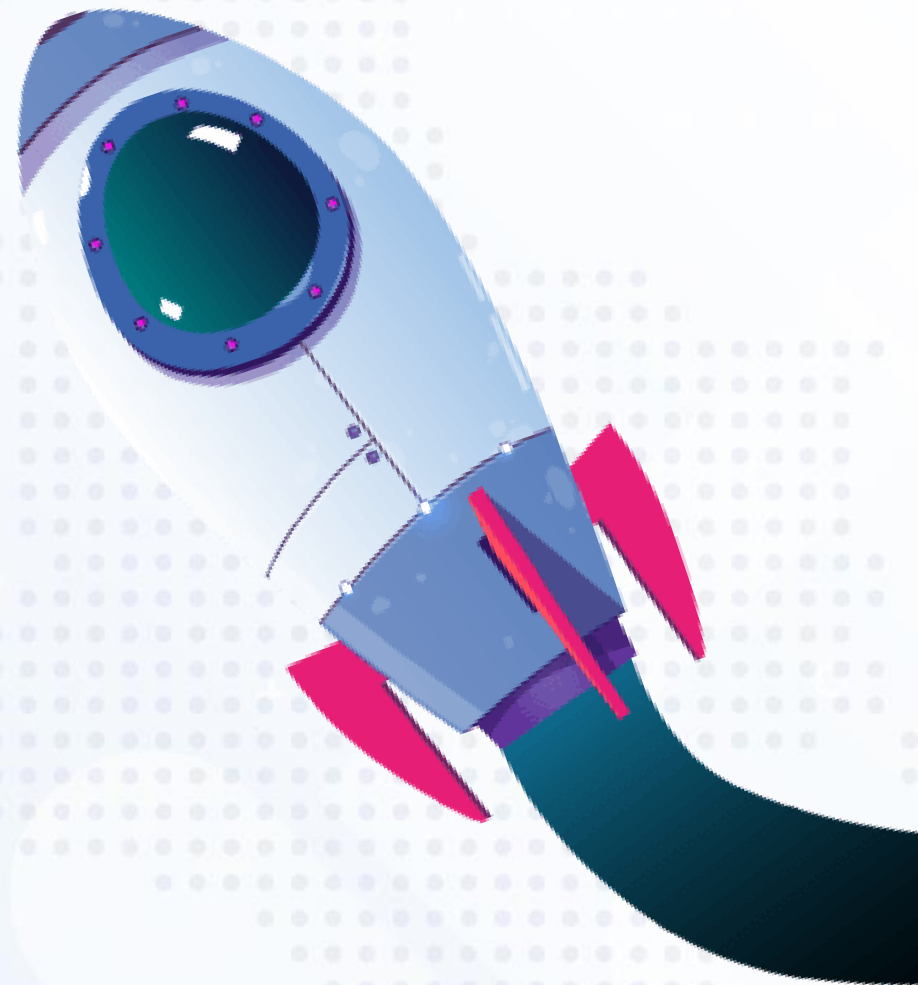
Operador.	Uso.
==	Igual que...
!=	Distinto que...
<	Menor que...
<=	Menor o igual que...
>	Mayor que...
>=	Mayor o igual que...

1.7. Condicionales

1.7.1. Condicional if

Ahora observaremos cómo se realiza en Java el Condicional if. Su sintaxis es la siguiente:

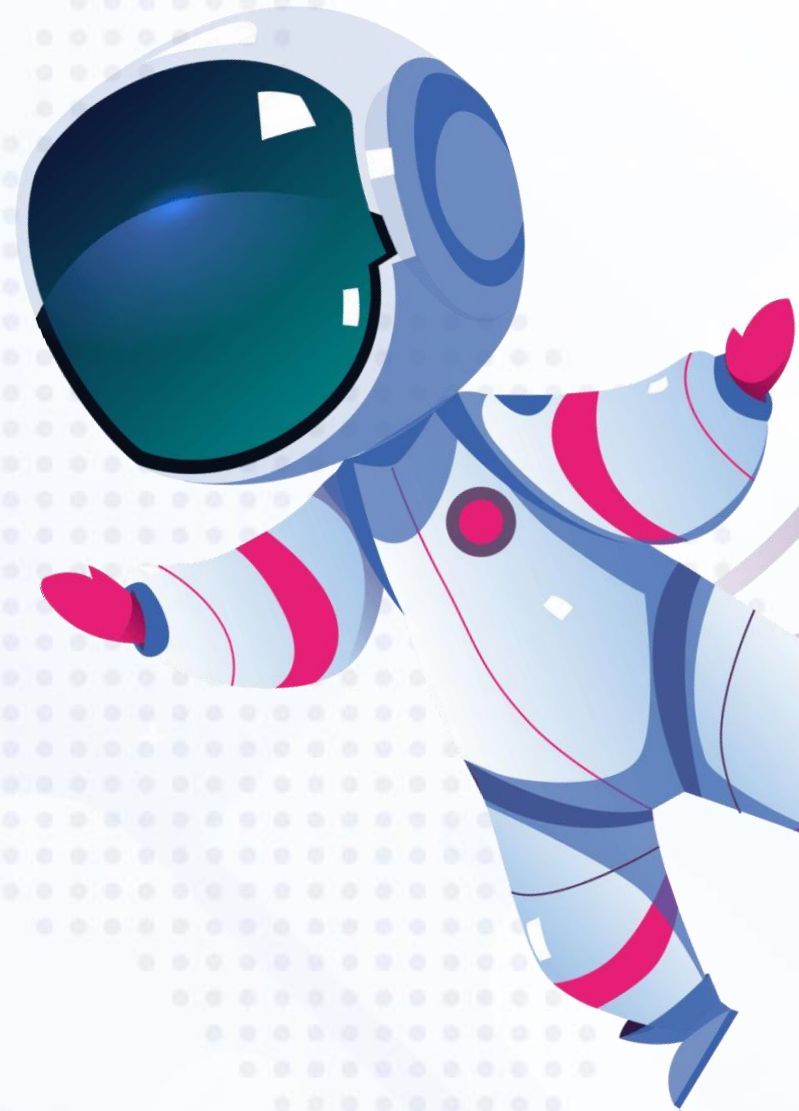
```
if ( Condición )  
{  
  Sentencias;  
}
```

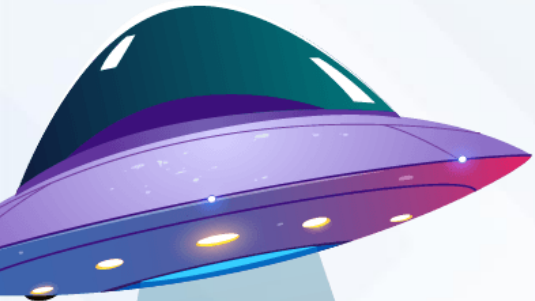


1.7.2. Condicional if-else

Una variante de la instrucción anterior es if else. Su sintaxis es la siguiente:

```
if ( Condición ) {  
  
Sentencias; // Si condición es cierta.  
} else {  
  
Sentencias; // Si condición es falsa.  
}
```





1.7.3. Condicional if-else if

Si una condición no se cumple se crea una nueva condición. Su sintaxis es la siguiente:

```
if ( Condición1 ) {  
    Sentencias;  
} else if ( Condición2 ) { // Si condición1 es falsa. Nueva  
condición  
    Una sentencia;  
} else if ( Condición3 ){ // Si condición2 es falsa. Nueva  
condición  
    Una sentencia;  
} else { // Si condición3 es falsa  
    Una sentencia;  
} // Llave de cierre final
```


1.7.4. Anidación de sentencias condicionales if ó if else

La multiplicidad de sentencias condicionales lleva a crear códigos con varios if con su else respectivos, tal y como se observa en la siguiente estructura:

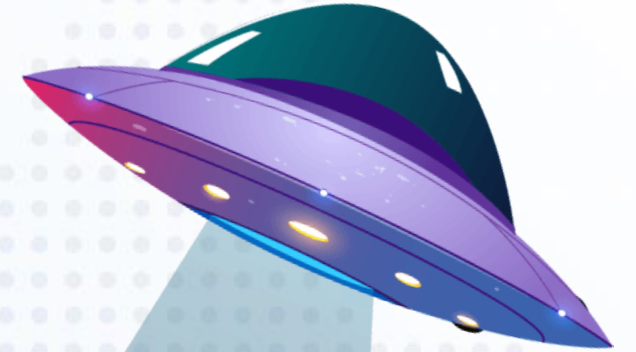
```
if ( Condición1 ) {  
  Sentencias;  
} else { // Si condición1 es falsa.  
  if ( Condición2 ) {  
    Sentencias;  
  } else { // Si condición2 es falsa.  
    if ( Condición3 ) {  
      Sentencias;  
    } else { // Si condición3 es falsa.  
      Sentencias;  
    }  
  }  
} // Llave de cierre final.
```

1.8. Bucles

1.8.1. Bucle mientras (While)

El bucle while es un ciclo de comandos que se cumplen siempre y cuando se cumpla una condición. En Java se puede expresar mediante la siguiente estructura:

```
while (condición)
{
    sentencias o instrucciones
}
```





1.8.2. Bucle para (for)

El bucle for ejecuta un bloque de instrucciones una cantidad de veces definida.

La estructura en Java se puede expresar como:

```
for ( Inic. de variables. ;condición;increment./decrement. ) {  
    sentencias o instrucciones;  
}  
Este es el aspecto normal de un bucle for:  
for ( i = 1 ;i <= 10  
    ; i++) {  
    sentencias o instrucciones;  
}
```