	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

I. RESULTADO DEL APRENDIZAJE

Que al finalizar la practica el estudiante conozca:

- Las generalidades sobre el lenguaje de programación Java.
- Utilice el lenguaje de programación Java para la creación de aplicaciones móviles.
- Conocer los fundamentos del entorno de desarrollo Android.
- Preparar el entorno de desarrollo con Android Studio.
- Aprender a configurar un emulador para probar aplicaciones.

II. INTRODUCCIÓN TEORICA

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.

Podemos mencionar algunas 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 ejecutaran en tiempo real muchas funciones al mismo tiempo.

Debemos de comprender algunos conceptos técnicos antes de comenzar a programar en 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 compilas 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 escritorio.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

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:

Tipo	Tamaño en bits	Rango de almacenamiento
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:

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

Booleanos y caracteres:

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


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";
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

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

Identificadores

Un identificador es un nombre que identifica a una variable, a un método o función miembro, a una clase. Todos los lenguajes tienen ciertas reglas para componer los identificadores:

- Todos los identificadores han de comenzar con una letra, el carácter subrayado (_) o el carácter dollar (\$).
- Puede incluir, pero no comenzar por un número
- No puede incluir el carácter espacio en blanco
- Distingue entre letras mayúsculas y minúsculas
- No se pueden utilizar las palabras reservadas como identificadores


Además de estas restricciones, hay ciertas convenciones que hacen que el programa sea más legible, pero que no afectan a la ejecución del programa. La primera y fundamental es la de encontrar un nombre que sea significativo, de modo que el programa sea lo más legible posible. El tiempo que se pretende ahorrar eligiendo nombres cortos y poco significativos se pierde con creces cuando se revisa el programa después de cierto tiempo.

Operadores

En una condición deben disponerse únicamente variables, valores constantes y operadores relacionales.

Operadores Relacionales:

Operador	Uso	Devuelve verdadero si
>	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

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Operadores Matemáticos:

Operador	Uso	Descripción
+	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:

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

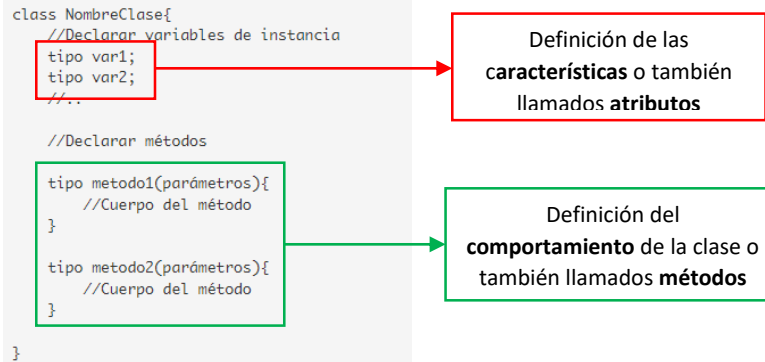
Nota: Los operadores lógicos siempre devuelven un valor booleano.

¿Qué es una clase 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.**

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

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");
    }
}

```

¿Qué son los 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.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

```
//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");
    }
}
```

Conversión de tipos de datos en java

En Java es posible transformar el tipo de una variable u objeto en otro diferente al original con el que fue declarado. Este proceso se denomina "conversión", "moldeado" o "tipado" y es algo que debemos manejar con cuidado pues un mal uso de la conversión de tipos es frecuente que dé lugar a errores. De forma general trataremos de atenernos a la norma de que "en las conversiones debe evitarse la pérdida de información". En la siguiente tabla vemos conversiones que son seguras por no suponer pérdida de información.


Tipo origen	Tipo destino
byte	double, float, long, int, char, short
short	double, float, long, int
char	double, float, long, int
int	double, float, long
long	double, float
float	Double

String a INT

```
String numCadena = "1";
int numEntero = Integer.parseInt(numCadena);
```

INT a String

```
int numEntero = 4;
String numCadena= String.valueOf(numEntero);
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

O

```
String numCadena= Integer.toString(numEntero);
```

III. DESARROLLO DE PRÁCTICA

Para realizar la práctica puede utilizar una herramienta online o la de su preferencia. Se sugiere utilizar:

<https://repl.it/>

Ejemplo #1: Lectura de datos de entrada

Para leer la entrada de consola, lo primero que se hace es construir un Scanner que este asociado al flujo de entrada estándar System.in

```
Scanner in = new Scanner(System.in);
```

Y ahora se utilizan los distintos métodos de la clase Scanner para leer la entrada. Por ejemplo, el método nextLine lee una línea de entrada.

```
System.out.print("¿Cómo te llamas? ");
String nombre = in.nextLine();
```


En este caso, se utiliza el método nextLine porque la entrada podría contener espacios. Para leer una sola palabra (delimitada por espacios de separación), se hace la llamada.

```
String nombreDepila=in.next();
```

Para leer un entero utilice el método nextInt.

```
System.out.print("¿Cuántos años tienes?");
Int edad = int.nextInt();
```

Digite el siguiente código y verifique su funcionamiento:

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        // Creando un objeto de la clase Scanner
        Scanner in = new Scanner(System.in);
        //obtener el primer dato
        System.out.println("¿Como te llamas? ");
        String nombre = in.nextLine();

        //obtener el segundo dato
        System.out.println("¿Cuántos años tienes? ");
        int edad = in.nextInt();

        //mostrar el resultado en la consola
        System.out.println("Hola, " + nombre + ". El año que viene tendrás " + (edad + 1) + " años");
    }
}
```

Resultado:

```
javac -classpath ./run_dir/junit-4.12.jar:target/dependency/* Main.java
java -classpath ./run_dir/junit-4.12.jar:target/dependency/* Main
¿Como te llamas?
Pedro
¿Cuántos años tienes?
25
Hola, Pedro. El año que viene tendrás 26 años
```

Ejemplo #2: Estructura de programación secuencial

Nota: Cuando en un problema sólo participan operaciones, entradas y salidas se la denomina una estructura secuencial.

Considere realizar el siguiente ejercicio, desarrolle un programa que reciba la carga de dos números enteros por teclado e imprimir su suma y su producto.

Digite el siguiente código y verifique su funcionamiento:

```
import java.util.Scanner;

public class SumaProductoNumeros {

    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        int num1, num2, suma, producto;
        System.out.print("Ingrese primer valor: ");
        num1 = teclado.nextInt();

        System.out.print("Ingrese segundo valor: ");
        num2 = teclado.nextInt();
        suma = num1 + num2;
        producto = num1 * num2;
        System.out.print("La suma de los dos valores es: ");
        System.out.println(suma);
        System.out.print("El producto de los dos valores es: ");
        System.out.println(producto);
    }
}
```


Ejemplo #3: Estructura condicionales simples y compuestas

No todos los problemas pueden resolverse empleando estructuras secuenciales. Cuando hay que tomar una decisión aparecen las estructuras condicionales.

Ejemplo #3.1

Ingresar el salario de una persona, si supera los 3000 dólares mostrar un mensaje en pantalla indicando que debe abonar impuestos. Podemos observar lo siguiente: Siempre se hace la carga del sueldo, pero si el sueldo que ingresamos supera 3000 dólares se mostrará por pantalla el mensaje "Esta persona debe abonar impuestos", en caso que la persona cobre 3000 o menos no aparece nada por pantalla.

Digite el siguiente código y verifique su funcionamiento:

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

```
import java.util.Scanner;
public class EstructuraCondicionalSimple1 {

    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        float sueldo;
        System.out.print("Ingrese el sueldo:");
        sueldo = teclado.nextFloat();
        if (sueldo > 3000) {
            System.out.println("Esta persona debe abonar impuestos");
        }
    }
}
```

Ejemplo #3.2

Desarrolle un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:

- Si el promedio es ≥ 7 mostrar "Promocionado".
- Si el promedio es ≥ 4 y < 7 mostrar "Regular".
- Si el promedio es < 4 mostrar "Reprobado".

Digite el siguiente código y verifique su funcionamiento:


```
import java.util.Scanner;
public class EstructuraCondicionalAnidada1 {

    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        int nota1, nota2, nota3;
        System.out.print("Ingrese primer nota: ");
        nota1 = teclado.nextInt();
        System.out.print("Ingrese segunda nota: ");
        nota2 = teclado.nextInt();
        System.out.print("Ingrese tercer nota: ");
        nota3 = teclado.nextInt();
        int promedio = (nota1 + nota2 + nota3) / 3;
        if (promedio >= 7) {
            System.out.print("Promocionado");
        } else {
            if (promedio >= 4) {
                System.out.print("Regular");
            } else {
                System.out.print("Reprobado");
            }
        }
    }
}
```

Estructuras repetitivas

Hasta ahora hemos empleado estructuras SECUENCIALES y CONDICIONALES. Existe otro tipo de estructuras tan importantes como las anteriores que son las estructuras REPETITIVAS.

Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Ejemplo #4: Estructura repetitiva while

Escribir un programa que solicite la carga de un valor positivo y nos muestre desde 1 hasta el valor ingresado de uno en uno. Ejemplo: Si ingresamos 30 se debe mostrar en pantalla los números del 1 al 30.

Digite el siguiente código y verifique su funcionamiento:

```
import java.util.Scanner;
public class EstructuraRepetitivaWhile2 {
    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        int n, x;
        System.out.print("Ingrese el valor final:");
        n = teclado.nextInt();
        x = 1;
        while (x <= n) {
            System.out.print(x);
            System.out.print(" - ");
            x = x + 1;
        }
    }
}
```

Ejemplo #5: Estructura repetitiva for


Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura while. Pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones. En general, la estructura for se usa en aquellas situaciones en las cuales CONOCEMOS la cantidad de veces que queremos que se ejecute el bloque de instrucciones.

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio. Este problema ya lo desarrollamos, lo resolveremos empleando la estructura for.

Digite el siguiente código y verifique su funcionamiento:

```
import java.util.Scanner;

public class EstructuraRepetitivaFor {
    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        int suma, f, valor, promedio;
        suma = 0;
        for (f = 1; f <= 10; f++) {
            System.out.print("Ingrese valor:");
            valor = teclado.nextInt();
            suma = suma + valor;
        }
        System.out.print("La suma es:");
        System.out.println(suma);
        promedio = suma / 10;
        System.out.print("El promedio es:");
        System.out.print(promedio);
    }
}
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Ejemplo #6: Estructura repetitiva do while

La estructura do while es otra estructura repetitiva, la cual ejecuta al menos una vez su bloque repetitivo, a diferencia del while o del for que podían no ejecutar el bloque. Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo. La condición de la estructura está abajo del bloque a repetir, a diferencia del while o del for que está en la parte superior.

Escribir un programa que solicite la carga de números por teclado, obtener su promedio. Finalizar la carga de valores cuando se cargue el valor 0.

Digite el siguiente código y verifique su funcionamiento:

```
import java.util.Scanner;


public class EstructuraRepetitivaDoWhile2 {

    public static void main(String[] ar) {
        Scanner teclado = new Scanner(System.in);
        int suma, cant, valor, promedio;
        suma = 0;
        cant = 0;
        do {
            System.out.print("Ingrese un valor (0 para finalizar):");
            valor = teclado.nextInt();
            if (valor != 0) {
                suma = suma + valor;
                cant++;
            }
        } while (valor != 0);
        if (cant != 0) {
            promedio = suma / cant;
            System.out.print("El promedio de los valores ingresados es:");
            System.out.print(promedio);
        } else {
            System.out.print("No se ingresaron valores.");
        }
    }
}
```

Ejemplo #7: Clases

Confeccionar un programa que permita cargar los nombres de 5 personas y su mail, luego implementar los siguientes métodos:

- Mostrar por pantalla los datos.
- Consulta del mail ingresando su nombre.
- Mostrar los emails que no tienen el carácter @.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Digite el siguiente código y verifique su funcionamiento:

```
import java.util.Scanner;

public class Cadena5 {

    private Scanner teclado;
    private String[] nombres;
    private String[] mail;


    public Cadena5() {
        teclado = new Scanner(System.in);
        nombres = new String[5];
        mail = new String[5];
        for (int f = 0; f < nombres.length; f++) {
            System.out.print("Ingrese nombre:");
            nombres[f] = teclado.nextLine();
            System.out.print("Ingrese mail:");
            mail[f] = teclado.nextLine();
        }
    }

    public void listar() {
        for (int f = 0; f < nombres.length; f++) {
            System.out.println(nombres[f] + " - " + mail[f]);
        }
    }

    public void consultaMail() {
        String aux;
        System.out.print("Ingrese el nombre de la persona:");
        aux = teclado.nextLine();
        boolean existe = false;
        for (int f = 0; f < nombres.length; f++) {
            if (aux.equals(nombres[f])) {
                System.out.println("Mail de la persona:" + mail[f]);
                existe = true;
            }
        }
        if (existe == false) {
            System.out.println("No existe una persona con ese nombre.");
        }
    }

    public void sinArroba() {
        for (int f = 0; f < mail.length; f++) {
            boolean tiene = false;
            for (int k = 0; k < mail[f].length(); k++) {
                if (mail[f].charAt(k) == '@') {
                    tiene = true;
                }
            }
            if (tiene == false) {
                System.out.println(mail[f] + " no tiene @");
            }
        }
    }

    public static void main(String[] ar) {
        Cadena5 cad = new Cadena5();
        cad.listar();
        cad.consultaMail();
        cad.sinArroba();
    }
}
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

INTRODUCCION TEORICA

Android

Sistema operativo basado en Linux, diseñado principalmente para móviles con pantalla táctil como teléfonos inteligentes o tabletas inicialmente desarrollados por Android, Inc., que Google respaldó financieramente y más tarde compró en 2005.

Versionamientos.


ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99.8%
4.2 Jelly Bean	17	99.2%
4.3 Jelly Bean	18	98.4%
4.4 KitKat	19	98.1%
5.0 Lollipop	21	94.1%
5.1 Lollipop	22	92.3%
6.0 Marshmallow	23	84.9%
7.0 Nougat	24	73.7%
7.1 Nougat	25	66.2%
8.0 Oreo	26	60.8%
8.1 Oreo	27	53.5%
9.0 Pie	28	39.5%
10. Android 10	29	8.2%

Fundamentos.

- Las herramientas de Android SDK generan un archivo. apk.
- Un APK tiene archivos de recursos y datos (paquete de Android)
-


Una vez instalada en el dispositivo, cada aplicación de Android se aloja en su propia zona de pruebas de seguridad:

- El sistema operativo Android es un sistema Linux multiusuario en el que cada aplicación es un usuario diferente.
- De forma predeterminada, el sistema le asigna a cada aplicación una ID de usuario de Linux única (solo el sistema utiliza la ID y la aplicación la desconoce). El sistema establece permisos para todos los archivos en una aplicación de modo que solo el ID de usuario asignado a esa aplicación pueda acceder a ellos.
- Cada proceso tiene su propio equipo virtual (EV), por lo que el código de una aplicación se ejecuta de forma independiente de otras aplicaciones.
- De forma predeterminada, cada aplicación ejecuta su proceso de Linux propio.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Componentes.

Componente	Descripción	Clase y etiqueta para declararlo en archivo de manifiesto.
Actividades (Activity)	Una <i>actividad</i> representa una pantalla con interfaz de usuario. Ej. App de correo, App de contactos, etc.	Activity <activity>
Servicios (Service)	Un <i>servicio</i> es un componente que se ejecuta en segundo plano para realizar operaciones prolongadas o tareas para procesos remotos. Un servicio no proporciona una interfaz de usuario. Por ejemplo, un servicio podría reproducir música en segundo plano mientras el usuario se encuentra en otra aplicación	Service <service>
Proveedores de contenido (ContentProvider)	Un <i>proveedor de contenido</i> administra un conjunto compartido de datos de la app. Puedes almacenar los datos en el sistema de archivos, en una base de datos SQLite, en la Web o en cualquier otra ubicación de almacenamiento persistente a la que tu aplicación pueda acceder. A través del proveedor de contenido, otras aplicaciones pueden consultar o incluso modificar los datos (si el proveedor de contenido lo permite)	ContentProvider <provider>
Receptores de mensajes (BroadcastReceiver)	Un <i>receptor de mensajes</i> es un componente que responde a los anuncios de mensajes en todo el sistema. Muchos mensajes son originados por el sistema; por ejemplo, un mensaje que anuncie que se apagó la pantalla, que la batería tiene poca carga o que se tomó una foto. Las aplicaciones también pueden iniciar mensajes; por ejemplo, para permitir que otras aplicaciones sepan que se descargaron datos al dispositivo y están disponibles para usarlos.	BroadcastReceiver <receiver>

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

PROCEDIMIENTO

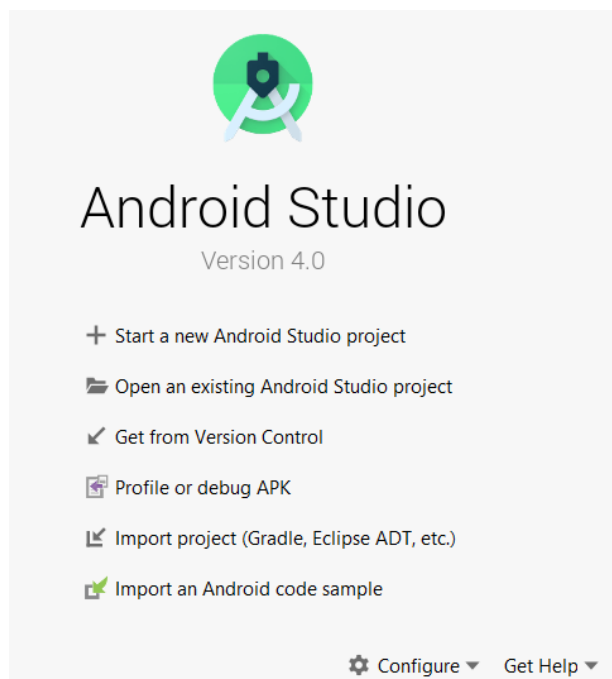
Para realizar la práctica debe tener instalado:


- **Java JDK** (Según el sistema operativo que posea en su computador)
 - Enlace de descarga: <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>
- **Android Studio.**
 - Enlace de descarga: <https://developer.android.com/studio>

Creación de un nuevo proyecto con Android Studio.

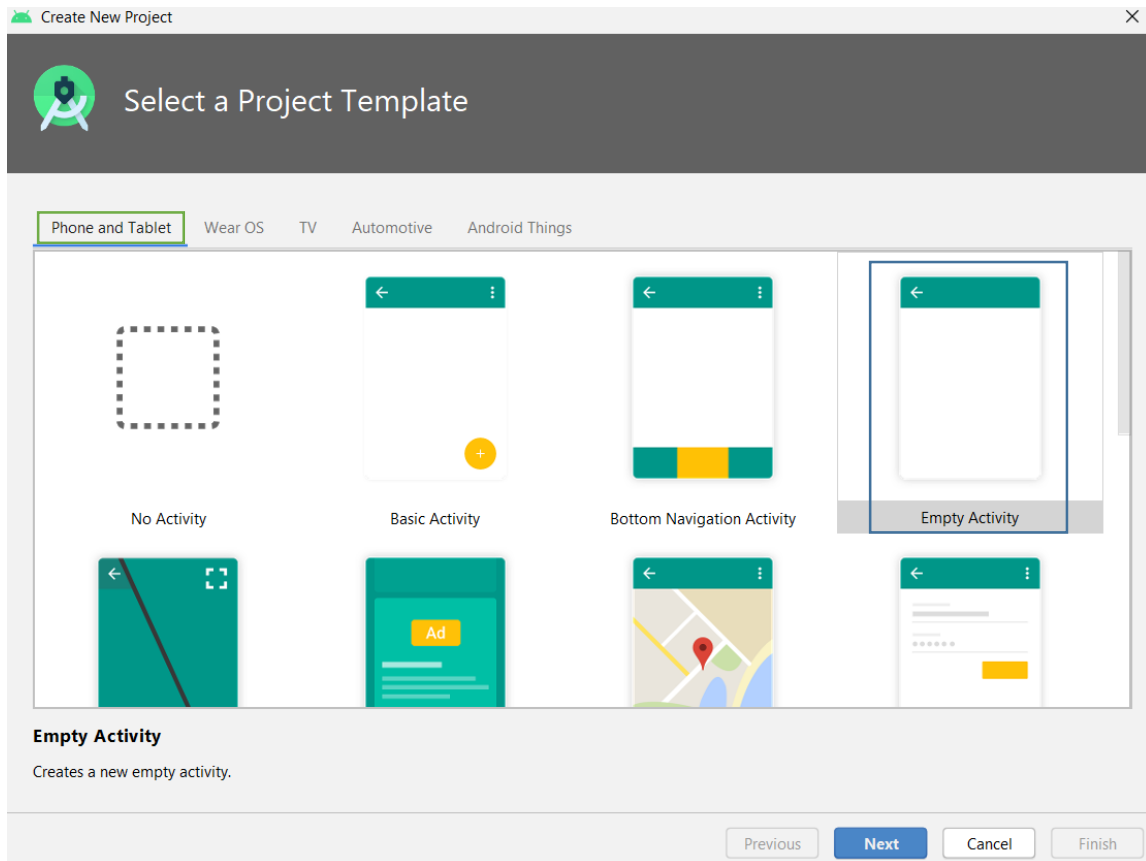
Teniendo instalado Java JDK y Android Studio siga los siguientes pasos:

1. Buscar la aplicación Android Studio. Esto permite ingresar a la pantalla de inicio de Android Studio como la siguiente:




	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

2. Seleccione la opción **“Start a new Android Studio Project”** o su equivalente en español, para proceder a crear nuestro primer proyecto. Se mostrará la siguiente pantalla para poder: seleccionar el tipo de dispositivo donde vamos a poder ejecutar nuestra aplicación Android, alguna plantilla de proyecto



predeterminada, etc.

3. Vamos a crear nuestro ejemplo para que pueda ejecutarse **en teléfonos y tabletas** y además vamos a seleccionar una de las plantillas que vienen por denominada **“Empty Activity”** (Actividad vacía)
4. Selecciona el botón **Next** (Siguiendo)

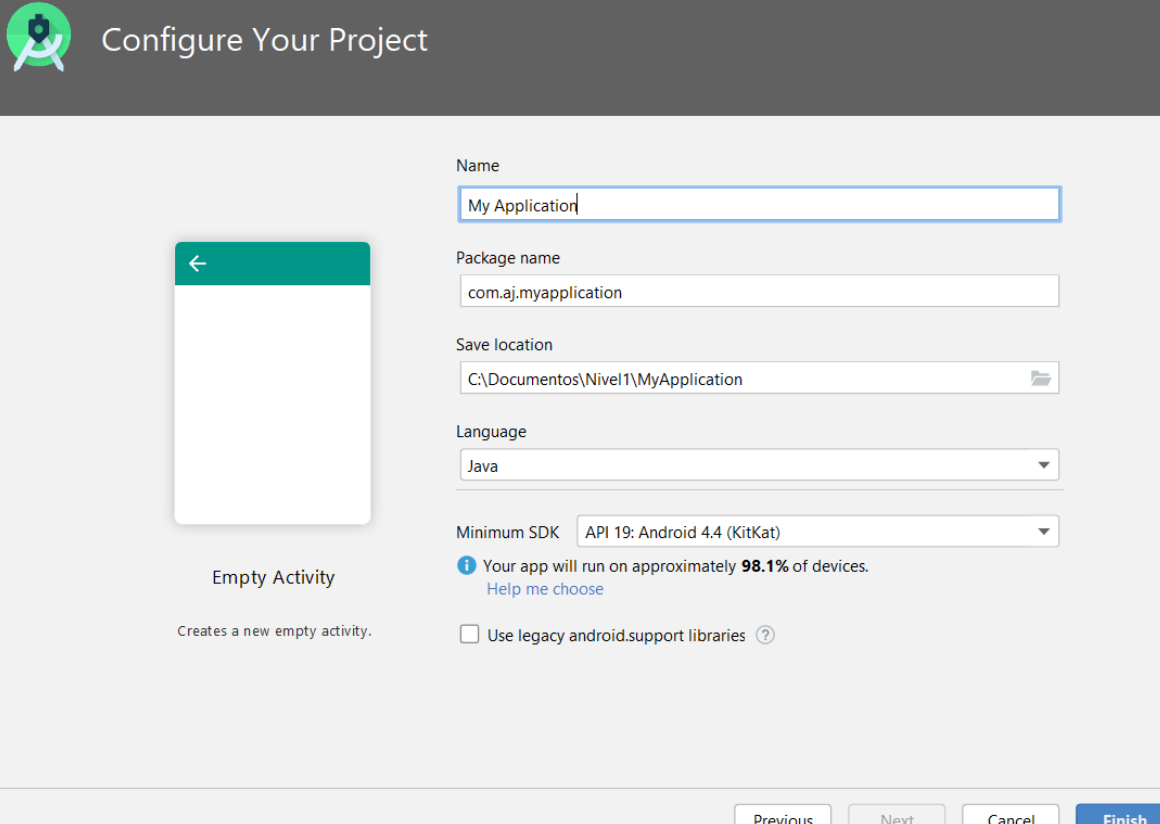
	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Deberá aparecer una pantalla como la siguiente:

Esta pantalla nos permite configurar:

- a- **El nombre de nuestra aplicación.** (Será el nombre que el usuario verá en su dispositivo cuando instale la aplicación)
- b- **El nombre del paquete.** Representa el identificador único de la aplicación y debes asegurarte de que no se repita con ninguna otra aplicación Android en el mundo. Este nombre de paquete será el nombre de la carpeta donde se instalará la aplicación en el dispositivo (dentro de \data\data) por lo tanto es imprescindible que selecciones un nombre que representa a la aplicación como única. También cuando quieras subir tu aplicación a la tienda de Google Play para distribuirla, Google realizará una validación evitando que este nombre de paquete ya exista previamente asignado a otra aplicación.
- c- **Ubicación del proyecto (Save location).** Es la carpeta donde se guardará tu proyecto fuente
- d- **Lenguaje.** Es el lenguaje que utilizarás para desarrollar tu aplicación (**Java** o Kotlin)
- e- **SDK Mínimo.** Es la versión mínima de Android con la cuál tu aplicación será compatible. Es decir, si un cliente quiere instalar tu aplicación en su dispositivo deberá tener instalado en su dispositivo ésta versión de Android como mínimo.

5- Selecciona **Finish** (Finalizar)



Configure Your Project

Name
My Application

Package name
com.aj.myapplication

Save location
C:\Documentos\Nivel1\MyApplication

Language
Java


Minimum SDK
API 19: Android 4.4 (KitKat)

Empty Activity
Creates a new empty activity.

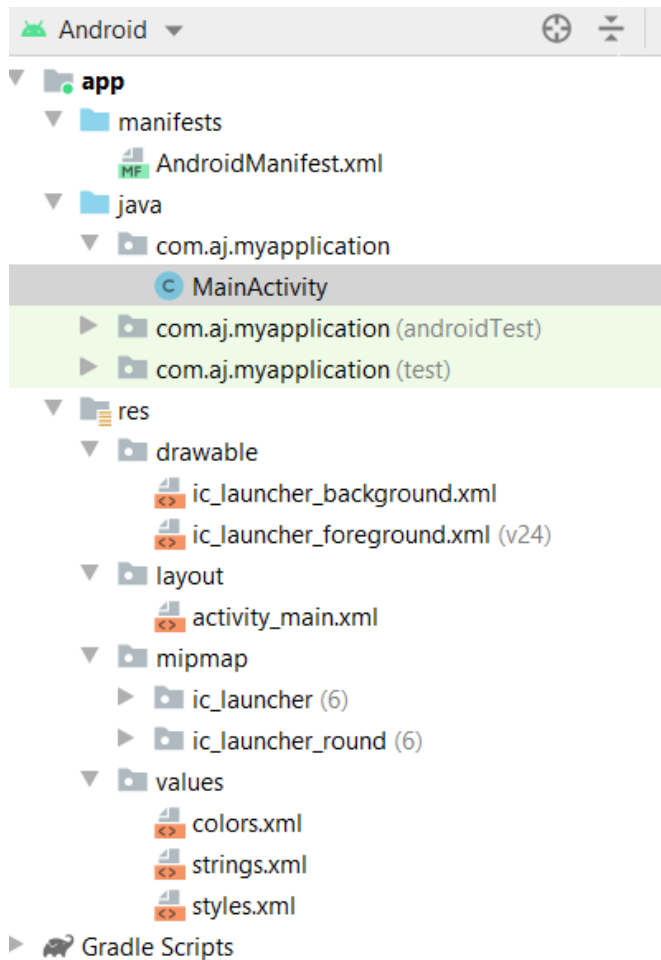
Summary: Your app will run on approximately **98.1%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries

Buttons: Previous, Next, Cancel, **Finish**

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Espera unos segundos mientras se configura el proyecto y al final deberás observar lo siguiente:



descripción breve.

AndroidManifest.xml: Archivo de configuración de la aplicación para definir algunos elementos como: permisos, actividades, servicios, u otros componentes que formaran parte de la aplicación.

\java\<nombre de paquete>

Paquete principal de la aplicación donde podremos encontrar la actividad principal de nuestro proyecto y donde podremos agregar otros paquetes y otras clases.

res\drawable

Carpeta que puede contener recursos de tipo imágenes.

res\layout

Carpeta que puede contener recursos que representan los diseños de pantalla de nuestra aplicación.

res\mipmap

Carpeta que contiene recursos que contienen imágenes que representan a nuestra aplicación.

res\values


Carpeta que puede contener recursos como listas de valores que podrán ser leídos por nuestra aplicación (listas de colores, estilos, cadenas, menús, etc.)

Configuración de emulador.

Un emulador es una máquina virtual que representa un dispositivo Android con una versión del sistema operativo y un hardware determinado.

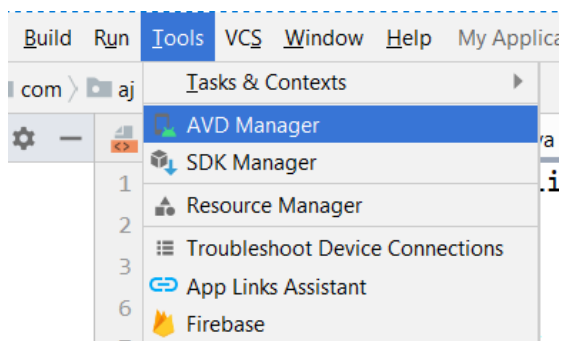
Con el emulador podremos hacer pruebas de nuestra aplicación antes de ser puesta en un entorno de producción y sin la necesidad (si así se requiere) de contar con un dispositivo físico.

Antes de poder ejecutar la aplicación que acabamos de crear con los pasos anteriores vamos a crear un emulador donde será ejecutada dicha aplicación.

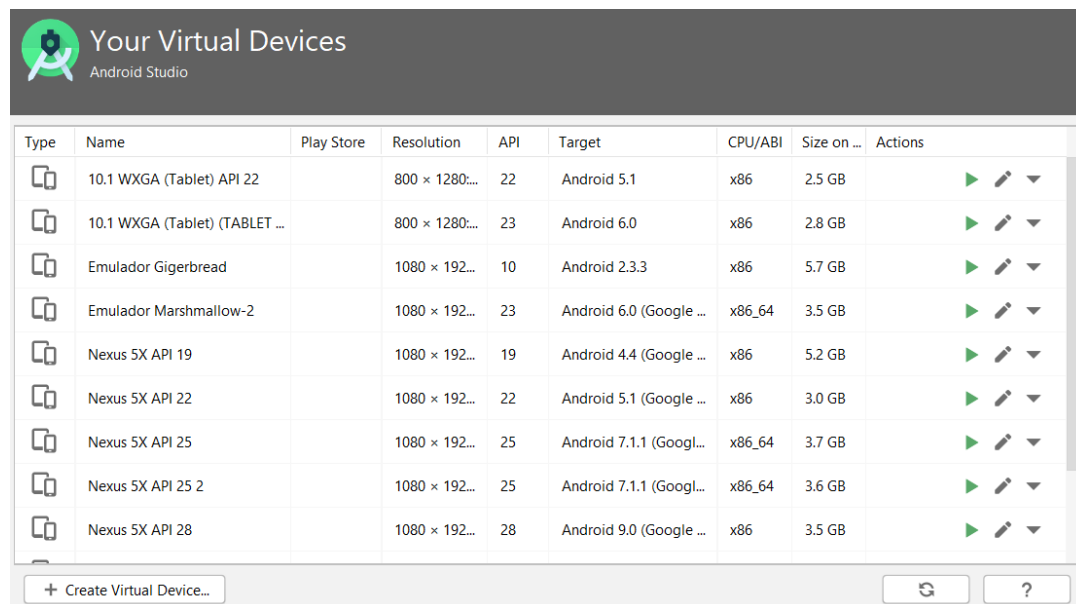
	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Creación de emulador.

1. Selecciona las opciones **Tools->AVD Manager** (Herramientas) en el menú principal de Android studio. . Esto permite ingresar a la ventana de administrador de Dispositivos Virtuales de Android (AVD)




Esto te permitirá acceder a la siguiente pantalla:



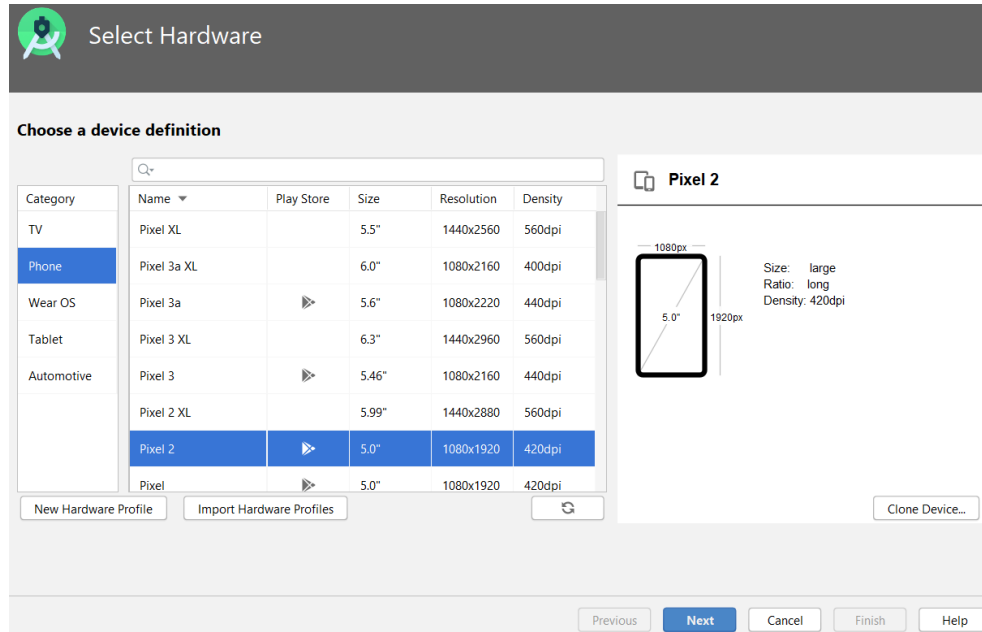
Esta consola te permite administrar todos tus dispositivos virtuales que tengas configurado en tu equipo.

Esto quiere decir que puedes: Agregar, eliminar, modificar características, activar dichos dispositivos virtuales.

Podrás contar con varios dispositivos (Ten cuidado con esto, pues cada dispositivo ocupa una carpeta con archivos en tu disco duro y el espacio utilizado será proporcional a las características que hayas definido para cada dispositivo virtual.)

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	
		GUIA DE LABORATORIO N° 1	

2- Selecciona el botón “Create Virtual Device...” (Crear dispositivo virtual)



Esta ventana permite definir las características de hardware que simulará tu dispositivo virtual:

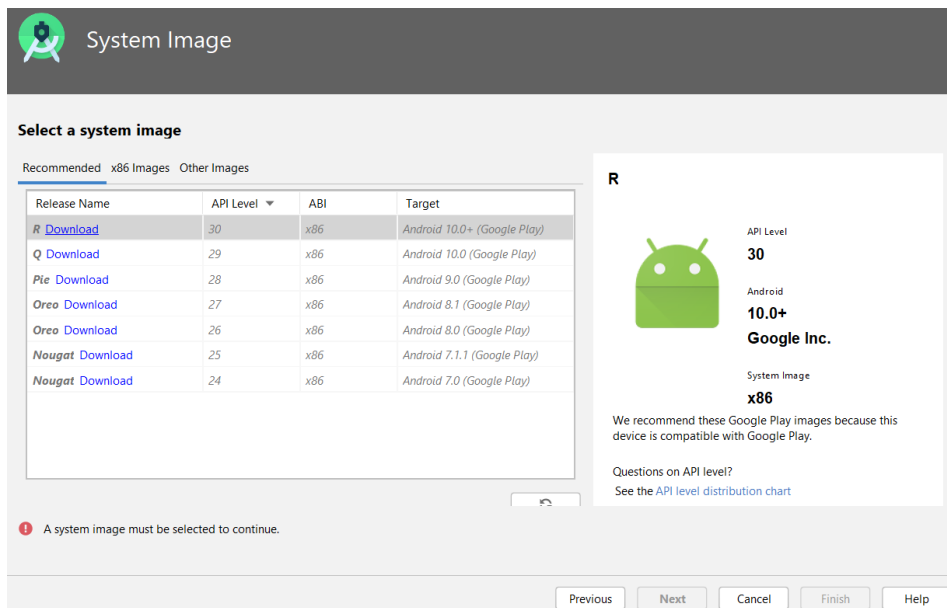
(categoría de dispositivo: teléfono, Tablet, etc)

Tamaño de la pantalla

3- Selecciona en Category : **Phone** y un modelo como el mostrado en pantalla : **pixel 2 (5.0")**

4- Selecciona el botón “Next” (Siguiente)


Ahora procederemos a seleccionar la imagen o versión del sistema operativo Android que deseamos emular en nuestro dispositivo.



En el tab “Recommended” podrás encontrar versiones de sistema operativo recomendado por el asistente.

Si tienes instalado/descargado en tu SDK estas versiones podrías seleccionar una de estas, de lo contrario observarás que a la para de cada versión de sistema operativo esta una opción “Download” (No la selecciones).

La descarga de una versión del sistema operativo de android puede tardarse varios minutos dependiendo de tu enlace de internet.

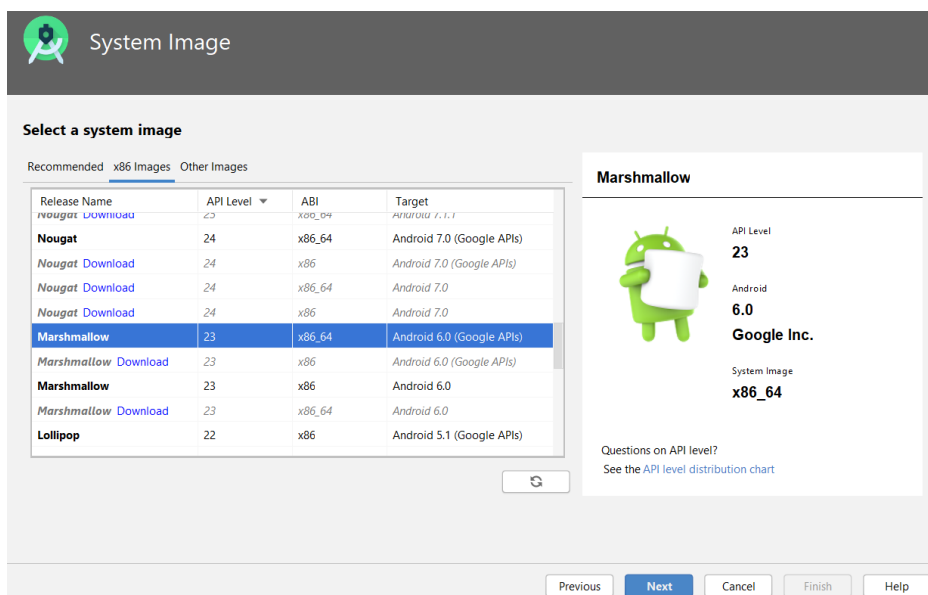
	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Puedes también revisar en la pestaña **x86-images** para ver que opciones tienes disponibles además y poder seleccionar una versión de android.

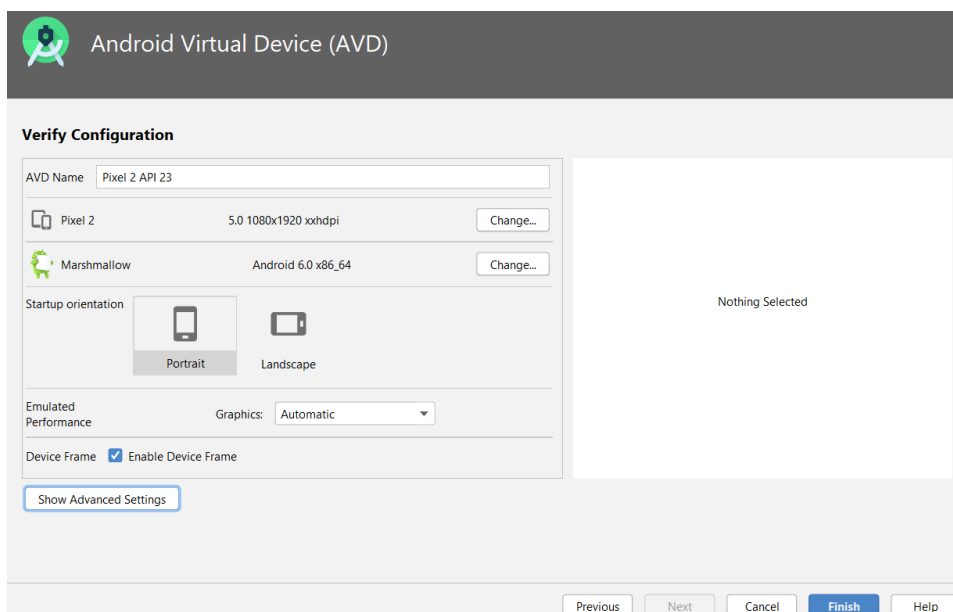
Busca una de las imágenes que ya tengas disponibles en tu computador (una que no tenga “download”), de lo contrario tendrás que descargar una.

5- En el ejemplo se selecciona **Marshmallow** como ejemplo. (Pero tú puedes seleccionar otra imagen que ya tengas disponible)

6- Seleccionar el botón **“Next”** (Siguiendo)




Ahora puedes darle un nombre a tu dispositivo virtual y configurar algunas características adicionales.



7- Haz un clic en **“Show advanced settings”** (Mostrar opciones avanzadas)

Para configurar algunas opciones como:

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Memoria Ram

Memoria interna

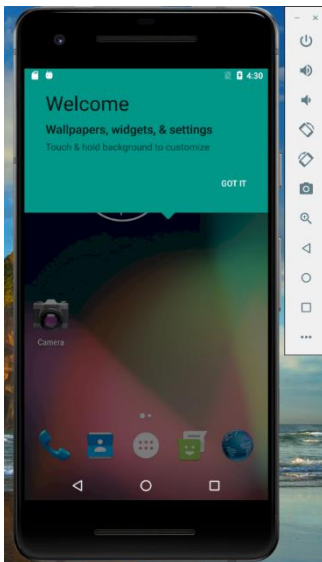
Memoria SD

Considerar NO colocar mucha memoria pues será consumida de tu computadora cada vez que inicies el dispositivo virtual.

RAM:	<input type="text" value="1536"/>	MB ▾
VM heap:	<input type="text" value="256"/>	MB ▾
Internal Storage:	<input type="text" value="800"/>	MB ▾
SD card:	<input checked="" type="radio"/> Studio-managed <input type="text" value="512"/> MB ▾	
	<input type="radio"/> External file <input type="text" value=""/>	
	<input type="radio"/> No SDCard	
<input checked="" type="checkbox"/> Enable Device Frame Custom skin definition		
	<input type="text" value="pixel_2"/> ▾	<input type="text" value="..."/>

8- Selecciona el botón **“Finish”** (Finalizar), espera unos segundos y regresaras a la ventana principal del AVD Manager. Ahora tu dispositivo virtual está listo. selecciónalo y haz clic en el botón **play** para ejecutarlo.


	NEXUS 5X API 26		1080 × 1920	26	Android 9.0 (Google ...	x86	5.5 GB	
	Pixel 2 API 23		1080 × 1920	23	Android 6.0 (Google ...	x86_64	2.5 GB	
	Pixel 2 API 23 - Marshmallow		1080 × 1920	23	Android 6.0 (Google ...	x86_64	8.0 GB	



Espera unos segundos y el dispositivo se mostrará en pantalla.

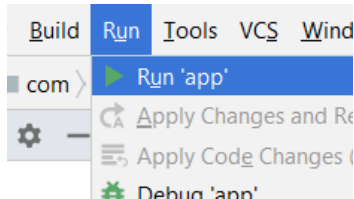
La primera vez que se ejecuta el dispositivo virtual puede tardarse un poco más, ten paciencia.

Puedes hacer clic en la opción **“GOT IT”** (para ocultar la pantalla de bienvenida)

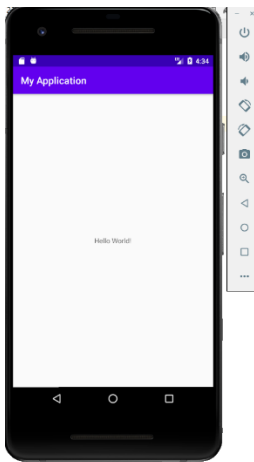
	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Ahora puedes ejecutar tu aplicación en este dispositivo virtual.

Puede usar las opciones Run->Run 'app' del menú principal de Android.



La aplicación se ejecutará en el emulador como se muestra en la pantalla:



No olvides cerrar tu emulador antes de cerrar apagar tu equipo y cerrar Android Studio.


Importante:

Cuando quieras ejecutar la aplicación directamente en tu dispositivo físico, puedes usar el cable USB para conectarlo a tu computador (Considera que tu computador tenga los drivers adecuados para que reconozca a tu dispositivo físico)

Cuando el dispositivo esté conectado puedes validar que Android Studio lo ha reconocido si se observa esto en la barra de herramientas:



Entonces cuando se ejecute, la aplicación se instalará y ejecutará en el dispositivo físico.

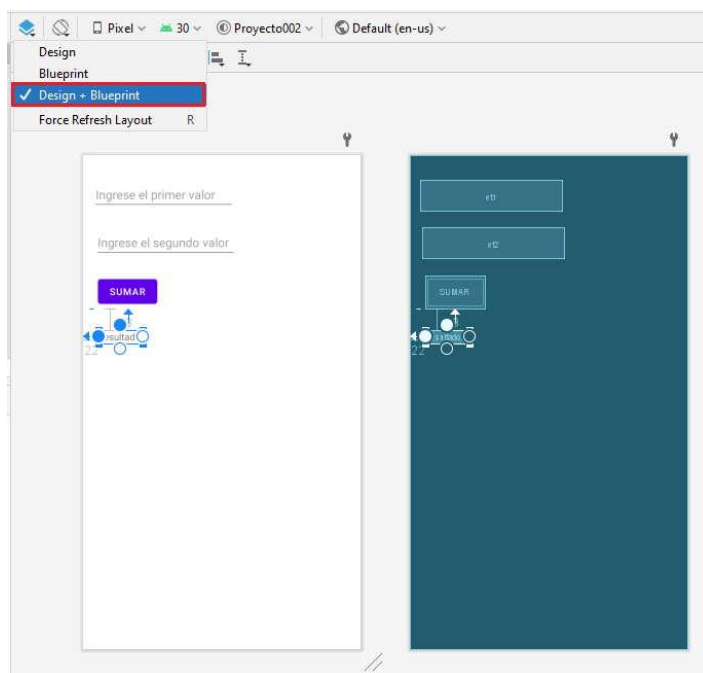
	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Primer ejemplo Capturar el clic de un botón

Problema:

Confeccionar un programa que permita la carga de dos números enteros en controles de tipo EditText (Number). Mostrar dentro de los mismos controles EditText mensajes que soliciten la carga de los valores. Disponer un Button para sumar los dos valores ingresados. Mostrar el resultado en un control de tipo TextView.


La interfaz visual debe quedar algo similar a esto:

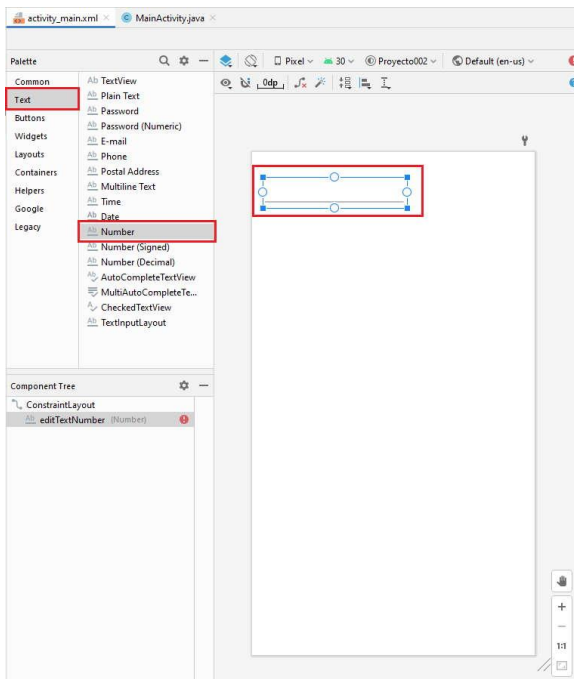


Recordar que si queremos ocultar o volver a mostrar el diseño "blueprint" tenemos tres íconos en la parte superior del diseñador.

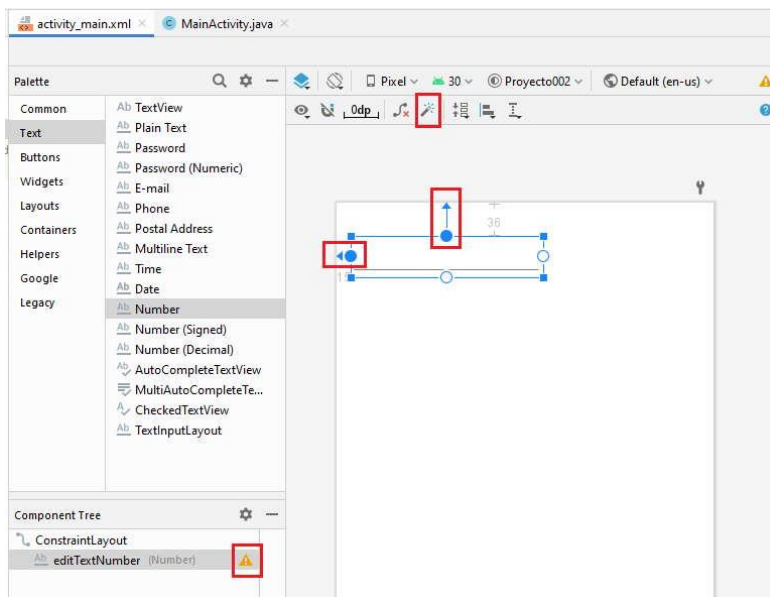
Crear un proyecto llamado: **primerappmovil**.


Veamos paso a paso como creamos la interfaz visual de nuestro programa. Primero borramos el TextView que aparece por defecto cuando se crea un proyecto con el Android Studio. Ahora desde la ventana "Palette" seleccionamos de la pestaña "Text" el control "Number" (es de la clase EditText) y lo arrastramos a la ventana de diseño de nuestra interfaz a la parte superior izquierda:

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

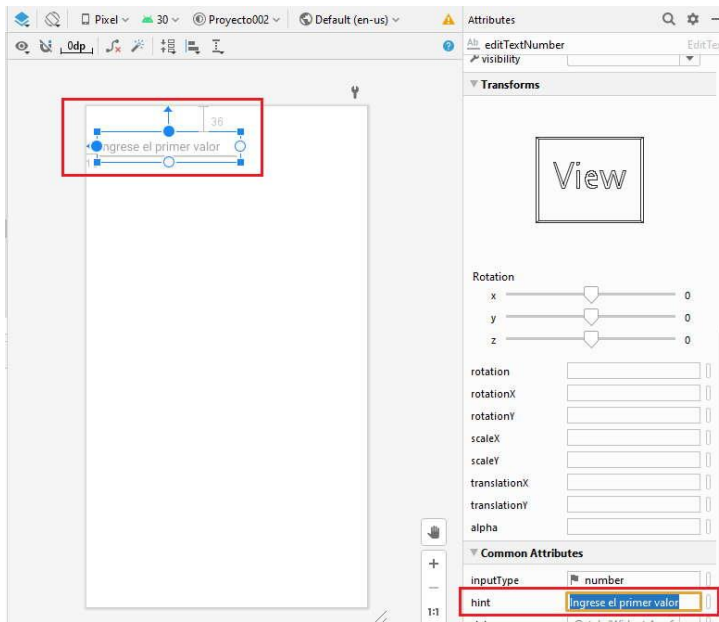


Como el control EditText se inserta en un contenedor ConstraintLayout, debemos indicar la posición relativa dentro del mismo, la forma más fácil es presionar el botón "Infer Constraints" para que se generen en forma automática (podemos hacerlo en forma manual presionando los círculos que aparecen en el EditText y desplazando con el mouse hasta los bordes

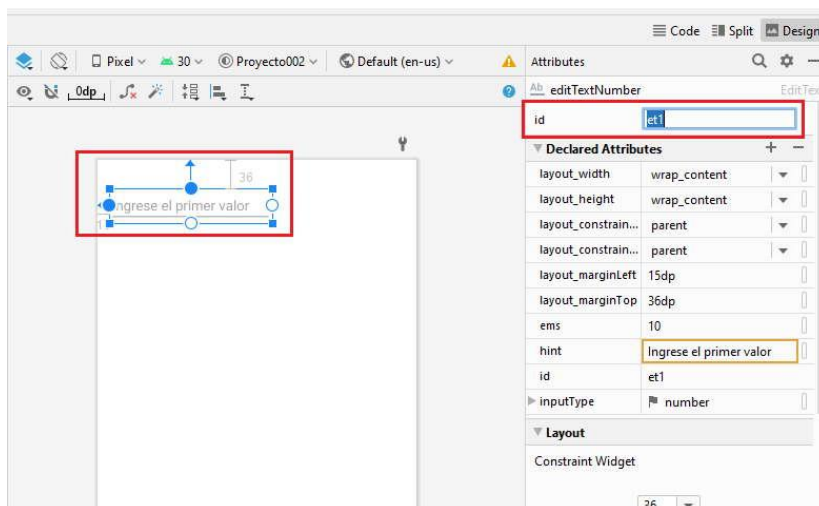


	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Ahora lo seleccionamos y en la ventana de propiedades (Properties) especificamos la propiedad hint, disponemos el texto "Ingrese el primer valor":




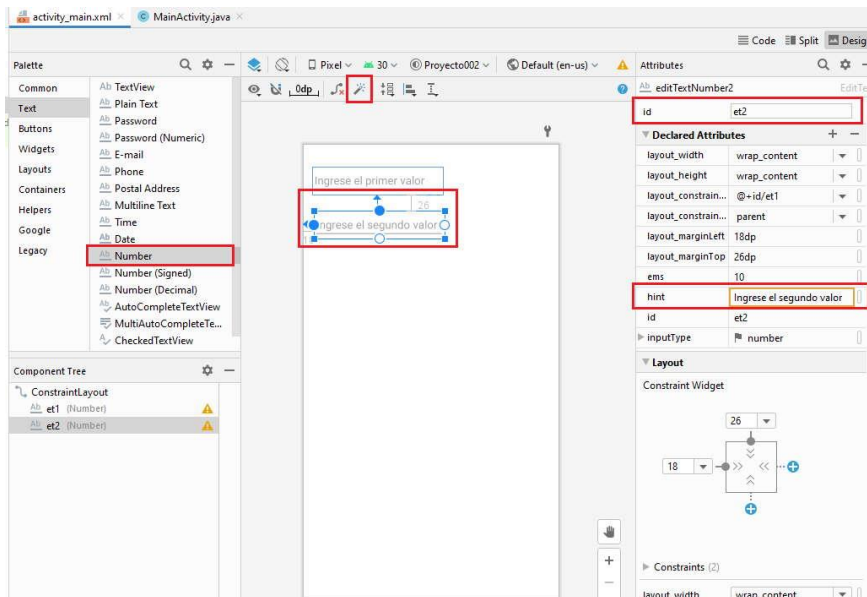
También vamos a especificar la propiedad "id", y le asignaremos el valor et1



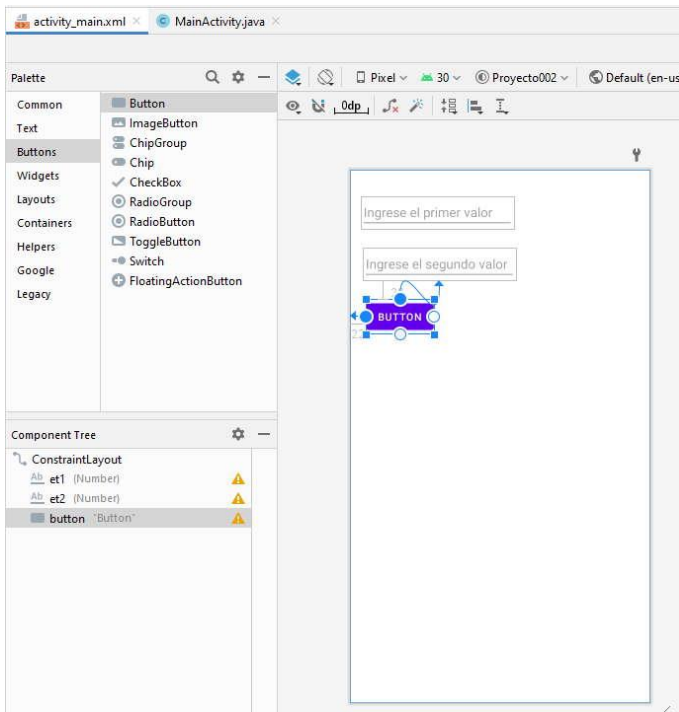
Entonces asignado como nombre a este objeto: et1 (recordemos que se trata de un objeto de la clase EditText), este nombre haremos referencia posteriormente desde el programa en Java.

Efectuamos los mismos pasos para crear el segundo EditText de tipo "Number" (iniciamos las propiedades respectivas) Definimos el id con el nombre et2 y la propiedad hint con el mensaje "Ingrese el segundo valor", el resultado visual debe ser algo semejante a esto:


	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

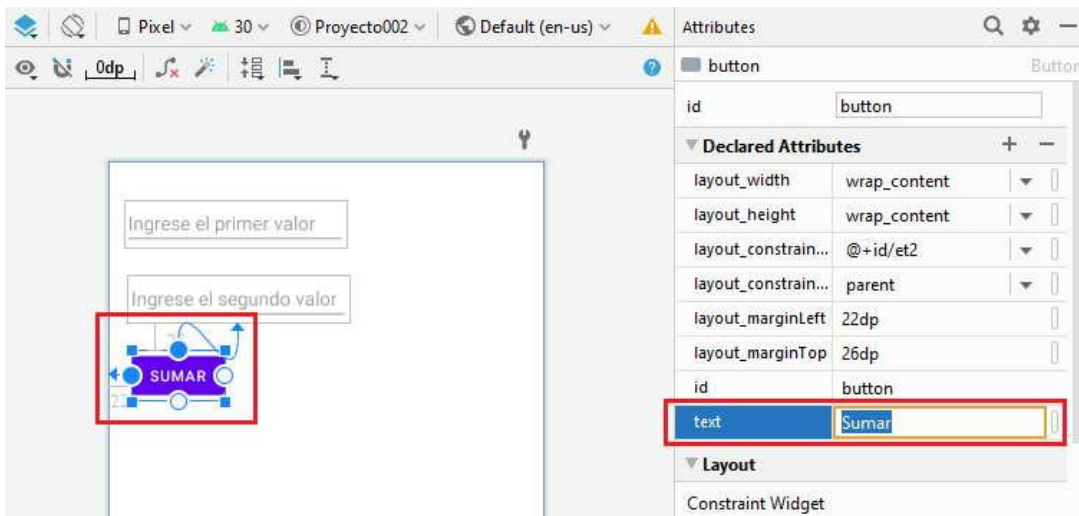


De la pestaña "Buttons" arrastramos un control de tipo "Button":

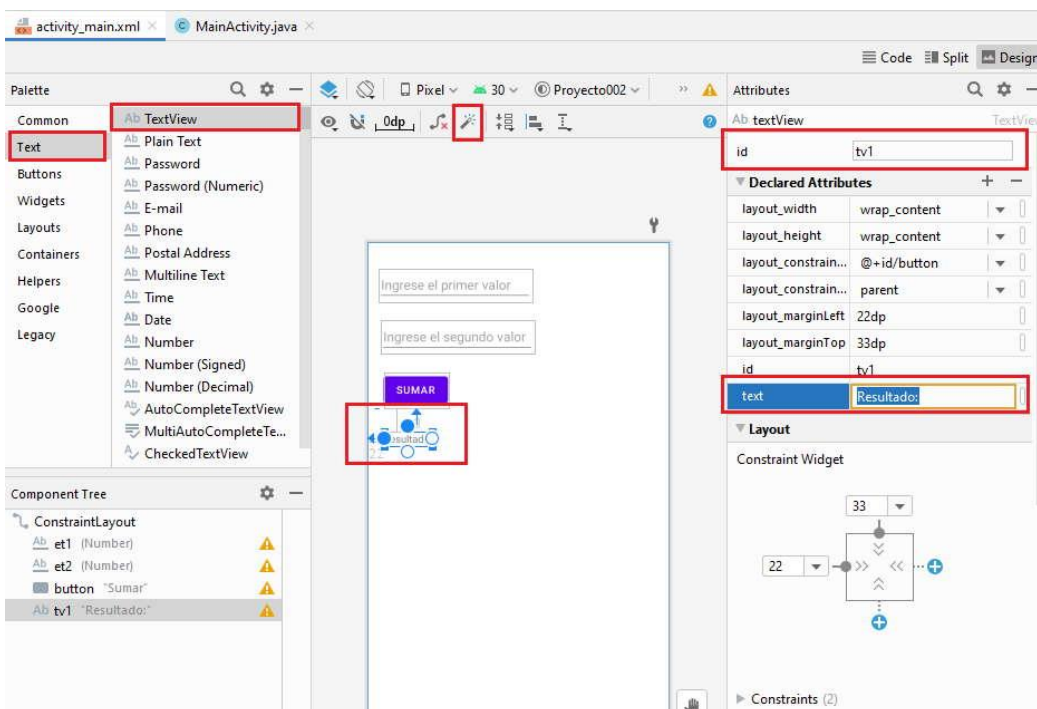



Iniciamos la propiedad text con el texto "Sumar" y la propiedad id la dejamos con el valor ya creado llamado "button":

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
			GUÍA DE LABORATORIO N° 1
	MATERIA DESARROLLO DE SOFTWARE PARA MÓVILES	PRÁCTICA INTRODUCCIÓN JAVA Y ANDROID	

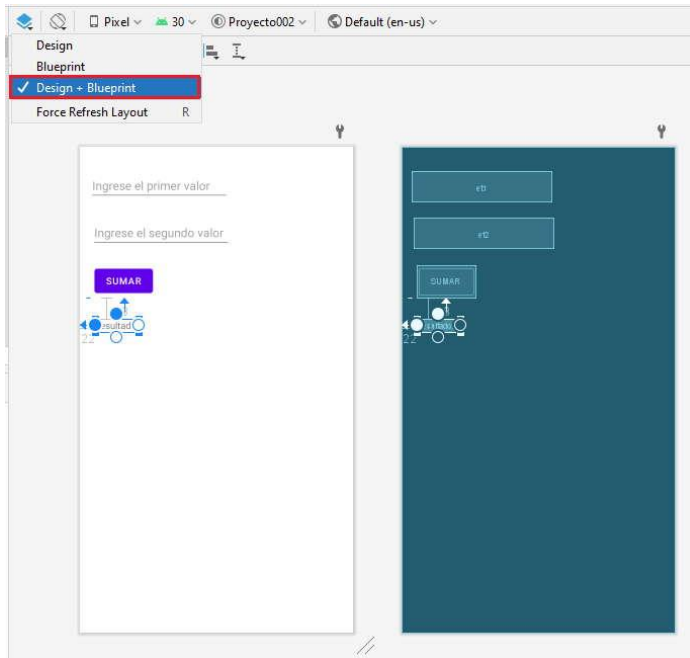


Para terminar con nuestra interfaz visual arrastramos una componente de tipo "TextView" de la pestaña "Text". Definimos la propiedad id con el valor "tv1" y la propiedad text con el texto "Resultado":



	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

La interfaz final debe ser semejante a esta:



Si en este momento ejecutamos la aplicación aparece la interfaz visual correctamente pero cuando presionemos el botón no mostrará la suma.

Hasta ahora hemos trabajado solo con el archivo xml (activity_main.xml) donde se definen los controles visuales de la ventana que estamos creando.


Abrimos el archivo **MainActivity.java**

```

2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14

```

La clase MainActivity hereda de la clase AppCompatActivity. La clase AppCompatActivity representa una ventana de Android y tiene todos los métodos necesarios para crear y mostrar los objetos que hemos dispuesto en el archivo xml.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

El código fuente de la clase MainActivity.java es:

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Como mínimo se debe sobrescribir el método onCreate heredado de la clase AppCompatActivity donde procedemos a llamar al método setContentView pasando como referencia un valor almacenado en una constante llamada activity_main contenida en una clase llamada layout que a su vez la contiene una **clase llamada R (veremos más adelante que el Android Studio se encarga de crear la clase R en forma automática y sirve como puente entre el archivo xml y nuestra clase MainActivity)**

Captura de eventos.

Ahora viene la parte donde definimos variables en java donde almacenamos las referencias a los objetos definidos en el archivo XML. Definimos tres variables, dos de tipo EditText y finalmente una de tipo TextView (estas dos clases se declaran en el paquete android.widget, es necesario importar dichas clases para poder definir las variables de dichas clases, la forma más fácil de importar las clases es una vez que definimos el objeto por ejemplo private EditText et1; veremos que aparece en rojo el nombre de la clase y nos invita el Android Studio a presionar las teclas "Alt" e "Intro" en forma simultánea. Luego el Android Studio codifica automáticamente la línea que importa la clase: import android.widget.EditText;):


```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private EditText et1;
    private EditText et2;
    private TextView tv1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Recordar que la forma más fácil de importar las clases EditText y TextView es escribir las tres líneas:

```
private EditText et1;
private EditText et2;
private TextView tv1;
```

y luego presionar las teclas "Alt" y "Enter" en cada nombre de clase que se debe importar

Esto hace que se escriban automáticamente los import:

```
import android.widget.EditText;
import android.widget.TextView;
```

Los nombres de los objetos en este caso coinciden con la propiedad id (no es obligatorio):

```
private EditText et1;
private EditText et2;
private TextView tv1;
```

Para la clase Button no es necesario definir un atributo.


En el método onCreate debemos enlazar estas variables con los objetos definidos en el archivo XML, esto se hace llamando al método findViewById:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    et1=findViewById(R.id.et1);
    et2=findViewById(R.id.et2);
    tv1=findViewById(R.id.tv1);
}
```

Al método **findViewById** debemos pasar la constante **creada en la clase R (recordemos que se crea automáticamente esta clase)** el nombre de la constante si debe ser igual con el nombre de la propiedad del objeto creado en el archivo XML.

Ya tenemos almacenados en las variables las referencias a los tres objetos que se crean al llamar al **método: setContentView(R.layout.main);** .

Ahora planteamos el método que se ejecutará cuando se presione el botón (el método debe recibir como parámetro un objeto de la clase View) En nuestro ejemplo lo llamé sumar:

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private EditText et1;
    private EditText et2;
    private TextView tv1;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

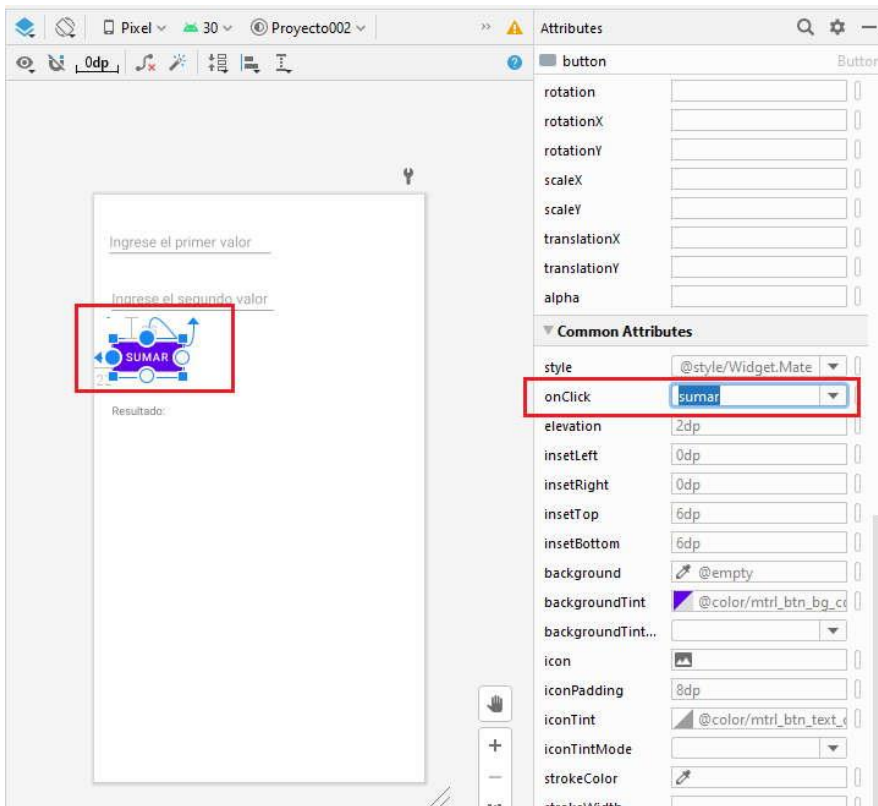
        et1=findViewById(R.id.et1);
        et2=findViewById(R.id.et2);
        tv1=findViewById(R.id.tv1);
    }

    //Este método se ejecutará cuando se presione el botón
    public void sumar(View view) {
    }

}
```

Debemos importar la clase View (presionamos las teclas "Alt" y luego "Enter" en forma simultanea)
 Ahora debemos ir al archivo XML (vista de diseño) e inicializar la propiedad onClick del objeto button con el nombre del método que acabamos de crear (este paso es fundamental para que el objeto de la clase Button pueda llamar al método sumar que acabamos de crear):

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	



Finalmente implementaremos la lógica para sumar los dos valores ingresados en los controles EditText:

```
public void sumar(View view) {


    String valor1=et1.getText().toString();
    String valor2=et2.getText().toString();

    int nro1=Integer.parseInt(valor1);
    int nro2=Integer.parseInt(valor2);

    int suma=nro1+nro2;

    String resu=String.valueOf(suma);
    tv1.setText(resu);
}
```

Extraemos el texto de los dos controles de tipo EditText y los almacenamos en dos variables locales de tipo String. Convertimos los String a tipo entero, los sumamos y el resultado lo enviamos al TextView donde se muestra la suma (previo a convertir la suma a String)

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

La clase completa queda entonces como:

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private EditText et1;

    private EditText et2;

    private TextView tv1;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        et1=findViewById(R.id.et1);

        et2=findViewById(R.id.et2);

        tv1=findViewById(R.id.tv1);

    }

    //Este método se ejecutará cuando se presione el botón

    public void sumar(View view) {

        String valor1=et1.getText().toString();

        String valor2=et2.getText().toString();

        int nro1=Integer.parseInt(valor1);

        int nro2=Integer.parseInt(valor2);


        int suma=nro1+nro2;

        String resu=String.valueOf(suma);

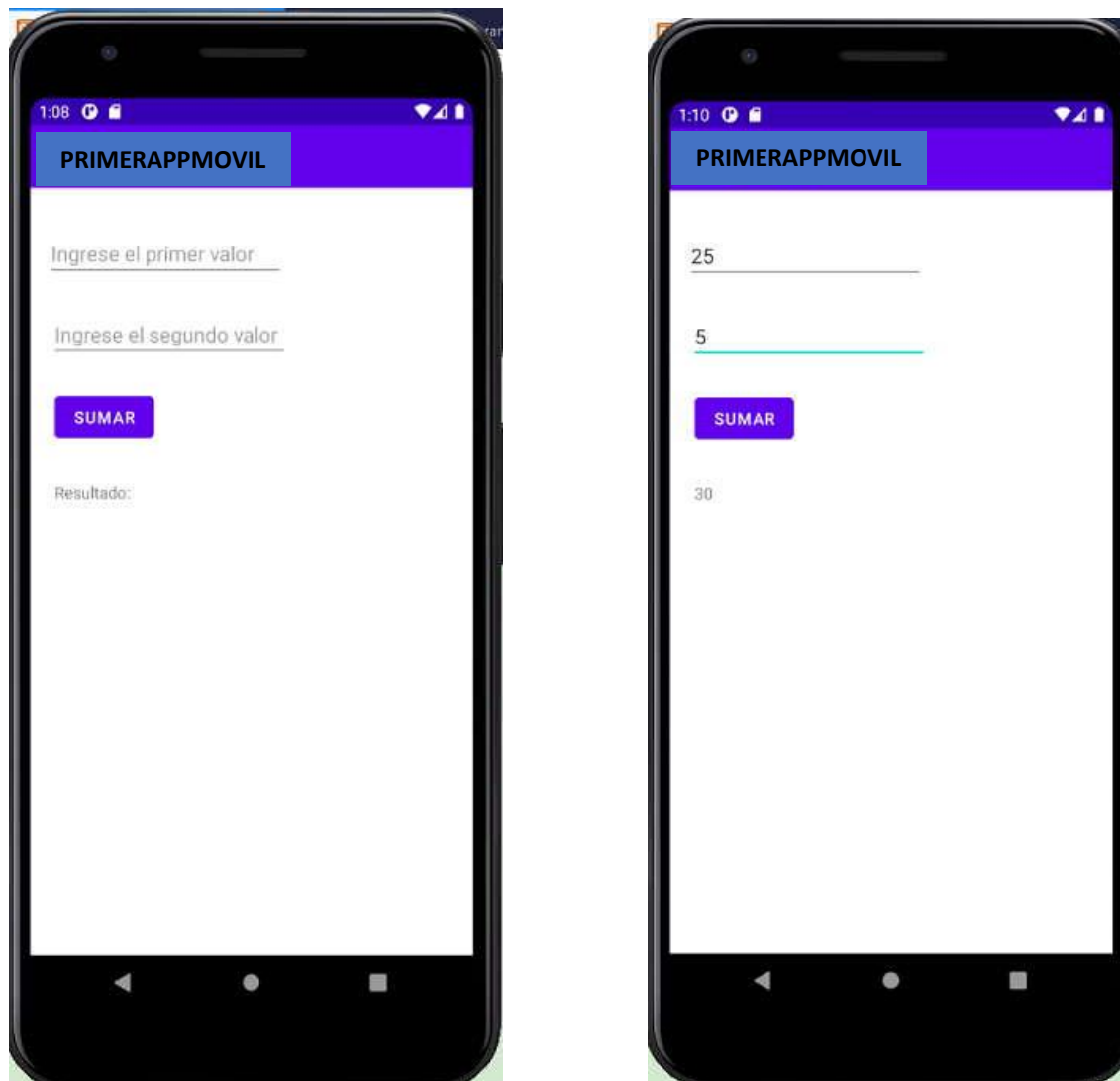
        tv1.setText(resu);

    }


}
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

Si ejecutamos nuestro programa podemos ver ahora que los controles EditText muestran los mensajes "Ingrese el primer valor" e "Ingrese el segundo valor" (la propiedad hint de los EditText muestran un mensaje que se borra automáticamente cuando el operador carga los enteros):



Luego de cargar dos valores al presionar el botón aparece en el TextView el resultado de la suma de los dos EditText.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 01-2022
	MATERIA	DESARROLLO DE SOFTWARE PARA MÓVILES	GUIA DE LABORATORIO N° 1
	PRÁCTICA	INTRODUCCIÓN JAVA Y ANDROID	

IV. DISCUSIÓN DE RESULTADOS

1. Crear un programa en consola que me permita saber si dos números son divisibles entre sí, para saber si un número es divisible por otro se tiene que obtener el modulo y si este es cero entonces este número es divisible por el otro.
2. Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.
3. Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares.
4. Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano. Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.

Nota: Analice cada ejercicio y de ser necesario considere utilizar clases para su desarrollo.

1. Crear un emulador con las siguientes características:
 - a. Hardware: pixel 3, tamaño de pantalla: mayor o igual a 5", memoria RAM: 500 MB, memoria interna: 7500 MB
 - b. Sistema operativo: Android Lollipop.
2. Crea otro proyecto llamado: Segunda App
 - a. SDK mínimo: API 20
 - b. Plantilla: **Empty Activity**
 - c. Cambia el mensaje "**Hello World**" por "<Tu nombre completo>"
 - d. Ejecuta la aplicación en el emulador creado en el paso 1
3. Crea otro proyecto llamado: OperacionesApp
 - a. Debe tener todas las operaciones básicas en una misma pantalla (+, *, -, /)
 - b. Deben de ser los mismo Text, solo debe tener un Button para cada operación (Sumar, Restar, Multiplicar, División)

V. BIBLOGRAFÍA

- Aprendiendo Java 2 en 21 Días Lemay, Laura
- Cómo Programar en Java Deitel, Harvey M.
- Desarrollo de aplicaciones para Android 2016 / Android application development, Anaya Multimedia
- El gran libro de Android: 7ª Edición (Jesús Tomás Gironés)