

## Contenido

TEORÍA.....	1
Videotutoriales de programación.....	2
0 - Aprende a programar.....	2
1 - IntelliJ, Debug, Librerías y Maven.....	3
2- Java básico. Variables, condicionales, bucles y arrays.....	4
3- Clase y objeto.....	5
4- Static como atributo.....	5
5- Colecciones.....	5
6- Herencia.....	6
7- Interfaces.....	6
8- Polimorfismo.....	6
<b>ACTIVIDADES REFUERZO.....</b>	<b>8</b>
Actividades grupo 1. IntelliJ, librerías y Maven.....	8
Actividades grupo 2. Java básico. Variables, condicionales, bucles y arrays.....	14
Actividades grupo 3. Clase y objetos.....	21
Actividades grupo 4. Static.....	23
Actividades grupo 5. Colecciones.....	26
Actividades grupo 6. Herencia.....	30
Actividades grupo 7. Interfaces.....	32
Actividades grupo 8. Polimorfismo.....	34
<b>ANEXO.....</b>	<b>35</b>

## OBJETIVO

Proporcionar una guía al alumnos sobre qué contenido ver sobre Java y que ejercicios puede realizar para practicar su uso. Se proporciona un base teórica que tienen su correspondencia con actividades de refuerzo, de tal manera que el alumno deberá primero informarse sobre la teoría para luego aplicarla realizando los ejercicios correspondientes

## TEORÍA

### Videotutoriales de programación

[Curso de LÓGICA DE PROGRAMACIÓN Desde Cero](#)

[JAVA Desde Cero: Primeros Pasos en una hora](#)

[Curso Java. ¿Se pide Java en las ofertas de empleo? ¿Merece la pena estudiar Java en 2022?. Vídeo 0](#)

Videos [0-56]

### 0 - Aprende a programar

Estructuras básicas y algoritmos:

<https://www.udemy.com/course/algoritmos-desde-cero/>

<https://editorial.uaa.mx/docs/algoritmos.pdf>





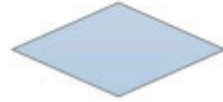
Diagramas de flujo:

<https://www.diagramadeflujo.net/receta-de-cocina/>

[http://lsi.vc.ehu.es/asignaturas/Inf\\_Bas/2005-06/comun/04%20Control%20flujo.pdf](http://lsi.vc.ehu.es/asignaturas/Inf_Bas/2005-06/comun/04%20Control%20flujo.pdf)

[http://lsi.vc.ehu.es/asignaturas/Inf\\_Bas/2005-06/comun/05%20Bucles.pdf](http://lsi.vc.ehu.es/asignaturas/Inf_Bas/2005-06/comun/05%20Bucles.pdf)

[https://www.areatecnologia.com/diagramas-de-flujo.htm#google\\_vignette](https://www.areatecnologia.com/diagramas-de-flujo.htm#google_vignette)

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Linea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

## 1 - IntelliJ, Debug, Librerías y Maven

¿Conoces IntelliJ sabes lo que es un jar/librería y Maven? (N)

- IntelliJ
  - <https://www.aluracursos.com/blog/intellij-idea-para-principiantes>
  - [❤️ Introducción IDE IntelliJ IDEA Apps JAVA programación Profesor Ángel Aguinaga CIPSA Bilbao](#)
- Establecer la JDK
  - <https://www.jetbrains.com/help/idea/sdk.html#access-external-documentation>
- Atajos IntelliJ
  - [10 atajos de teclado IMPRESCINDIBLES para NAVEGACIÓN](#)
  - <https://cheatography.com/cangueler/cheat-sheets/intellij-keyboard-shortcuts/>

- <https://adictosaltrabajo.com/2016/05/18/conviertete-en-un-dios-de-la-productividad-en-intellij-idea/>
- <https://www.jetbrains.com/help/idea/mastering-keyboard-shortcuts.html>

Ir a clase: ctrl + N

Ir a fichero: ctrl + Mayusc +N

Formatear código: ctrl +alt + L

Organizar Imports: ctrl + alt + o

- Debug:
  - [DEPURACIÓN \(DEBUG\) en JAVA con INTELLIJ IDEA - Tutorial Completo Fácil](#)
  - <https://www.jetbrains.com/help/idea/debugging-your-first-java-application.html#analyzing-state>
- Jar/Librería:
  - [Curso Java. Despliegue Aplicaciones. Archivos JAR I. Vídeo 137](#)
  - Export:  
[https://www.jetbrains.com/help/idea/compiling-applications.html#package\\_into\\_jar](https://www.jetbrains.com/help/idea/compiling-applications.html#package_into_jar)
  - Import:  
<https://www.jetbrains.com/help/idea/working-with-module-dependencies.html>
- Maven:
  - [Curso de Maven - 1 Introducción](#)
  - <https://www.arquitecturajava.com/que-es-maven/>

## 2- Java básico. Variables, condicionales, bucles y arrays

<https://www.w3schools.com/java/default.asp>

## 3- Clase y objeto

[https://www.w3schools.com/java/java\\_classes.asp](https://www.w3schools.com/java/java_classes.asp)

<https://www.programarya.com/Cursos/Java/Objetos-y-Clases>

## 4- Static como atributo

<https://refactorizando.com/directiva-static-java/>

[https://www.youtube.com/watch?v=29TnNxNzvZo&ab\\_channel=CharlyCimino](https://www.youtube.com/watch?v=29TnNxNzvZo&ab_channel=CharlyCimino)

## 5- Colecciones

<https://www.w3schools.blog/list-set-map-java>

Listas:

[https://www.youtube.com/watch?v=D9R2tn65v-4&list=PLTt\\_5WPJqE-ID0Ze4VgZBum\\_JetdRByhL&index=1&ab\\_channel=PedroCamacho](https://www.youtube.com/watch?v=D9R2tn65v-4&list=PLTt_5WPJqE-ID0Ze4VgZBum_JetdRByhL&index=1&ab_channel=PedroCamacho)

ArrayList:

[https://www.youtube.com/watch?v=0My1c9f95iU&list=PLTt\\_5WPJqE-ID0Ze4VgZBum\\_JetdRByhL&index=2&ab\\_channel=PedroCamacho](https://www.youtube.com/watch?v=0My1c9f95iU&list=PLTt_5WPJqE-ID0Ze4VgZBum_JetdRByhL&index=2&ab_channel=PedroCamacho)

Conjuntos (Set):

[https://www.youtube.com/watch?v=nyU6\\_GQykKY&list=PLTt\\_5WPJqE-ID0Ze4VgZBum\\_JetdRByhL&index=6&ab\\_channel=PedroCamacho](https://www.youtube.com/watch?v=nyU6_GQykKY&list=PLTt_5WPJqE-ID0Ze4VgZBum_JetdRByhL&index=6&ab_channel=PedroCamacho)

Mapa (Map):

[https://www.youtube.com/watch?v=CJNRdDEpHwQ&list=PLTt\\_5WPJqE-ID0Ze4VgZBum\\_JetdRByhL&index=8&ab\\_channel=PedroCamacho](https://www.youtube.com/watch?v=CJNRdDEpHwQ&list=PLTt_5WPJqE-ID0Ze4VgZBum_JetdRByhL&index=8&ab_channel=PedroCamacho)

## 6- Herencia

[https://www.w3schools.com/java/java\\_inheritance.asp](https://www.w3schools.com/java/java_inheritance.asp)

[https://www.youtube.com/watch?v=l4o7fvSQvBA&ab\\_channel=TodoCode](https://www.youtube.com/watch?v=l4o7fvSQvBA&ab_channel=TodoCode)

## 7- Interfaces

Teoría: [https://www.youtube.com/watch?v=hfwtzjOhvKk&ab\\_channel=TodoCode](https://www.youtube.com/watch?v=hfwtzjOhvKk&ab_channel=TodoCode)

Práctica: [https://www.youtube.com/watch?v=O3hKxRLkLVU&ab\\_channel=TodoCode](https://www.youtube.com/watch?v=O3hKxRLkLVU&ab_channel=TodoCode)

<https://www.freecodecamp.org/espanol/news/interfaces-java-explicadas-con-ejemplos/>

[https://www.w3schools.com/java/java\\_interface.asp](https://www.w3schools.com/java/java_interface.asp)

## 8- Polimorfismo

<https://www.arquitecturajava.com/java-polimorfismo-herencia-y-simplicidad/>



## ACTIVIDADES REFUERZO

### Actividades grupo 1. IntelliJ, librerías y Maven

#### Actividad 1.1

Descarga IntelliJ:

- Crea un proyecto Java (no Maven) llamado “calculadora”.
- Dentro de él crea un paquete llamado com.fp
- Dentro de ese paquete crea una clase Calculadora que tendrá los métodos:
  - sumar, dividir, multiplicar y restar
- Crea un main y utiliza el debugger de IntelliJ para depurar el código, ver las variables y comprobar si funciona bien.

Guía: teoría IntelliJ, librerías, Maven y Debug.

#### Actividad 1.2

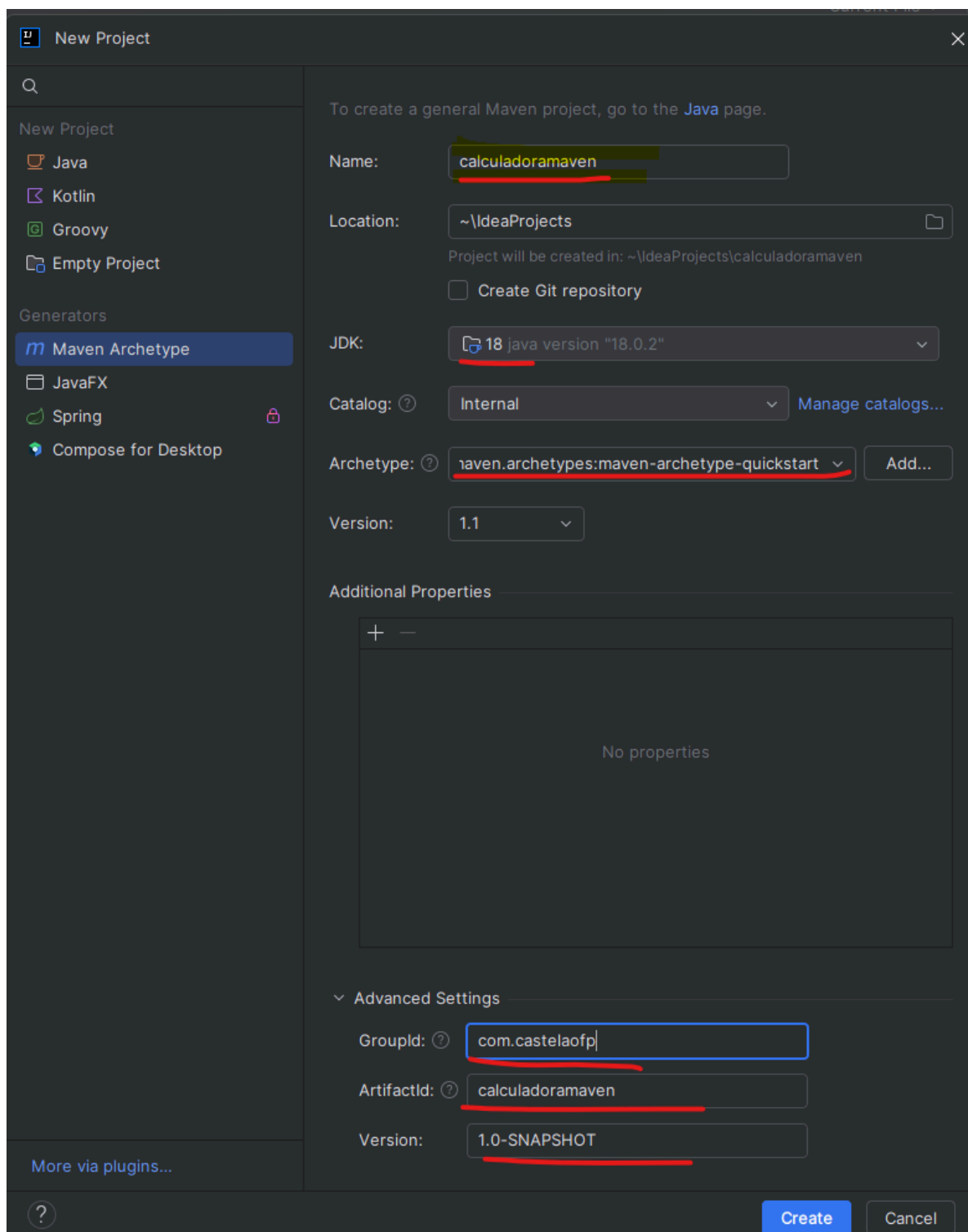
- En el proyecto anterior de la actividad 1.1, crear un jar llamado calculadora.jar
- Crea un nuevo proyecto Java (no Maven) llamado “matemáticas”.
- Dentro de él crea un paquete llamado com.fp
- Importa el jar de la actividad 1.1
- Crear una clase en el paquete creado anteriormente y utiliza la clase Matemáticas

Guía: teoría IntelliJ, librerías, Maven y Debug.

#### Actividad 1.3

Crea un proyecto Maven en IntelliJ con estos datos:





**Group id:** com.fp

**Artifact id:** calculadoramaven

- Crea un paquete com.fp
- Copia la clase Calculadora del anterior proyecto dentro de dicho paquete

#### **Actividad 1.4**

Crea un proyecto Maven en IntelliJ con estos datos:

**Group id:** com.fp

**Artifact id:** matematicasmaven

- Crea un paquete com.fp
- Importa como dependencia el proyecto del anterior ejercicio com.fp:calculadoramaven
- Utiliza la clase calculadora dentro de tu proyecto matemáticas

#### **Actividad 1.5**

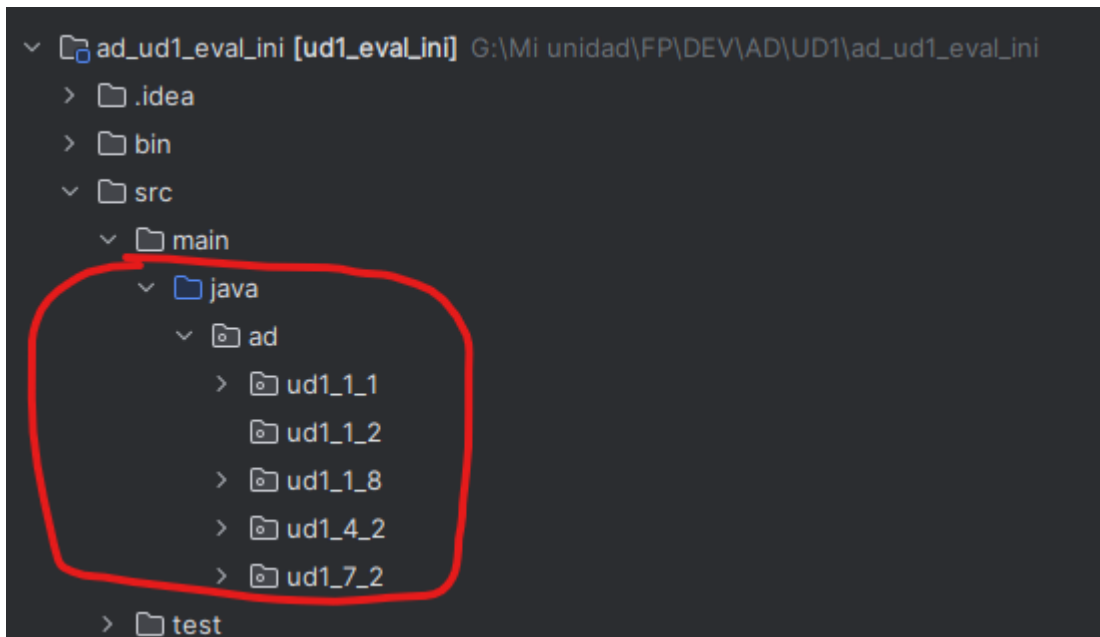
Crea un proyecto Maven en IntelliJ con estos datos:

**Group id:** ad

**Artifact id:** ud1\_eval\_ini

Será el proyecto dónde ubicamos las siguientes actividades que realicemos a lo largo de esta evaluación inicial.

Cuando tengas que crear código en las siguientes actividades, hazlo dentro de su subpaquete con el código de la actividad a la que pertenece.



### Actividad 1.6

Crea y ejecuta un “Hola Mundo” en IntelliJ .

Dentro del proyecto anterior:

**Group id:** ad

**Artifact id:** ud1\_eval\_ini

Tienes que:

- Crear un paquete siguiendo la nomenclatura indicada en la actividad anterior ad.ud1\_1\_6. Recuerda hacer esto mismo para los ejercicios venideros.
- Crear la clase HolaMundo que escriba por pantalla la frase “Hola Mundo”

### Actividad 1.7:

Crea un programa que escriba por pantalla el valor enésimo de la secuencia de Fibonacci. El valor n se recibirá como argumento por pantalla. Hazlo en su paquete correspondiente ad.ud1\_1\_7

Ejemplo:

Introduce el número de la secuencia de Fibonacci que deseas ver: 6

El número 6 de la secuencia de Fibonacci es: 8

### Actividades complementarias

- a. Conocer los atajos de IntelliJ

### Actividad 1.8

Dentro del proyecto anterior:

**Group id:** ad

**Artifact id:** ud1\_eval\_ini

Queremos comprobar si un determinado texto, es numérico o no. Como somos un poco vagos, queremos aprovechar el código de alguien que ya haya hecho una función semejante. Buscando en google vemos que existe una clase StringUtils del proyecto commons-lang3 que ya contiene lo que buscamos:

```
public class App {  
    public static void main(String[] args) {  
        boolean numeric = StringUtils.isNumeric("AAAA");  
        System.out.println(numeric);  
    }  
}
```

Vamos a aprender a añadir una dependencia de terceros o jar, para eso debemos seguir los siguientes pasos:

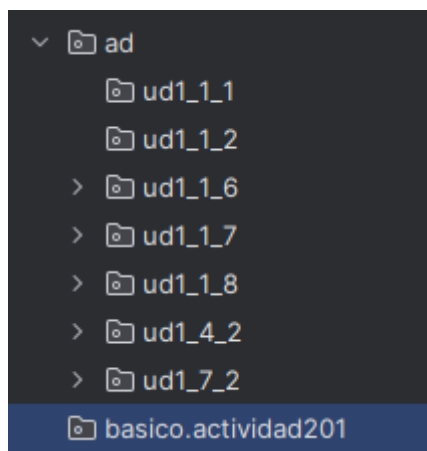
- Crea el paquete: ad.ud1\_1\_8 en tu proyecto
- Importa la dependencia Maven en el de commons-lang3, que nos provee Maven de su repositorio central: <https://mvnrepository.com/>
- Acude a esta página y busca la dependencia indicada en el anterior paso

- Debería redirigir aquí:  
<https://mvnrepository.com/artifact/org.apache.commons/commons-lang3/3.13.0>
- Añade esa dependencia a tu POM, de manera semejante a como ya has hecho anteriormente
- Crea una clase App en tu paquete que realice la comprobación de si un texto es numérico o no

## Actividades grupo 2. Java básico. Variables, condicionales, bucles y arrays

### Actividad 2.0.1 Tipos Básicos

Crea un paquete llamado basico.actividad201



Escribe los siguientes tipos de datos en Java

char, String, int, double, BigDecimal, boolean

Intenta hacer conversiones entre ellos e imprime sus valores por consola, concretamente las siguientes:

String -> int

int->String

int -> double

double ->int

boolean-> String

String -> boolean

### Actividad 2.0.2 Operadores

Crea un paquete llamado basico.actividad202

Escriba un programa Java para imprimir la suma (suma), multiplicar, restar, dividir y el resto de dos números.

Datos de prueba:

Ingrese el primer número: 125

Ingrese el segundo número: 24

Salida esperada :

$125 + 24 = 149$

$125 - 24 = 101$

$125 \times 24 = 3000$

$125/24 = 5$

$125 \bmod 24 = 5$

### **Actividad 2.0.3 Condicionales**

Crea un paquete llamado `basico.actividad203`

Escriba un programa Java para imprimir reciba 2 enteros e indique lo siguiente:

- a) Cual de ellos es el mayor
- b) Para cada uno de ellos, si es par o impar

### **Actividad 2.1 OrdenarMayorMenor**

Crea un paquete llamado `basico.actividad21`

Escribe un programa en Java que reciba 3 números enteros.

Debe mostrar por la salida los elementos ordenados de mayor a menor.

Ejemplo:

Introduzca 3 elementos:

A=3

B=5

C=2

Los elementos ordenados de mayor a menor son: B,A,C

Hint:

Una manera de hacerlo es anidando if else, dónde vaya comparando 2 números, así cubriría las 6 posibilidades existentes:

A,B,C

A,C,B

B,A,C

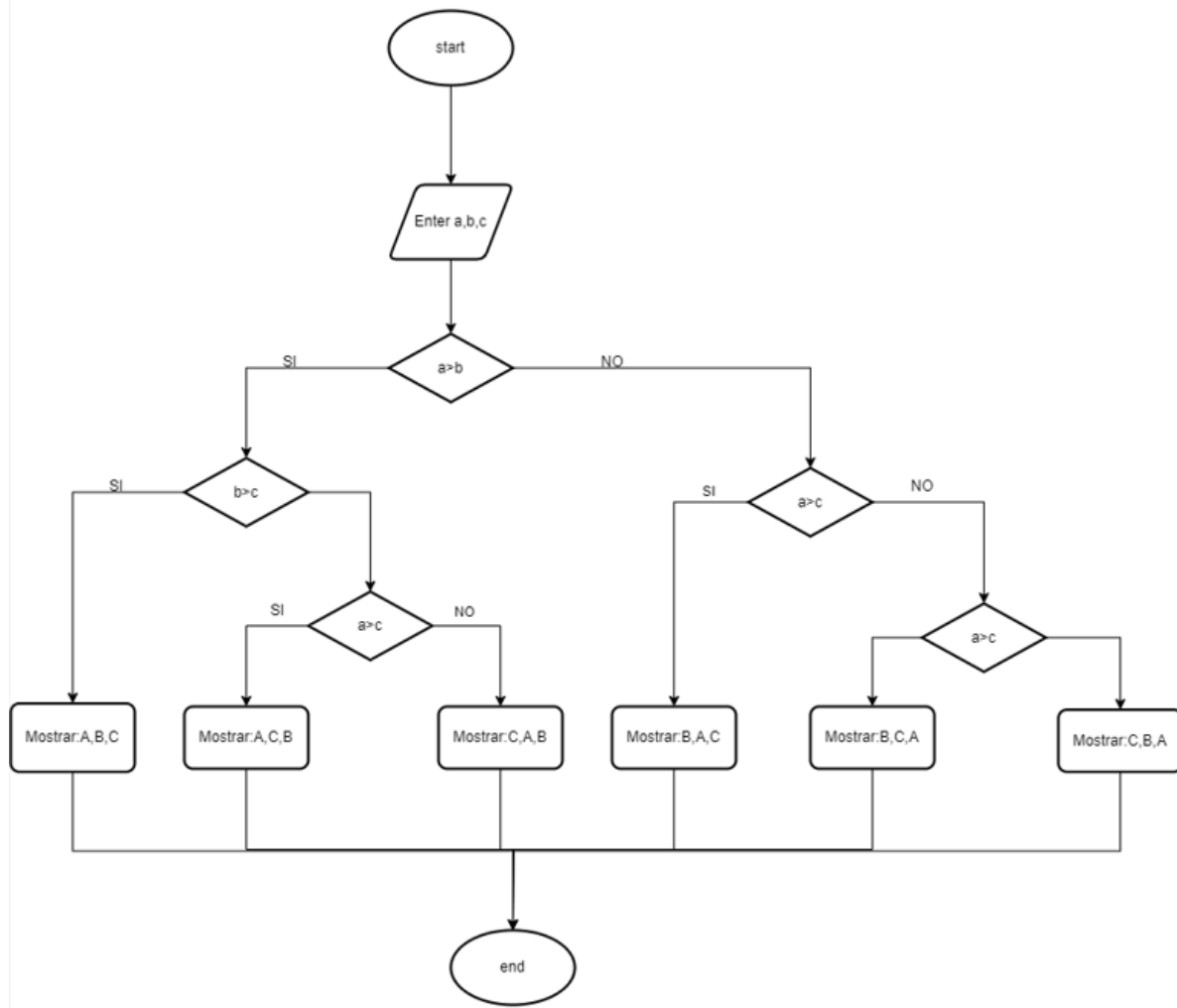
B,C,A

C,A,B

C,B,A



### Actividad 2.1 Ordenar mayor menor



### Actividad 2.2 Calificaciones

Crea un paquete llamado basico.actividad22

Escribir un programa que reciba 1 número entero que representa una nota.

Debe mostrar por la salida la calificación correspondiente.

Insuficiente (<5), Suficiente (5), Bien (6), Notable (7 o 8), SobreSaliente (9,10)

### Actividad 2.3. Sumar enteros

Crea un paquete llamado `basico.actividad23`

Escribe un programa en Java que reciba un número entero.

Debe mostrar por la salida la suma de todos los números anteriores al número recibido.

Datos de prueba:

Ingrese un número: 4

Salida: 6

Ejemplos aclaratorios:

1=> 0

2=> 1

3=> 1+2=3

4 => 1+2+3 = 6

5 => 1+2+3+4=10

### Actividad 2.4. Tabla multiplicar

Crea un paquete llamado `basico.actividad24`

Dentro escribe un programa Java que tome un número como entrada e imprima su tabla de multiplicar hasta 10.

Datos de prueba:

Ingrese un número: 8

Salida esperada :

8 x 1 = 8

8 x 2 = 16

8 x 3 = 24

...

8 x 10 = 80

### **Actividad 2.5 Promedio**

Crea un paquete llamado `basico.actividad25`

Dentro escribe un programa que realice el cálculo de la media aritmética de 3 números enteros recibidos como entrada.

### **Actividad 2.6 Número Primo**

Crea un paquete llamado `basico.actividad26`

Realiza una función que calcule si un número es primo o no (es aquel que solo es divisible por 1 o por si mismo)

Ahora haz una función que sume todos los números que son primos hasta 100

### **Actividad 2.7 Cuenta números**

Crea un paquete llamado `basico.actividad27`

Realizar un algoritmo que pida números (se pedirá por teclado la cantidad de números a introducir). El programa debe informar de cuántos números introducidos son mayores que 0, menores que 0 e iguales a 0.

### **Actividad 2.8 ¿Solo vocales?**

Crea un paquete llamado `basico.actividad28`

En este ejercicio debemos pedir al cliente que introduzca una cadena de caracteres y posteriormente debemos analizarla para ver si se han incluido sólo vocales o hay algún otro tipo de carácter.

### **Actividad 2.9 Distribuir Números**

Haz un programa que reciba como argumento con Scanner un número. El programa creará 2 arrays, uno para los elementos pares y otro para los impares hasta el número ofrecido como argumento. Finalmente recorrerá ambos arrays para ver su contenido

### **Actividad 2.10 Voltear matriz**

Crea un paquete llamado `basico.actividad210`

Escribe un programa para rotar los argumentos de una matriz. La matriz inicial la definirás hardcodeda en el programa

Datos de prueba: {20, 30, 40}

Salida esperada: {40, 30, 20}

### **Actividad 2.11 Multiplicar matriz**

Crea un paquete llamado `basico.actividad211`

Escriba un programa Java para multiplicar los elementos correspondientes de dos matrices de enteros. La matriz serán siempre de una dimensión y tendrán 4 elementos

Salida de muestra:

Matriz1: [1, 3, -5, 4]

Matriz2: [1, 4, -5, -2]

Resultado: 1 12 25 -8

## Actividades grupo 3. Clase y objetos

### Actividad 3.1

Crea un paquete llamado `clases.actividad31`

Crear una clase círculo que represente esta figura geométrica. Crea métodos para calcular su perímetro y área

### Actividad 3.2

Crea un paquete llamado `clases.actividad32`

Crea una clase llamada Cuenta que tendrá los siguientes atributos: titular y cantidad (puede tener decimales).

El titular será obligatorio y la cantidad es opcional. Crea dos constructores que cumpla lo anterior.

Crea sus métodos `get`, `set` y `toString`.

Tendrá dos métodos especiales:

- `ingresar(double cantidad)`: se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
- `retirar(double cantidad)`: se retira una cantidad a la cuenta, si restando la cantidad actual a la que nos pasan es negativa, la cantidad de la cuenta pasa a ser 0.

Crea una clase `App`, siempre es mejor tener la ejecución en una clase independiente, dónde crees una cuenta sobre la que realices ingresos y retiros de dinero

### Actividad 3.3

En la clase anterior, queremos crear un nuevo método para transferir dinero entre cuentas.

```
public void transferencia(double cantidad, Cuenta cuentaDestino) {
```

Implementa este método para pasar dinero de la cuenta origen a la cuenta destino.

Crea 2 cuentas, inicialas con un saldo y haz una transferencia de una a otra.

### Actividad 3.4

Crea un paquete llamado `basico.actividad34`

Se te ha solicitado completar la clase NaveEspacial que simula el movimiento de una nave espacial en un plano bidimensional.

La clase tendrá métodos para moverse a izquierda, derecha, arriba y abajo. Ten en cuenta que la nave espacial no pueda salirse de los límites definidos por los valores xMax, xMin, yMax y yMin.

Si al intentar movernos superamos el límite de la coordenada correspondiente, entonces no nos moveremos y devolveremos false en los métodos.

Si al intentar movernos podemos hacerlo sin problema, entonces avanzaremos o retrocederemos en el eje correspondiente y devolveremos true.

La clase NaveEspacial tiene los siguientes atributos:

- x: posición horizontal actual de la nave.
- y: posición vertical actual de la nave.
- xMax: límite máximo de la posición horizontal.
- xMin: límite mínimo de la posición horizontal.
- yMin: límite mínimo de la posición vertical.
- yMax: límite máximo de la posición vertical.

Crea un App dónde crees una nave espacial con los atributos anteriores y pruebes los diversos casos posibles de movimiento.

## Actividades grupo 4. Static

Resumen:

static como atributo es un valor compartido por todas las instancias de un objeto

### Actividad 4.2

Ejercicio: Tienda de Productos

#### Descripción:

Crea un paquete llamado `staticproperty.actividad42`

En este ejercicio, simularás una tienda de productos electrónicos. Crearás una clase llamada `Producto` que represente un producto en la tienda. Cada producto tendrá un nombre y un importe. Además, crearás una clase llamada `Tienda` que tendrá un contador estático para realizar un seguimiento del total de ventas en la tienda y una variable para almacenar el importe total de las ventas realizadas.

#### Instrucciones:

Crea una clase llamada `Producto` con variables de instancia para el nombre y el precio del producto,

Define un constructor para la clase `Producto` que tome el nombre y el precio del producto como parámetros.

Crea una clase llamada `Tienda` con dos variables estática para realizar el seguimiento de:

- número de ventas
- importe total de las ventas realizadas

Define un método estático llamado `realizarVenta(Producto p)` en la clase `Tienda` que tome el producto como parámetro e incremente el importe total y el número de ventas.

En el método `main`, crea varios objetos `Producto`, realiza ventas y luego imprime tanto el total de productos vendidos como el total de ventas en la tienda.

```
1 package ad.udl_repaso.basico.actividad42;
2
3 public class App {
4
5     public static void main(String[] args) {
6         Producto producto1 = new Producto("Laptop", 800);
7         Producto producto2 = new Producto("Teléfono", 500);
8         Producto producto3 = new Producto("Tablet", 300);
9
10        Tienda.realizarVenta(producto1);
11        Tienda.realizarVenta(producto2);
12        Tienda.realizarVenta(producto3);
13
14        System.out.println("Numero de ventas: " + Tienda.getNumeroVentas());
15        System.out.println("Total de ventas: " + Tienda.getTotalImporteVendido());
16
17    }
18
19 }
20
```

Console X Problems Debug Shell Search Terminal Call Hierarchy Coverage Markdown View His

<terminated> App (14) [Java Application] C:\Program Files\Java\jdk1.8.0\_202\bin\javaw.exe (8 abr 2024 18:53:07 – 18:53:08) [pid: 25700]

Numero de ventas: 3.0  
Total de ventas: 1600.0

### Actividad 4.3

Crea un paquete llamado staticproperty.actividad43

Intenta escribir un programa para representar el consumo de energía de una instalación eléctrica. Para ello, se hará una clase que representa los aparatos conectados en la instalación. Cada aparato tiene un consumo eléctrico determinado. Al encender un aparato eléctrico, el consumo de energía se incrementa en la potencia de dicho aparato. Al apagarlo, se decrementa el consumo. Inicialmente, los aparatos están todos apagados. Además, se desea consultar el consumo total de la instalación.

Dicho todo ésto, haz un programa que declare dos aparatos eléctricos, una bombilla de 150 vatios y una plancha de 2000 vatios. El programa deberá imprimir el consumo nada más crear los objetos. Después, se enciende la bombilla y la plancha, y el programa imprime el consumo. Luego se apaga la bombilla, y se vuelve a imprimir el consumo.

### Cómo puede hacerse

Según el enunciado, la clase que modela a los aparatos eléctricos tiene dos datos, su potencia y si están encendidos (o no). Esta clase podéis llamarla AparatoElectrico. Por otro lado, el consumo total debe ser un atributo accesible a todos los aparatos eléctricos, lo que se consigue con un elemento de clase. Para ello, se declara el atributo consumoTotal como double y, como no, static. Así, todas las instancias de AparatoElectrico comparten el mismo atributo.



Para consultar el consumo total de la instalación, se proporciona un método que devuelva el valor de consumoTotal. Este método ha de ser también de clase (o sea, static) y podéis llamarlo consumo()).

Cuando un aparato eléctrico se enciende puede incrementar el consumoTotal con la potencia que consume. Esto lo llevará a cabo con el método enciende(). Correspondientemente, cuando el aparato se apaga, se resta su potencia al consumo total. Y de nuevo, sólo si estaba encendido. Esto lo hará el método apaga()

## Actividades grupo 5. Colecciones

### Actividad 5.1 Probar colecciones

Crea un paquete llamado colecciones.actividad51

Revisa en la teoría aquellos videos de las colecciones que necesites repasar.

Copia en IntelliJ los ejemplos de w3schools, entiendelos, ejecútalos e intenta utilizar más opciones del API de cada colección.

<https://www.w3schools.blog/list-set-map-java>

¿Qué pasa si añades en un Set un elemento que ya existe en la lista? ¿Permite duplicados?

¿Qué sucede si añades a un Map un elemento con clave que ya existe?

Replica los ejemplos anteriores para List, Set y Mapa, pero utilizando en vez de un String en las colecciones, un objeto. P.ej:

Alumno (String nombre, int id)

### Actividad 5.2

Crea un paquete llamado colecciones.actividad52

Haz una clase App con un main, que realice la siguiente tarea:

Tiene que ofrecer al usuario la opción de realizar un CRUD (Create, Remove, Read, Update) sobre una lista de artistas musicales, el artista se define como un String

1. Nuevo artista
2. Consultar artistas
3. Eliminar artista
4. Actualizar artista
5. Salir del programa

Introduzca una opción:

....

#### Implementación:

- 1 Nuevo artista => el programa solicitará que se introduzca un String con el nombre del artista
- 2 El programa mostrará por consola todos los artistas introducidos hasta el momento. Su posición en la lista (id) y nombre
- 3 El programa solicitará que introduzcamos la posición en la lista (id) para borrar ese artista.
- 4 El programa solicitará el id del artista a actualizar y posteriormente un String con el nombre del nuevo artista
- 5 Salir del programa

### Actividad 5.3

Crea un paquete llamado colecciones.actividad53

Haz una clase App con un main, que realice la siguiente tarea:

Debemos montar un software de encriptación/desencriptación para los teléfonos de nuestros agentes secretos:

Implementación:

- El software tiene 2 modos de funcionamiento, dependiendo del valor introducido por el usuario por teclado: C (Cifrado), D (Descifrado).
- Después de introducir el modo de funcionamiento, el usuario introducirá el número de teléfono a cifrar descifrar
- El mapa cifrado descifrado sería el siguiente

0	1
1	9
2	3
3	7
4	8
5	6
6	2
7	4
8	5
9	0

Ejemplo:

-Introduzca el modo de funcionamiento:

-C

-Introduzca el teléfono: 615051847

-Teléfono cifrado: 296169584

### **Cómo puede hacerse**

Podemos tener 2 mapas creados de antemano, uno para cifrar y otro para descifrar.

Recibimos el teléfono como String, iteramos sobre él para obtener un character en concreto. Ese character se puede convertir a entero: <https://www.javatpoint.com/java-char-to-int>

### **Actividad 5.4**

Crea un paquete colecciones.actividad54

Haz una clase App Main que cumpla lo siguiente:

Queremos hacer una app para agregar cosas que nos gustan del 0 al 10. Para ello, en bucle se nos pedirá una tarea y la puntuación que le asignamos. Saldremos de dicho bucle cuando en la tarea escribamos la palabra SALIR. Una vez que hayamos escrito SALIR, se mostrarán las tareas asignadas a cada puntuación

Ejemplo salida:

```
--- Tareas por Puntuación ---  
Puntuación 5: [FREGAR, CORRER]  
Puntuación 8: [IR AL CINE, IR AL TEATRO]  
Puntuación 10: [VER EL MOVIL]
```

### **Cómo puede hacerse**

Puedes tener un mapa con clave la puntuación y valor la lista de tareas

## Actividades grupo 6. Herencia

### Actividad 6.1

Crea un paquete llamado herencia.actividad61

Copia en IntelliJ los ejemplos de w3schools, entiendelos, ejecutalos. Agrega un método más que devuelva el número de ruedas de un vehículo. Imprime las ruedas de myCar.

Seguramente no te funcionen desde IntelliJ, es debido a cómo está estructurado el código, separa cada Clase en su correspondiente fichero, de manera que tengas 3 clases:

- Vehicle
- Car
- App: contendrá un método static void main(String[] args) dónde instances el Coche y realices el código que vienen en el ejemplo

[https://www.w3schools.com/java/java\\_inheritance.asp](https://www.w3schools.com/java/java_inheritance.asp)

### Actividad 6.2

Crea un paquete llamado herencia.actividad62

Crea una aplicación Java para gestionar empleados en una empresa. Hay tres tipos de empleados: Empleado, EmpleadoTemporal y EmpleadoPermanente. Todos los empleados tienen un nombre, un salario y un método calcularSalarioAnual() que calcula el salario.

La clase Empleado es la clase base. Debe tener los siguientes atributos y métodos:

- Atributos: nombre (String), salario (double). Salario se refiere al salario base mensual.
- Métodos:
  - Constructor: Un constructor que inicializa el nombre y el salario del empleado.
  - calcularSalarioAnual(): Un método que devuelve el salario anual del empleado.

La clase EmpleadoTemporal hereda de Empleado. Además de los atributos y métodos de Empleado, tiene un atributo adicional duracionContrato (int) que representa la duración

del contrato en meses. Sobrescribe el método `calcularSalarioAnual()` para calcular el salario de acuerdo con la duración del contrato (por ejemplo, el salario base por mes multiplicado por la duración del contrato).

La clase `EmpleadoPermanente` también hereda de `Empleado`. Además de los atributos y métodos de `Empleado`, tiene un atributo adicional `beneficios` (`double`) que representa el valor de los beneficios adicionales que recibe el empleado permanente. Sobrescribe el método `calcularSalarioAnual()` para calcular el salario sumando el salario base y los beneficios.

Crea una clase `Main` que tenga el método `main()` para probar tu implementación. Crea instancias de cada tipo de empleado, asigna valores y muestra los salarios calculados para cada uno.

Este ejercicio proporciona una buena práctica para comprender los conceptos de herencia en Java y cómo se pueden sobrescribir métodos en las clases derivadas.

## Actividades grupo 7. Interfaces

### Actividad 7.1

Crea un paquete llamado `interfaz.actividad71`

Revisa primero el apartado de teoría correspondiente.

En IntelliJ, crea los ejemplos proporcionados en el ejemplo de w3schools:
















[https://www.w3schools.com/java/java\\_interface.asp](https://www.w3schools.com/java/java_interface.asp)

### Actividad 7.2

Crea un paquete llamado `interfaz.actividad72`

Vamos a realizar una caravana de vehículos, la típica que vemos todos los años en la tele en verano desplazándose a las playas.

Para eso nos han facilitado una interfaz que deben implementar todos los vehículos a motor, ya que debe controlarse su velocidad en todo tipo de carretera:

<b>NUEVOS LÍMITES DE VELOCIDAD</b>	Autovías y autopistas	Carreteras convencionales	Carreteras convencionales con separación física entre los dos sentidos	Sin pavimentar
 Turismos y motos				
 Autobuses, caravanas y tte. escolar				
 Camiones y furgonetas				
Excepción: Se puede superar en 20 km/h el límite genérico de 90 km/h en adelantamientos				

```
public interface Vehiculo {  
  
    int velAutopista();  
    int velCarretera();  
    int velCarreteraSepFisica();  
    int velSinPavimentar();  
}
```



```
}
```

Implementa esa interfaz para los vehículos del gráfico e implementa su velocidad. Pueden ser 3 clases: Turismo, Autobus y Camion.

Crea un App con un main dónde vamos a crear una lista “caravana de vehículos”. Rellénala con varios vehículos de diversa índole

```
List<Vehiculo> caravana = new ArrayList<Vehiculo>  
caravana.add(new Turismo());  
caravana.add(new Camion());  
...
```

Crea un método que reciba como argumento nuestra caravana e imprima por consola su velocidad por autopista.

## Actividades grupo 8. Polimorfismo

Revisa la teoría propuesta en la documentación, cuando creas que la hayas entendido, realiza la actividad propuesta.

### Actividad 8.1

Crea un paquete llamado polimorfismo.actividad81

Sobre el ejemplo de este blog:

<https://www.arquitecturajava.com/java-polimorfismo-herencia-y-simplicidad/>

Entiéndelo, haz preguntas a tu profesor e implementalo en IntelliJ.

### Actividad 8.2

Crea un paquete llamado polimorfismo.actividad82

En la carpeta de Drive de la carpeta AD\TEMARIO\UD1\ACTIVIDADES\ hay un documento llamado Polimorfismo. Realiza su implementación

## ANEXO

### Más ejercicios

Nota: Realiza estos ejercicios en un paquete aparte dónde no interfiera con los ejemplos realizados en clase.

Java Básico

<https://www.w3resource.com/java-exercises/basic/index.php>

Bucles

<https://manolohidalgo.com/ejercicios-bucles-en-java/>

Arrays

<https://dam.org.es/ejercicios-de-arrays-resueltos/>

Ejercicios POO:

- <https://www.discoduroderoer.es/ejercicios-propuestos-y-resueltos-programacion-orientado-a-objetos-java/>
- <http://puntocomnoesunlenguaje.blogspot.com/p/blog-page.html>

Static:

- <https://javaparajavatos.wordpress.com/2016/01/11/ejercicio-de-ampliacion-de-clases-static-i>

Colecciones:

- <http://myfpschool.com/ejercicios-de-java-collections/>

Ejercicios todo tipo:

- <https://www.geeksforgeeks.org/java-exercises/#array-programs-in-java>
- <https://github.com/PedroProfe/Programacion/blob/main/Ejercicio%20colaboraci%C3%B3n%20entre%20clases%20solucion.pdf>
- <https://elhacker.info/manuales/Lenguajes%20de%20Programacion/Java/Ejercicios-de-Programacion-en-Java.pdf>