Técnico en

DESARROLLO DE SOFTWARE

Programación Orientada a Objetos I

Programación Orientada a Objetos I

Unidad I

1. Conceptos básicos

Clase

Son modelos que permiten mostrar el constructor del objeto, el estado y el comportamiento de cada uno de los objetos al momento de ser creados. Cada vez que se construye un objeto de una clase, se construye una instancia de esa clase.

Una instancia es una variable de tipo objeto. Una clase puede tener muchas instancias y cada una es un objeto diferente.

Objeto

Un objeto consta de un estado y de un comportamiento que tienen datos almacenados y tareas que realizan en tiempo de ejecución. Cada uno de los objetos creados son capaces de recibir mensajes, procesar datos y de interconectarse con otros objetos creados en el mismo programa.

Atributo

También son conocidos como propiedades de los objetos. Son las características de los objetos. Al definir un atributo, normalmente se especifica su nombre y su tipo. Los atributos son las variables donde se almacenarán los datos relacionados con los objetos.

Método

Es la funcionalidad asociada al objeto. Son las acciones que puede llevar a cabo el objeto.

Ejemplo:

Tenemos a nuestra clase ***perro***, en la cual vamos a definir la construcción de nuestro objeto ***perro***, el cual tiene un estado y un comportamiento. Nuestro objeto ***perro*** tiene varios atributos, entre ellos podemos mencionar que tiene un nombre, tiene un color, tiene una raza que lo distingue de los demás y puede tener más, pero para nuestro ejemplo bastan estos. Además tiene varias acciones que lleva a cabo nuestro objeto ***perro*** tales como: ladrar, correr, comer, jugar, truco y los que nosotros queramos definir.

2. Pilares de la programación orientada a objetos

Abstracción

Consiste en captar las características específicas de un objeto, así como su comportamiento con los atributos que se consideren necesarios. En la abstracción no es necesario entrar a tanto detalle, solo es necesario indicar “¿Qué es?” y “¿Qué hace?” pero no es necesario colocar “¿Cómo lo hace?”.

Ejemplo:

Continuando con el ejemplo anterior, vamos a continuar con nuestra clase ***perro*** y nuestro objeto ***perro***. Al expresarnos de la abstracción de un objeto nos referimos a los atributos y métodos de nuestro objeto, sus características específicas.

Objeto: Perro

Atributos

nombre

color

raza

Métodos

ladrar

correr

comer

jugar

truco

Encapsulamiento

Es una propiedad que ayuda a mantener juntos en una única entidad a los atributos o propiedades y los métodos o funciones que definen el comportamiento del objeto. También nos permite evitar que cualquiera pueda modificar los atributos del objeto de forma directa, a menos que lo lleve a cabo mediante métodos.

Estas propiedades son:

- **public**: los atributos del objeto se pueden modificar en cualquier clase, en cualquier programa.

- **protected**: los atributos del objeto se pueden modificar desde desde cualquier clase del mismo paquete, pero no en otras clases de otros paquetes.
- **private**: los atributos del objeto se pueden modificar únicamente en su clase donde se encuentran definidos.

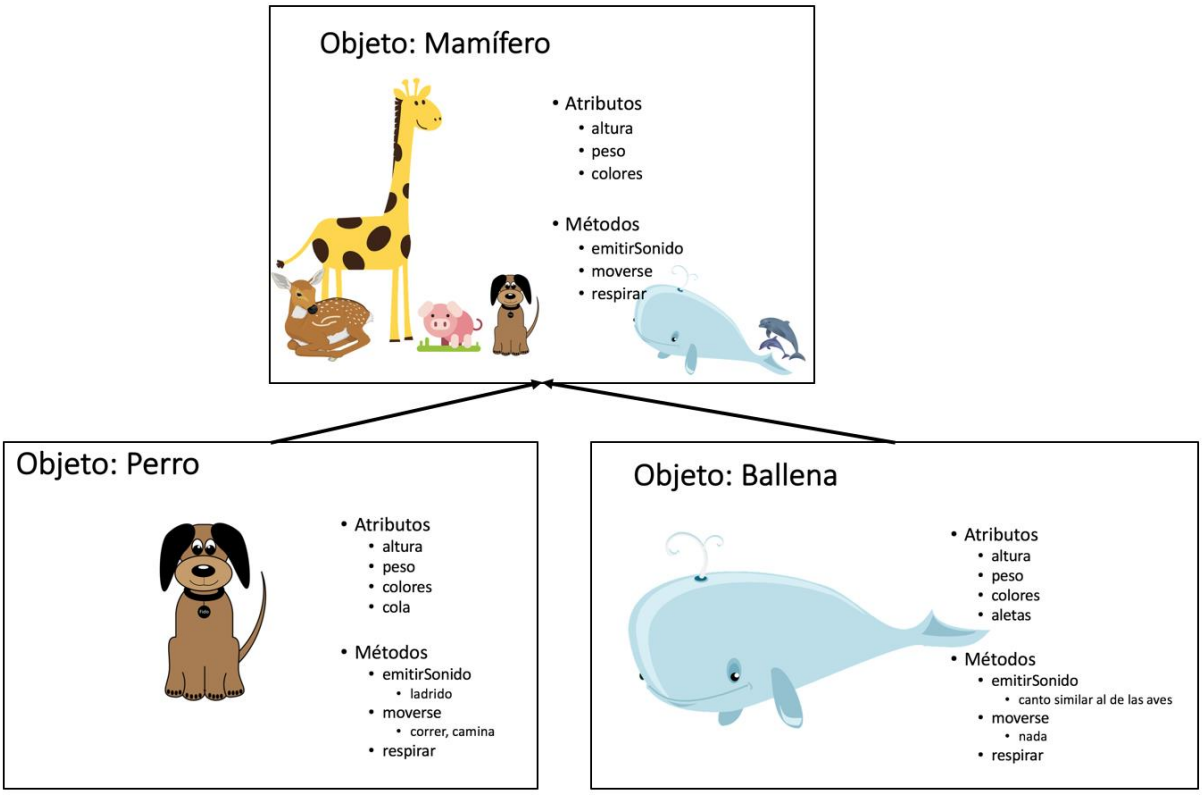
Más adelante estaremos viendo ejemplos de encapsulamiento.

Herencia

Es una propiedad que permite que los objetos sean creados a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a los ya existentes. Al heredar métodos, estos pueden ser iguales a los métodos de su clase padre o pueden modificarse para mejorarlos. Más adelante vamos a entrar en detalle a la herencia.

Ejemplo:

Si tenemos nuestra clase ***mamífero***, pero queremos crear una clase por cada tipo de animal mamífero que existe, entonces al crear nuestras clases ***perro*** y ***ballena*** que son heredadas de ***mamífero***, las clases ***perro*** y ***ballena*** están heredando todos los atributos y métodos de la clase ***mamífero***, pero tienen la posibilidad de que ***perro*** tenga nuevos atributos y métodos que lo van a diferenciar de ***ballena*** y de cualquier otra subclase heredada de mamífero.



Polimorfismo

En la programación orientada a objetos, las subclases de una clase pueden definir sus propios comportamientos únicos y aún así compartir algunas de las mismas funcionalidades de la clase principal.

Ejemplo:

Si continuamos con nuestro ejemplo, la clase ***mamífero*** tiene diferentes métodos, pero si nos enfocamos en el método ***emitirSonido***, podemos notar que la acción que ejecuta este método para ***perro*** es distinta a la acción que ejecuta el método con el mismo nombre para ***ballena***, teniendo para ambos un resultado distinto, a pesar de que la acción se llame de la misma forma.

3. Java

Java es un lenguaje de programación orientado a objetos, compilado, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico creado en 1995 por Sun Microsystems el cual está basado en C y C++. Desde el 2010, Sun Microsystems le pertenece a Oracle. Su extensión es ".java" y la extensión de sus archivos ya compilados es ".class".

Tipos de datos

A continuación se muestran los distintos tipos de datos que maneja Java:

	Nombre	Tipo	Tamaño	Rango
Tipos Primitivos (No poseen métodos, no son objetos y no necesitan de una invocación para ser creados)	byte	Entero	1 byte	-128 a 127
	short	Entero	2 bytes	-32,768 a 32,767
	int	Entero	4 bytes	-2^{31} a $2^{31}-1$
	long	Entero	8 bytes	-2^{63} a $2^{63}-1$
	float	Decimal simple	4 bytes	floating point de 32 bits
	double	Decimal doble	8 bytes	floating point de 64 bits
	char	Carácter simple	2 bytes	'\u0000' o 0 al '\uffff' o 65,535
	boolean	Booleano	1 byte	true o false
Tipos Objeto (Tienen métodos y necesitan de una invocación para ser creados)	Tipos de la biblioteca estándar de Java	String, ArrayList, TreeSet, entre otros.		
	Tipos definidos por el programador /	Perro, Gato, Ballena, Alumno, Profesor, o el que ustedes vayan a definir en sus tareas.		

	usuario	
	arrays	Serie de elementos o formación tipo vector o matriz.
	Tipos wrapper: Equivalentes a los tipos primitivos pero como objetos.	Byte
		Short
		Integer
		Long
		Float
		Double
		Character
		Boolean

3.1. Sintaxis

Asignación de datos

La asignación de datos se maneja muy similar a como lo hemos manejado hasta el momento en JavaScript en los cursos de Lógica de la Programación 1 y 2, por lo que no tendrán inconveniente en la asignación de datos y en las distintas operaciones y expresiones de comparación o en los ciclos. A continuación algunos ejemplos:

Tipo primitivo: int numero = 9;

Tipo Objeto: Perro maggie = new Perro (“Maggie”, 8, “Beagle”); //nuevo objeto tipo Perro asignado a la variable maggie.

Para la asignación de datos, siempre es necesario colocar el tipo de dato de nuestra variable al inicio, luego colocamos el nombre de la variable; como siguiente paso colocamos el signo de igualdad (=) y por último colocamos el valor a asignar a la variable.

Comentarios

Para colocar comentarios de una sola línea colocamos dos diagonales // previo al inicio de nuestro comentario. Ejemplo:

```
// Este es un comentario de una línea
```

Ahora bien, si queremos colocar un comentario de varias líneas, entonces es necesario colocar el comentario de la siguiente forma:

```
/* Diagonal y asterisco al inicio de nuestro comentario
*  y al inicio de cada línea colocamos asterisco
*  por cada línea comentada y al final lo cerramos con asterisco y diagonal
*/
```

Operadores Booleanos

Seguiremos trabajando con los mismos operadores booleanos y los mismos operadores de comparación que utilizábamos en los cursos de Lógica de la Programación 1 y 2, ya que el manejo de las comparaciones es muy igual a JavaScript.

Condicional AND (&&):

Indica que la condición es verdadera si las dos o más expresiones a evaluar son verdaderas.

```
if (x < 18 && x > 14) ... // Si ambos son verdaderos, entonces entra a la condición.
```

Condicional OR (||):

Indica que la condición es verdadera si al menos una de las expresiones a evaluar es verdadera.

```
if (x < 18 || x > 50) … // Si uno de los dos es verdadero, entonces entra a la condición.
```

Existen más condicionales, pero estos dos son los más comunes. Si quiere aprender sobre más operaciones Booleanos o más operaciones en Java, puede revisar el libro "Java in a Nutshell" o buscar en la documentación oficial de Java, pero para este curso esto será suficiente.

Condiciones

Las condiciones las seguiremos trabajando al igual que en JavaScript, por lo que solo haremos un recordatorio de las mismas mediante un ejemplo:

- **If...else:**

```
if (edad > 18) {
    System.out.println("Es mayor de edad");
} else {
    System.out.println("Es menor de edad");
}
```

- **If..else if:**

```
if (edad > 18 && edad < 60) {
    System.out.println("Es mayor de edad");
} else if (edad >= 60) {
    System.out.println("Es de la tercera edad");
} else {
   System.out.println("Es menor de edad");
}
```

- **switch:**

```
switch (dia) {
case 0: {
    System.out.println("Hoy es domingo!");
    break;
}
case 1: {
    System.out.println("Hoy es lunes");
    break;
}
case 5: {
     System.out.println("Viernes!!!");
     break;
}
default: System.out.println("Dia no registrado");
}
```

Ciclos

La sintaxis de los ciclos la seguiremos utilizando tal y como lo hemos hecho en JavaScript, solo que ahora vamos a utilizar también el ciclo foreach, el cual va a ejecutarse para cada uno de los valores de un arreglo. A continuación un ejemplo de cada uno de los ciclos. No entraremos en detalle porque esto ya lo vimos en cursos anteriores.

- **While**

Ejecutar ciclo mientras se cumpla la función. Recordar cambiar el valor que condiciona para evitar ciclos infinitos.

```
int count = 0;
while (count <= 5 ) {
count++;
System.out.println(count);
}
```

- **Do … While**

Ejecutar ciclo al menos una vez y la segunda ejecución se llevará a cabo solo si se cumple la condición.

```
int contador = 1;
do {
    System.out.println(“Ingreso”);
} while (contador < 1);
```

- **For**

Ejecuta el ciclo mientras se cumpla la condición y puede llevar a cabo codigo al iniciar y ejecutar código por iteración.

```
for (int contador = 0; contador < 10; contador++) {
   System.out.println(“Imprimiendo “ + contador);
}
```

- **Foreach**

Ejecuta el ciclo mientras nuestra colección de objetos tenga datos y ejecuta las instrucciones que nosotros le indiquemos para cada uno de los datos de nuestra colección de datos.

```
String[] dias = {"Lunes", "Martes", "Miércoles", "Jueves", "Viernes",
"Sábado", "Domingo"};
for (String dia : dias) {
System.out.println(dia);
}
```

3.2. Estructura de una clase

Básicamente una clase está construida por los parámetros del objeto, el constructor de cada objeto y sus métodos. A continuación, se muestra la clase de ejemplo Perro.java:

```
/*
*  Clase Perro
*/

public class Perro {

  String nombre, color, raza;  // definición de atributos

  public Perro(String ingNombre, String ingColor, String ingRaza) {
//Constructor
    //asignación de valores a los atributos
    nombre = ingNombre;
    color = ingColor;
    raza = ingRaza;
  }
  public void setNombre(String name) {  //método 1 que no devuelve datos
        nombre = name;
  }
  public String getNombre() { //método 2 que devuelve un dato tipo String
        return nombre;
  }
}
```

Básicamente la clase es creada con la instrucción “public class Nombre”, para que la clase pueda ser pública, el nombre del archivo debe ser el mismo que el nombre de la clase.

Luego inicializamos los atributos agrupados (o no) por su tipo de dato.

Como siguiente punto definimos el constructor de nuestro objeto. Pueden ser uno o más constructores por objeto; pueden tener argumentos o no. Al tener argumentos, se sugiere que aquí se inicialice los valores de los atributos del objeto.

Luego ya podemos crear los métodos, los cuales están definidos de la siguiente forma:

```
[public/private/protected] [tipo de dato que nuestro método va a devolver o void en caso no vaya a devolver ningún dato] nombreMetodo(argumentos indicando el tipo de dato de cada argumento a recibir o paréntesis vacíos en caso no reciba argumentos) {
    Instrucciones a llevar a cabo
    return variable si fuera el caso
}
```

3.3. Creación y uso de objetos

Para el ejemplo anterior, si queremos crear un objeto Perro y manejar sus datos, debemos crear un objeto de la siguiente forma para poder manipular el objeto, manipular sus datos y ejecutar sus acciones:

```
//crear objeto mediante la llamada de su constructor el cual se llama Perro y recibe tres parámetros de tipo String. Al finalizar recordemos colocar punto y coma.

Perro perro1 = new Perro(“Max”, “negro”, “común”);
```

Para mandar a llamar a los métodos de la clase perro, simplemente colocamos el nombre de la variable seguida por un punto y luego el nombre del método. El método se debe mandar a llamar con los parámetros según esté implementado el método y en caso no tenga parámetros entonces se manda a llamar vacío. Ejemplo:

```
System.out.println("Su perro se llama " + perro1.getNombre());
```

El ejemplo anterior se encargará de imprimir en pantalla el texto “Su perro se llama” y a esto le va a concatenar el valor que devuelve el método “getNombre()” sin parámetros para el objeto perro1, que en este caso será “Max”, ya que en nuestro constructor dijimos que nuestro perro se llama “Max” y el método getNombre lo que hace es que devuelve el valor del atributo “nombre” de nuestro objeto.

Paquetes

En Java, un paquete contiene un conjunto de clases y sirve para agrupar las distintas partes de un programa. Para este curso no vamos a crear paquetes, pero si utilizaremos paquetes que nos brinda Java, por lo que en este curso utilizaremos los más comunes:

Ingreso y salida de datos: para poder utilizar el ingreso y egreso de datos, necesitamos importar la clase java.io, ya que esta nos brinda clases que nos permiten realizar la comunicación con el teclado y con la pantalla. Para el ingreso de datos del teclado también utilizaremos la clase Scanner del paquete java.util, por lo que para poder hacer uso de estas clases, necesitamos colocar lo siguiente al inicio de nuestra clase que ejecutará las solicitudes de ingreso o egreso de datos:

```
import java.io.BufferedOutputStream; //Salida de datos
import java.io.InputStream; //Ingreso de datos
import java.util.Scanner; //Ingreso de datos
```

Para imprimir en pantalla utilizaremos el siguiente comando el cual finaliza con un salto de línea:

```
System.out.println(“Texto a imprimir”);
```

Y para recibir del teclado crearemos un objeto tipo Scanner y luego mandaremos a llamar a System.in el cual recibe bytes del teclado:

```
Scanner scanner = new Scanner( System.in );
        System.out.println("Ingrese su nombre");
String nombre = scanner.nextLine();
```

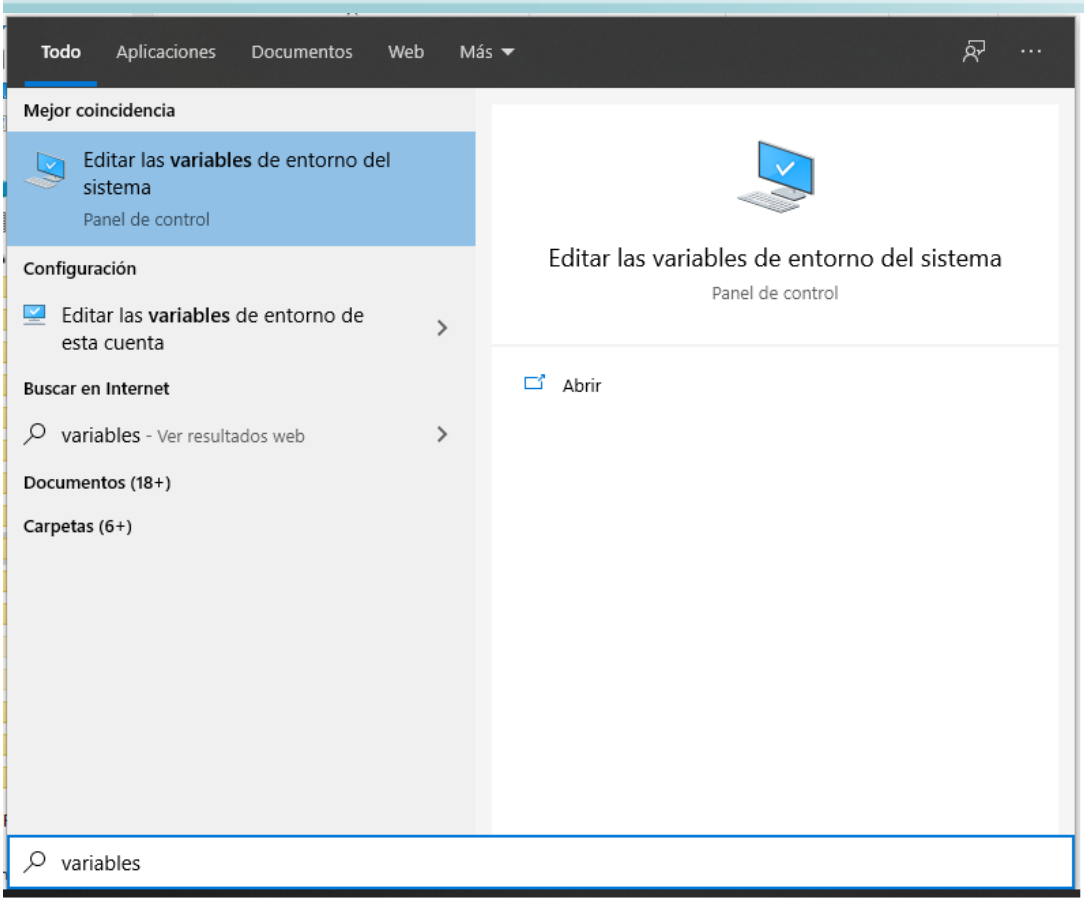
4. Instalando Java en Windows.

Primero que nada, hay que descargar el Java Development Kit, este incluye el Java Runtime Environment. Para eso vamos a dirigirnos a https://jdk.java.net/ aqui podemos descargar la versión que prefiramos, ya que no vamos a entrar a ver las muchas otras características de Java.

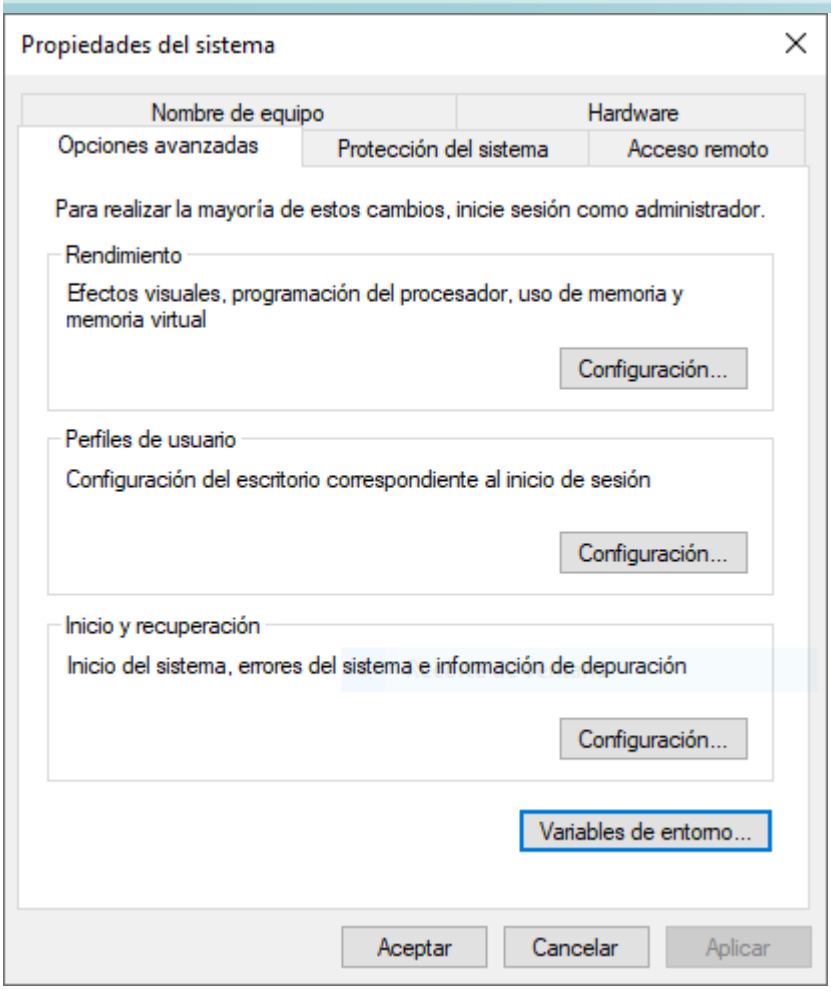
Las diferencias entre el JDK que nos ofrece OpenJDK y el que nos ofrece Oracle son mínimas, sin embargo una de las más importantes es que OpenJDK es totalmente Open Source, por lo que la podemos usar sin ningún tipo de licencia ni otros requerimientos.

Una vez que tengamos descargado el SDK, hay que extraer el contenido en una carpeta, como por ejemplo: C:\Java\jdk-14.0.1\

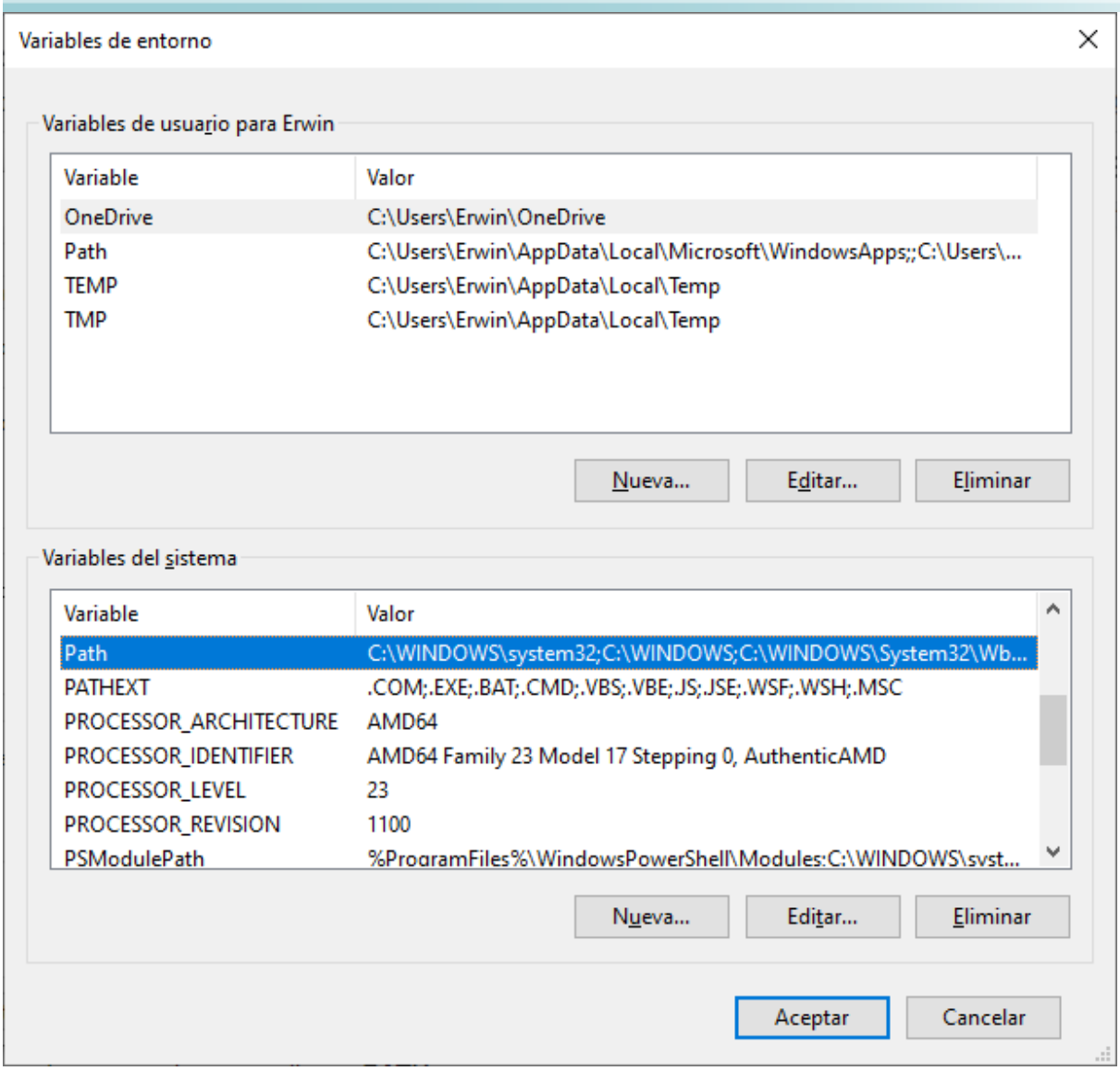
Ahora tenemos que configurar las variables de entorno para poder utilizar java desde cualquier sitio



Luego elegimos modificar las variables de entorno

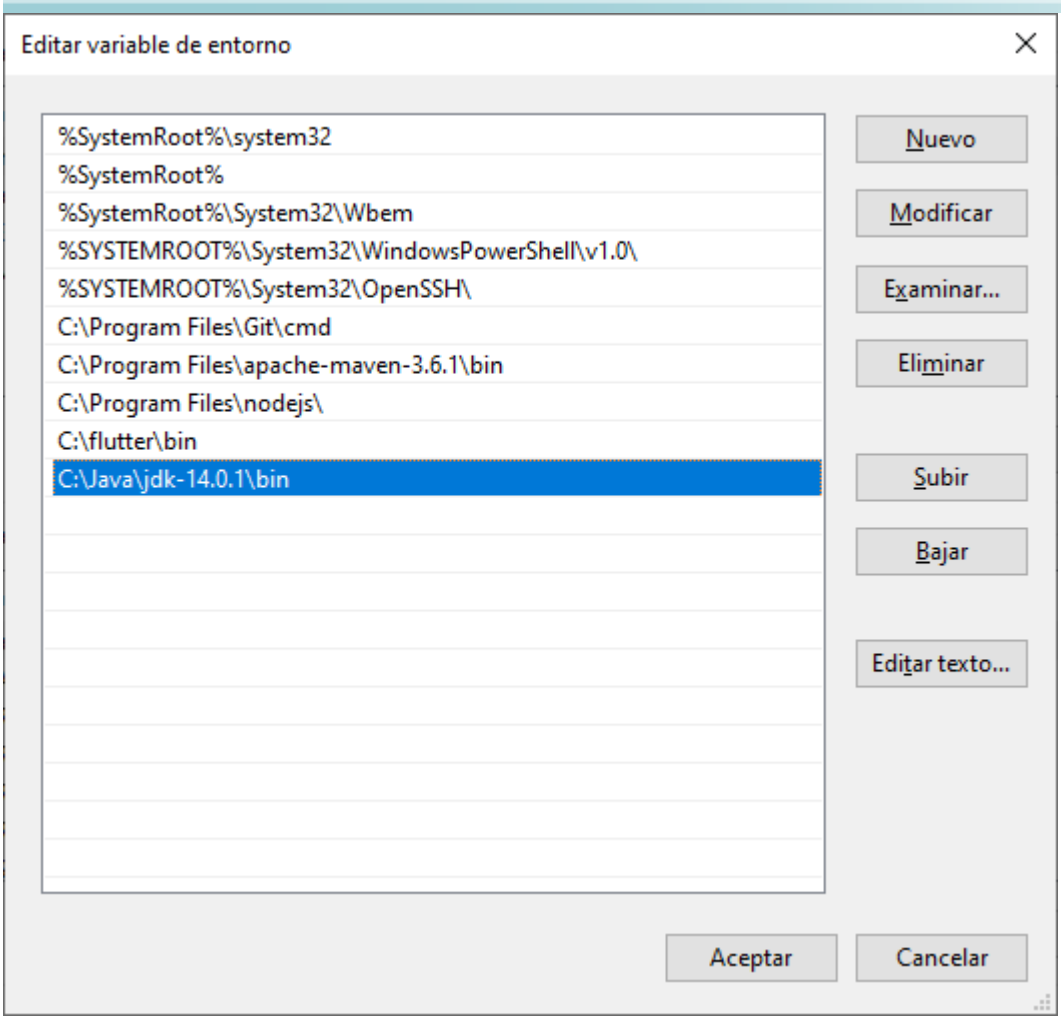


Ahora buscamos la que se llama ***PATH***



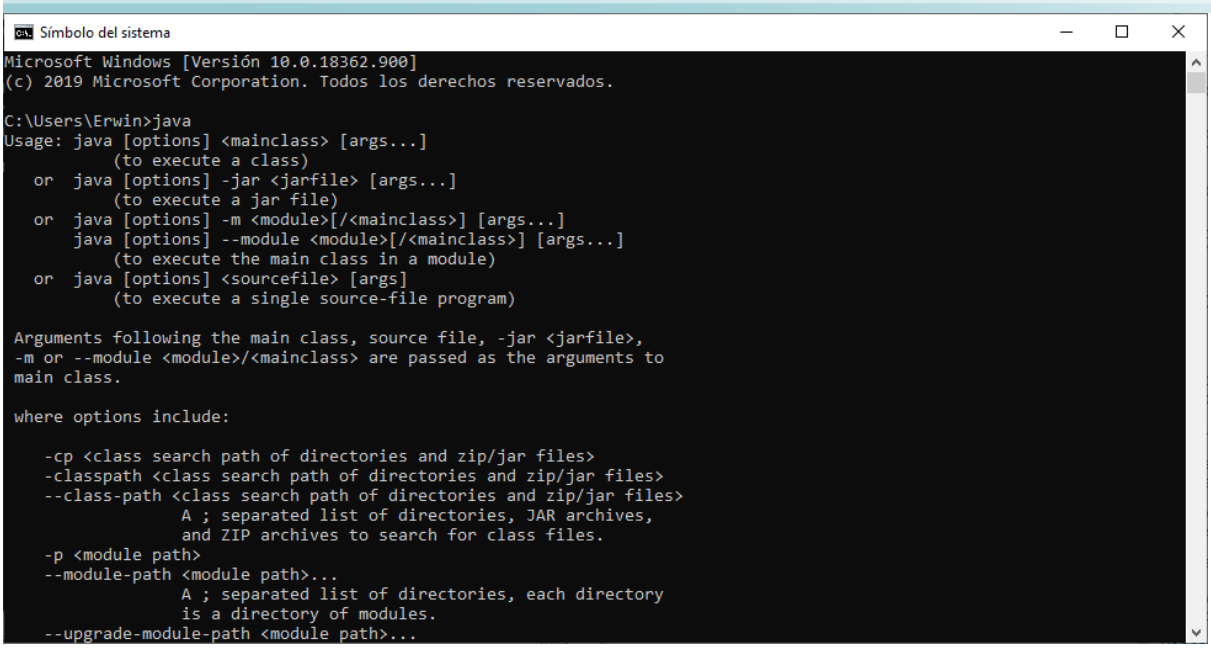
Elegimos ***Editar*** y luego ***Nuevo***

Nos mostrará una nueva línea en las direcciones, ahí debemos colocar la dirección a la carpeta bin de la descarga de Java. Por ejemplo C:\Java\jdk-14.0.1\bin

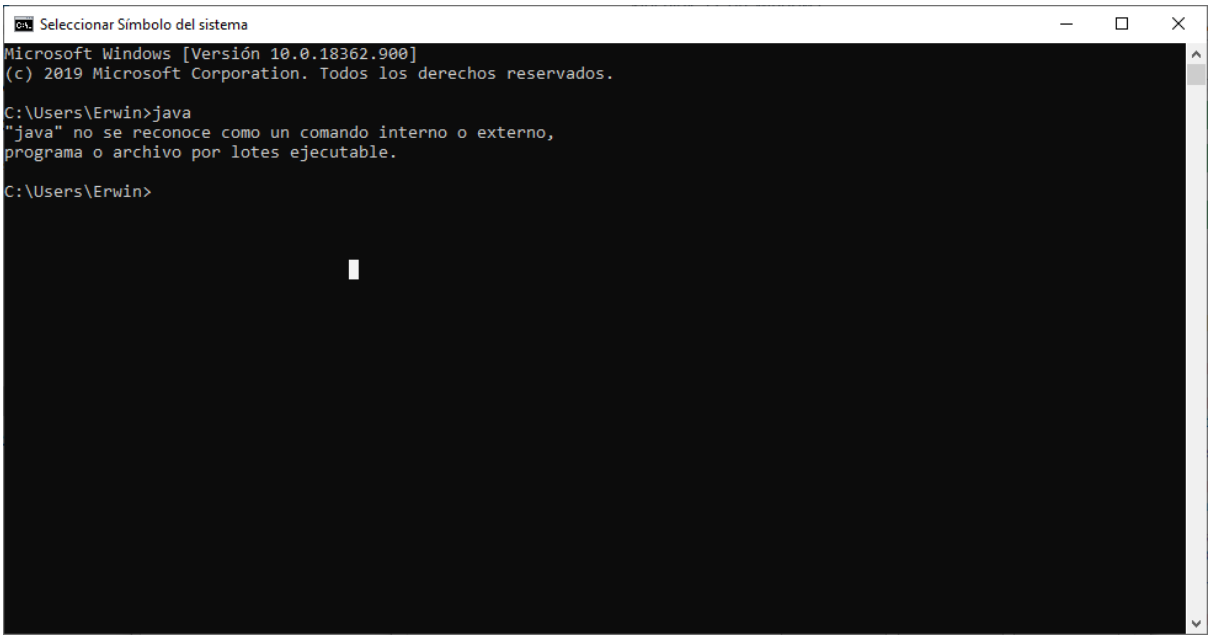


Luego de hacer clic en ***Aceptar.*** Vamos a crear una ***Nueva*** variable de entorno llamada ***JAVA_HOME***

Luego de aceptar y guardar los cambios, podemos abrir una línea de comando e ingresar el comando “java”.
Si lo configuramos bien veremos un mensaje parecido al siguiente



Si en cambio no está bien configurado, veremos un mensaje parecido al siguiente



En este caso hay que revisar que todos los pasos fueron ejecutados correctamente

Para más instrucciones en la instalación de Java en otros sistemas puede visitar https://openjdk.java.net/install/

Y si por alguna razón no puede realizar la instalación correctamente, puede visitar https://https://adoptopenjdk.net para descargar un instalador.

5. Instalando Java en Linux (En una máquina Virtual)

Linux es uno de los ambientes de desarrollo más utilizados en cuestiones de desarrollo.

Sin embargo, lo más común es que nuestras computadoras tengan alguna versión de Windows, que es el Sistema Operativo más común de todos.

Aquí vamos a ver cómo usar una máquina virtual para instalar Ubuntu, una distribución de Linux, y que tengamos un entorno en el cual poder trabajar.

Si ya tienen un sistema operativo de Ubuntu, pueden saltarse esta sección, el comando que necesitan para instalar Java es

```
sudo apt install default-jdk
```

cuando termine de cargar podemos revisar que la instalación sea correcta con el comando

```
java -version
```

Instalando una máquina VIrtual

Lo primero que hacemos es descargar una máquina virtual, como por ejemplo VirtualBox, disponible en https://www.virtualbox.org.

El siguiente paso es descargar una distribución de Linux, por ejemplo Ubuntu, disponible en https://ubuntu.com/download/desktop.

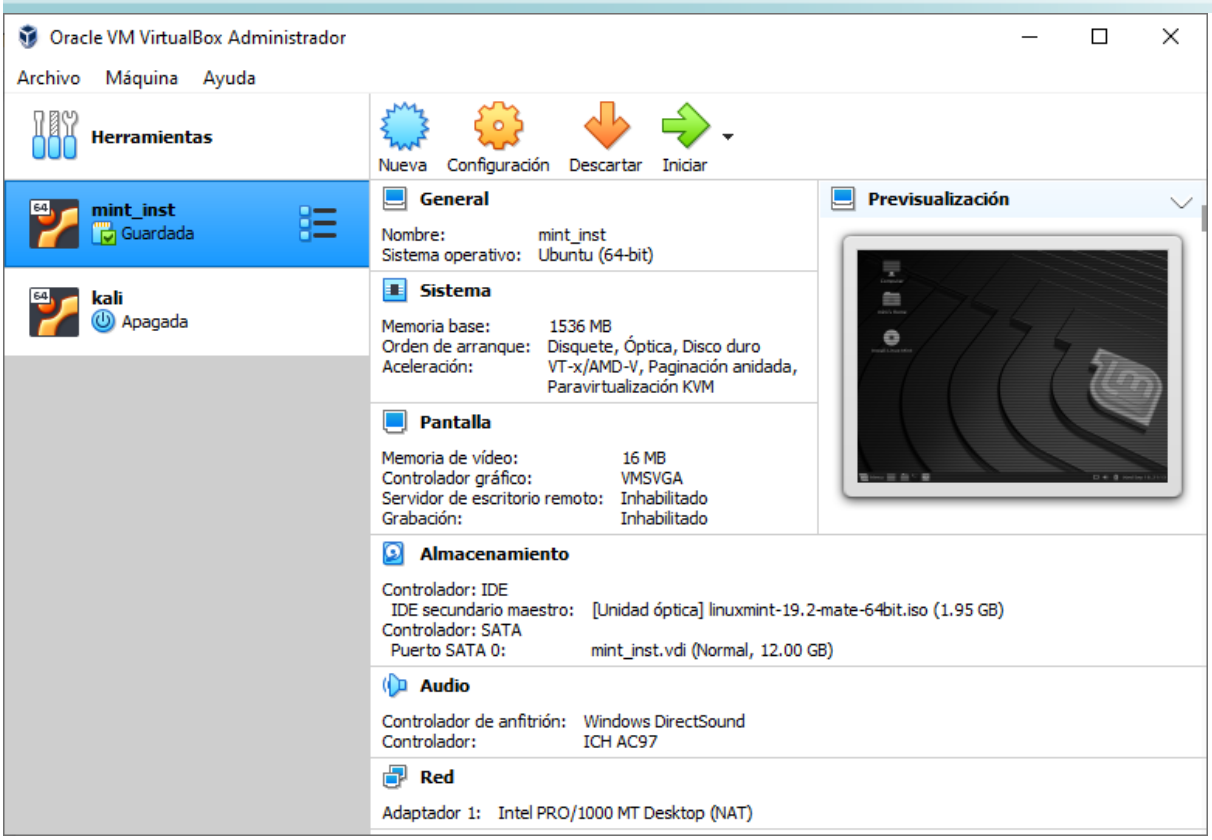
Estos archivos son bastante grandes, hablamos de múltiples GB, si la conexión de internet que utilizas no es estable, hay opciones de descarga, la más confiable es, probablemente, descargarlo como un Torrent.

Ahora vamos a instalar la máquina virtual, haciendo uso del instalador es bastante sencillo, no es necesario modificar los parámetros que trae por defecto.

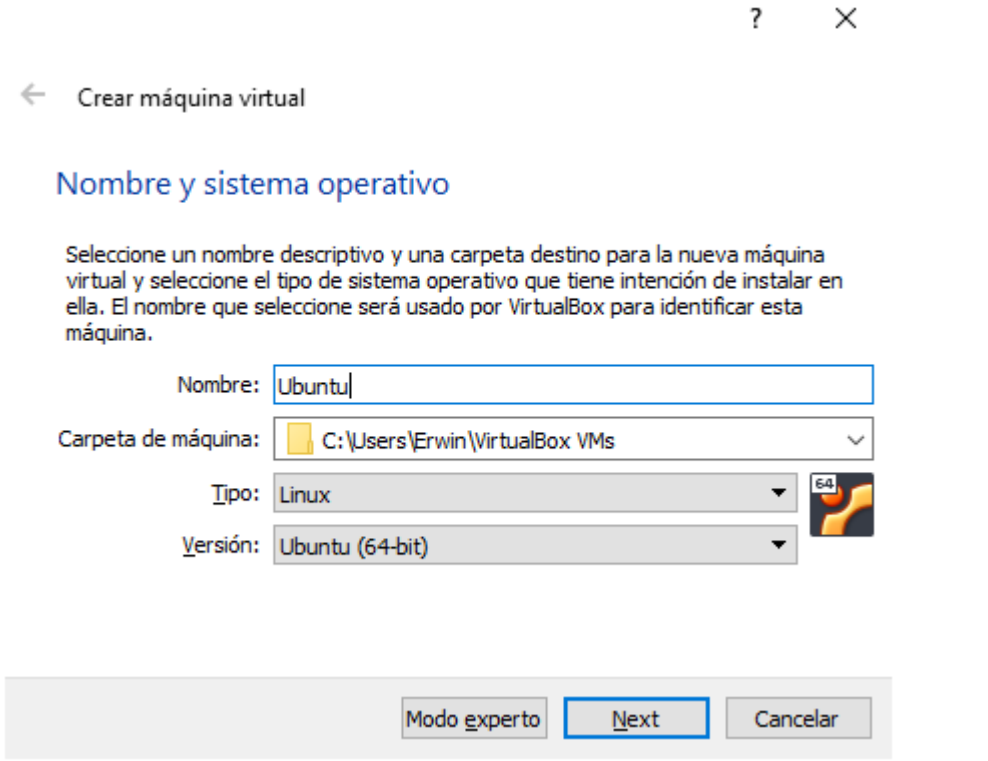
Ahora vamos a montar la imagen de Ubuntu en la máquina virtual. ¿Qué significa montar una imagen? Básicamente, instalar un sistema operativo en una máquina virtual.

Entonces, vamos a seguir los siguientes pasos.

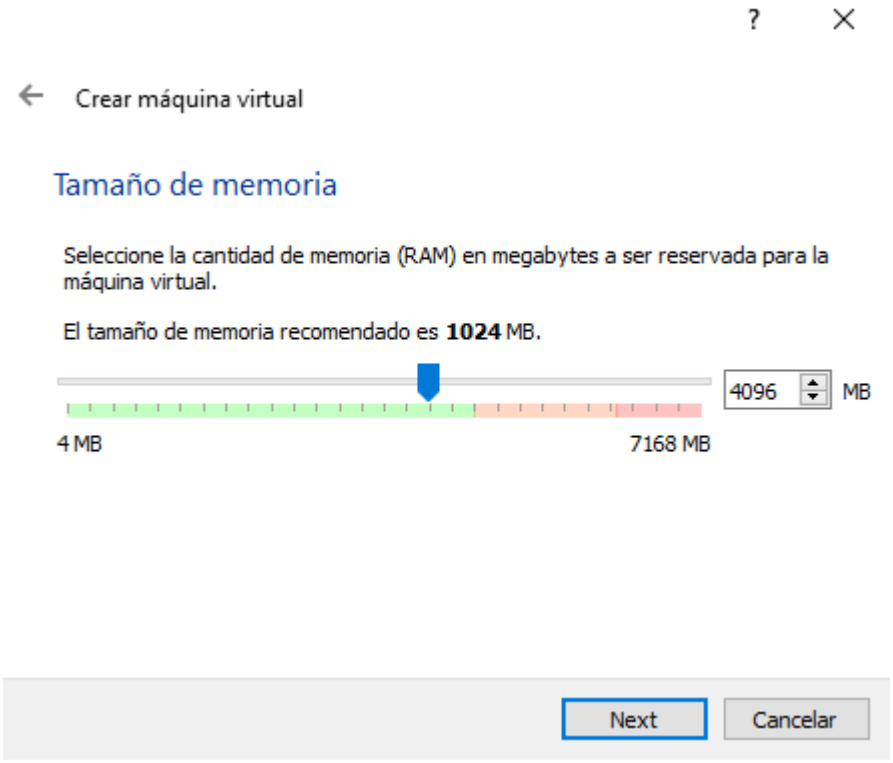
Primero seleccionamos Nueva máquina virtual



Luego tenemos que colocarle un nombre, y seleccionar un tipo de sistema operativo. Si colocamos de nombre Ubuntu, la máquina virtual nos seleccionará el tipo correcto.



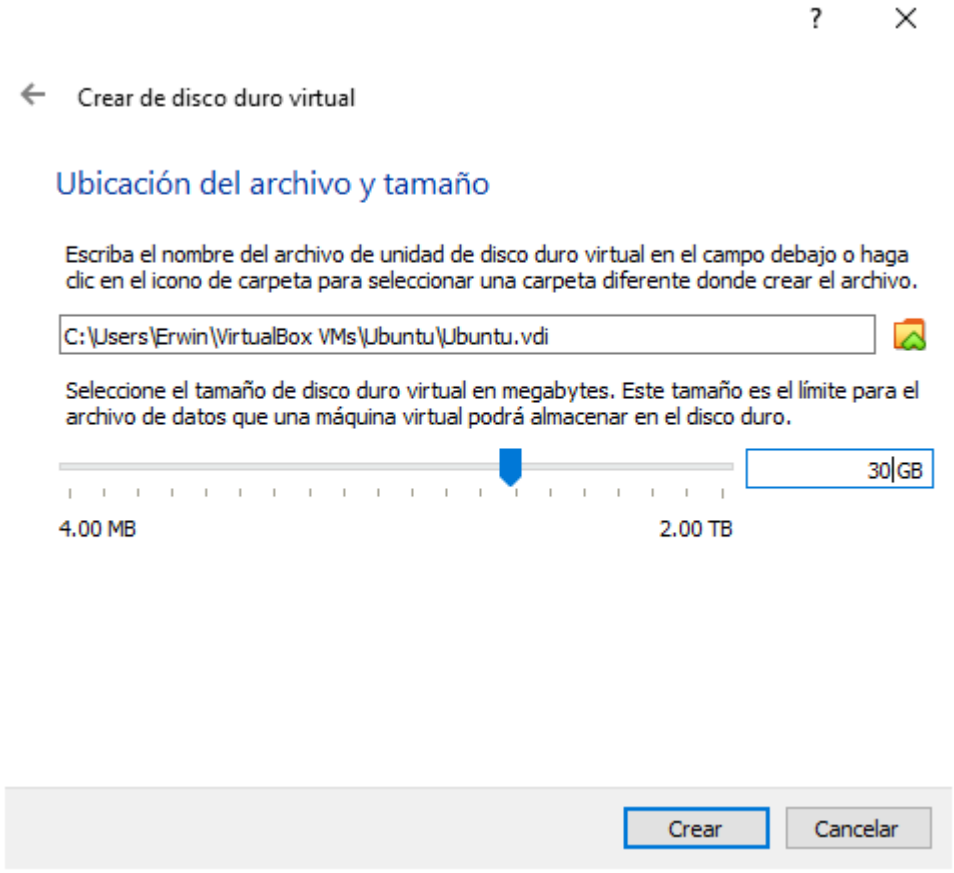
Le damos a Next.



Nos pedirá reservar memoria RAM para la máquina virtual. El recomendado de Ubuntu es de 4GB, sin embargo, reservar mucha memoria para la máquina virtual puede causar problemas en nuestra computadora cada vez que usemos la máquina virtual.

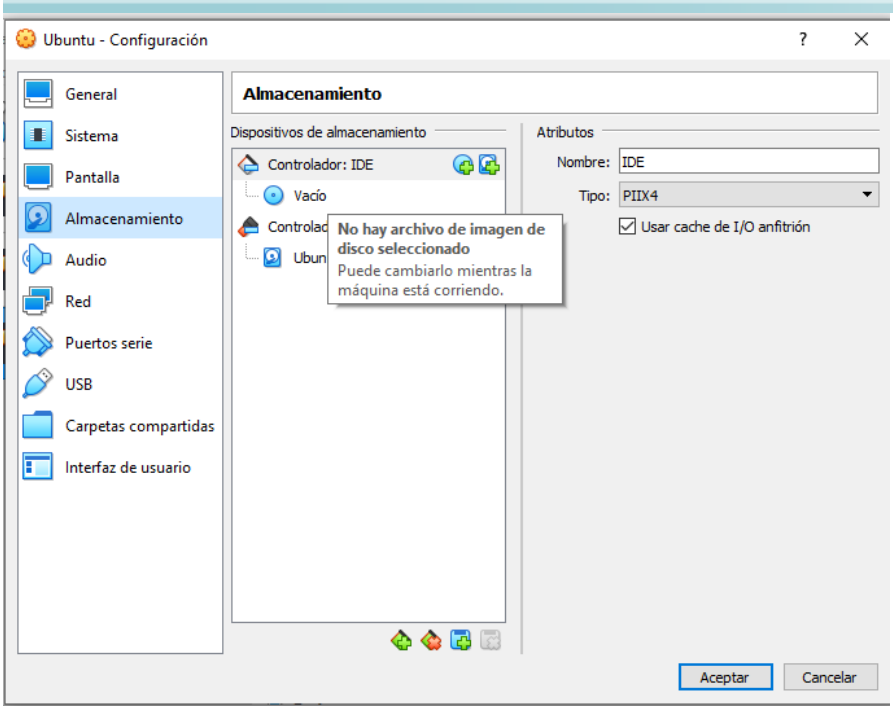
Se puede colocar un poco menos, pero el rendimiento de la máquina virtual se verá afectado.

Luego tenemos que crear un disco virtual, podemos dejar las opciones por defecto, hasta que nos pide el tamaño del disco que queremos crear. El recomendado de Ubuntu es de 25GB.

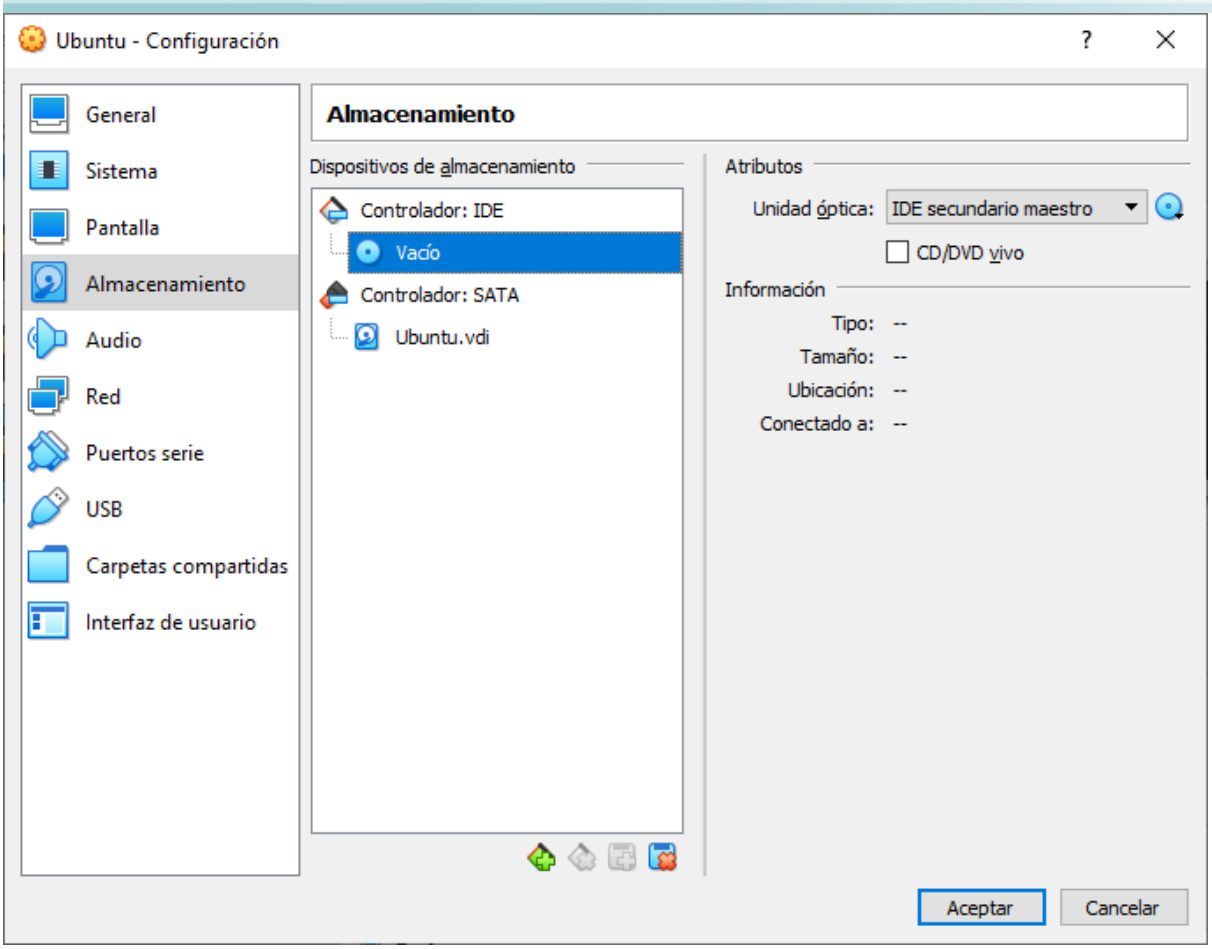


Le damos a Crear y aparecerá en el menú principal. Ahora vamos a montar la imagen del sistema operativo que descargamos con anterioridad.

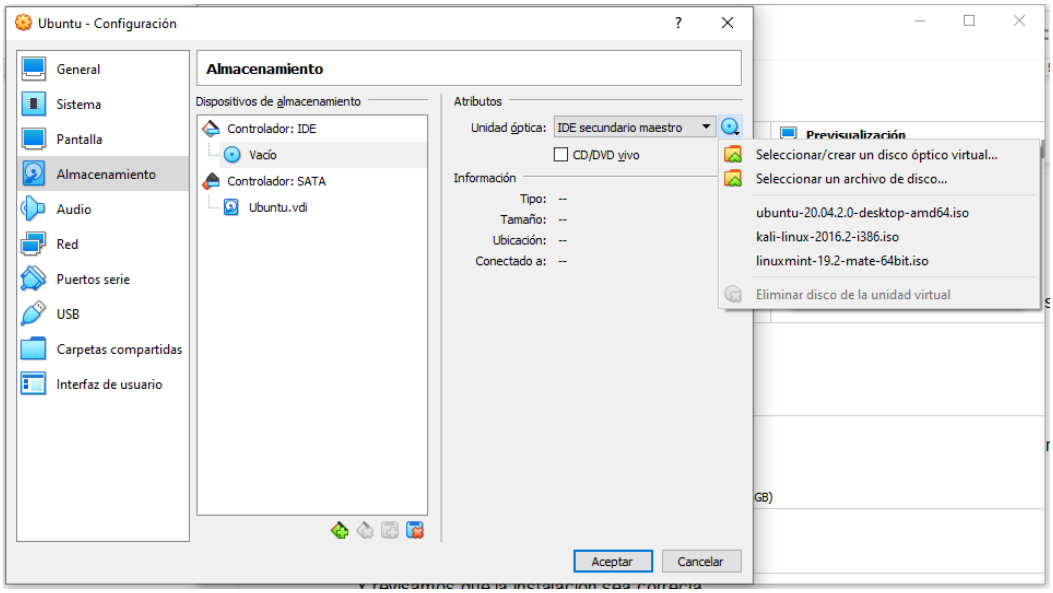
Elegimos la máquina que acabamos de crear y le damos clic al botón de configuración. En el menú derecho vamos a *Almacenamiento*. Ahí veremos algo como lo siguiente



Si colocamos el cursor sobre donde dice *Vacío,* podemos ver que dice que no hay un archivo seleccionado. Hacemos clic sobre donde dice *Vacío* y nos mostrará lo siguiente



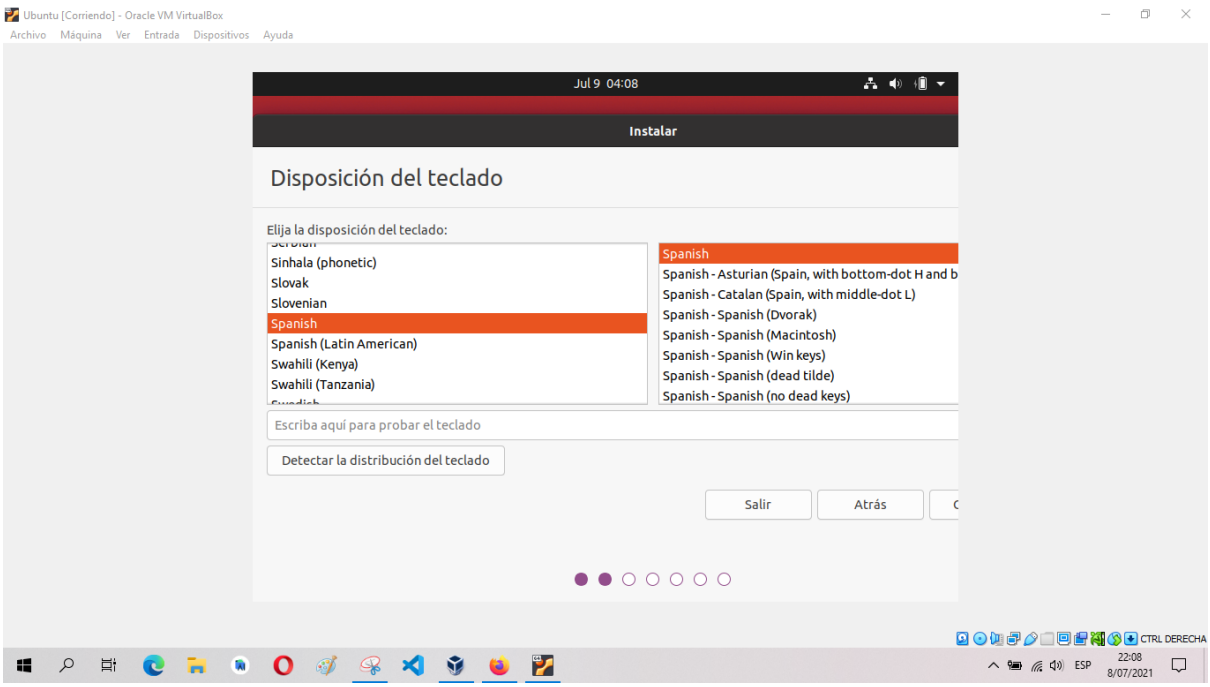
Hacemos clic en el icono de disco que está a la derecha en la sección de Atributos. Nos desplegará un menú y elegimos *Seleccionar un archivo de Disco.*



Ahora buscamos y seleccionamos el archivo de Ubuntu que descargamos con anterioridad. Le damos a *Aceptar* y ahora sí podemos Iniciar la máquina virtual.

Aquí seleccionamos nuestro idioma y elegimos la opción de Instalar Ubuntu

Si seleccionamos el idioma español, vemos que la ventana es muy pequeña y no muestra el contenido completo. Esto lo solucionaremos más adelante. Por ahora podemos arrastrar la ventana para ver los botones y seleccionar el botón de continuar.



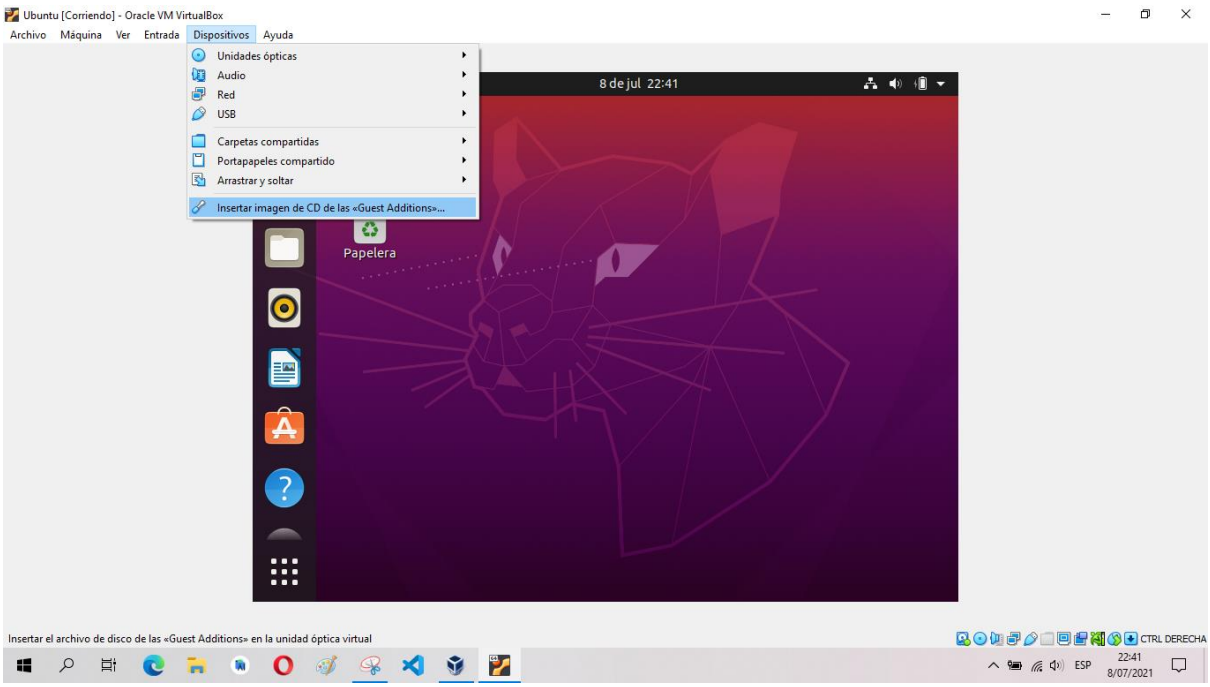
No es necesario que cambiemos ninguna de las configuraciones.

En este momento nos va a decir que si queremos borrar los archivos del disco duro para instalar Ubuntu, no hay problema en hacer esto, está haciendo uso del espacio que habíamos reservado cuando montamos la imagen. Nuestros archivos estarán a salvo.

La instalación puede tomar bastante tiempo, así que hay que tener paciencia.

Configurando nuestra máquina virtual e Instalando Java

Ahora sí vamos a arreglar la cuestión de la pantalla. Primero que nada tenemos que iniciar sesión, una vez que nos encontremos en el escritorio lo que tenemos que hacer es seleccionar en el menú: Dispositivos > Insertar el disco de <<Guest Additions>>



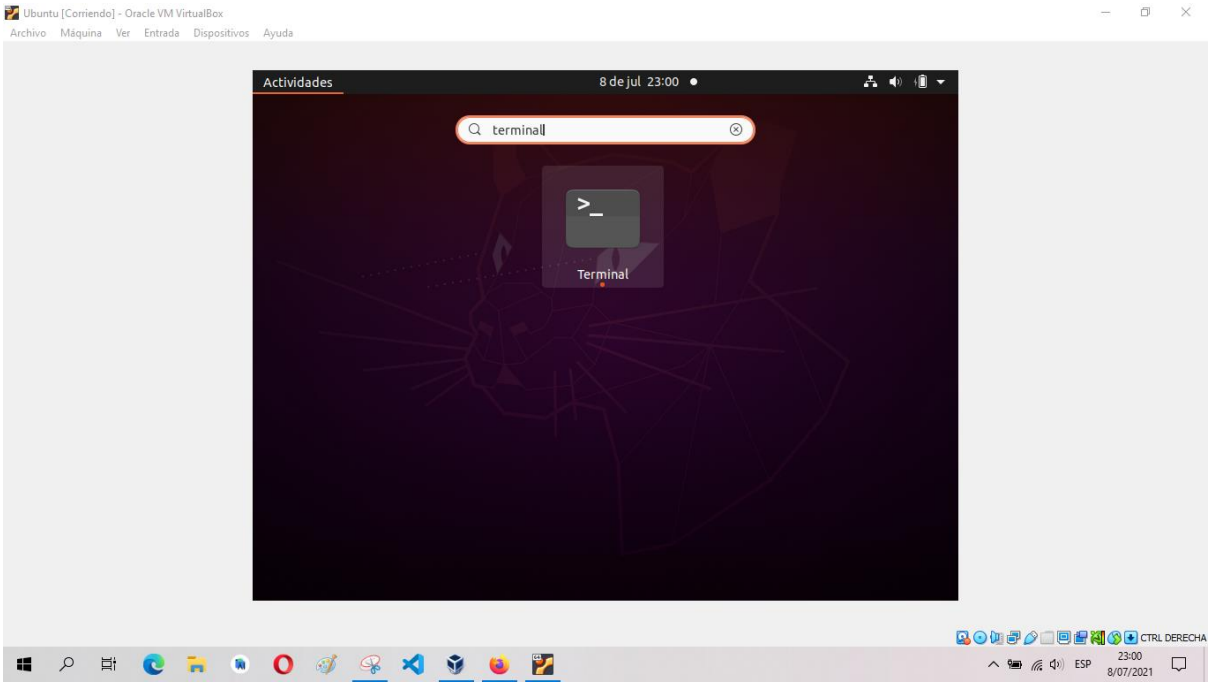
Nos mostrará una advertencia, le damos a ejecutar. Nos pedirá que ingresemos la contraseña y luego procederá a instalar las configuraciones necesarias.

Si les da un error parecido al siguiente

```
Verifying archive integrity... All good.
Uncompressing VirtualBox 6.1.22 Guest Additions for Linux.....
VirtualBox Guest Additions installer
Copying additional installer modules ...
Installing additional modules ...
VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel modules.
This system is currently not set up to build kernel modules.
```

```
Please install the gcc make perl packages from your distribution.
VirtualBox Guest Additions: Running kernel modules will not be replaced until the system is restarted
VirtualBox Guest Additions: Starting.
Press Return to close this window...
```

Lo que tenemos que hacer es abrir una terminal (Abajo a la derecha para mostrar todas las aplicaciones y escribimos terminal)

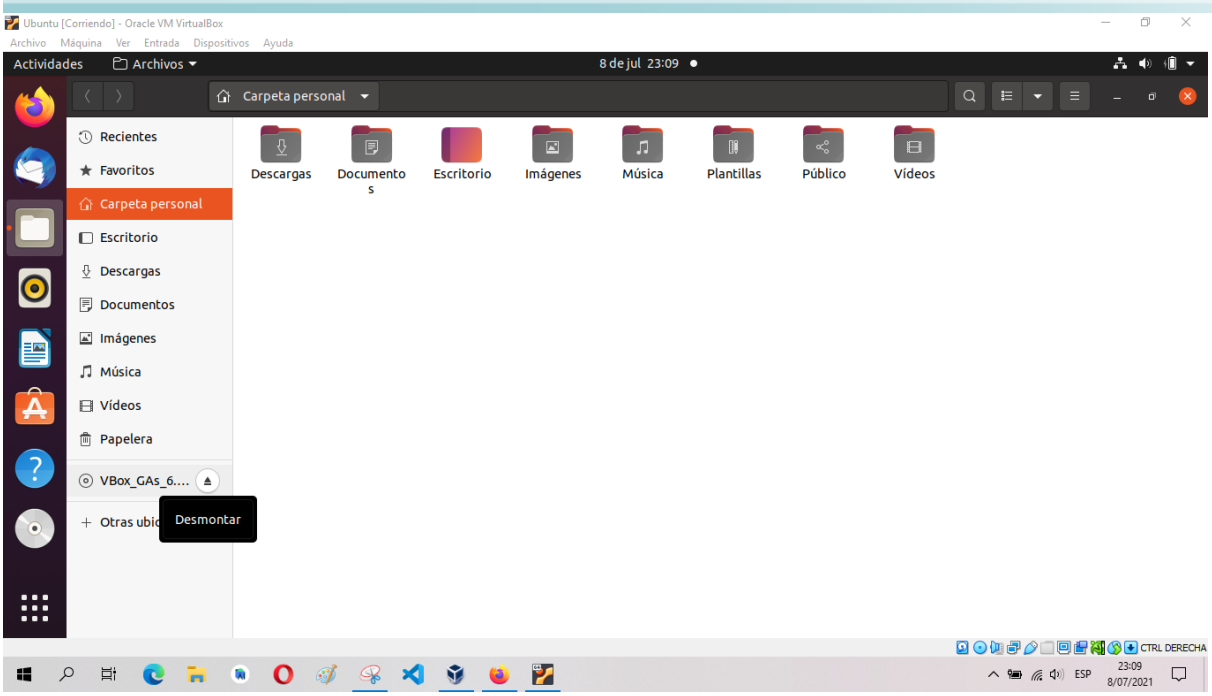


Ahí colocamos lo siguiente

```
sudo apt-get update
sudo apt-get install build-essential gcc make perl dkms
```

Luego tenemos que reiniciar la máquina, lo podemos hacer colocando el comando reboot en la terminal.

Si nos dio error tenemos que quitar las Guest Additions y tratar nuevamente.



Abrimos el explorador de archivos y seleccionamos desmontar el CD. Luego lo tratamos de instalar desde Dispositivos > Insertar el disco de <<Guest Additions>> nuevamente.

Cuando esté instalado deberíamos ver la pantalla completa.

Ahora acá podemos instalar Java con el siguiente comando

```
sudo apt install default-jdk
```

cuando termine de cargar podemos revisar que la instalación sea correcta con el comando

```
java -version
```

6. Mi primer programa en Java

Vamos a abrir un archivo de texto y lo vamos a guardar con la extensión Perro.java y luego le colocaremos el siguiente contenido. Esta es la clase Perro que ya vimos con anterioridad:

```
/*
*  Clase Perro
*/
public class Perro {
  String nombre, color, raza;
  public Perro(String nuevoNombre, String nuevoColor, String nuevaRaza) {
     nombre = nuevoNombre;
     color = nuevoColor;
     raza = nuevaRaza;
  }
  public void setNombre(String name) {
        nombre = name;
  }
  public String getNombre() {
        return nombre;
  }
}
```

Como siguiente paso vamos a abrir un nuevo documento de texto y vamos a crear nuestra clase con la cual vamos a ejecutar nuestro ejemplo. El archivo se va a llamar Ejemplo1.java y le colocaremos el siguiente contenido:

```
import java.io.BufferedOutputStream;
import java.io.InputStream;
import java.util.Scanner;

class Ejemplo1 {

  public static void main(String[] args) {
        Scanner scanner = new Scanner( System.in );
        System.out.println("Ingrese el nombre de su perro");
        String input1 = scanner.nextLine();
        System.out.println("Ingrese el color de su perro");
        String input2 = scanner.nextLine();
        System.out.println("Ingrese la raza de su perro");
        String input3 = scanner.nextLine();
        Perro perro1 = new Perro(input1, input2, input3);
        System.out.println("Su perro se llama " + perro1.getNombre());
  }

}
```

Para ejecutar nuestros programas debemos de ejecutar lo siguiente:

```
javac Perro.java
javac Ejemplo1.java
```

Estas dos instrucciones no nos deberían de mostrar ningún error, pero si llegara a haber algún error, entonces en la consola nos indica en dónde se encuentra el error. Se recomienda iniciar la corrección de errores desde la primera línea, ya que es posible que aunque tenga 500 errores, el primer error sea el causante de los siguientes 499 y al corregir el primero se corrigen los demás.

Luego ejecutamos nuestro programa con la siguiente instrucción:

```
java Ejemplo1
```

y nos debería de mostrar algo similar a esto:

```
Ingrese el nombre de su perro
Maggie
Ingrese el color de su perro
beige
Ingrese la raza de su perro
Beagle
Su perro se llama Maggie
```

Lo que está en negrilla es lo que el programa estará imprimiendo y lo que no está en negrilla es lo que se ingresó mediante el teclado.

Referencias

- API de Java: https://docs.oracle.com/javase/7/docs/api/
- Java in a Nutshell - Benjamin J. Evans & David Flanagan: http://www.r-5.org/files/books/computers/languages/java/main/Benjamin_Evans_David_Flanagan-Java_in_a_Nutshell_6th_ed-EN.pdf