

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

## I. Objetivos de Aprendizaje

- Abordar los fundamentos del lenguaje de programación Java.
- Desarrollar programas de ejemplo utilizando Java

## II. Introducción Teórica

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems y actualmente es propiedad de Oracle.

Se puede utilizar para desarrollar aplicaciones móviles (especialmente aplicaciones de Android), aplicaciones de escritorio, aplicaciones web, realizar conexiones a bases de datos entre otras funcionalidades.

### Características de Java

1. Lenguaje Simple: "Se lo conoce como lenguaje simple porque viene de la misma estructura de c y c++; ya que c++ fue un referente para la creación de java por eso utiliza determinadas características de c++ y se han eliminado otras."
2. Orientado a Objetos.
3. Multihilos: Java tiene una facilidad de cumplir varias funciones al mismo tiempo, gracias a su función de multihilos ya que por cada hilo que el programa tenga se ejecutarán en tiempo real muchas funciones al mismo tiempo.

### Entornos de desarrollo y componentes de Java

#### ➤ JRE

El JRE (Java Runtime Environment, o Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma

#### ➤ JDK

Es el acrónimo de "Java Development Kit", es decir Kit de desarrollo de Java. Se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java.

#### ➤ API

Las siglas API tienen su origen en Application Program Interface y consiste en un conjunto de librerías de código java compiladas que son ofrecidas a los desarrolladores.

La API puede utilizarse según el ámbito de su desarrollo:

1. **Java ME** (Java Platform, Micro Edition) o J2ME — orientada a entornos de limitados recursos, como teléfonos móviles, PDAs (Personal Digital Assistant), etc.
2. **Java SE** (Java Platform, Standard Edition) o J2SE — para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO 01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

escritorio.

3. **Java EE** (Java Platform, Enterprise Edition) o J2EE — orientada a entornos distribuidos empresariales o de Internet.

Java cuenta con un conjunto de tipos de datos primitivos y los cuales son fundamentales para definir otros tipos de datos estructurados o complejos. Existe un total de ocho tipos de datos primitivos, las cuales se dividen a continuación:

### Tipos numéricos enteros

En Java existen cuatro tipos destinados a almacenar números enteros. La única diferencia entre ellos es el número de bytes usados para su almacenamiento y, en consecuencia, el rango de valores que es posible representar con ellos. Todos ellos emplean una representación que permite el almacenamiento de números negativos y positivos. El nombre y características de estos tipos son los siguientes:

<b>Tipo</b>	<b>Tamaño en bits</b>	<b>Rango de almacenamiento</b>
byte	8	-128 a 127
short	16	-32,768 a 32,767
int	32	-2,147,483,648 a 2,147,483,647
long	64	-9,223,372,036,854,775,808L a 9,223,372,036,854,775,807L

### Tipos numéricos en punto flotante

Los tipos numéricos en punto flotante permiten representar números tanto muy grandes como muy pequeños además de números decimales. Java dispone de 2 tipos concretos en esta categoría:

<b>Tipo</b>	<b>Tamaño en bits</b>	<b>Rango de almacenamiento</b>
float	32	+/- 3.4E+38F (6-7 dígitos importantes)
double	64	+/- 1.8E+308 (15 dígitos importantes)

### Booleanos y caracteres

<b>Tipo</b>	<b>Tamaño en bits</b>	<b>Rango de almacenamiento</b>
char	16	Conjunto de caracteres Unicode ISO
boolean	1	verdadero o falso

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b>

## Tipos de datos estructurados

Los tipos de datos estructurados se denominan así porque en su mayor parte están destinados a contener múltiples valores de tipos más simples, primitivos. También se les llama muchas veces "tipos objeto" porque se usan para representar objetos.

### a. Cadenas de caracteres

Las cadenas de carácter son una instancia de la clase String y se delimitan entre comillas dobles, en lugar de simple como los caracteres individuales. El siguiente es un ejemplo de la utilización de una cadena de caracteres:

```
String nombreMateria = "Desarrollo de Software para móviles";
```

### b. Vectores o arrays

Los vectores son colecciones de datos de un mismo tipo, también son conocidos como arrays o arreglos. Un vector es una estructura de datos en la que cada elemento le corresponde una posición identificada por uno o más índices numéricos enteros. Los elementos de un vector se empiezan a numerar en la posición cero y permiten gestionar desde una sola variable múltiples datos del mismo tipo.

Nota: en algunas ocasiones es habitual llamar matrices a los vectores, sin embargo nos referimos a vectores que trabajan con dos dimensiones.

## Valores por defecto en una variable sin inicializar

- Object referencia un valor null (es decir no referencia ningún objeto)
- byte, short, int, long valor por defecto es 0
- float, double valor por defecto es 0.0
- boolean valor por defecto es false

## Operadores de Relación

<b>Operador</b>	<b>Uso</b>	<b>Devuelve verdadero si</b>
>	ope1 > ope2	ope1 es mayor que ope2
>=	ope1 >= ope2	ope1 es mayor o igual que ope2
<	ope1 < ope2	ope1 es menor que ope2
<=	ope1 <= ope2	ope1 es menor o igual que ope2
= =	ope1 = = ope2	ope1 y ope2 son iguales
! =	ope1 != ope2	ope1 y ope2 son distintos

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO 01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

## Operadores Matemáticos

<b>Operador</b>	<b>Uso</b>	<b>Descripción</b>
+	ope1 + ope2	Suma ope1 y ope2 (*)
-	ope1 - ope2	Resta ope1 de ope2
*	ope1 * ope2	Multiplica ope1 y ope2
/	ope1 / ope2	Divide ope1 por ope2
%	ope1 % ope2	Obtiene el módulo de dividir ope1 por ope2
++	ope++	Incrementa ope en 1; evalúa el valor antes de incrementar
++	++ope	Incrementa ope en 1; evalúa el valor después de incrementar
--	ope--	Decrementa ope en 1; evalúa el valor antes de incrementar
--	--ope	Decrementa ope en 1; evalúa el valor después de incrementar

Hay que tener en cuenta que al disponer una condición debemos seleccionar que operador relacional se adapta a la pregunta.

Ejemplos:

- Se ingresa un número multiplicarlo por 10 si es distinto a 0. (!=)
- Se ingresan dos números mostrar una advertencia si son iguales. (==)

## Operadores lógicos

<b>Operador</b>	<b>Uso</b>	<b>Devuelve verdadero si</b>
&&	ope1 && ope2	ope1 y ope2 son verdaderos
	ope1    ope2	uno de los dos es verdadero
!	! ope	ope es falso (niega ope)

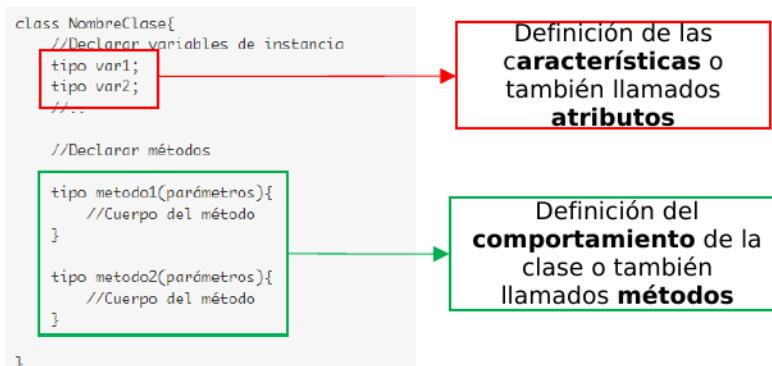
**Nota:** Los operadores lógicos siempre devuelven un valor booleano.

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO 01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

## Clases en Java

Una clase es una plantilla (características y comportamiento) que define la forma de un objeto. Especifica los datos y el código que operará en esos datos. Java usa una especificación de clase para construir objetos. Los objetos son instancias de una clase. Por lo tanto, **una clase es esencialmente un conjunto de planes que especifican cómo construir un objeto.**

A continuación, se muestra una forma general simplificada de una definición de clase:



## Definición de una Clase

```

class Vehiculo {
    int pasajeros; // números de pasajeros
    int capacidad; // capacidad del combustible en galones
    int mpg; // combustible consumido en millas por galon
}

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

        Vehiculo minivan = new Vehiculo();
        int rango;

        // asignando valores a los campos de minivan
        minivan.pasajeros = 9;
        minivan.capacidad = 15;
        minivan.mpg = 20;

        // Calcular el rango asumiendo un tanque lleno
        rango = minivan.capacidad * minivan.mpg;

        System.out
            .println("La Minivan puede llevar " + minivan.pasajeros + " pasajeros con un rango de " + rango + " millas");
    }
}

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO 01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

## Objetos en Java

Un objeto es una instancia de una clase. Por ejemplo: si hablamos de un molde de galletas, esta sería nuestra clase y cada galleta creada con el molde sería nuestro objeto. En el siguiente ejemplo veremos cómo creamos objetos a partir de la clase vehículo.

```
//Este programa crea dos objetos Vehiculo
class Vehiculo {
    int pasajeros; //números de pasajeros
    int capacidad; //capacidad del combustible en galones
    int mpg; //combustible consumido en millas por galon
}

//Esta clase declara un objeto de tipo Vehiculo

class DosVehiculo {

    public static void main(String[] args) {
        Vehiculo minivan = new Vehiculo();
        Vehiculo sportscar = new Vehiculo();

        int rango1, rango2;

        //asignando valores a los campos de minivan
        minivan.pasajeros = 9;
        minivan.capacidad = 15;
        minivan.mpg = 20;

        //asignando valores a los campos de sportscar
        sportscar.pasajeros = 10;
        sportscar.capacidad = 25;
        sportscar.mpg = 30;

        //Calcular el rango asumiendo un tanque lleno
        rango1 = minivan.capacidad * minivan.mpg;
        rango2 = sportscar.capacidad * sportscar.mpg;

        System.out.println("La Minivan puede llevar " + minivan.pasajeros + " pasajeros con un rango de " + rango1 + " millas");
        System.out.println("El Sportscar puede llevar " + sportscar.pasajeros + " pasajeros con un rango de " + rango2 + " millas");
    }
}
```



### **III. Desarrollo de la Práctica**

#### **Tipos de datos en Java**

En el siguiente programa se crearan una seria de variables de los distintos tipos de datos que soporta Java tales como int, long, float, double, char y boolean.

En la linea 1 a 4 se hace uso de /\* \*/ para incluir un comenario en un programa Java, en este caso todo el contenido entre los juegos de caracteres antes mencionado y que puede estar constituido por una o mas lineas de codigo, se consideraran como comentario y no sera compilado por el compilador de Java.

En las lineas 13 a 14 se puede ver ejemplos de comentarios en Java para una sola linea de codigo, todo el contenido despues de los caracteres // es considerado comenario.

Digitar con un editor de texto tal como notepad el codigo del programa Java TiposDatos.java o crear un proyecto en Netbeans, compilar el programa y ejecutarlo.

#### **TiposDatos.java**

```
1. /*
2. Asignatura Java Avanzada
3. Programa de ejemplo tipos de datos en Java
4. */
5.
6. public class TiposDatos{
7.     public static void main(String[] args)
8.     {
9.         int var01 = 2000;
10.        long var02 = 10000;
11.        float var03 = 6.5F;
12.        double var04 = 10.7;
13.        char var05 = '\u0022'; // Se asigna a var05 un valor
   constante Unicode
14.        char var06 = 'k';           // Se asigna a var06 un
   valor constante caracter
15.        boolean var07 = true;
16.
17.        System.out.println("var01 = " + var01);
```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

**GUIA DE  
LABORATORIO  
Nº 1**

**MATERIA** DESARROLLO DE SOFTWARE PARA MÓVILES

**PRÁCTICA** INTRODUCCIÓN JAVA Y KOTLIN - Parte I

```
18.         System.out.println("var02 = " + var02);
19.         System.out.println("var03 = " + var03);
20.         System.out.println("var04 = " + var04);
21.         System.out.println("var05 = " + var05);
22.         System.out.println("var06 = " + var06);
23.         System.out.println("var07 = " + var07);
24.     }
25. }
```

Modifique la linea 11 del programa `TiposDatos.java`, de tal manera que el contenido de la linea de codigo en mención sea: `float var03 = 6.5;`

Compilar una el programa modificado y ejecutarlo. Analizar el mensaje que proporciona el compilador de Java despues de haber realizado la modificación solicitada.

A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the command "java Tiposdatos" being run. The output displays several variable assignments: var01 = 2000, var02 = 10000, var03 = 6.5, var04 = 10.7, var05 = "", var06 = k, and var07 = true. The prompt at the bottom right indicates the user has reached the end of the scrollable output.

```
C:\Windows\system32\cmd.exe
E:\Java-Avanzado\walter-sanchez>java Tiposdatos
var01 = 2000
var02 = 10000
var03 = 6.5
var04 = 10.7
var05 =
var06 = k
var07 = true

E:\Java-Avanzado\walter-sanchez>
```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

### Operadores Aritmeticos

En el siguiente programa se implementa un programa básico para resolver un sistema de dos ecuaciones con dos incógnitas. Los operadores aritméticos más corrientes + - \* / se utilizan en Java para la suma, resta, multiplicación y división para cantidades enteras y punto flotante como en otros lenguajes de programación, en el caso del operador módulo % este solo se utiliza para cantidades enteras. Digitar con un editor de texto tal como notepad el código del programa Java SimultaneoEcuaciones.java o crear un proyecto en Netbeans, compilar el programa y ejecutarlo.

### SimultaneoEcuaciones.java

```

1. public class SimultaneoEcuaciones{
2.     public static void main(String[] args)
3.     {
4.         float x1 = 2, y1 = 4, c1= -2;
5.         float x2 = -3, y2 = 5, c2= 10;
6.
7.         float x = (c1*y2 - c2*y1) / (x1*y2 - x2*y1);
8.         float y = (x1*c2 - x2*c1) / (x1*y2 - x2*y1);
9.
10.        System.out.println("x = " + x);
11.        System.out.println("y = " + y);
12.
13.    }
14. }
```



```

C:\WINDOWS\system32\cmd.exe
E:\Java-Avanzado\walter-sanchez>java SimultaneoEcuaciones
x = -2.2727273
y = 0.6363636
E:\Java-Avanzado\walter-sanchez>
```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

### Constantes en Java

En java se utiliza la palabra clave final para declarar y definir constantes como se muestra en la linea 5 del programa Constantes.java.

Digitar con un editor de texto tal como notepad el codigo del programa Java Constantes.java o crear un proyecto en Netbeans, compilar el programa y ejecutarlo.

### Constantes.java

```

1. public class Constantes
2. {
3.     public static void main(String[] args)
4.     {
5.         final float pi = 3.14159265F;
6.         float radio = 3.5F;
7.         double area;
8.
9.         System.out.println("area    circunferencia = " +
pi*radio*radio);
10.
11.    }
12. }
```



```

C:\WINDOWS\system32\cmd.exe
E:\Java-Avanzado\walter-sanchez>java Constantes
area circunferencia = 38.484512
E:\Java-Avanzado\walter-sanchez>
```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

### Cadenas en Java

Para la creación de cadenas en Java no existe un tipo de dato para tal propósito, en su lugar la biblioteca estandar de Java tiene una clase predefinida llamada String. Recordar que una cadena es una secuencia de caracteres encerrada entre comillas dobles.

En la tabla 1 se hace referencia a algunas funciones miembros de la clase String invocadas en el codigo del programa Cadenas.java

<b>Función miembro clase String</b>	<b>Descripción</b>
char charAt(int indice)	Devuelve el carácter de la posición especificada
boolean equals(Object otro)	Devuelve true si la cadena es igual a otra
int length()	Devuelve la longitud de la cadena
String substring(int inicio, int final)	Devuelve una cadena de caracteres nueva va desde el indice inicio hasta el indice final exclusive de la cadena

**Tabla 1.** Algunas funciones miembros de la clase String

Digitar con un editor de texto tal como notepad el codigo del programa Java Cadenas.java o crear un proyecto en Netbeans, compilar el programa y ejecutarlo.

### Cadenas.java

```

1. public class Cadenas{
2.     public static void main(String[] args)
3.     {
4.         String cadena01 = "Universidad";
5.         String cadena02 = "Don Bosco";
6.         String cadena03 = "Escuela";
7.         String cadena04 = "de";
8.             String cadena05 = "Computacion";
9.             String cadena06 = cadena03 + " " + cadena04 + " " +
cadena05;
10.            String cadena07 = cadena06 + " Java Avanzado " + "Ciclo
01-" + 2011;
11.
12.            System.out.println(cadena06);

```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>	<b>GUIA DE LABORATORIO N° 1</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b>	

```
13.         System.out.println(cadena07);
14.
15.         String cadena08 = cadena05.substring(0,5);
16.         System.out.println(cadena08);
17.
18.         System.out.println("Cadena 06: " + cadena06);
19.         System.out.println("Longitud de Cadena 06: " +
cadena06.length());
20.         System.out.println("El caracter 4 de Cadena 06 es: " +
cadena06.charAt(4));
21.
22.         cadena07 = cadena07.substring(23,36) + " - Programacion
Orientada a Objetos";
23.         System.out.println(cadena07);
24.
25.         System.out.println(cadena07 == cadena06);
26.
27.         String cadena09 = cadena06.substring(11,22);
28.         System.out.println(cadena09.equals("Computacion"));
29.
30.     }
31. }
```

```
C:\WINDOWS\system32\cmd.exe
E:\Java-Avanzado\walter-sanchez>java Cadenas
Escuela de Computacion
Escuela de Computacion Java Avanzado Ciclo 01-2011
Compu
Cadena 06: Escuela de Computacion
Longitud de Cadena 06: 22
El caracter 4 de Cadena 06 es: e
Java Avanzado - Programacion Orientada a Objetos
false
true

E:\Java-Avanzado\walter-sanchez>
```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

### Estructuras de Selección

El programa EstructuraCondicional01.java es un ejemplo del uso de la estructura de selección if-else, compilar y ejecutar el programa en mención.

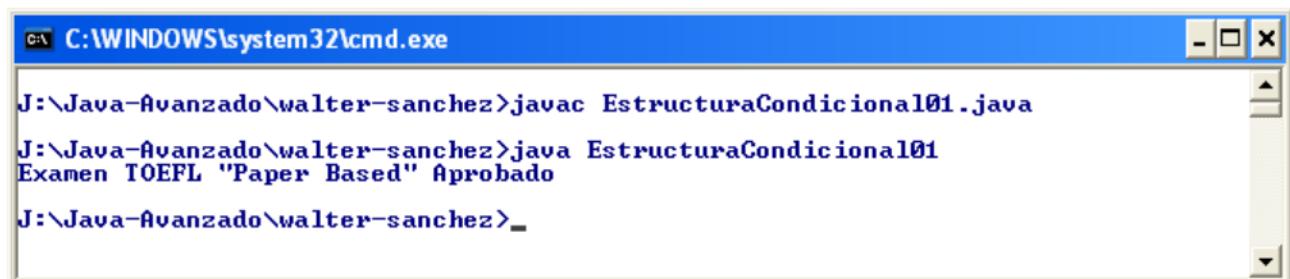
En este programa se verifica si la nota obtenida por un estudiante en la prueba TOEFL paper based es la mínima aceptada o superior según estándares promedios de Universidades Europeas o Norteamericanas.

### EstructuraCondicional01.java

```

1. public class EstructuraCondicional01
2. {
3.     public static void main(String[] args)
4.     {
5.         int NotaTOEFL = 550;
6.
7.         if(NotaTOEFL >= 550)
8.             System.out.println("Examen      TOEFL      \"Paper      Based\""
9.                             "Aprobado");
10.            else
11.                System.out.println("Examen      TOEFL      \"Paper      Based\""
12.                               "Reprobado");
13.    }
}

```



```

C:\WINDOWS\system32\cmd.exe
J:\Java-Avanzado\walter-sanchez>javac EstructuraCondicional01.java
J:\Java-Avanzado\walter-sanchez>java EstructuraCondicional01
Examen TOEFL "Paper Based" Aprobado
J:\Java-Avanzado\walter-sanchez>_

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

El programa EstructuraCondicional02.java es un ejemplo del uso de la estructura de selección if-else en "escalera", compilar y ejecutar el programa en mención. Este programa hace la conversión de escala de notas en el sistema de educación nacional a Norteamericano (EEUU).

#### **EstructuraCondicional02.java**

```

1. public class EstructuraCondicional02
2. {
3.     public static void main(String[] args)
4.     {
5.         int NotaEstudiante = 75;
6.
7.         if (NotaEstudiante >= 90)
8.             System.out.println("A");
9.         else if (NotaEstudiante >= 80)
10.            System.out.println("B");
11.        else if (NotaEstudiante >= 70)
12.            System.out.println("C");
13.        else if (NotaEstudiante >= 60)
14.            System.out.println("D");
15.        else
16.            System.out.println("F");
17.    }
18. }
```



```

C:\WINDOWS\system32\cmd.exe
J:\Java-Avanzado\walter-sanchez>javac EstructuraCondicional02.java
J:\Java-Avanzado\walter-sanchez>java EstructuraCondicional02
C
J:\Java-Avanzado\walter-sanchez>_

```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	<b>GUIA DE LABORATORIO N° 1</b>
PRÁCTICA	INTRODUCCIÓN JAVA Y KOTLIN - Parte I	

El programa EstructuraCondicional03.java es un ejemplo del uso de la estructura de selección multiple switch, compilar y ejecutar el programa en mención.

**EstructuraCondicional03.java**

```
1. public class EstructuraCondicional03
2. {
3.     public static void main(String[] args)
4.     {
5.         int mes = 9;
6.         switch(mes)
7.         {
8.             case 1:
9.                 System.out.println("Mes de Enero");
10.                break;
11.            case 2:
12.                System.out.println("Mes de Febrero");
13.                break;
14.            case 3:
15.                System.out.println("Mes de Marzo");
16.                break;
17.            case 4:
18.                System.out.println("Mes de Abril");
19.                break;
20.            case 5:
21.                System.out.println("Mes de Mayo");
22.                break;
23.            case 6:
24.                System.out.println("Mes de Junio");
25.                break;
26.            case 7:
27.                System.out.println("Mes de Julio");
28.                break;
29.            case 8:
30.                System.out.println("Mes de Agosto");
31.                break;
32.            case 9:
33.                System.out.println("Septiembre");
34.                break;
35.            case 10:
36.                System.out.println("Octubre");
37.                break;
```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>	<b>GUIA DE LABORATORIO N° 1</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b>	

```
38.         case 11:
39.             System.out.println("Noviembre");
40.             break;
41.         case 12:
42.             System.out.println("Diciembre");
43.             break;
44.         default:
45.             System.out.println("Mes invalido");
46.         }
47.     }
48. }
```

```
C:\WINDOWS\system32\cmd.exe
J:\Java-Avanzado\walter-sanchez>javac EstructuraCondicional03.java
J:\Java-Avanzado\walter-sanchez>java EstructuraCondicional03
Septiembre
J:\Java-Avanzado\walter-sanchez>
```

### Estructuras de Repetición

El programa EstructuraRepetitiva01.java es un ejemplo del uso de la estructura de repetición do-while, compilar y ejecutar el programa en mención. Este programa calcula el factorial de un numero entero.

### EstructuraRepetitiva01.java

```
1. public class EstructuraRepetitiva01
2. {
3.     public static void main(String[] args)
4.     {
5.
6.         long factorial = 1;
7.         int i = 1;
8.         int n = 5;
9.
10.        do{
11.            factorial = factorial * i;
```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>	<b>GUIA DE LABORATORIO N° 1</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b>	

```
12.             i++;
13.         }while(i < n + 1);
14.         System.out.println("Factorial de " + n + " es " +
factorial);
15.     }
16. }
```

```
C:\WINDOWS\system32\cmd.exe
J:\Java-Avanzado\walter-sanchez>javac EstructuraRepetitiva01.java
J:\Java-Avanzado\walter-sanchez>java EstructuraRepetitiva01
Factorial de 5 es 120
J:\Java-Avanzado\walter-sanchez>
```

El programa EstructuraRepetitiva02.java es un ejemplo del uso de la estructura de repetición for, compilar y ejecutar el programa en mención. Este programa calcula el factorial de un numero entero.

**EstructuraRepetitiva02.java**

```
1. public class EstructuraRepetitiva02
2. {
3.     public static void main(String[] args)
4.     {
5.
6.         long factorial = 1;
7.         int n = 9;
8.
9.         for(int i = 1;i < n + 1;i++)
10.             factorial = factorial * i;
11.
12.         System.out.println("Factorial de " + n + " es " +
factorial);
13.     }
14. }
```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

```

C:\WINDOWS\system32\cmd.exe
J:\Java-Avanzado\walter-sanchez>javac EstructuraRepetitiva02.java
J:\Java-Avanzado\walter-sanchez>java EstructuraRepetitiva02
Factorial de 9 es 362880
J:\Java-Avanzado\walter-sanchez>

```

El programa EstructuraRepetitiva03.java es un ejemplo del uso de la estructura de repetición while, compilar y ejecutar el programa en mención. Este programa calcula el factorial de un numero entero.

#### EstructuraRepetitiva03.java

```

1. public class EstructuraRepetitiva03
2. {
3.     public static void main(String[] args)
4.     {
5.
6.         long factorial = 1;
7.         int n = 7;
8.         int i = 1;
9.
10.        while(i < n + 1){
11.            factorial = factorial * i;
12.            i++;
13.        }
14.
15.        System.out.println("Factorial de " + n + " es " +
factorial);
16.    }
17. }

```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

**MATERIA** DESARROLLO DE SOFTWARE PARA MÓVILES

**PRÁCTICA** INTRODUCCIÓN JAVA Y KOTLIN - Parte I

**GUIA DE  
LABORATORIO  
Nº 1**

The screenshot shows a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The command line shows the execution of a Java program named 'EstructuraRepetitiva03.java'. The output indicates that the factorial of 7 is 5040.

```
C:\Windows\system32\cmd.exe
J:\Java-Avanzado\walter-sanchez>javac EstructuraRepetitiva03.java
J:\Java-Avanzado\walter-sanchez>java EstructuraRepetitiva03
Factorial de 7 es 5040
J:\Java-Avanzado\walter-sanchez>_
```

### Arreglos

En el programa `GraficoBarrasNotas.java` se presenta un ejemplo de uso de arreglos donde se genera un grafico de barras desde la linea de comandos del Sistema Operativo en funcion de la distribucion de notas de un grupo de clases.

El arreglo `array` indica que un estudiante obtuvo 10 de nota, dos estudiantes obtuvieron nota entre 90-99, seis estudiantes obtuvieron nota entre 80-89, y asi sucesivamente.

El parametro en mención significa que el método `main` recibe un array de cadenas de caracteres que son los argumentos especificados desde la linea de comandos

### `GraficoBarrasNotas.java`

```
1. public class GraficoBarrasNotas
2. {
3.     public static void main( String args[] )
4.     {
5.         int array[] = { 0, 0, 0, 0, 0, 0, 3, 12, 6, 2, 1 };
6.
7.         System.out.println( "Distribucion de Notas" );
8.
9.         for ( int contador = 0; contador < array.length; contador++ )
10.            {
11.                if ( contador == 10 )
12.                    System.out.printf( "%5d: ", 100 );
13.                else
14.                    System.out.printf( "%02d-%02d: ", contador * 10,
15.                           contador * 10 + 9 );
16.            }
17.        }
18.    }
```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>	<b>GUIA DE LABORATORIO N° 1</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b>	

```
16.                     for ( int asteriscos = 0; asteriscos <
    array[contador]; asteriscos++ )
17.                         System.out.print( "*" );
18.
19.                         System.out.println();
20.                     }
21.                 }
22.             }
```

The screenshot shows a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The command line shows the execution of a Java program named 'GraficoBarrasNotas.java'. The output displays a distribution of grades from 00-09 to 100, represented by asterisks (\*). The distribution is as follows:

Grade Range	Count
00-09	1
10-19	1
20-29	1
30-39	1
40-49	1
50-59	1
60-69	7
70-79	7
80-89	5
90-99	2
100	1

E:\Java-Avanzado\walter-sanchez>javac GraficoBarrasNotas.java  
E:\Java-Avanzado\walter-sanchez>java GraficoBarrasNotas  
Distribucion de Notas  
00-09:  
10-19:  
20-29:  
30-39:  
40-49:  
50-59:  
60-69: \*\*\*\*\*  
70-79: \*\*\*\*\*  
80-89: \*\*\*  
90-99: \*\*  
100: \*



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>	<b>GUIA DE LABORATORIO N° 1</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b>	

### **Clases y Herencia**

La clase Empleado define una clase base que será utilizada para crear clases derivadas para un cierto tipo de empleados específicos tales como una secretaria, un vigilante, un motorista, un gerente, entre otros tipos, la clase base empleado define los campos y métodos de un perfil de empleado genérico. Digitar el código de la clase Empleado siguiente:

#### **Empleado.java**

```
1. import java.util.*;
2. class Empleado
3. {
4.     public Empleado(String n, double s, int anio, int mes, int
5.                     dia)
6.     {
7.         nombre = n;
8.         salario = s;
9.         GregorianCalendar calendario = new GregorianCalendar(anio,
10.                                         mes - 1, dia);
11.        fechaContratacion = calendario.getTime();
12.    }
13.    public String obtenerNombre()
14.    {
15.        return nombre;
16.    }
17.    public double obtenerSalario()
18.    {
19.        return salario;
20.    }
21.    public Date obtenerFechaContratacion()
22.    {
23.        return fechaContratacion;
24.    }
25.    public void aumentarSalario(double porcentaje)
26.    {
27.        double aumento = salario * porcentaje / 100;
28.        salario += aumento;
29.    }
30.    private String nombre;
31.    private double salario;
32. }
```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

La clase Gerente es una clase derivada de la clase Empleado, en este caso a manera de ejemplo debido a su naturaleza, el empleado Gerente obtiene su sueldo bruto mensual a partir de su sueldo base mas un bono, por esa razón la clase Gerente redefine la función obtenerSalario() de la clase Empleado.

El resto de métodos miembros de la clase Empleado es reutilizada a través de la herencia por la clase Gerente, exactamente la clase Gerente hereda de la clase Empleado los siguientes métodos:

- 1) obtenerFechaContratacion()
- 2) obtenerNombre()
- 3) obtenerNombre()
- 4) aumentarSalario(double porcentaje)

Digitar el código de la clase Gerente siguiente:

#### **Gerente.java**

```

1. class Gerente extends Empleado
2. {
3.     public Gerente(String n, double s, int anio, int mes, int dia)
4.     {
5.         super(n, s, anio, mes, dia);
6.         bono = 0;
7.     }
8.
9.     public double obtenerSalario()
10.    {
11.        double salarioBase = super.obtenerSalario();
12.        return salarioBase + bono;
13.    }
14.
15.    public void obtenerBono(double b)
16.    {
17.        bono = b;
18.    }
19.    private double bono;
20. }
```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b></p>

En la clase PruebaEmpleadoGerente se crea un arreglo de tipo Empleado para contener tanto objetos de tipo Empleado como objetos de Tipo Gerente. Digitar el código de la clase PruebaEmpleadoGerente. Compilar las clases digitadas anteriormente y ejecutar el programa PruebaEmpleadoGerente como muestra la **figura 1**.

#### **PruebaEmpleadoGerente.java**

```

1. public class PruebaEmpleadoGerente
2. {
3.     public static void main(String[] args)
4.     {
5.
6.         Gerente jefe = new Gerente("Carlos Martinez", 800, 1987,
7.                                     12, 15);
8.
9.         Empleado[] personal = new Empleado[3];
10.
11.        personal[0] = jefe;
12.        personal[1] = new Empleado("Tomas Castillo", 350, 1989, 10,
13.                                   1);
14.        personal[2] = new Empleado("Roberto Mendoza",
15.                                   400, 1990, 3, 15);
16.
17.        for (int i = 0; i < personal.length; i++)
18.        {
19.            Empleado e = personal[i];
20.            System.out.println("nombre = " + e.obtenerNombre() +
21.                               ", salario = " + e.obtenerSalario());
}
}
}

```

 <p><b>UNIVERSIDAD DON BOSCO</b> VITAM IMPENDERE VERO</p>	<b>UNIVERSIDAD DON BOSCO</b> <b>FACULTAD DE INGENIERÍA</b> <b>ESCUELA DE COMPUTACIÓN</b>	
	<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>
	<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte I</b>

```
C:\WINDOWS\system32\cmd.exe
K:\WalterSanchez\Clase05\EjemploEmpleado>javac *.java
K:\WalterSanchez\Clase05\EjemploEmpleado>java PruebaEmpleadoGerente
nombre = Carlos Martinez, salario = 900.0
nombre = Tomas Castillo, salario = 350.0
nombre = Roberto Mendoza, salario = 400.0
K:\WalterSanchez\Clase05\EjemploEmpleado>
```

Figura 1

 <p><b>UNIVERSIDAD DON BOSCO</b> VITAM IMPENDERE VERO</p>	<b>UNIVERSIDAD DON BOSCO</b> <b>FACULTAD DE INGENIERÍA</b> <b>ESCUELA DE COMPUTACIÓN</b>		<b>CICLO</b> <b>01-2024</b>
	<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>	<b>GUIA DE LABORATORIO</b> <b>Nº 1</b>
	<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b>	

## I. Objetivos de Aprendizaje

- Abordar los fundamentos del lenguaje de programación Kotlin.
- Desarrollar programas de ejemplo utilizando Kotlin

## II. Introducción Teórica

Kotlin es un lenguaje de programación estático de código abierto que admite la programación funcional y orientada a objetos. Proporciona una sintaxis y conceptos similares a los de otros lenguajes, como C#, Java y Scala, entre muchos otros. Cuenta con variantes que se orientan a la JVM (Kotlin/JVM), JavaScript (Kotlin/JS) y el código nativo (Kotlin/Native).

Kotlin es administrado por Kotlin Foundation, un grupo creado por JetBrains y Google, que se ocupa de continuar el desarrollo del lenguaje. Google admite oficialmente Kotlin para el desarrollo de Android, lo cual significa que la documentación y las herramientas de Android están diseñadas para ser compatibles con Kotlin.

Algunas API de Android, como Android KTX, son específicas de Kotlin, pero la mayoría están escritas en Java y se pueden llamar desde Java o Kotlin. La interoperabilidad de Kotlin con Java es fundamental para su crecimiento. Eso quiere decir se puede llamar al código Java desde Kotlin, y viceversa, y de esa manera aprovechar todas tus bibliotecas de Java existentes.

## Tipos de datos en Kotlin

<b>Tipos de datos básicos en Kotlin</b>	
<b>Tipo de dato</b>	<b>Tamaño en bits</b>
Double	64
Float	32
Long	64
Int	32
Short	16
Byte	8

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

## Variables Inmutables

Las variables inmutables solo pueden ser inicializadas una vez, esto hace que estas variables sean realmente **constantes**.

```
val variable02: Int = 10 // Declaración e inicialización al mismo tiempo
```

```
val variable02: Int // Declaración
```

```
variable02 = 10 // Inicialización
```

## Variables Mutables

```
var intVariable01 = 5
```

```
var intVariable02 = 5L
```

```
var doubleVariable = .02
```

### Ejemplo de uso de tipos de datos básicos en Kotlin

```
var varlong: Long = 9999L
```

```
var varint: Int = 999
```

```
var varshort: Short = 99
```

```
var varbyte: Byte = 9
```

```
var varfloat: Float = 9.99F
```

```
var vardouble: Double = 9.99999
```

```
var varhex: Int = 0xFF
```

```
var varbinaria:Int = 0b1001001
```

```
var bool: Boolean = true
```

```
var bool2: Boolean = 1 > 100
```

```
var varchar: Char = 'a'
```

```
var varstring: String = "Kotlin"
```

```
val cadenaMultiple: String = """
```

Asignatura: Desarrollo de software para móviles

Código: DSM104

Escuela: Ingeniería en Ciencias de la Computación""""

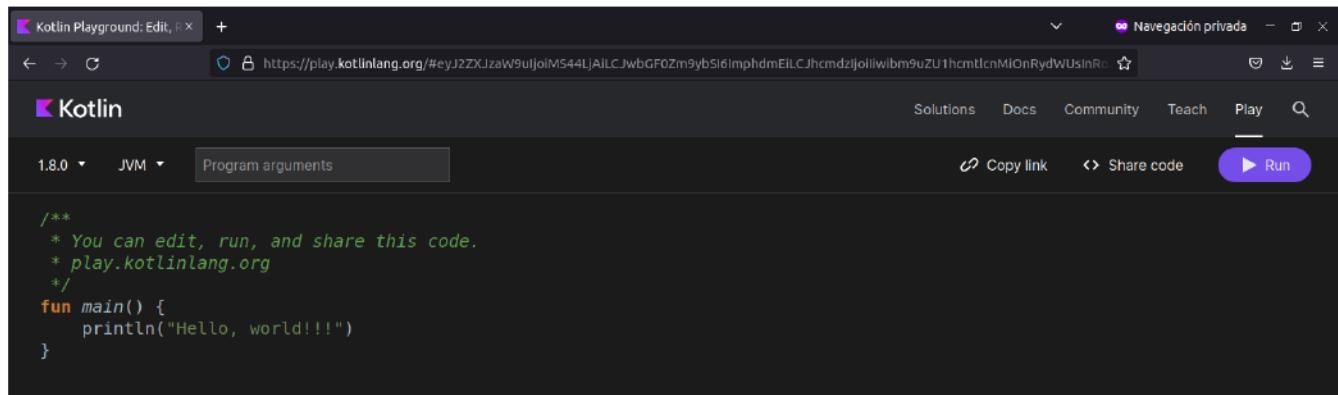
 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

## Operadores numéricos en Kotlin

Operadores Numéricos
+, -, *, /, %
+=, -=, *=, /=, %=
==, !=
<, >, <=, >=

## Entornos de Desarrollo Kotlin

<https://play.kotlinlang.org/>



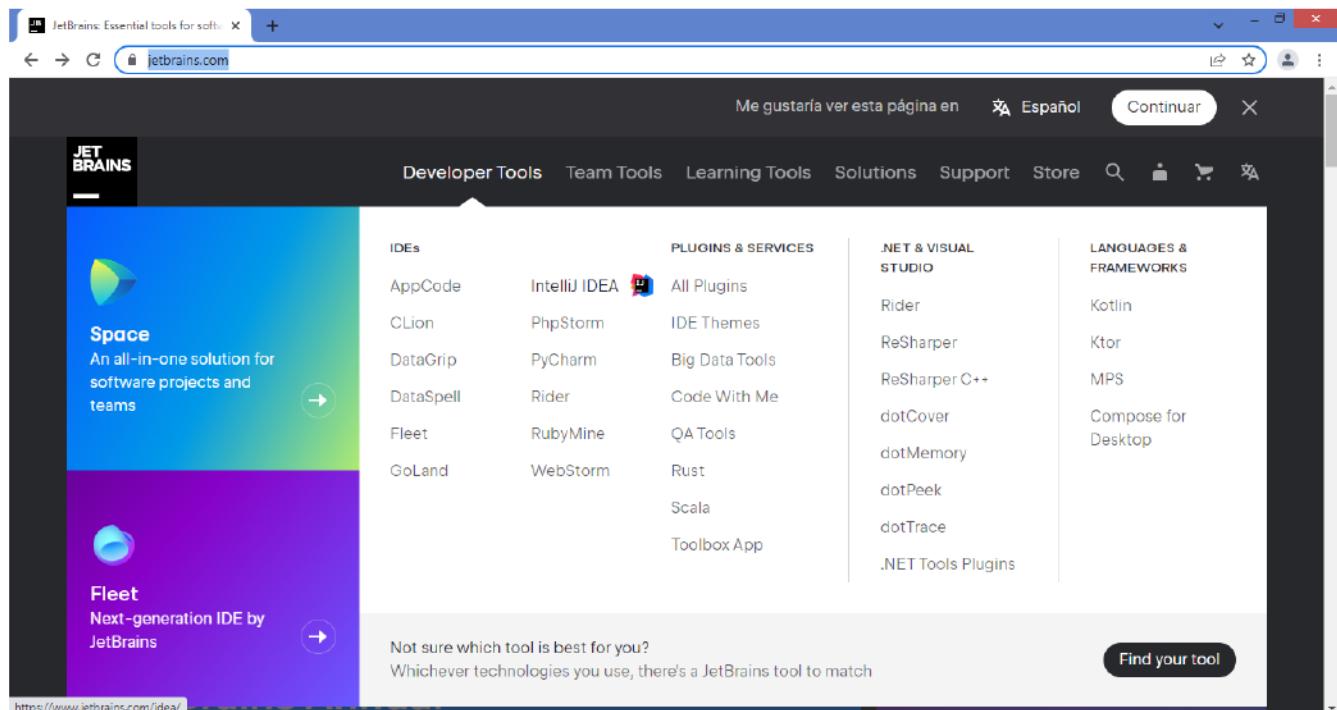
The screenshot shows the Kotlin Playground interface. At the top, there's a navigation bar with tabs for 'Edit' (selected), 'Run', 'Solutions', 'Docs', 'Community', 'Teach', and 'Play'. Below the navigation bar, there are dropdown menus for '1.8.0' and 'JVM'. A 'Program arguments' input field is present. On the right side, there are buttons for 'Copy link', 'Share code', and a purple 'Run' button. The main area contains the following Kotlin code:

```
/*
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */
fun main() {
    println("Hello, world!!!")
}
```

 <p><b>UNIVERSIDAD DON BOSCO</b> FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN</p>	<b>CICLO 01-2024</b>
<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b>

## IntelliJ IDEA

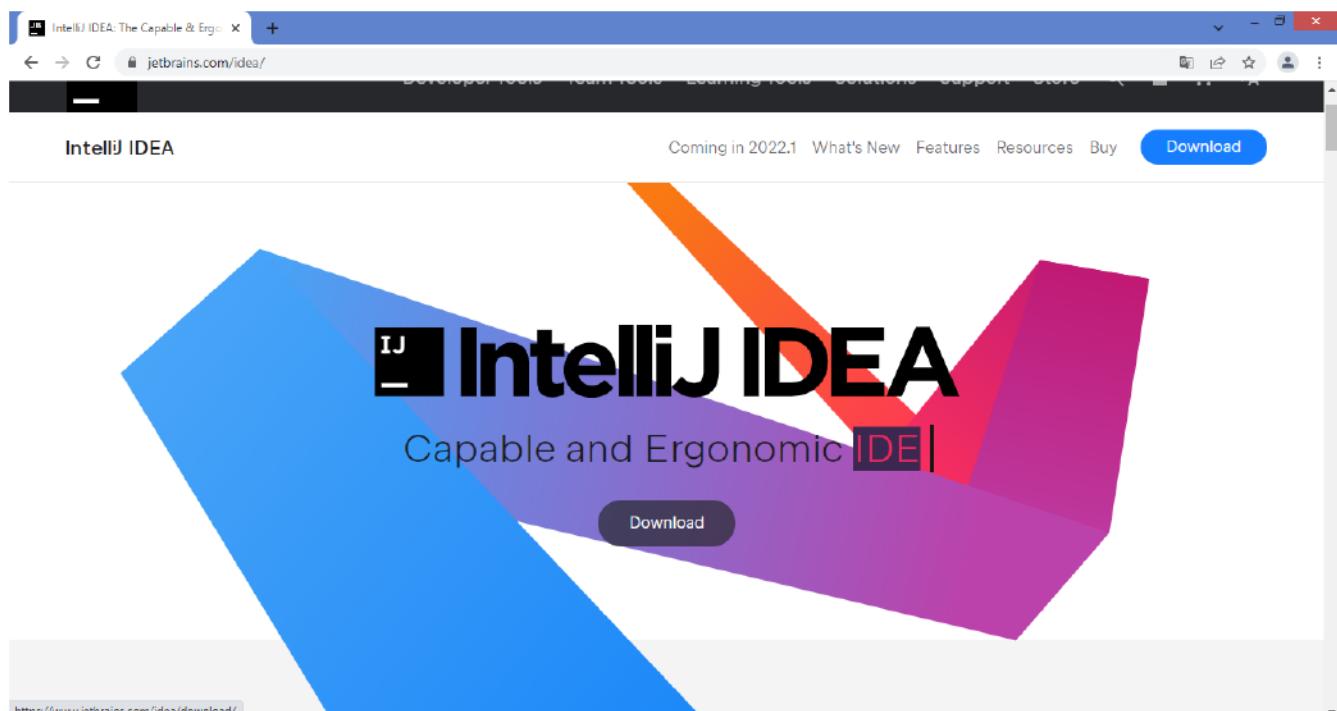
<https://www.jetbrains.com/>



The screenshot shows the JetBrains homepage with a navigation bar at the top. Below the navigation, there's a large grid of tools categorized into several groups:

- IDEs:** AppCode, IntelliJ IDEA (highlighted), CLion, DataGrip, DataSpell, Fleet, GoLand.
- PLUGINS & SERVICES:** All Plugins, IDE Themes, Big Data Tools, Code With Me, QA Tools, Rust, Scala, Toolbox App.
- .NET & VISUAL STUDIO:** Rider, ReSharper, ReSharper C++, dotCover, dotMemory, dotPeek, dotTrace, .NET Tools Plugins.
- LANGUAGES & FRAMEWORKS:** Kotlin, Ktor, MPS, Compose for Desktop.

On the left side of the grid, there are two promotional boxes: "Space" (an all-in-one solution for software projects and teams) and "Fleet" (next-generation IDE by JetBrains). A message at the bottom of the grid says: "Not sure which tool is best for you? Whichever technologies you use, there's a JetBrains tool to match". A "Find your tool" button is located in the bottom right corner of the grid area.



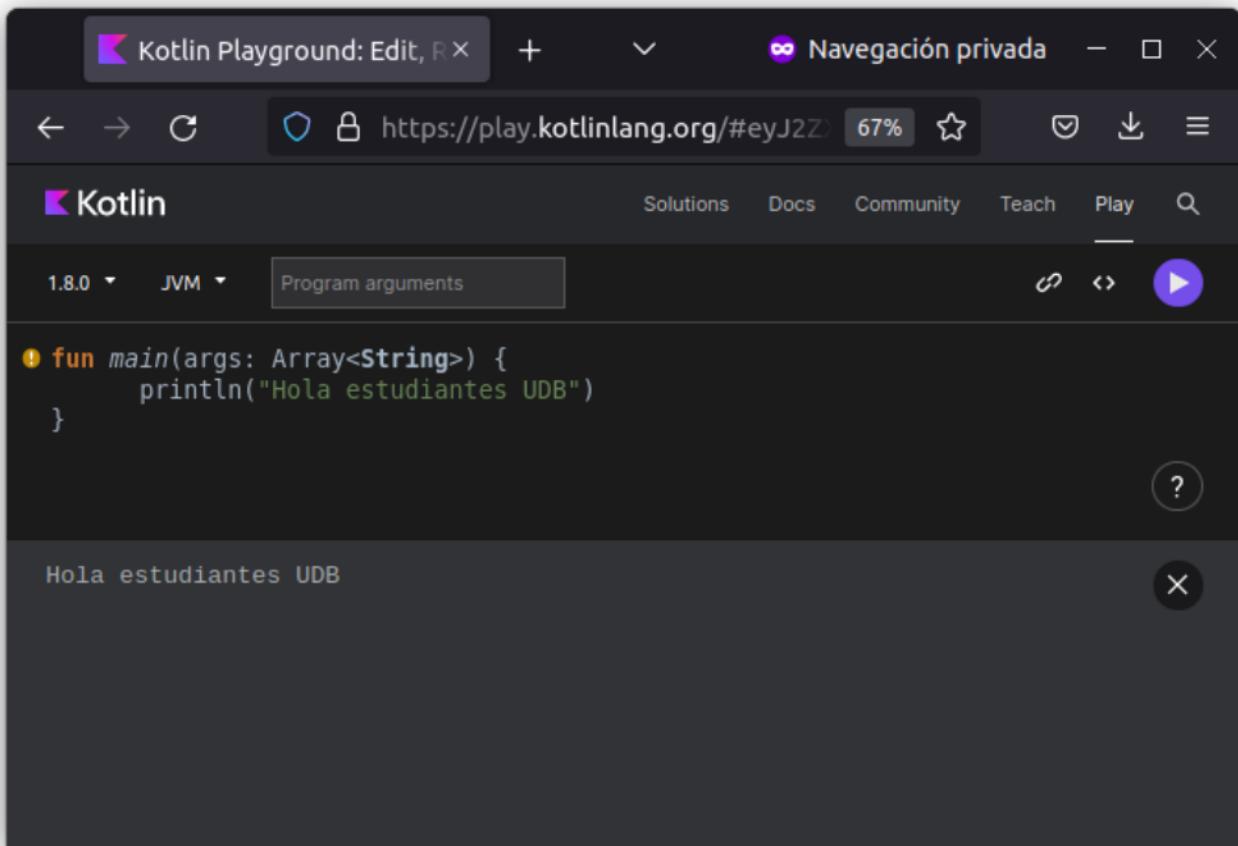
The screenshot shows the IntelliJ IDEA product page. At the top, there's a navigation bar with links to Developer Tools, Team Tools, Learning Tools, Solutions, Support, and Store. Below the navigation, the page title is "IntelliJ IDEA" and there's a "Coming in 2022.1" notice. The main content features a large, stylized "IJ" logo followed by the text "IntelliJ IDEA" and "Capable and Ergonomic IDE". A prominent "Download" button is located in the center of the page. The background has a geometric pattern of overlapping blue, orange, and pink shapes.

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

### III. DESARROLLO DE PRÁCTICA

Desde el entorno de desarrollo <https://play.kotlinlang.org/> digitar y ejecutar los siguientes programas de ejemplo en Kotlin:

**Ejemplo 1.** Primer programa en Kotlin e implementación de la función main



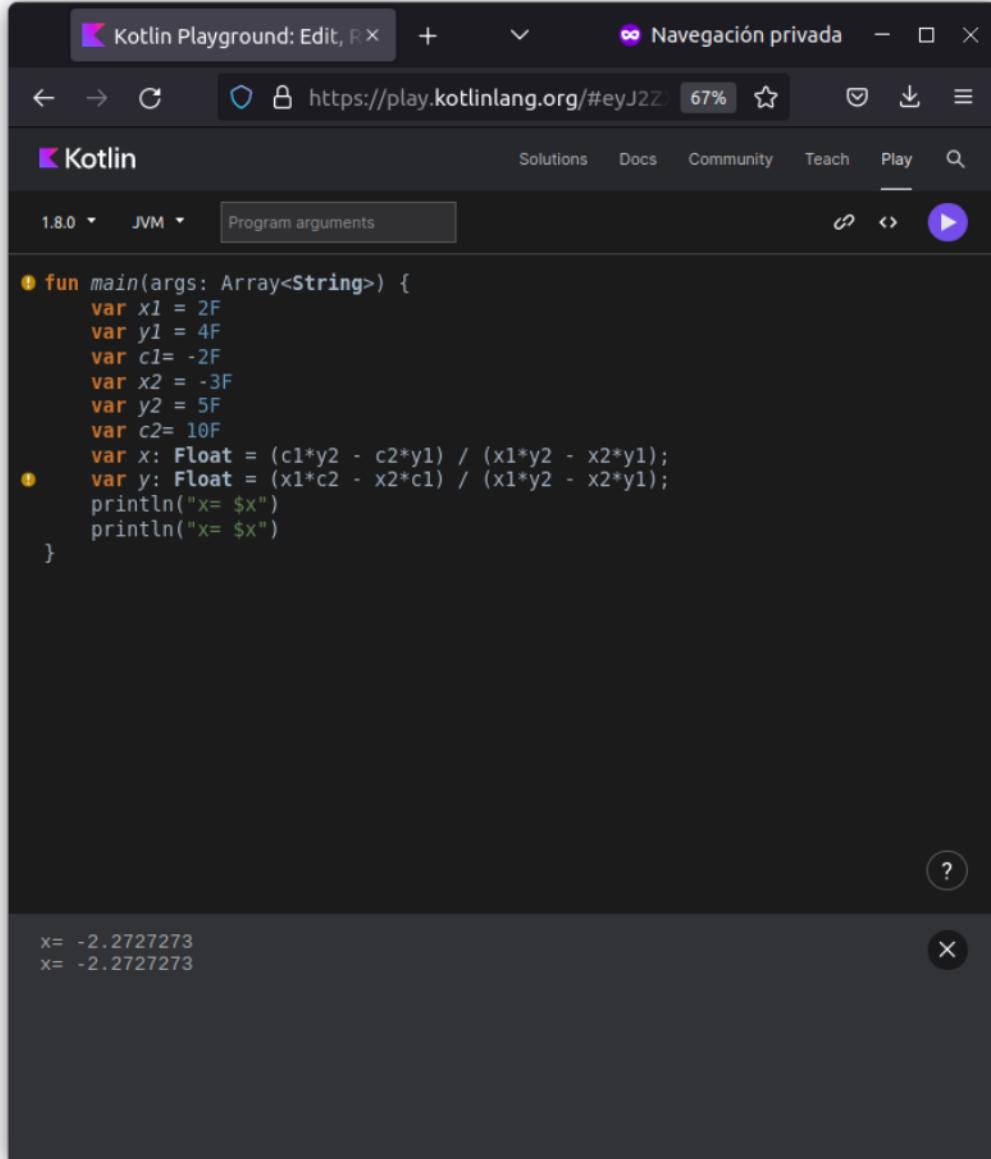
The screenshot shows the Kotlin Playground interface. At the top, it displays "Kotlin Playground: Edit, R X" and "Navegación privada". Below the header, there's a toolbar with navigation icons and a URL bar showing "https://play.kotlinlang.org/#eyJ2Z...". The main workspace is titled "Kotlin" and shows version "1.8.0" and "JVM". It contains a code editor with the following Kotlin code:

```
fun main(args: Array<String>) {
    println("Hola estudiantes UDB")
}
```

Below the code, the output window shows the result: "Hola estudiantes UDB".

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO 01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

## Ejemplo 2. Declaración de variables



The screenshot shows the Kotlin Playground interface. The code in the editor is:

```

! fun main(args: Array<String> {
    var x1 = 2F
    var y1 = 4F
    var c1= -2F
    var x2 = -3F
    var y2 = 5F
    var c2= 10F
    var x: Float = (c1*y2 - c2*y1) / (x1*y2 - x2*y1);
    var y: Float = (x1*c2 - x2*c1) / (x1*y2 - x2*y1);
    println("x= $x")
    println("y= $y")
}

```

The output window below shows the results of the program execution:

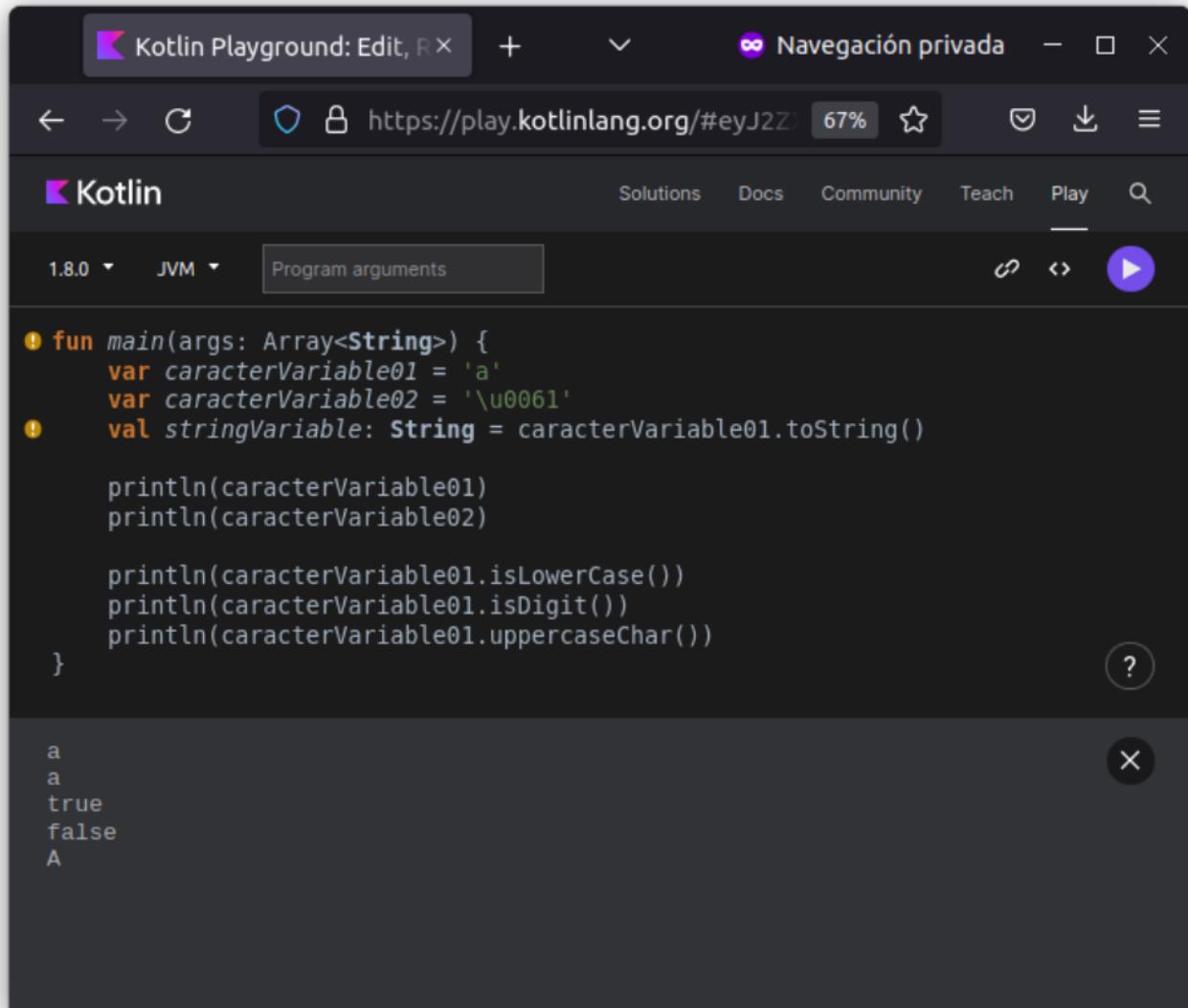
```

x= -2.2727273
y= -2.2727273

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b>

### Ejemplo 3. Declaración de variables



The screenshot shows the Kotlin Playground interface. The code in the editor is:

```

fun main(args: Array<String>) {
    var caracterVariable01 = 'a'
    var caracterVariable02 = '\u0061'
    val stringVariable: String = caracterVariable01.toString()

    println(caracterVariable01)
    println(caracterVariable02)

    println(caracterVariable01.isLowerCase())
    println(caracterVariable01.isDigit())
    println(caracterVariable01.uppercaseChar())
}

```

The output window below the editor shows the results of the program's execution:

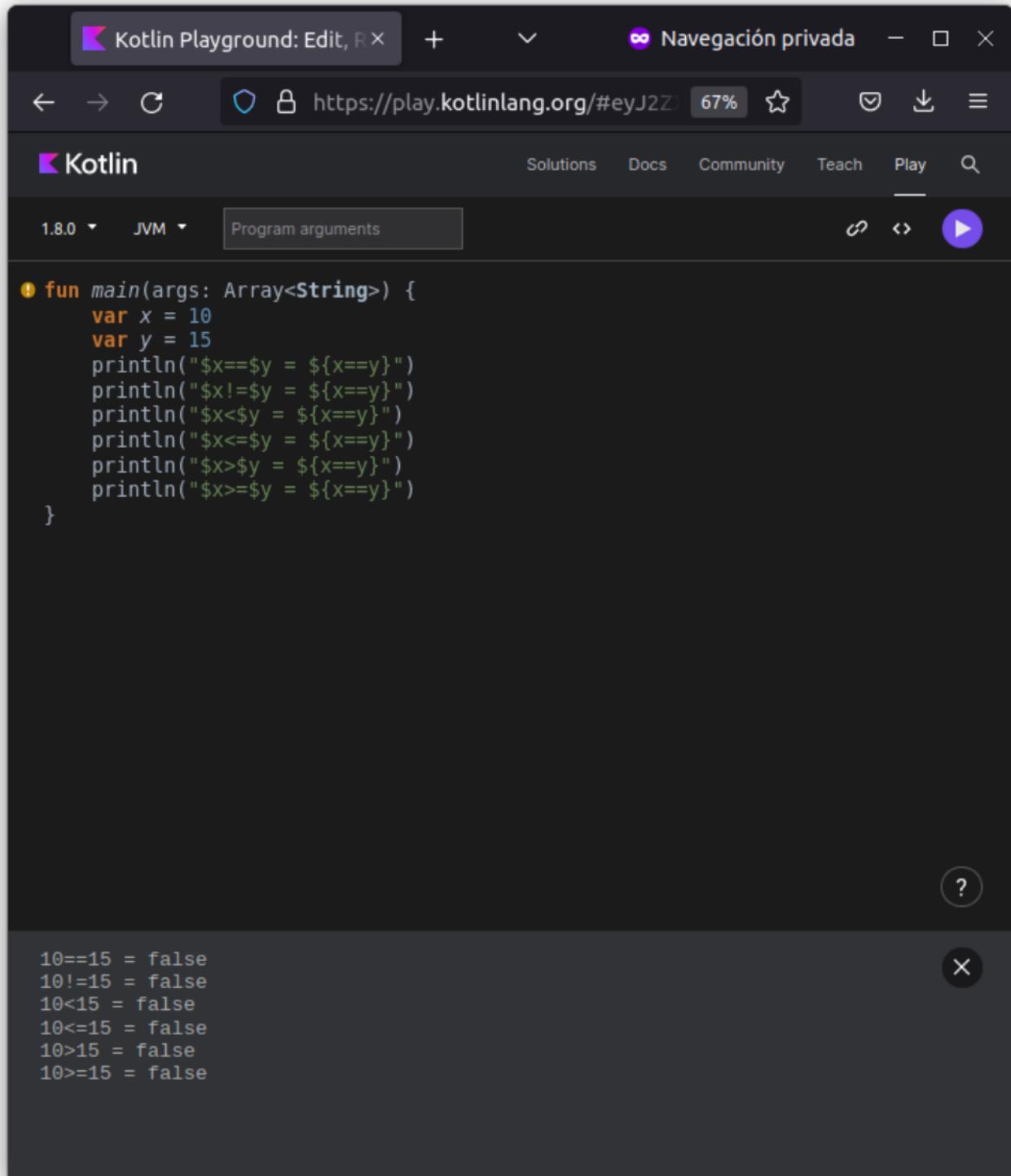
```

a
a
true
false
A

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

#### Ejemplo 4. Declaración de variables y operadores de relación



The screenshot shows the Kotlin Playground interface. The top bar includes the title "Kotlin Playground: Edit, R X", a "+" button, a dropdown menu, and "Navegación privada". Below the bar are standard browser controls (back, forward, search, etc.). The main header says "Kotlin" with version "1.8.0" and "JVM" selected. A "Program arguments" input field is present. On the right, there are icons for copy, share, and run.

```

fun main(args: Array<String>) {
    var x = 10
    var y = 15
    println("$x==$y = ${x==y}")
    println("$x!=$y = ${x==y}")
    println("$x<$y = ${x==y}")
    println("$x<=$y = ${x==y}")
    println("$x>$y = ${x==y}")
    println("$x>=$y = ${x==y}")
}

```

The output window at the bottom displays the results of the println statements:

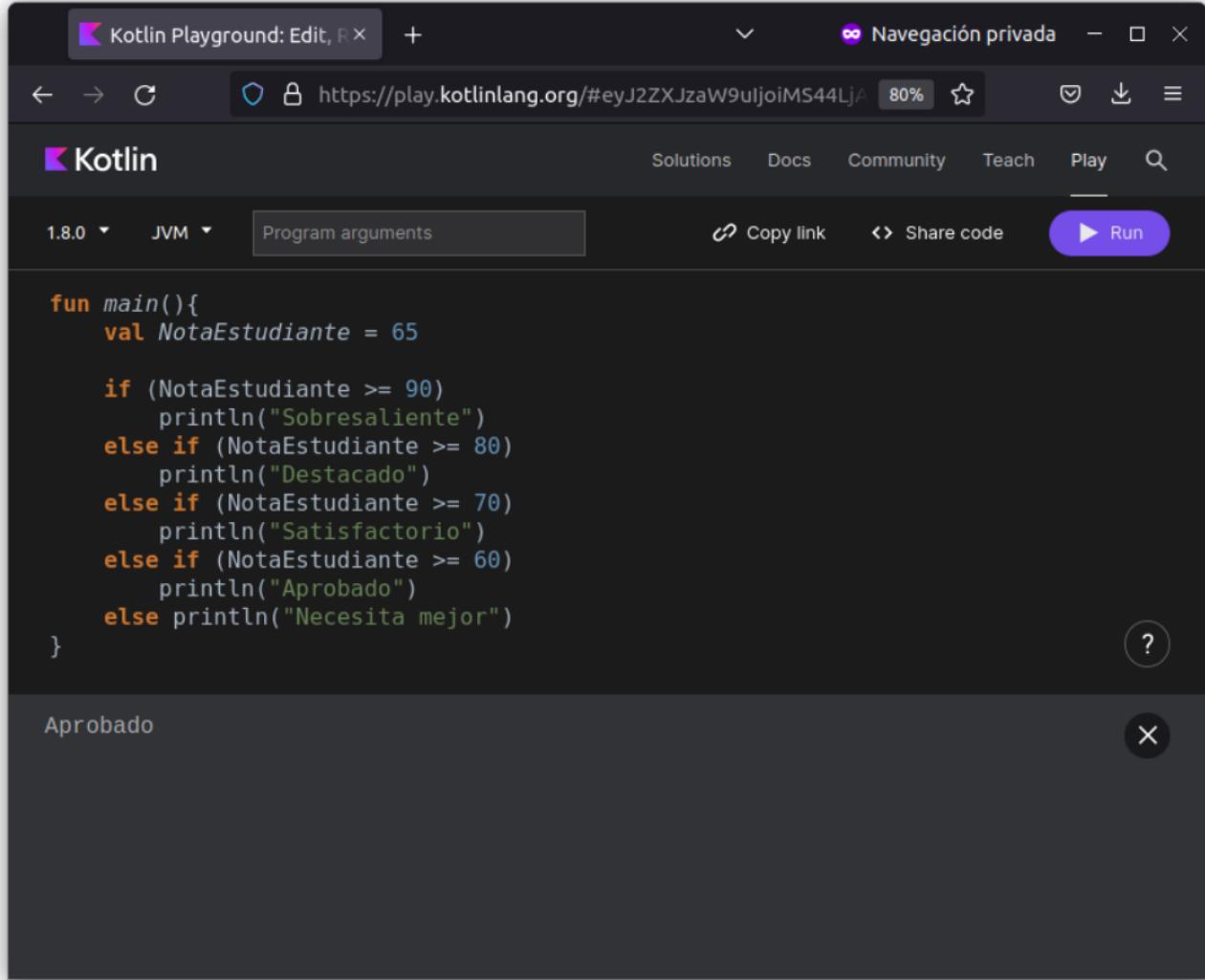
```

10==15 = false
10!=15 = true
10<15 = true
10<=15 = true
10>15 = false
10>=15 = false

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

### Ejemplo 5. Estructura condicional if-else



The screenshot shows the Kotlin Playground interface. The code in the editor is:

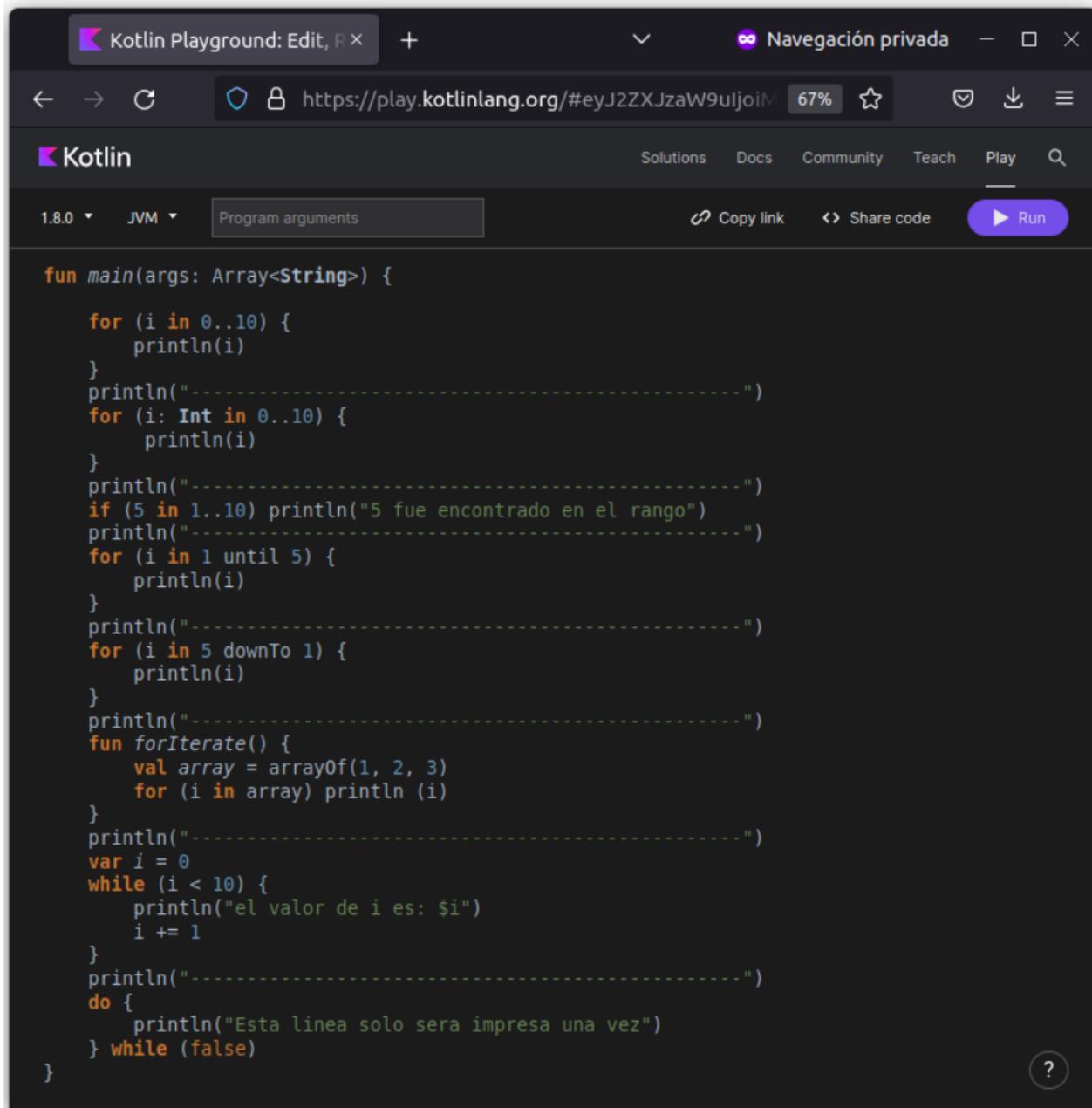
```
fun main(){
    val NotaEstudiante = 65

    if (NotaEstudiante >= 90)
        println("Sobresaliente")
    else if (NotaEstudiante >= 80)
        println("Destacado")
    else if (NotaEstudiante >= 70)
        println("Satisfactorio")
    else if (NotaEstudiante >= 60)
        println("Aprobado")
    else println("Necesita mejor")
}
```

The output window below the editor shows the result: "Aprobado".

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO 01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

## Ejemplo 6. Estructura for



The screenshot shows the Kotlin Playground interface. The URL in the address bar is <https://play.kotlinlang.org/#eyJ2ZXJzaW9uLjoiM...>. The code editor contains the following Java/Kotlin code:

```

fun main(args: Array<String>) {

    for (i in 0..10) {
        println(i)
    }
    println("-----")
    for (i: Int in 0..10) {
        println(i)
    }
    println("-----")
    if (5 in 1..10) println("5 fue encontrado en el rango")
    println("-----")
    for (i in 1 until 5) {
        println(i)
    }
    println("-----")
    for (i in 5 downTo 1) {
        println(i)
    }
    println("-----")
    fun forIterate() {
        val array = arrayOf(1, 2, 3)
        for (i in array) println (i)
    }
    println("-----")
    var i = 0
    while (i < 10) {
        println("el valor de i es: $i")
        i += 1
    }
    println("-----")
    do {
        println("Esta linea solo sera impresa una vez")
    } while (false)
}

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

### Ejemplo 7. Estructura condicional when

Kotlin Playground: Edit, R X + Navegación privada - □ ×

← → C 🔒 https://play.kotlinlang.org/#eyJ2Z... 67% ⭐

Kotlin Solutions Docs Community Teach Play

1.8.0 JVM Program arguments

fun main(args: Array<String>) {
 var varnumero: Int = 3
 var varany: Any = "DSM104"
 var a: Int = 50
 var b: Int = 75

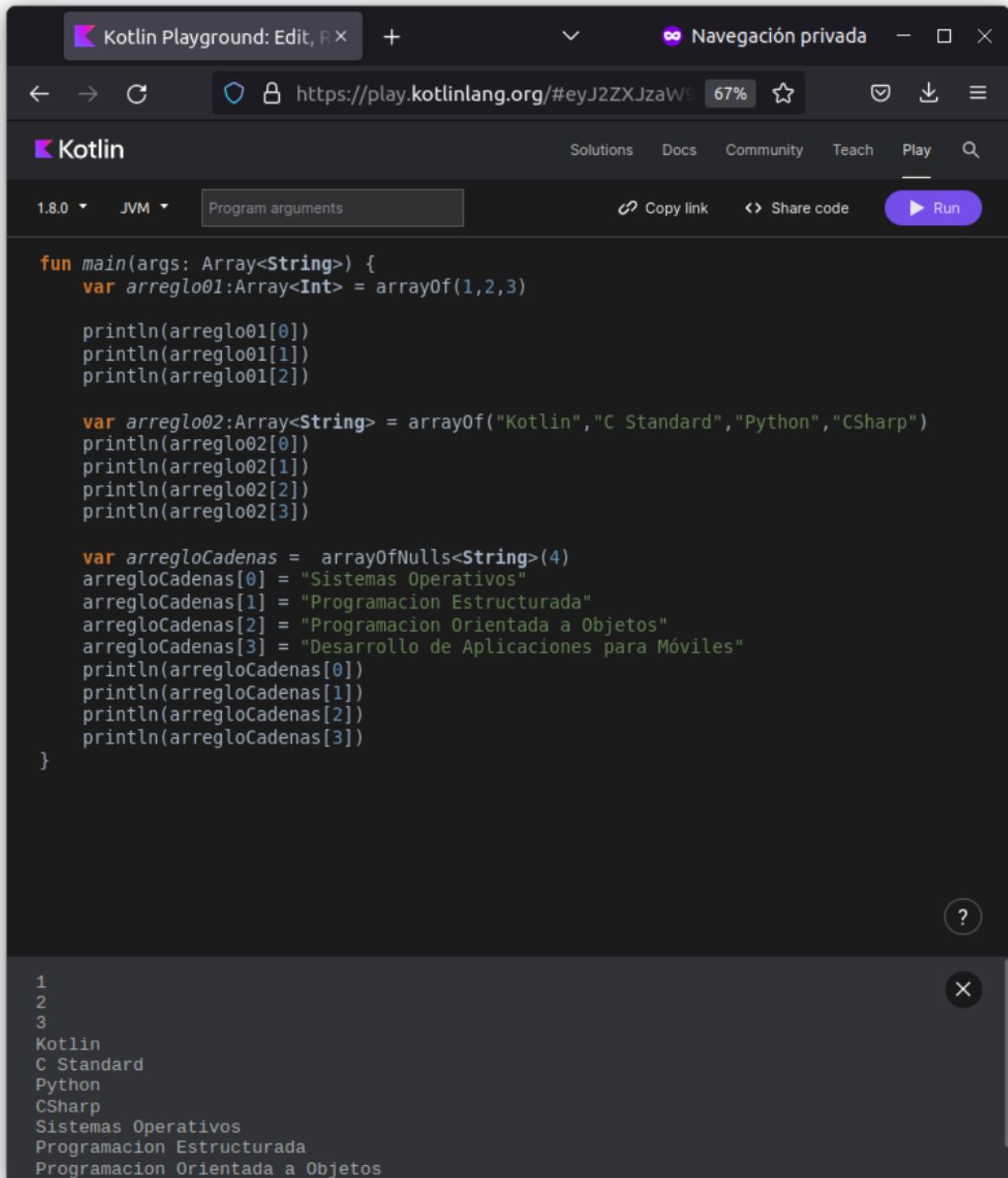
 when(varnumero) {
 1 -> println("varnumero tiene el valor: 1")
 2,3,4,5 -> println("varnumero esta en el rango de 2 a 5")
 else -> "varnumero tiene un valor mayor de 5"
 }

 when (varany) {
 is Int -> println("varany es de tipo Int")
 is Double -> println("varany es de tipo Int Double")
 is String -> println("varany es de tipo string")
 is Long -> println("varany es de tipo Long")
 }
 when {
 a \* b > 100 -> println("el producto de a y b es mayor que 100")
 a + b > 100 -> println("la suma de a y b es mayor que 100")
 a < b -> println("a es menor que b")
 }
}

varnumero esta en el rango de 2 a 5
varany es de tipo string
el producto de a y b es mayor que 100

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

### Ejemplo 8. Arreglos



The screenshot shows the Kotlin Playground interface. The code editor contains the following Kotlin code:

```

fun main(args: Array<String>) {
    var arreglo01:Array<Int> = arrayOf(1,2,3)

    println(arreglo01[0])
    println(arreglo01[1])
    println(arreglo01[2])

    var arreglo02:Array<String> = arrayOf("Kotlin","C Standard","Python","CSharp")
    println(arreglo02[0])
    println(arreglo02[1])
    println(arreglo02[2])
    println(arreglo02[3])

    var arregloCadenas = arrayOfNulls<String>(4)
    arregloCadenas[0] = "Sistemas Operativos"
    arregloCadenas[1] = "Programacion Estructurada"
    arregloCadenas[2] = "Programacion Orientada a Objetos"
    arregloCadenas[3] = "Desarrollo de Aplicaciones para Móviles"
    println(arregloCadenas[0])
    println(arregloCadenas[1])
    println(arregloCadenas[2])
    println(arregloCadenas[3])
}

```

The output window at the bottom displays the results of the printed statements:

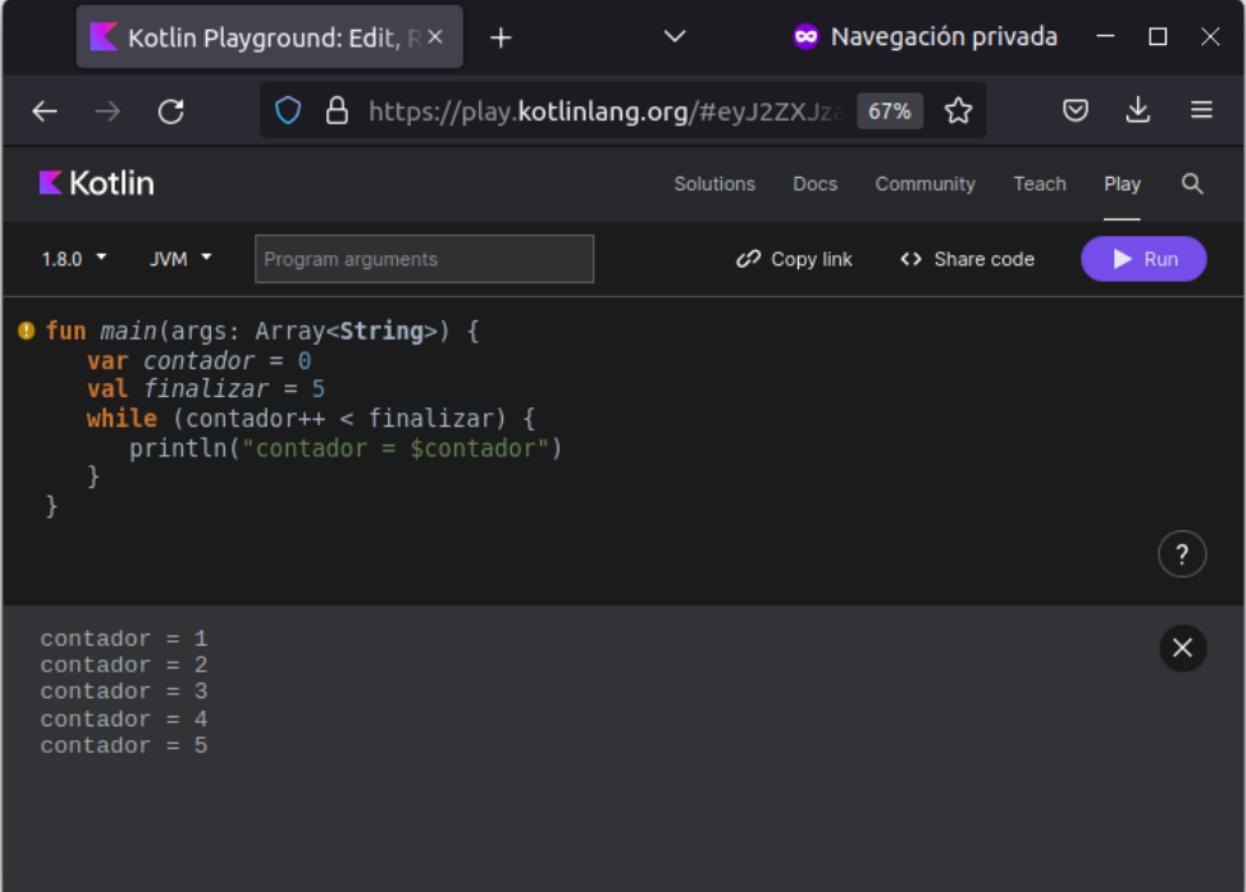
```

1
2
3
Kotlin
C Standard
Python
CSharp
Sistemas Operativos
Programacion Estructurada
Programacion Orientada a Objetos

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

### Ejemplo 9. Estructura while



The screenshot shows a web-based Kotlin playground interface. At the top, it displays the URL <https://play.kotlinlang.org/#eyJ2ZXJZdWVudC1jb2RlIjoiMS4uLmFzLmNvbS5qcy5zaG9wLmNvbmQucGhwIiwic2V0IjoiMS4uLmFzLmNvbS5qcy5zaG9wLmNvbmQucGhwIiwidXNlciI6InVzZXIifQ==>. Below the header, there's a navigation bar with links for Solutions, Docs, Community, Teach, Play, and a search icon. The main area has dropdown menus for Java Version (1.8.0) and JVM. A text input field labeled "Program arguments" is empty. To the right of the code editor are buttons for "Copy link", "Share code", and a large purple "Run" button. The code itself is a simple Kotlin main function that prints a sequence of numbers from 1 to 5 using a while loop:

```

fun main(args: Array<String>) {
    var contador = 0
    val finalizar = 5
    while (contador++ < finalizar) {
        println("contador = $contador")
    }
}

```

The output window below the code shows the printed values:

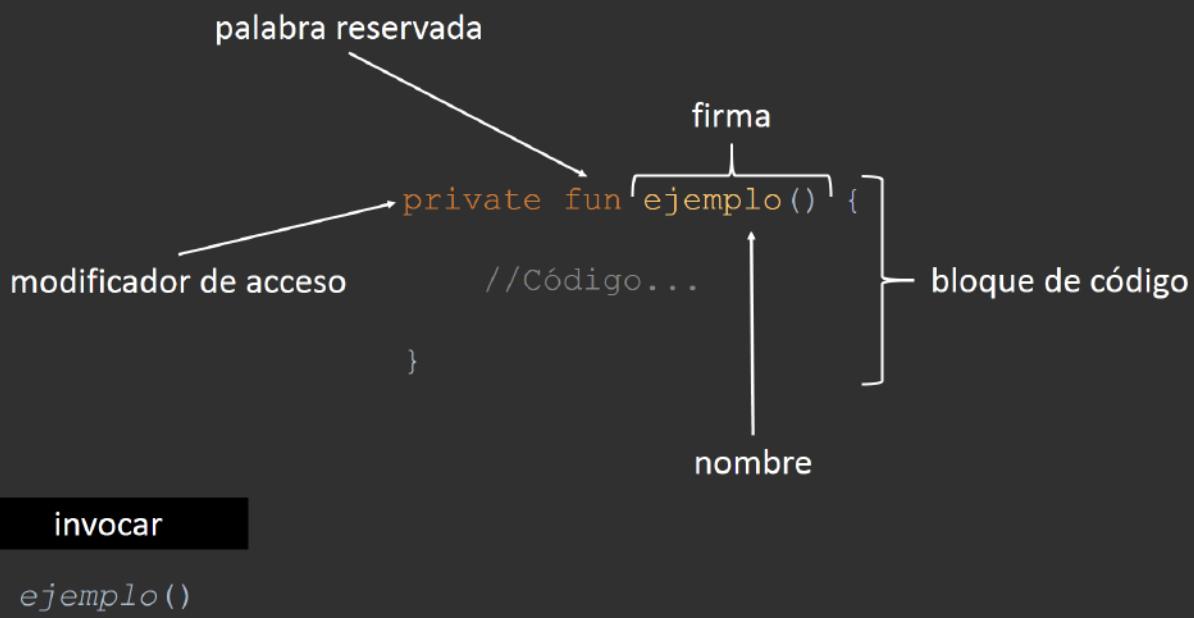
```

contador = 1
contador = 2
contador = 3
contador = 4
contador = 5

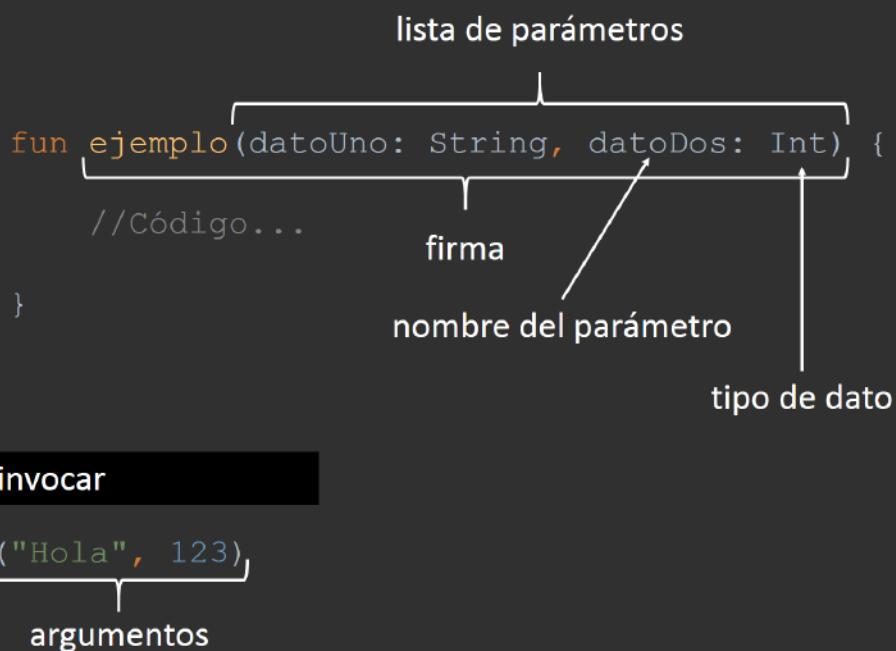
```



## Funciones en Kotlin



## Funciones en Kotlin



 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

## Ejemplo 10. Funciones

Kotlin Playground: Edit, R X + Navegación privada - □ ×

← → C 🔒 https://play.kotlinlang.org/#eyJZXJzaW9... 67% ⭐

Kotlin Solutions Docs Community Teach Play 🔍

1.8.0 JVM Program arguments Copy link Share code Run

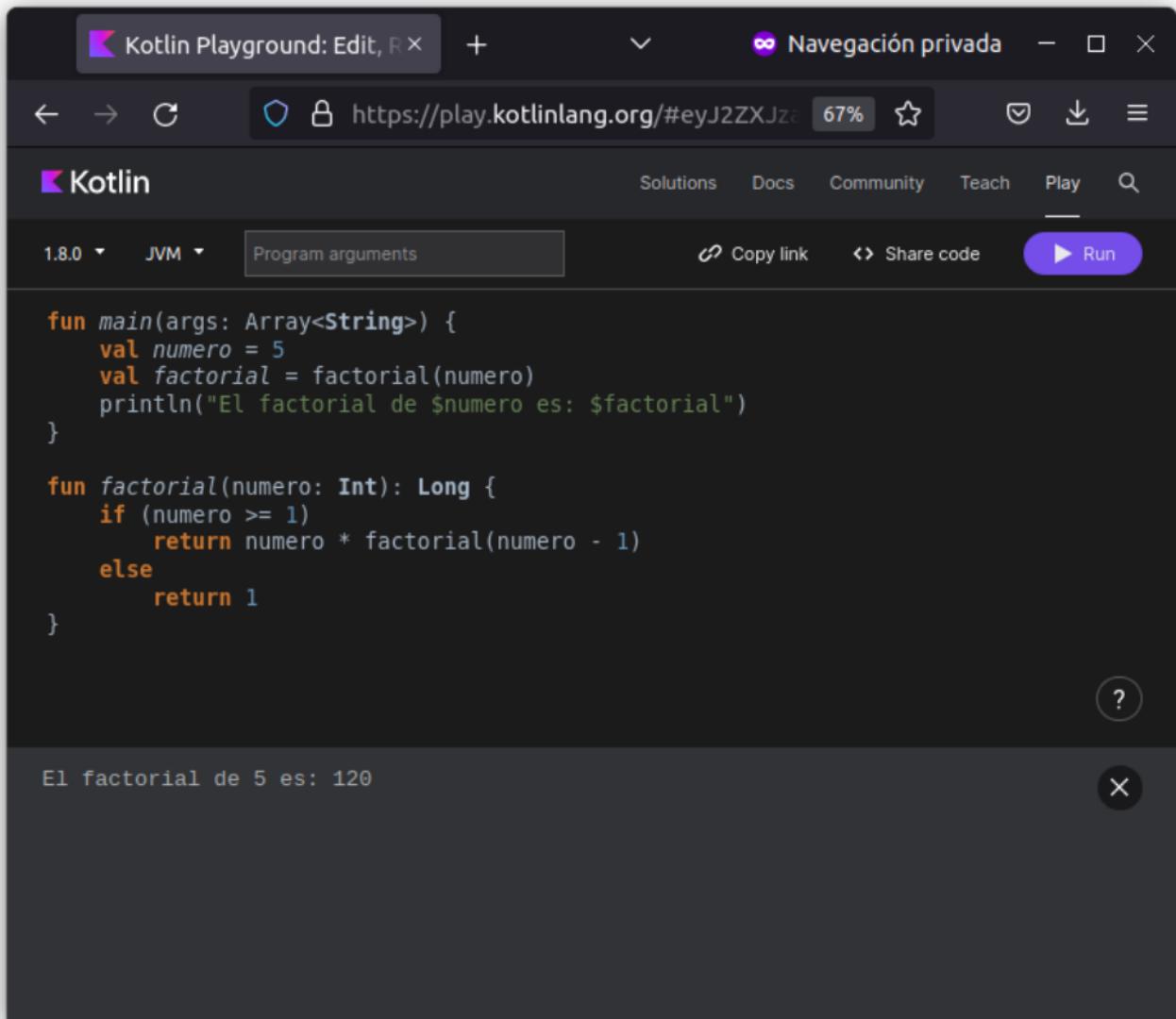
```
! fun main(args: Array<String>) {
    mostrarSaludo("Hola estudiantes DSM104", 5)
}

fun mostrarSaludo(mensaje: String, conteo: Int) : Unit {
    var contador = 1
    while(contador++ <= conteo ) {
        println(mensaje)
    }
}
```

Hola estudiantes DSM104  
 Hola estudiantes DSM104  
 Hola estudiantes DSM104  
 Hola estudiantes DSM104  
 Hola estudiantes DSM104

 <p><b>UNIVERSIDAD DON BOSCO</b> <b>FACULTAD DE INGENIERÍA</b> <b>ESCUELA DE COMPUTACIÓN</b></p>	<b>CICLO 01-2024</b>
<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b>

## Ejemplo 11. Funciones



The screenshot shows the Kotlin Playground interface. The code in the editor is:

```

fun main(args: Array<String>) {
    val numero = 5
    val factorial = factorial(numero)
    println("El factorial de $numero es: $factorial")
}

fun factorial(numero: Int): Long {
    if (numero >= 1)
        return numero * factorial(numero - 1)
    else
        return 1
}

```

The output window at the bottom displays the result:

```

El factorial de 5 es: 120

```

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

## Ejemplo 12. Clases

Kotlin

1.8.0 ▾ JVM ▾ Program arguments

```

import java.util.*

class Nota(var nombre:String, var contenido:String, var fechaCreacion:Date){

    init{
        nombre = nombre.toUpperCase()
    }

    constructor(nombre: String, contenido: String):this(nombre, contenido, Date())

    fun imprimirTotalCaracterContenido(){
        println(contenido.length)
    }
    fun contarPalabrasContenido(): Int {
        val palabras = contenido.split(" ")
        return palabras.size
    }

    fun contarPalabrasContenido(palabra:String): Int {
        val palabras = contenido.split(" ")
        var contador = 0
        palabras.forEach(){
            if(it.toUpperCase() == palabra.toUpperCase()) contador++
        }

        return contador
    }

}

fun main(){
    val nota1 = Nota("Kotlin","Kotlin es un lenguaje de programación orientada a objetos. ",Date())
    println(nota1.contarPalabrasContenido())
    println(nota1.contarPalabrasContenido("kotlin"))
    println(nota1.contenido)
}

```

10  
1  
Kotlin es un lenguaje de programación orientada a objetos.

 <p><b>UNIVERSIDAD DON BOSCO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>ESCUELA DE COMPUTACIÓN</b></p>	<p><b>CICLO</b>  <b>01-2024</b></p>
<p><b>MATERIA</b></p>	<p><b>DESARROLLO DE SOFTWARE PARA MÓVILES</b></p>
<p><b>PRÁCTICA</b></p>	<p><b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b></p>

### Ejemplo 13. Clases y Herencia

```

Kotlin

1.8.0 JVM Program arguments

import java.time.LocalDate
import java.time.Period
import java.time.format.DateTimeFormatter

open class Persona(
    var nombre: String,
    var apellidos: String,
    var direccion: String,
    var telefono: String?,
    var fechaNacimientoTexto: String
) {
    var fechaNacimientoDate: LocalDate = LocalDate.parse(fechaNacimientoTexto,DateTimeFormatter.ofPattern(FORMATO_FECHA))

    fun obtenerEdad(): Int{
        return Period.between(fechaNacimientoDate, LocalDate.now()).years
    }

    fun obtenerNombre(): String{
        return nombre
    }
    fun obtenerApellidos(): String{
        return apellidos
    }
    fun obtenerDireccion(): String{
        return direccion
    }
    fun obtenerfechaNacimiento(): String{
        return fechaNacimientoTexto
    }
    companion object{
        const val FORMATO_FECHA = "dd/MM/yyyy"
    }
}

class Empleado(
    nombre: String,
    apellidos: String,
    direccion: String,
    telefono: String?,
    fechaNacimientoTexto: String
) : Persona(nombre, apellidos, direccion, telefono, fechaNacimientoTexto) {

}

fun main(){
    val persona = Persona("Walter Ovidio","Sanchez Campos","Colonia Los Abedules","00000000","20/02/1991")
    println(persona.obtenerNombre())
    println(persona.obtenerApellidos())
    println(persona.obtenerDireccion())
    println(persona.obtenerfechaNacimiento())
    println(persona.obtenerEdad())
    val empleado = Empleado("Jaime David","Campos Sanchez","Colonia Los Abedules","00000000","20/02/1995")
    println(empleado.obtenerNombre())
    println(empleado.obtenerApellidos())
    println(empleado.obtenerDireccion())
    println(empleado.obtenerfechaNacimiento())
    println(empleado.obtenerEdad())
}

```



**UNIVERSIDAD DON BOSCO  
FACULTAD DE INGENIERÍA  
ESCUELA DE COMPUTACIÓN**

**CICLO  
01-2024**

<b>MATERIA</b>	<b>DESARROLLO DE SOFTWARE PARA MÓVILES</b>	<b>GUIA DE LABORATORIO Nº 1</b>
<b>PRÁCTICA</b>	<b>INTRODUCCIÓN JAVA Y KOTLIN - Parte II</b>	

Walter Ovidio  
Sanchez Campos  
Colonia Los Abedules  
20/02/1991  
31  
Jaime David  
Campos Sanchez  
Colonia Los Abedules  
20/02/1995  
27



## **V. BIBLIOGRAFÍA**

- Cómo Programar en Java Deitel, Harvey M.
- Kotlin docs 1.8.0. <https://kotlinlang.org/docs/home.html>