

Practica 1: Entorno y Lenguaje de Programacion

Adolfo Roman Jimenez

September 20, 2021

1 Objetivo

Identificar y probar el entorno de ejecución y el lenguaje de programación orientado a objetos a utilizar durante el curso.

2 Actividades

- Probar los conceptos basicos del entorno y lenguaje
- Revisar la instalacion y configuracion del entorno de ejecucion
- Realizar un programa en el lenguaje de programación usando el entorno de ejecución, utilizando la sintaxis básica (notación, palabras reservadas, comentarios, etc.)

3 Manual de Practicas - Desarrollo

3.1 Instalacion

3.1.1 Java Virtual Machine (JVM) y Programas en Java

La Maquina Virtual de Java, compila y ejecuta el programa que a diferencia de otros lenguajes de programacion, no se ejecuta directamente en el sistema, sino dentro de la propia maquina virtual.

Por ejemplo, en el lenguaje de programacion C, se pueden crear funciones en bibliotecas distintas que se compilan creando archivos `.obj` y que despues el mismo compilador combina en un unico archivo `.out` o `.exe` que es el programa que contiene el lenguaje de maquina que ejecutara el sistema operativo.

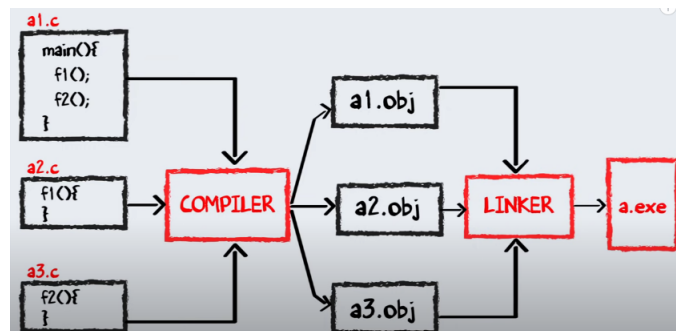


Figure 1: Proceso de compilacion en C

En el caso de Java, el proceso de compilacion y ejecucion primeramente se lleva a cabo cuando guardamos algun archivo con la extension `archivo.java`. Despues, al momento de compilar, se genera el `archivo.class` que es lo que Java usa para ejecutar dentro del JVM.

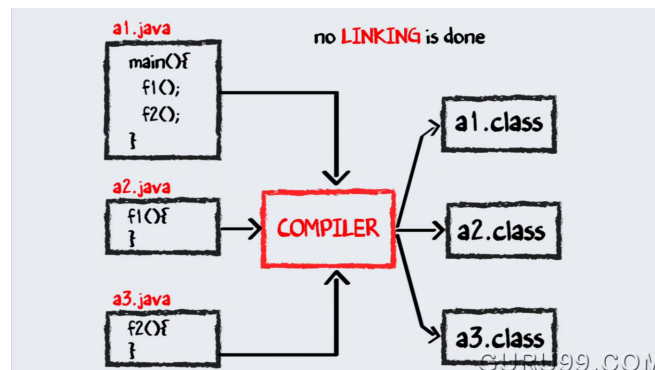


Figure 2: Proceso de compilacion en Java

Despues de esto, los archivos `.class` no se mezclan para formar un ultimo archivo ejecutable, sino que se transmiten a la *Java Virtual Machine* que dentro de su propio entorno, carga primero los archivos `.class` para posicionarlos en el *heap* de la memoria, mientras sus instancias las deposita en el *stack*. Este programa se va convirtiendo en *bytecode* que despues el *JIT* dentro la maquina virtual convertira en lenguaje de maquina ejecutable.

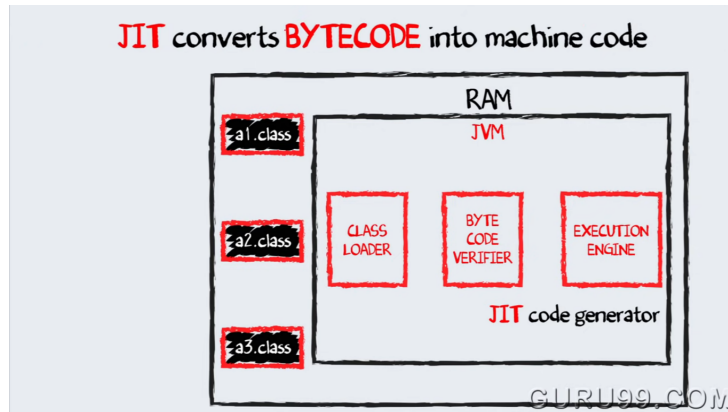


Figure 3: Proceso de compilacion en Java

3.1.2 Herramientas de Desarrollo (JDK)

Las herramientas de desarrollo en Java (*Java Development Kit*) proporcionan el entorno de desarrollo que usaremos para poder programar en Java. En mi caso usaremos el kit de desarrollo *Amazon Corretto 11* que es bastante usado debido a la popularidad de *Amazon Web Services*.

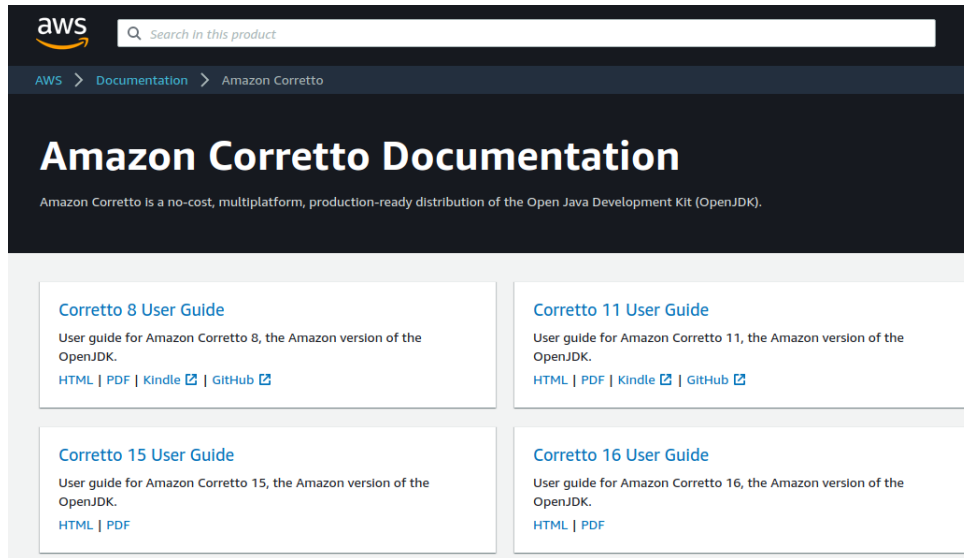


Figure 4: Pagina web de Amazon Corretto

Despues de instalar *Amazon Corretto 11* en la computadora, verificamos que la instalacion.

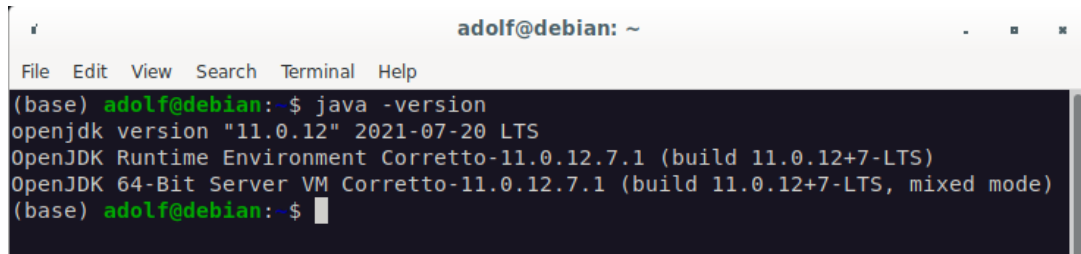


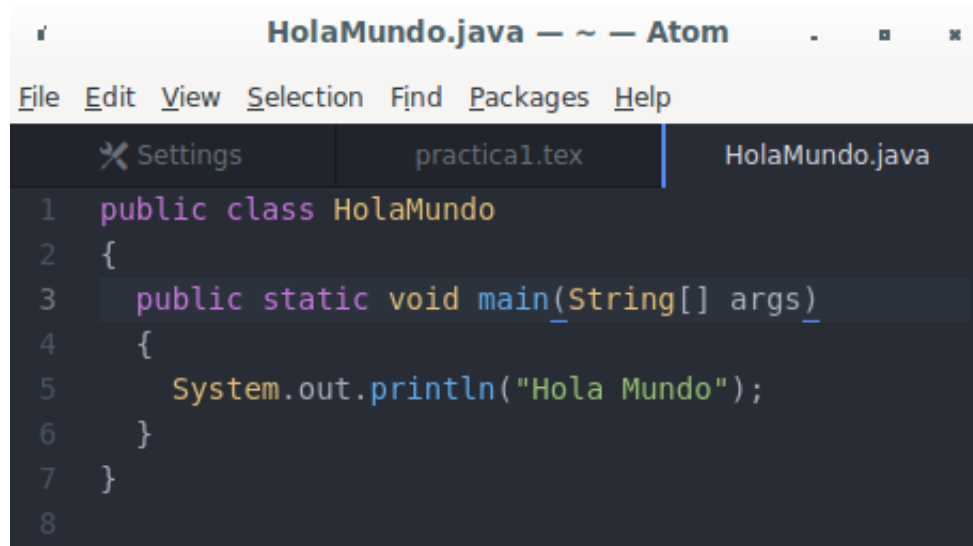
Figure 5: Amazon Corretto 11 instalado en Debian 11

3.2 Programando en Java

3.2.1 Codificacion

Usando el editor *Atom* creamos nuestro primer programa `HolaMundo.java`.

Es importante que el nombre del archivo sea exactamente el mismo que el de la clase con la que se trabaja, pues de lo contrario el compilador nos mostrara un error al momento de ejecutarlo.

A screenshot of the Atom text editor. The title bar at the top reads "HolaMundo.java — ~ — Atom". Below the title bar is a menu bar with "File", "Edit", "View", "Selection", "Find", "Packages", and "Help". The editor area shows a file named "HolaMundo.java" with the following code:

```
1 public class HolaMundo
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hola Mundo");
6     }
7 }
8
```

Figure 6: Primer programa en Java con el editor Atom

3.2.2 Compilacion y Programacion

Despues de escribir el programa, procedemos a compilarlo con el *JDK* que acabamos de instalar.

El comando para compilar es `javac` seguido del nombre del archivo con la extension `.java`.

Si la compilacion fue exitosa, el programa no nos mostrara ningun tipo de error y acto seguido podremos apreciar que se crea un archivo del mismo nombre que nuestro archivo original pero ahora con una extension `.class` que para ejecutarla, lo unico que debemos de hacer es ingresar el comando `java` y despues de este el nombre del archivo `.class` pero sin la extension como se muestra en la siguiente imagen.

```
adolfo@debian: ~/Desktop
File Edit View Search Terminal Help
(base) adolfo@debian:~/Desktop$ javac HolaMundo.java
(base) adolfo@debian:~/Desktop$ ls
'Adolf!'      Dropbox      Java
Avanzadas    'Extra info' Libros
'Ciencias de la Computacion' HolaMundo.class 'Libros 2022-1'
CPCFI        HolaMundo.java matlab.desktop
Deutsch      'Huawei - HCIA Storage'
(base) adolfo@debian:~/Desktop$ java HolaMundo
Hola Mundo
(base) adolfo@debian:~/Desktop$
```

Figure 7: Compilacion y Ejecucion de HolaMundo en Java

3.2.3 Entorno de Desarrollo Integrado (IDE)

El entorno de desarrollo integrado (*Integrated Development Environment*) es un programa que amplía las herramientas para desarrolladores del lenguaje bajo el que se trabaje y que también provee de entornos de ejecución y corrección de software integrados dentro del mismo programa, lo cual hace de un IDE una herramienta que simplemente facilita el trabajo del desarrollador al tener la mayoría o todas las opciones necesarias a la mano dentro un mismo entorno.

En mi caso preferí utilizar el entorno *IntelliJIDE* debido a que me pareció el más adecuado para mi sistema operativo.

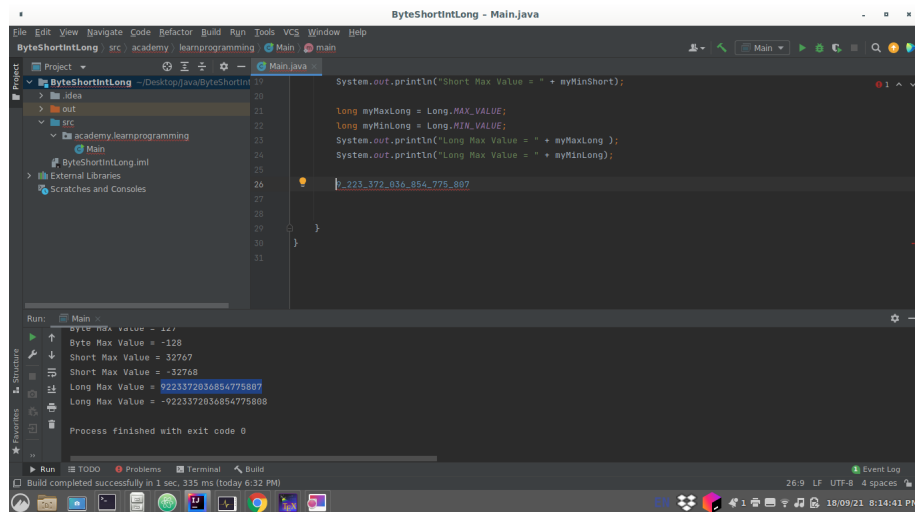


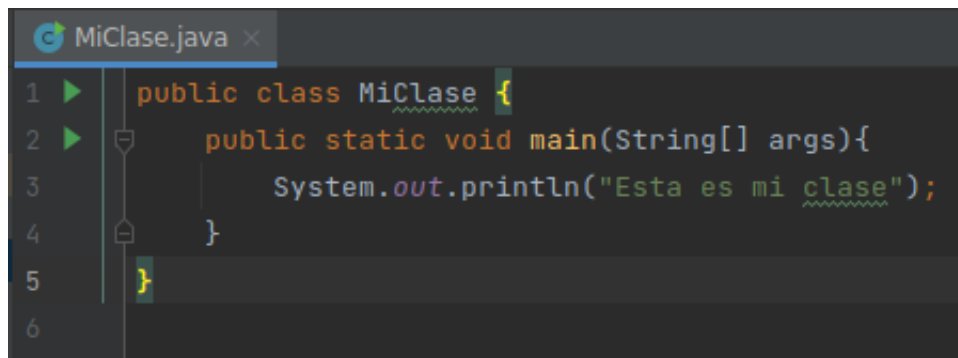
Figure 8: IntelliJIDE para Debian 11

3.3 Estructura de un Programa en Java

3.3.1 Estructura de Clase

Como ya se habia mencionado, un programa en Java para que pueda compilar correctamente, inicialmente debe de tener una clase que se llame de la misma forma que el archivo.

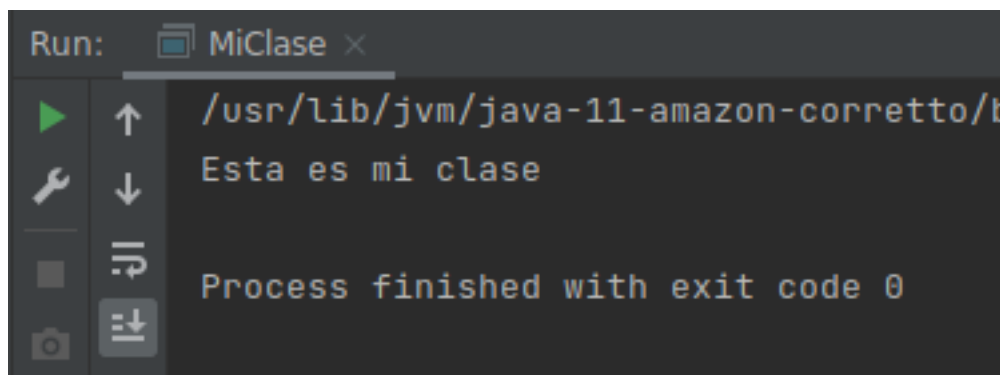
La clase principal debe ser publica y debe de contener un metodo *main* junto con el identificador *class*.



```
MiClase.java x
1 public class MiClase {
2     public static void main(String[] args){
3         System.out.println("Esta es mi clase");
4     }
5 }
6
```

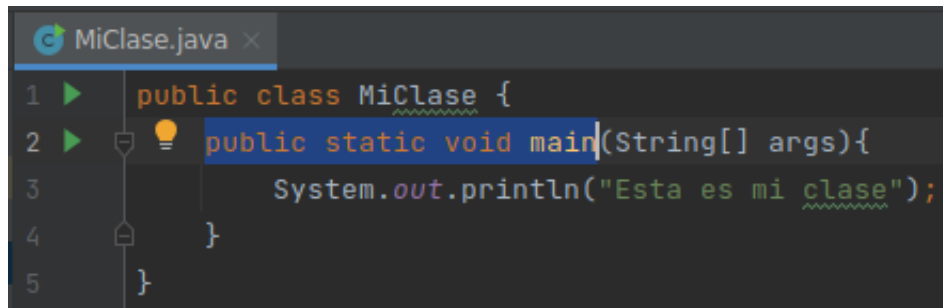
Figure 9: Programa con clase de nombre "MiClase" en IntelliJIDEA

Al compilar el programa podemos ver que el programa se ejecuta de forma correcta en el IDE.



```
Run: MiClase x
/usr/lib/jvm/java-11-amazon-corretto/t
Esta es mi clase
Process finished with exit code 0
```

Figure 10: Ejecucion de "MiClase" en IntelliJIDEA



```

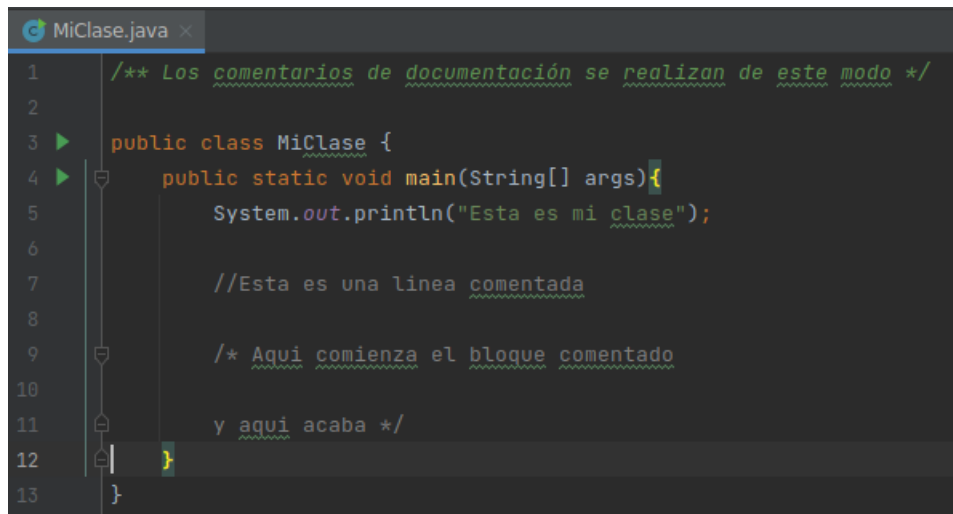
MiClase.java x
1  ▶ public class MiClase {
2  ▶  public static void main(String[] args){
3      System.out.println("Esta es mi clase");
4  }
5  }

```

Figure 11: Contenido de "main" en la clase principal

3.3.2 Comentarios

Existen diversas formas de insertar comentarios en un programa de Java para indicar sobre diversos temas, ya sea como instrucciones e indicaciones para los programadores que no conozcan el programa o como información sobre la autoría de este, credenciales, permisos, etc.



```

MiClase.java x
1  /** Los comentarios de documentación se realizan de este modo */
2
3  ▶ public class MiClase {
4  ▶  public static void main(String[] args){
5      System.out.println("Esta es mi clase");
6
7      //Esta es una línea comentada
8
9      /* Aquí comienza el bloque comentado
10
11     y aquí acaba */
12  }
13 }

```

Figure 12: Comentarios dentro de clase "MiClase"

3.3.3 Palabras Reservadas

Similar a los lenguajes que ya se han usado en el pasado, el lenguaje de Java, tambien tiene una serie de palabras que estan reservadas y no se pueden usar, pues corresponden a la sintaxis del lenguaje como tal.

Estas palabras incluyen fundamentalmente metodos y tipos de datos que el lenguaje usa para su correcto funcionamiento.

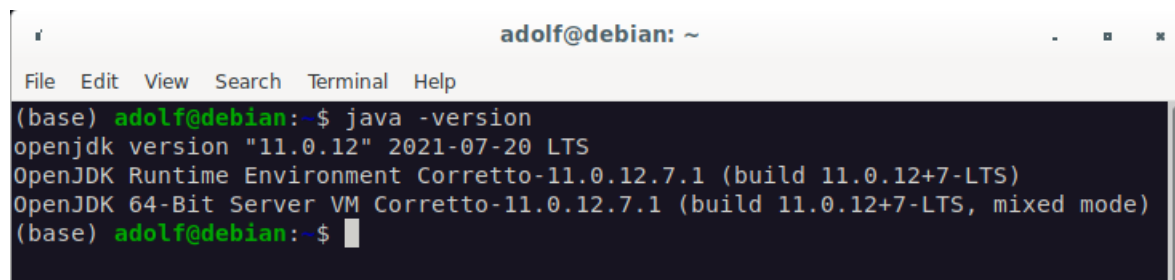
abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while
assert	enum				

Figure 13: Set de Palabras Reservadas en Java

4 Practica Sugerida - Desarrollo

4.1 Ejercicio 1

Para este ejercicio se nos pide instalar el *JDK* de *Oracle* recomendado en la practica. Sin embargo, yo decidi instalar *Amazon Coretto 11* ya que me se adecua mejor al sistema operativo que tengo.



```
adolfo@debian: ~  
File Edit View Search Terminal Help  
(base) adolfo@debian:~$ java -version  
openjdk version "11.0.12" 2021-07-20 LTS  
OpenJDK Runtime Environment Corretto-11.0.12.7.1 (build 11.0.12+7-LTS)  
OpenJDK 64-Bit Server VM Corretto-11.0.12.7.1 (build 11.0.12+7-LTS, mixed mode)  
(base) adolfo@debian:~$
```

Figure 14: JDK, Amazon Coretto 11

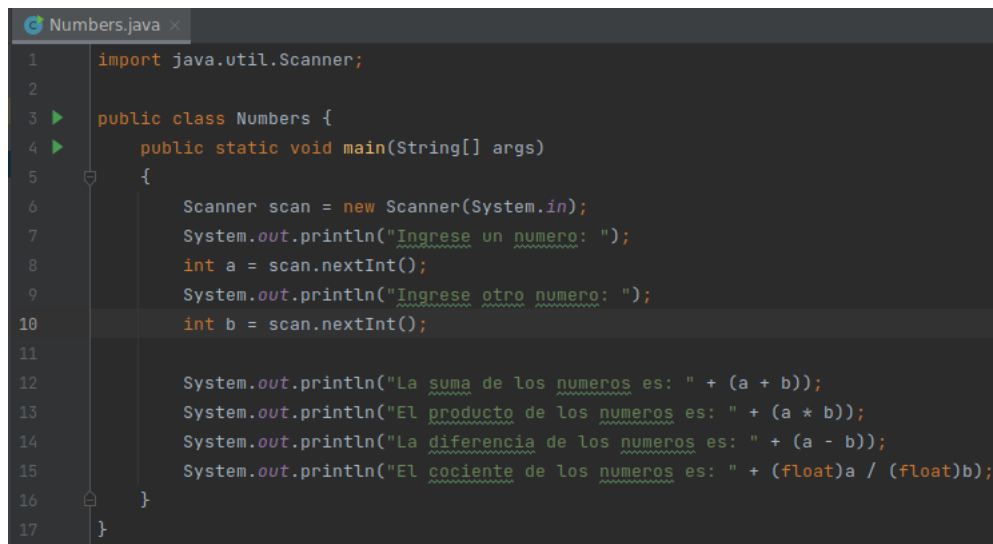
4.2 Ejercicio 2

En este ejercicio debemos de escribir un programa que pida al usuario 2 numeros y que de estos numeros imprima la suma, el producto, la diferencia y el cociente de estos.

Para lograrlo, importe la biblioteca *Scanner* que me permitio al momento de declararla, crear una instancia llamada simplemente *scan* con la cual utilice el metodo *nextInt()* para obtener los inputs del usuario.

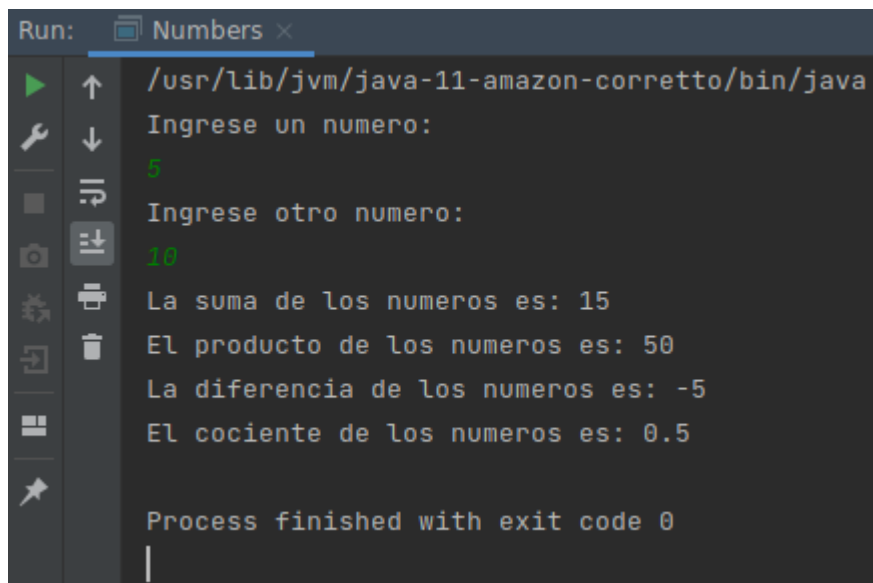
Antes de cada solicitud de *input* imprimir una leyenda con `System.out.println` solicitando al usuario ingresar un numero en cada de los requerimientos y el resultado lo deposite en dos variable de tipo `int` **a** y **b** respectivamente.

Despues con la ayuda de nuevo de `System.out.println` simplemente indique que era el tipo de resultado que se estaba imprimiendo y realice la operacion dentro de la misma linea a imprimir como argumento. En el caso del promedio entre dos enteros, tuve que hacer un *cast* de ambos tipos de dato entero para que el resultado pudiera presentarse como un numero no entero.



```
1  import java.util.Scanner;
2
3  public class Numbers {
4      public static void main(String[] args)
5      {
6          Scanner scan = new Scanner(System.in);
7          System.out.println("Ingrese un numero: ");
8          int a = scan.nextInt();
9          System.out.println("Ingrese otro numero: ");
10         int b = scan.nextInt();
11
12         System.out.println("La suma de los numeros es: " + (a + b));
13         System.out.println("El producto de los numeros es: " + (a * b));
14         System.out.println("La diferencia de los numeros es: " + (a - b));
15         System.out.println("El cociente de los numeros es: " + (float)a / (float)b);
16     }
17 }
```

Figure 15: Codigo en IntelliJIDE del Ejercicio 1



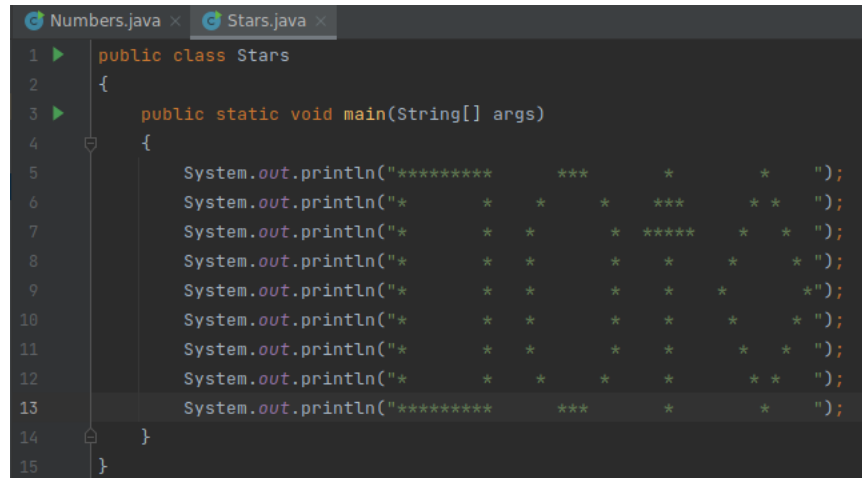
```
Run: Numbers x
/usr/lib/jvm/java-11-amazon-corretto/bin/java
Ingrese un numero:
5
Ingrese otro numero:
10
La suma de los numeros es: 15
El producto de los numeros es: 50
La diferencia de los numeros es: -5
El cociente de los numeros es: 0.5

Process finished with exit code 0
```

Figure 16: Ejecucion del ejercicio dentro del IDE

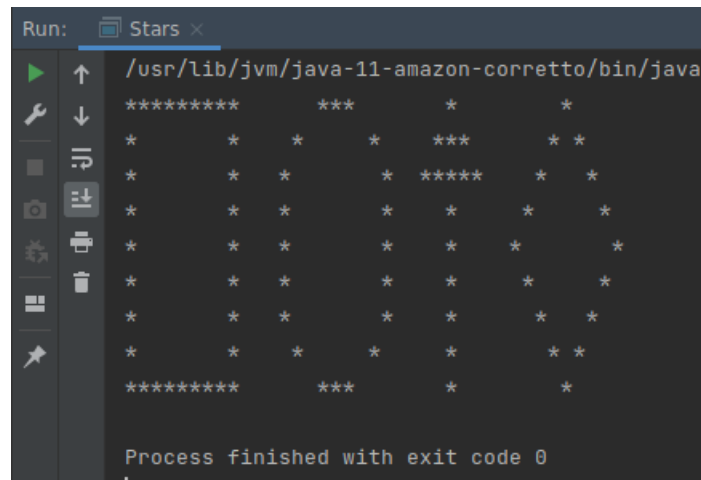
4.3 Ejercicio 3

Para este ejercicio que nos pedia imprimir una serie de asteriscos en forma de cuadrado, ovalo, flecha y diamante, simplemente utilice distintos comandos `System.out.println` para linea por linea, imprimir la forma solicitada y que el resultado fuera similar al requerido por la practica.



```
1 public class Stars
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("*****      ***      *      *");
6         System.out.println("*      *      *      *      ***      * *");
7         System.out.println("*      *      *      *      *****      * *");
8         System.out.println("*      *      *      *      *      *      * *");
9         System.out.println("*      *      *      *      *      *      * *");
10        System.out.println("*      *      *      *      *      *      * *");
11        System.out.println("*      *      *      *      *      *      * *");
12        System.out.println("*      *      *      *      *      *      * *");
13        System.out.println("*****      ***      *      *");
14    }
15 }
```

Figure 17: Hard-coding de las figuras solicitadas



```
Run: Stars x
/usr/lib/jvm/java-11-amazon-corretto/bin/java
*****      ***      *      *
*      *      *      *      ***      * *
*      *      *      *      *****      * *
*      *      *      *      *      *      * *
*      *      *      *      *      *      * *
*      *      *      *      *      *      * *
*      *      *      *      *      *      * *
*      *      *      *      *      *      * *
*****      ***      *      *

Process finished with exit code 0
```

Figure 18: Output del ejercicio 3

4.4 Ejercicio 4

En este ejercicio que fue similar al ejercicio 2, tuve que importar un par de clases adicionales para poder escoger el numero minimo y numero maximo que el ejercicio solicita que fueron las librerias *Arrays* y *Collections*.

La clase *Arrays* contiene diversos metodos para manipular arreglos, como de ordenamiento o de busqueda y la clase *Collections* contiene metodos estaticos que operan y regresan colecciones de elementos.

Declare la clase *Scanner* al principio del programa tambien y despues utilice la instancia *scan* para solicitar a usuario las 3 variables enteras requeridas.

Despues de que el usuario ingresa las variables y las deposito dentro de las variables enteras **a**, **b** y **c**, creo un arreglo de enteros con la clase *Integer* y deposito ahi el valor de las variables obtenidas como elementos del arreglo.

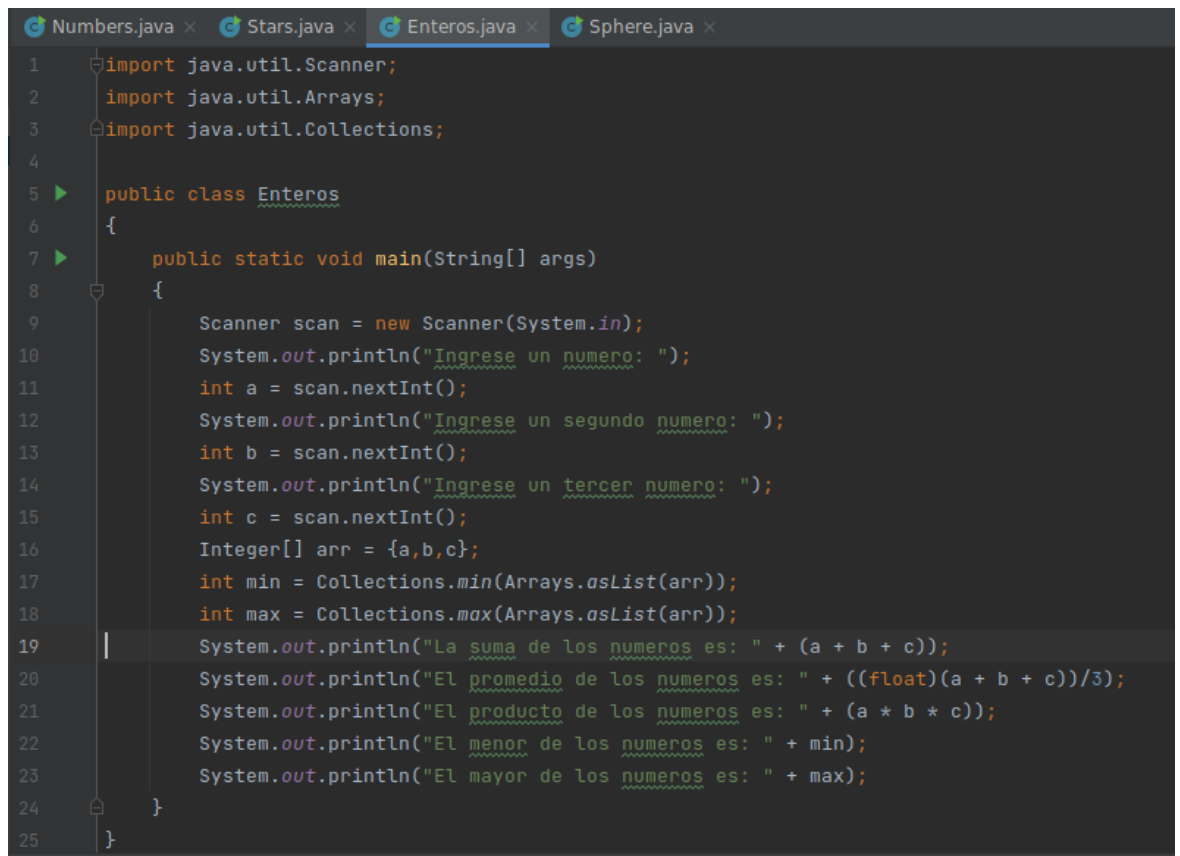
Una vez esto, creo dos varibales enteras para obtener el numero minimo y maximo del arreglo **min** y **max** y las igualo a los siguiente comandos dependiendo el caso:

- `Collections.min(Arrays.asList(arr))`
- `Collections.max(Arrays.asList(arr))`

Lo que hace cada una de estas directivas, es que la clase *Arrays* genera una lista del arreglo **arr** y la clase *Collections* aplica el metodo de obtener el valor maximo o valor minimo de esa lista, dependiendo de lo que el programador haya determinado.

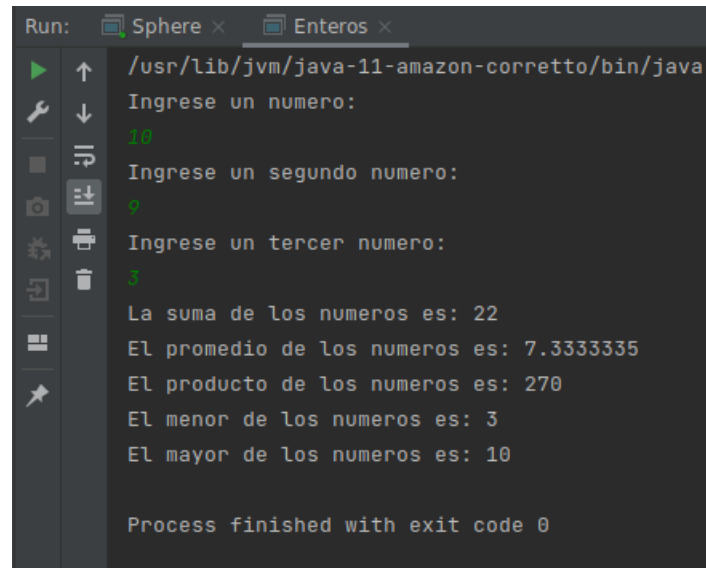
Con respecto al demas funcionamiento del programa, este es similar al del ejercicio 2 como ya lo habiamos mencionado. De igual forma para los distintos *outputs*, declaramos `System.out.println` y ejecutamos la operacion requerida en alguno de los argumentos de la funcion para que esta imprima el resultado como salida.

Solo en los casos de valor menor o mayor esto no se da, pues la operacion para determinar que elemento es mayor que el otro, se realiza fuera de las funciones de impresion a pantalla y en estas unicamente se ejecutan los resultados en las variables declaradas.



```
1 import java.util.Scanner;
2 import java.util.Arrays;
3 import java.util.Collections;
4
5 public class Enteros
6 {
7     public static void main(String[] args)
8     {
9         Scanner scan = new Scanner(System.in);
10        System.out.println("Ingrese un numero: ");
11        int a = scan.nextInt();
12        System.out.println("Ingrese un segundo numero: ");
13        int b = scan.nextInt();
14        System.out.println("Ingrese un tercer numero: ");
15        int c = scan.nextInt();
16        Integer[] arr = {a,b,c};
17        int min = Collections.min(Arrays.asList(arr));
18        int max = Collections.max(Arrays.asList(arr));
19        System.out.println("La suma de los numeros es: " + (a + b + c));
20        System.out.println("El promedio de los numeros es: " + ((float)(a + b + c))/3);
21        System.out.println("El producto de los numeros es: " + (a * b * c));
22        System.out.println("El menor de los numeros es: " + min);
23        System.out.println("El mayor de los numeros es: " + max);
24    }
25 }
```

Figure 19: Código fuente de la clase Enteros para ejercicio 4



```
Run: Sphere x Enteros x
/usr/lib/jvm/java-11-amazon-corretto/bin/java
Ingrese un numero:
10
Ingrese un segundo numero:
0
Ingrese un tercer numero:
3
La suma de los numeros es: 22
El promedio de los numeros es: 7.3333335
El producto de los numeros es: 270
El menor de los numeros es: 3
El mayor de los numeros es: 10

Process finished with exit code 0
```

Figure 20: Output de clase Enteros

4.5 Ejercicio 5

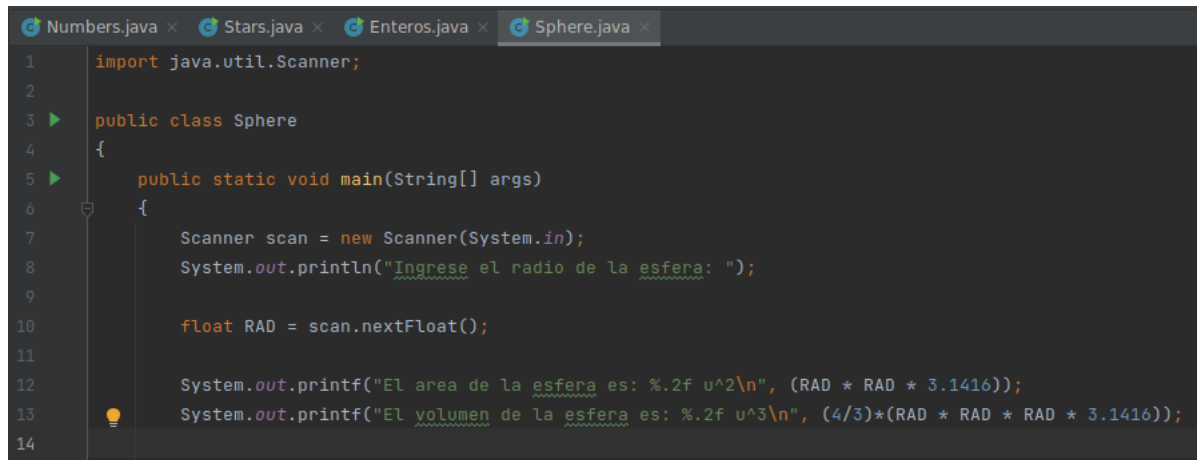
Este ejercicio no fue tan complicado pues es similar a los anteriores en terminos de recopilar datos del usuario y llevar a cabo las operaciones dentro de las funciones de impresion de lineas.

Algo que lleve a cabo, fue de reducir el numero de decimales a 2 para cualquier tipo de respuesta que involucre numero flotante.

En cada una de las lineas que imprimen la respuesta, en esta ocasion se usa un metodo diferente que es:

`System.out.printf`

y para el cual, es necesario incluir un salto de linea al final del comando pues esta funcion, no incluye un salto de linea como si lo incluia el metodo `println`.

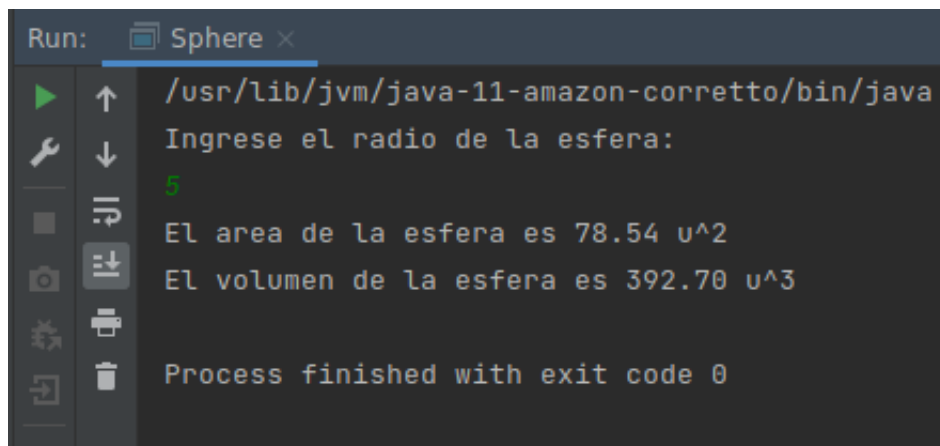


```

1  import java.util.Scanner;
2
3  public class Sphere
4  {
5      public static void main(String[] args)
6      {
7          Scanner scan = new Scanner(System.in);
8          System.out.println("Ingrese el radio de la esfera: ");
9
10         float RAD = scan.nextFloat();
11
12         System.out.printf("El area de la esfera es: %.2f u^2\n", (RAD * RAD * 3.1416));
13         System.out.printf("El volumen de la esfera es: %.2f u^3\n", (4/3)*(RAD * RAD * RAD * 3.1416));
14     }

```

Figure 21:Codigo fuente de la clase Sphere



```

Run: Sphere x
/usr/lib/jvm/java-11-amazon-corretto/bin/java
Ingrese el radio de la esfera:
5
El area de la esfera es 78.54 u^2
El volumen de la esfera es 392.70 u^3

Process finished with exit code 0

```

Figure 22: Output de clase Sphere5

5 Conclusiones

Me gusto mucho la practica, pues ademas de que es una de las primeros documentos que es escribo en \LaTeX con el cual me he podido ir acostumbrando a redactar documentos academicos de una mejor manera, tambien aprendi sobre el entorno y el lenguaje de Java.

Me gusta mucho el *IntelliJ IDEA* pues me parece un entorno bastante amigable y sencillo de usar.

Por otro lado, en cuanto al lenguaje de Java, me parece que la sintaxis no es tan compleja, pues es muy parecida a la sintaxis de *C* que ya usamos anteriormente, pero creo lo que me emociona en realidad, es la naturaleza del lenguaje que es 100 por ciento orientado a objetos y espero poder aprender mas al respecto durante el semestre.

Por lo mientras me parece genial que Java contenga diversas clases que podemos importar y funcionan justo como las bibliotecas en *C* o *Python* que por lo regular usamos, pues contienen mucho codigo que podemos utilizar para facilitarnos la vida dentro del lenguaje.

Me siento emocionado por aprender mas de Java y su funcionalidad como lenguaje Orientado a Objetos.

$\sqrt{2}$ no es un numero racional