

# **PROGRAMACIÓN JAVA**

## **EJERCICIOS 1**

---

**1º DESARROLLO DE  
APLICACIONES  
MULTIPLATAFORMA (D.A.M.)**

**2021/2022**

## INDICE

1. CONCEPTOS BÁSICOS .....	1
2. CONDICIONALES .....	2
3. BUCLES.....	6
4. FUNCIONES .....	11
5. VECTORES Y MATRICES (TABLAS). ....	14
6. CADENA DE CARACTERES .....	17
8. CLASES .....	23
9. HERENCIA .....	29

## 1. Conceptos Básicos

1. Diseñar una aplicación que calcule la longitud y el área de una circunferencia. Para ello, el usuario debe introducir el radio (que puede contener decimales).

$$\text{Longitud} = 2\pi \cdot \text{radio}$$
$$\text{Área} = \pi \cdot \text{radio}^2$$

```
<terminated> E1 [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\
Escribe el radio: 14
La longitud de la circunferencia es: 87.96459430051421
El area del circulo es: 615.7521601035994
```

1. Escribir un programa que pida un número al usuario e indique mediante un literal booleano (true o false) si el número es par.

```
<terminated> E2 [Java Application]
Escribe un numero : 19
Es par: false
```

2. Un frutero necesita calcular los beneficios anuales que obtiene de la venta de manzanas y peras. Por este motivo es necesario diseñar una aplicación que solicite las ventas (en kilos) de cada semestre para cada fruta. La aplicación mostrara el importe total sabiendo que el precio del kilo de manzanas está fijado en 2.35 Euros y el kilo de peras en 1,95 Euros.

```
Problems @ Javadoc Declaration Console X
<terminated> E3 [Java Application] C:\Program Files\Java\jdk-17.0.1\bin
Para las manzanas:
Ventas (en kilos) del primer semestre: 5
Ventas (en kilos) del segundo semestre: 9
Para las peras:
Ventas (en kilos) del primer semestre: 7
Ventas (en kilos) del segundo semestre: 4
El importe total es de: 54.349999999999994 euros
```

3. Escribir un programa que pida un numero al usuario y muestre su valor absoluto

### Nota: Operador ternario

*Este operador devuelve un valor que se selecciona de entre dos posibles. La selección dependerá de la evaluación de una expresión relacional o lógica que, como hemos visto, puede tomar dos valores: verdadero o falso.*

*El operador tiene la siguiente sintaxis:*

*expresión condicional ? valor1 : valor2*

La evaluación de la expresión decidirá cuál de los dos posibles valores se devuelven. En el caso de que la expresión resulte cierta se devuelve valor1, y cuando la expresión resulte falsa, valor2.

Ejemplos:

```
int a, b;
```

```
a = 3 < 5 ? 1 : -1; // 3 < 5 es cierto: a toma el valor 1
```

```
b = a == 7 ? 10 : 20; // a (que vale 1) == 7 es falso: b toma el valor 20
```

```
Escribe un número (positivo o negativo):  
-42  
El valor absoluto de -42 es 42
```

4. Escribir un programa que solicite las notas del primer, segundo y tercer trimestre (notas enteras que se solicitarán al usuario). El programa debe mostrar la nota media del curso como se utiliza en el boletín de calificaciones (solo la parte entera) y como se usa en el expediente académico (con decimales).

```
Nota primer trimestre: 8  
Nota segundo trimestre: 7  
Nota tercer trimestre: 6  
Boletín de calificaciones: 7  
Expediente académico: 7.0
```

5. Realizar un programa que pida como entrada un número decimal y lo muestre redondeado al entero más próximo.

```
Escriba un numero decimal (con punto): 54.75  
54.75 redondeo es: 55
```

## 2. Condicionales

6. Diseñar una aplicación que solicite al usuario un número e indique si es par o impar.

```
<terminated> E7 [Java Application]  
Introduce un numero: 423  
Es impar
```

8. Pedir dos números enteros y decir si son iguales o no.

```
Introduzca un número: 28
Introduzca otro número: 37
Los números son distintos:
```

9. Solicitar dos números distintos y mostrar cuál es el mayor.

```
<terminated> E9 [Java Application] C
Introduzca un número: 163
Introduzca otro número: 254
254 es mayor que 163
```

10. Implementar un programa que pida por teclado un número decimal e indique si es un número casi-cero, que son aquellos, positivos o negativos, que se acercan a 0 por menos de 1 unidad, aunque curiosamente el 0 no se considera un número casi-cero. Ejemplos de números casi-cero son: el 0,3, el -0,99 o el 0,123; algunos números que no se consideran casi-ceros son: el 12,3, el 0 o el -1.

```
<terminated> E10 [Java Application] C:\Program Files\Java\jdk-17.0.1\bin
Introduzca un número real positivo o negativo: -27.33
No es un casi-cero
```

11. Pedir dos números y mostrarlos ordenados de forma decreciente.

```
Introduzca un número: 45
Introduzca otro: 86
86, 45
```

12. Realizar de nuevo la Actividad resuelta 9 considerando el caso de que los números introducidos sean iguales.

```
<terminated> E11 [Java Application]
Introduzca un número: 45
Introduzca otro número: 89
89 es mayor que 45
```

13. Pedir tres números y mostrarlos ordenados de mayor a menor

```
<terminated> E13 [Java Application] C:\
Introduzca primer número: 77
Introduzca segundo número: 43
Introduzca tercer número: 2
77, 43, 2
```

14. Pedir los coeficientes de una ecuación de segundo grado y mostrar sus soluciones reales. Si no existen, habrá que indicarlo. Hay que tener en cuenta que las soluciones de una ecuación de segundo grado,  $ax^2 + bx + c = 0$ , son:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
Problems @ Javadoc Declaration Console
<terminated> main [Java Application] C:\Program Files\Java\
Introduzca primer coeficiente (a): 10.6
Introduzca segundo coeficiente: (b) : 5.2
Introduzca tercer coeficiente: (c) : 4
No existen soluciones reales
```

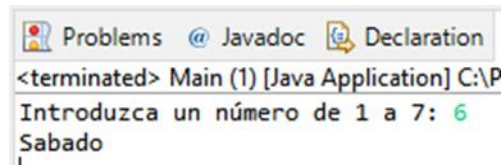
15. Escribir una aplicación que indique cuántas cifras tiene un número entero introducido por teclado, que estará comprendido entre 0 y 99 999.

```
Problems @ Javadoc Declaration Console X
<terminated> Main [Java Application] C:\Program Files\Java\jd
Introduzca un número entre 0 y 99.999: 5847
Tiene 4 cifras
```

16. Pedir una nota entera de 0 a 10 y mostrarla de la siguiente forma: insuficiente (de 0 a 4), suficiente (5), bien (6), notable (7 y 8) y sobresaliente (9 y 10).

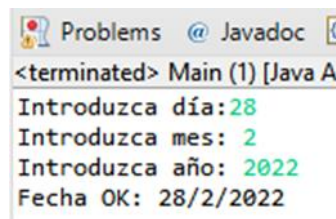
```
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Appli
Introduzca una nota: 6
Bien
```

17. Idear un programa que solicite al usuario un número comprendido entre 1 y 7, correspondiente a un día de la semana. Se debe mostrar el nombre del día de la semana al que corresponde. Por ejemplo, el número 1 corresponde a «lunes» y el 6 a «sábado».



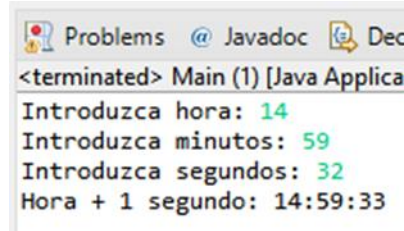
```
Problems @ Javadoc Declaration
<terminated> Main (1) [Java Application] C:\P
Introduzca un número de 1 a 7: 6
Sabado
```

18. Pedir el día, mes y año de una fecha e indicar si la fecha es correcta. Hay que tener en cuenta que existen meses con 28, 30 y 31 días (no se considerará los años bisiestos).



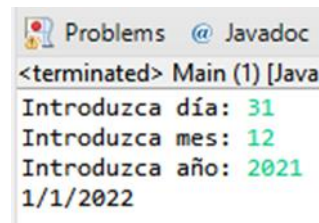
```
Problems @ Javadoc
<terminated> Main (1) [Java A
Introduzca día: 28
Introduzca mes: 2
Introduzca año: 2022
Fecha OK: 28/2/2022
```

19. Escribir un programa que pida una hora de la siguiente forma: hora, minutos y segundos. El programa debe mostrar qué hora será un segundo más tarde. Por ejemplo: hora actual [10:41:59] → hora actual +1 segundo: [10:42:00]



```
Problems @ Javadoc Dec
<terminated> Main (1) [Java Applica
Introduzca hora: 14
Introduzca minutos: 59
Introduzca segundos: 32
Hora + 1 segundo: 14:59:33
```

20. Crear una aplicación que solicite al usuario una fecha (día, mes y año) y muestre la fecha correspondiente al día siguiente.



```
Problems @ Javadoc
<terminated> Main (1) [Java
Introduzca día: 31
Introduzca mes: 12
Introduzca año: 2021
1/1/2022
```



### 3. Bucles

21. Diseñar un programa que muestre, para cada número introducido por teclado, si es par, si es positivo y su cuadrado. El proceso se repetirá hasta que el número introducido sea 0.

```
<terminated> Main (1) [Java Application]
Introduzca número: 7
Es par?: false
Es positivo?: true
Cuadrado: 49
Introduzca otro número: 122
Es par?: true
Es positivo?: true
Cuadrado: 14884
Introduzca otro número: 0
```

22. Implementar una aplicación para calcular datos estadísticos de las edades de los alumnos de un centro educativo. Se introducirán datos hasta que uno de ellos sea negativo, y se mostrará: la suma de todas las edades introducidas, la media, el número de alumnos y cuantos son mayores de edad.

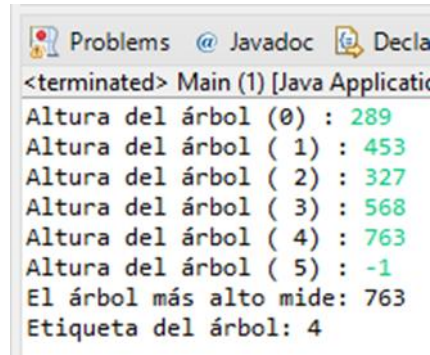
```
<terminated> Main (1) [Java Application]
Introduzca edad: 18
Introduzca edad: 22
Introduzca edad: 21
Introduzca edad: 17
Introduzca edad: 16
Introduzca edad: 23
Introduzca edad: -1
Suma de todas las edades: 117
Media: 19.5
Número total de alumnos: 6
Mayores de edad: 4
```

23. Codificar el juego «el número secreto», que consiste en acertar un número entre 1 y 100 (generado aleatoriamente). Para ello se introduce por teclado una serie de, números, para los que se indica: «mayor» o «menor», según sea mayor o menor con respecto al número secreto. El proceso termina cuando el usuario acierta o cuando se rinde, (introduciendo un - 1).

```
<terminated> Main (1) [Java Application] C:\Program
Introduzca un número entre 1 y 100: 50
Mayor
Introduzca otro número: 75
Menor
Introduzca otro número: 60
Menor
Introduzca otro número: 55
Enhorabuena . . .
```

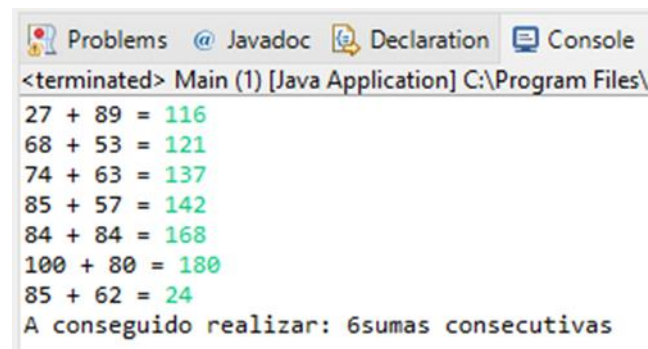


24. Un centro de investigación de la flora urbana necesita una aplicación que muestre cuál es el árbol más alto. Para ello se introducirá por teclado la altura (en centímetros) de cada árbol (terminando la introducción de datos cuando se utilice -1 como altura). Los árboles se identifican mediante etiquetas con números únicos correlativos, comenzando en 0. Diseñar una aplicación que resuelva el problema planteado.



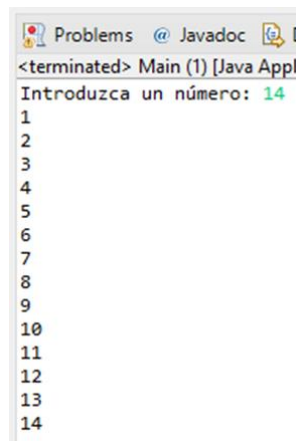
```
Problems @ Javadoc Decla
<terminated> Main (1) [Java Applicati
Altura del árbol (0) : 289
Altura del árbol ( 1) : 453
Altura del árbol ( 2) : 327
Altura del árbol ( 3) : 568
Altura del árbol ( 4) : 763
Altura del árbol ( 5) : -1
El árbol más alto mide: 763
Etiqueta del árbol: 4
```

25. Desarrollar un juego que ayude a mejorar el cálculo mental de la suma. El jugador tendrá que introducir la solución de la suma de dos números aleatorios comprendidos entre 1 y 100. Mientras la solución introducida sea correcta, el juego continuará. En caso contrario, el programa terminará y mostrará el número de operaciones realizadas correctamente.



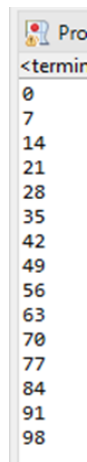
```
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\
27 + 89 = 116
68 + 53 = 121
74 + 63 = 137
85 + 57 = 142
84 + 84 = 168
100 + 80 = 180
85 + 62 = 24
A conseguido realizar: 6sumas consecutivas
```

26. Escribir una aplicación para aprender a contar, que pedirá un número n y mostrará todos los números del 1 a n.



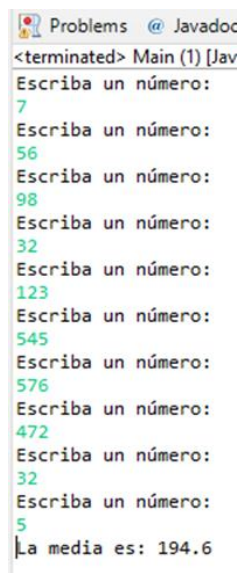
```
Problems @ Javadoc Decla
<terminated> Main (1) [Java Appl
Introduzca un número: 14
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

27. Escribir todos los múltiplos de 7 menores que 100



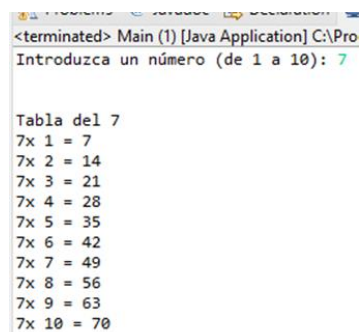
```
<terminar>  
0  
7  
14  
21  
28  
35  
42  
49  
56  
63  
70  
77  
84  
91  
98
```

28. Pedir diez números enteros por teclado y mostrar la media.



```
<terminated> Main (1) [Java  
Escriba un número:  
7  
Escriba un número:  
56  
Escriba un número:  
98  
Escriba un número:  
32  
Escriba un número:  
123  
Escriba un número:  
545  
Escriba un número:  
576  
Escriba un número:  
472  
Escriba un número:  
32  
Escriba un número:  
5  
La media es: 194.6
```

29. Implementar una aplicación que pida al usuario un número comprendido entre 1 y 10. Hay que mostrar la tabla de multiplicar de dicho número, asegurándose de que el número introducido se encuentra en el rango establecido.



```
<terminated> Main (1) [Java Application] C:\Pro  
Introduzca un número (de 1 a 10): 7  
  
Tabla del 7  
7x 1 = 7  
7x 2 = 14  
7x 3 = 21  
7x 4 = 28  
7x 5 = 35  
7x 6 = 42  
7x 7 = 49  
7x 8 = 56  
7x 9 = 63  
7x 10 = 70
```

30. Diseñar un programa que muestre la suma de los 10 primeros números impares.

```
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jd
La suma de los 10 primeros impares es: 100.0
```

31. Pedir un número y calcular su factorial. Por ejemplo, el factorial de 5 se denota 5! y es 1 igual a  $5 \times 4 \times 3 \times 2 \times 1 = 120$ .

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\
Introduzca un número: 37
El factorial de 37 es: 1.3763753091226346E43
```

32. Pedir 5 calificaciones de alumnos y decir al final si hay algún suspenso.

```
Problems @ Javadoc Declaration
<terminated> Main (1) [Java Application] C:\f
Introduzca nota (de 0 a 10): 7
Introduzca nota (de 0 a 10): 3
Introduzca nota (de 0 a 10): 5
Introduzca nota (de 0 a 10): 6
Introduzca nota (de 0 a 10): 8
Hay alumnos suspensos
```

33. Dadas 6 notas, escribir la cantidad de alumnos aprobados, condicionados (nota igual a cuatro) y suspensos.

```
<terminated> Main (1) [Java Applica
Nota del alumno número 1:
7
Nota del alumno número 2:
5
Nota del alumno número 3:
4
Nota del alumno número 4:
2
Nota del alumno número 5:
8
Nota del alumno número 6:
9
Aprobados: 4
Suspensos: 1
Condicionados: 1
```

34. Diseñar una aplicación que muestre las tablas de multiplicar del 1 al 10.

```
<terminated> Main (1) |
4x10=40

Tabla del 5
5x1=5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50

Tabla del 6
6x1=6
6x2=12
6x3=18
6x4=24
6x5=30
6x6=36
6x7=42
6x8=48
6x9=54
6x10=60

Tabla del 7
7x1=7
7x2=14
7x3=21
7x4=28
```

35. Pedir por consola un numero n y dibujar un triángulo rectángulo de n elementos de lado, utilizando para ello asteriscos (\*). Por ejemplo, para n = 4:

```
*****
****
***
**
*
```

```
<terminated> Main (1) |
Escriba n: 5
* * * * *
* * * *
* * *
* *
*
```

## 4. Funciones

36. Diseñar la función `eco ()` a la que se le pasa como parámetro un número `n`, y muestra por pantalla `n` veces el mensaje «Eco ... ».

```
<terminated> Main (1) [Java Application] C:\Program
Introduzca un número : 7
| - -Antes de llamar a la función - -
Eco . . :
Eco . . :
Eco . . :
Eco . . :
Eco . . :
Eco . . :
Eco . . :
--Después de llamar a la función--
```

37. Escribir una función a la que se le pasen dos enteros y muestre todos los números comprendidos entre ellos.

```
<terminated> Main (1) [Java Application]
Introduzca primer número: 3
Introduzca segundo número: 15
3
4
5
6
7
8
9
10
11
12
13
14
15|
```

38. Realizar una, función que calcule y muestre el área o el volumen de un cilindro, según se especifique. Para distinguir un caso de otro se le pasará como opción un número: 1 (para el área) o 2 (para el volumen). Además, hay que pasarle a la función el radio de la base y la altura.

$$\text{área} = 2\pi \cdot \text{radio} \cdot (\text{altura} + \text{radio})$$

$$\text{volumen} = \pi \cdot \text{radio}^2 \cdot \text{altura}$$

```

Introduzca radio: 53
Introduzca altura: 89
Qué desea calcular (1 (volumen)/ 2 (area) : 1
El volumen es de: 785401.304990102

```

39. Diseñar una función que recibe como parámetros dos, números enteros y devuelve el máximo de ambos.

```

Introduzca un número: 86
Introduzca otro número: 41
El número mayor es: 86

```

40. Crear una función que, mediante un booleano, indique si el carácter que se pasa como parámetro de entrada corresponde con una vocal.

```

La i es una vocal true
La E es una vocal true
La f es una vocal false

```

41. Diseñar una función con el siguiente prototipo:

boolean esPrimo(int n)

que devolverá true si n es primo y false en caso contrario.

```

^terminated^ main (1)
1 es compuesto
2 es primo
3 es primo
4 es compuesto
5 es primo
6 es compuesto
7 es primo
8 es compuesto
9 es compuesto
10 es compuesto
11 es primo
12 es compuesto
13 es primo
14 es compuesto
15 es compuesto

```

42. Escribir una función a la que se le pase un número entero y devuelva el número de divisores primos que tiene.

```

^terminated^ main (1) por
Divisores de 24: 2

```

43. Diseñar la función calculadora (), a la que se le pasan dos números reales (operandos) y qué operación se desea realizar con ellos. las operaciones disponibles son: sumar, restar, multiplicar o dividir. Estas se especifican mediante un número: 1 para la suma, 2 para la resta, 3 para la multiplicación y 4 para la división. La función devolverá el resultado de la operación mediante un número real.

```
7.0
-1.0
12.0
0.75
```

44. Repetir la Actividad resuelta 39 con una versión que calcule el máximo de tres números.

```
<terminated> main (1)
El mayor es: 9
```

45. Diseñar una función que calcule  $a^n$ , donde a es real y n es entero no negativo. Realizar una versión iterativa y otra recursiva.

a)

```
<terminated> Main (1) [Java Application] C:\Program Files\Ja
Introduzca base (real): 7.0
Introduzca exponente (entero no negativo): 3
7.0 elevado a 3 = 343.0
```

b)

```
<terminated> main (1) [Java Application] C:\Program Files\Ja
Introduzca base (real): 8.8
Introduzca el exponente: 4
El resultado es: 5996.953600000003
```



46. Escribir una función que calcule de forma recursiva el máximo común divisor de dos números. Para ello sabemos:

$$mcd(a, b) = \begin{cases} mcd(a - b, b) & \text{si } a \geq b \\ mcd(a, b - a) & \text{si } b > a \\ a & \text{si } b = 0 \\ b & \text{si } a = 0 \end{cases}$$

```
<terminated> Main (1) [Java Application] C:\Program Fil
Introduzca primer número: 128
Introduzca segundo número : 242
El mcd es 2
```

47. Diseñar una función recursiva que calcule el enésimo término de la serie de Fibonacci. En esta serie el enésimo valor se calcula sumando los dos valores anteriores de la serie. Es decir:

fibonacci(n) = fibonacci(n - 1) + fibonacci(n - 2)  
 fibonacci(0) = 1  
 fibonacci(1) = 1

```
<terminated> Main (1) [Java Application] C:\Program Fil
Vamos a calcular fibonacci( n )
Introduzca n (se recomienda n < 40 ) : 22

fibonacci(22) = 28657
```

## 5. Vectores y Matrices (Tablas).

48. Crear una tabla de longitud 10 que se inicializará con números aleatorios comprendidos entre 1 y 100. Mostrar la suma de todos los números aleatorios que se guardan en la tabla.

```
<terminated> Main (1) [Java Application] C:\Program Fil
La suma de los valores aleatorios es: 399
```

49. Diseñar un programa que solicite al usuario que introduzca por teclado 5 números decimales. A continuación mostrar, los números en el mismo orden que se han introducido.

```
<terminated> Main (1) [Java Applica
Introduzca un número: 2.0
Introduzca un número: 3.2
Introduzca un número: 4.7
Introduzca un número: 9.2
Introduzca un número: 3.4
[2.0, 3.2, 4.7, 9.2, 3.4]
```

50. Escribir una aplicación que solicite al usuario cuántos números desea introducir. A continuación, introducir por teclado esa cantidad de números enteros, y por último, mostrar en el orden inverso al introducido.

```
Cuántos números desea introducir:
3
Introduzca un número: 75
Introduzca un número: 42
Introduzca un número: 38
Los números en orden inverso son:
38
42
75|
```

51. Diseñar la función: `int máximo (int t [])`, que devuelva el máximo valor contenido en la tabla `t`.

```
<terminated> Main (1) [Java Ap
El valor maximo es: 85
```

52. Escribir la función `int [] rellenaPares(int longitud, int fin)`, que crea y devuelve una tabla ordenada de la longitud especificada, que se encuentra rellena con números pares aleatorios comprendidos en el rango desde 2 hasta fin (inclusive).

```
<terminated> main (1) [Java Application]
El valor del numero par es: 10
El valor del numero par es: 12
El valor del numero par es: 24
El valor del numero par es: 26
El valor del numero par es: 28
El valor del numero par es: 28
El valor del numero par es: 30
El valor del numero par es: 32
```

53. Definir una función que tome como parámetros dos tablas, la primera con los 6 números de una apuesta de la primitiva, Y la segunda (ordenada) con los 6 números de la combinación ganadora. La función devolverá el número de aciertos.

```
El numero de aciertos es: 2
```

54. Leer y almacenar `n` números enteros en una tabla, a partir de la que se construirán otras dos tablas con los elementos con valores pares e impares de la primera, respectivamente.

Las tablas pares e impares deben mostrarse ordenadas.

```
Escriba n: 5
Introduzca un número: 6
Introduzca un número: 35
Introduzca un número: 98
Introduzca un número: 54
Introduzca un número: 23
Pares: [6, 98, 54]
Impares: [35, 23]
```

55. Diseñar una aplicación para gestionar un campeonato de programación, donde se introduce la puntuación (enteros) obtenidos por 5 programadores, conforme van terminando su prueba. La aplicación debe mostrar las puntuaciones ordenadas de los 5 participantes. En ocasiones, cuando finalizan los 5 participantes anteriores, se suman al campeonato programadores de exhibición, cuyos puntos se incluyen con el resto. La forma de especificar que no intervienen más programadores de exhibición es introducir como puntuación un -1. La aplicación debe mostrar, finalmente, los puntos ordenados de todos los participantes.

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-17
Puntos programador (1): 20
Puntos programador (2): 30
Puntos programador (3): 15
Puntos programador (4): 12
Puntos programador (5): 25
Puntuación: [12, 15, 20, 25, 30]
Puntos del programador de exhibición: 30
Puntos del programador de exhibición: 24
Puntos del programador de exhibición: 15
Puntos del programador de exhibición: -1
Puntuacion final: [12, 15, 15, 20, 24, 25, 30, 30]
```

56. Escribir la función:

`int[] eliminarMayores(int t[], int valor )`

que crea y devuelve una copia de la tabla t donde se han eliminado todos los elementos que son mayores que valor.

```
<terminated> Main (1) [Java Application] C
El valor del vector es: [2, 4]
```

57. Desarrollar el Juego «la cámara secreta>>, que consiste en abrir una cámara mediante su combinación secreta, que está formado por una combinación de dígitos del uno al cinco. El jugador especificará cuál es la longitud de la combinación; a mayor longitud, mayor será la dificultad del Juego. La aplicación genera, de forma aleatoria, una combinación secreta que el usuario tendrá que acertar. En cada intento se muestra

como pista, para cada dígito de la combinación introducida por el jugador, si es mayor, menor o igual que el correspondiente en la combinación secreta.

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (16 ene 2022 13:20:43 - 13:20:44)
Longitud de la combinación secreta :
4
[5, 5, 2, 2]
Escriba una combinación
3
4
1
2
Pistas:
3 mayor
4 mayor
1 mayor
2 igual
Escriba una combinación:
5
5
2
2
| La cámara está abierta!
```

58. Crear una tabla bidimensional de longitud 5 x 5 y rellenarla de la siguiente forma: el elemento de la posición  $[n][m]$  debe contener el valor  $10 \times n + m$ . Después se debe mostrar su contenido.

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (16 ene 2022 13:20:43 - 13:20:44)
[[0, 1, 2, 3, 4], [10, 11, 12, 13, 14], [20, 21, 22, 23, 24], [30, 31, 32, 33, 34], [40, 41, 42, 43, 44]]
0 1 2 3 4
10 11 12 13 14
20 21 22 23 24
30 31 32 33 34
40 41 42 43 44
```

## 6. Cadena de caracteres

59. Escribir un programa que muestre todos los caracteres Unicode junto a su code point, cuyo valor esté comprendido entre `\u0000` y `\uFFFF`.

```

\terminated:
\ua0:
\ua1: ¡
\ua2: ¢
\ua3: £
\ua4: ¤
\ua5: ¥
\ua6: ¦
\ua7: §
\ua8: ¨
\ua9: ©
\uaa: º
\uab: «
\uac: ¬
\uad: -
\uae: ®
\uaf: ¯
\ub0: °
\ub1: ±
\ub2: ²
\ub3: ³
\ub4: ´
\ub5: µ
\ub6: ¶
\ub7: ·
\ub8: ¸
\ub9: ¹

```

60. Introducir por teclado dos frases e indicar cuál de ellas es la más corta, es decir, la que contiene menos caracteres.

```

\terminated: main() {
Primera frase:
Cuando el grajo vuela bajo
Segunda frase:
Hace un frio del carajo
Hace un frio del carajo es más corta que Cuando el grajo vuela bajo
}

```

61. Diseñar el juego «Acierta la contraseña». La mecánica del juego es la siguiente: el primer jugador introduce la contraseña; a continuación, el segundo jugador debe teclear palabras hasta que la acierte. Realizar dos versiones; en la primera se facilita el juego indicando si la palabra introducida es mayor o menor alfabéticamente que la contraseña. En la segunda, el programa mostrará la longitud de la contraseña y una cadena con los caracteres acertados en sus lugares respectivos y asteriscos en los no acertados.

**Solución a)**

```

Jugador 1. Introduzca la contraseña: agua
Jugador 2. Palabra : vino
La contraseña es menor que: vino
Jugador 2. Palabra : cerveza
La contraseña es menor que: cerveza
Jugador 2. Palabra : agua
¡Acertaste!

```

**Solución b)**

```

La contraseña tiene 4 caracteres
Jugador 2. Palabra: pepe
***
Jugador 2. Introduzca palabra de nuevo: aula
a**a
Jugador 2. Introduzca palabra de nuevo: agua
¡Acertaste!

```

62. Diseñar una aplicación que pida al usuario que introduzca una frase por teclado e indique cuántos espacios en blanco tiene.

```

<terminated> main (1) [Java Application] C:\Program
Escriba una frase: En Abril aguas mil
Tiene: 3 espacios en blanco

```

63. Diseñar una función a la que se le pase una cadena de caracteres y la devuelva invertida. Un ejemplo, la cadena «Hola mundo» quedaría «odnum aloH».

```

<terminated> Main (1) [Java Application] C:\F
Escriba una cadena: buenas tardes
sedrat saneub

```

64. Escribir un programa que pida el nombre completo al usuario y lo muestre sin vocales (mayúsculas, minúsculas y acentuadas). Por ejemplo, “Alvaro Perez se mostrará «lvr Prz »».

```

<terminated> main (1) [Java Application] C:\Program Files\Java
Escriba su nombre completo : Pablo Rodriguez
| Pbl Rdrgz

```

65. Diseñar un programa que solicite al usuario una frase y una palabra. A continuación buscará cuántas veces aparece la palabra en la frase.

```

<terminated> main (1) [Java Application] C:\Program Files\Java\jdk-1.8.0_60\bin\javaw.exe
Introduzca una frase: un tigre, dos tigres, tres tigres un trigar
Introduzca una palabra: tigres
"tigres" está 2 veces

```

66. Los habitantes de Javalandia tienen un idioma algo extraño; siempre comienzan sus frases con «Javalin, Javalon», para después hacer una pausa más o menos

larga (la pausa se representa mediante espacios en blanco o tabuladores) y a continuación expresan el mensaje. Existe un dialecto que no comienza sus frases con la muletilla anterior, pero siempre las terminan con un silencio, más o menos prolongado y la coletilla «javalen len len». Se pide diseñar un traductor que, en primer lugar nos diga si la frase introducida está escrita en el idioma de Javalandia (en cualquiera de sus dialectos), y en caso afirmativo, nos muestre solo el mensaje sin muletillas.

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (20 e
Escriba una frase: Javalín, javalón vaya marrón
vaya marrón
Escriba una frase: Con diez cañones por banda,
viento en popa a toda vela, No está escrito en el idioma de Javalandia
```

67. Introducir por teclado una frase palabra a palabra, y mostrar la frase completa separando las palabras introducidas con espacios en blanco. Terminar de leer la frase cuando alguna de las palabras introducidas sea la cadena «fin» escrita con cualquier combinación de mayúsculas y minúsculas. La cadena «fin» no aparecerá en la frase final.

```
Escriba una palabra : colorin
Escriba una palabra: colorado
Escriba una palabra: este
Escriba una palabra: cuento
Escriba una palabra: se
Escriba una palabra: ha
Escriba una palabra: acabado
Escriba una palabra: FIN
| colorin colorado este cuento se ha acabado
```

68. Realizar un programa que lea una frase del teclado y nos indique si es palíndroma, es decir, que la frase sea igual leyendo de izquierda a derecha que de derecha a izquierda, sin tener en cuenta los espacios. Un ejemplo de frase palíndroma es: «Dábale arroz a la zorra el abad».

Las vocales con tilde hacen que los algoritmos consideren una frase palíndroma como si no lo fuese. Por esto, supondremos que el usuario introduce la frase sin tildes.

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-17.0
Introduzca una frase: dabale arroz a la zorra el abad
La frase es palíndroma
```

69. Se dispone de la siguiente relación de letras:



conjunto 1:	e	i	k	m	p	q	r	s	t	u	v
conjunto 2:	p	v	i	u	m	t	e	r	k	q	s

Con ellas es posible codificar un texto, convirtiendo cada letra del conjunto 1 en su correspondiente del conjunto 2. El resto de las letras no se modifican. Los conjuntos se utilizan tanto para codificar mayúsculas como minúsculas, mostrando siempre la codificación en minúsculas.

Un ejemplo: la palabra «PaquiTo» se codifica como «matqvko».

Realizar un programa que codifique un texto. Para ello implementar la siguiente función:

```
char codifica(char conjunto1[], char conjunto2[], char c)
```

que devuelve el carácter c codificado según los conjuntos 1 y 2 que se le pasen.

```
Introduzca un texto a codificar: las oscuras golondrinas
lar orcqear golondevnar
```

70. Un anagrama es una palabra que resulta del cambio del orden de los caracteres de otra. Ejemplos de anagramas para la palabra roma son: amor, ramo o mora. Construir un programa que solicite al usuario dos palabras e indique si son anagramas una de otra.

```
Escriba una palabra:
roma
Escriba otra:
amor
Son anagramas
```

71. Diseñar un algoritmo que lea del teclado una frase e indique, para cada letra que aparece en la frase, cuántas veces se repite. Se consideran iguales las letras mayúsculas y las minúsculas para realizar la cuenta. Un ejemplo sería:

Frase: En un lugar de La Mancha.

Resultado:

a: 4 veces  
c: 1 vez  
d: 1 vez  
e: 2 veces

```

Introduzca una frase: Soldadito marinero
La letra a se repite 2 veces
La letra d se repite 2 veces
La letra e se repite 1 veces
La letra i se repite 2 veces
La letra l se repite 1 veces
La letra m se repite 1 veces
La letra n se repite 1 veces
La letra o se repite 3 veces
La letra r se repite 2 veces
La letra s se repite 1 veces
La letra t se repite 1 veces

```

72. Implementar el juego del anagrama, que consiste en que un jugador escribe una palabra y la aplicación muestra un anagrama (cambio del orden de los caracteres) generado al azar. A continuación, otro jugador tiene que acertar cuál es el texto original. La aplicación no debe permitir que el texto introducido por el jugador 1 sea la cadena vacía. Por ejemplo, si el jugador 1 escribe «teclado», la aplicación muestra como pista un anagrama al azar, como por ejemplo: «etcloda».

```

1 import java.util.Scanner;
2 public class Main {
3     public static void main(String[] args) {
4         String original; //texto original que introduce el jugador 1
5         String intento; //intento de acertar la palabra original del jugador 2
6         do {
7             System.out.print("Jugador 1. Introduzca una palabra: ");
8             original = new Scanner(System.in).next();
9             while (original.isEmpty());
10
11             String anagrama = creaAnagrama(original);
12             System.out.println("A qué palabra corresponde el anagrama: " + anagrama);
13             do {
14                 System.out.println("Jugador 2. ¿Cuál es el original?");
15                 intento = new Scanner(System.in).next();
16             } while (!original.equals(intento)); //mientras no acierte el texto original
17             System.out.println("Muy bien ... "); //si salimos del bucle es que ha acertado
18         }
19
20         /* La función creaAnagrama() crea y devuelve un anagrama del texto original
21         * pasado como parámetro. El algoritmo para construir el anagrama es:
22         * 1. Convertir el String original en una tabla, que es más cómoda para
23         * intercambiar caracteres.
24         * 2. Elegir dos caracteres (sus índices) al azar e intercambiarlos.
25         * 3. Repetir el punto 2. Cuantas más veces se repita, mayor es el desorden.
26         * Repetiremos tantas veces como la longitud del texto original. */
27         static String creaAnagrama(String original) {
28             char anagrama[] = original.toCharArray(); //una tabla es más cómoda para modificar
29
30             // realizamos un intercambio al azar por cada carácter que forma el texto
31             for (int numCambios = 0; numCambios < anagrama.length; numCambios++) {
32                 int i = (int) (Math.random() * anagrama.length); //índice al azar
33                 int j = (int) (Math.random() * anagrama.length); //índice al azar
34                 char aux = anagrama[i]; //intercambiarnos anagrama[i] y anagrama[j]
35                 anagrama[i] = anagrama[j];
36                 anagrama[j] = aux;
37             }
38             return String.valueOf(anagrama); //devolvemos un String a partir de la tabla
39         }
40     }
41 }

```

```

<terminated> Main [Java Application] C:\Program Files\Java\
Jugador 1. Introduzca una palabra: piano
A qué palabra correspondo el anagrama: paino
Jugador 2. ¿Cuál es el original?
piano
Muy bien ...

```

73. Modificar la Actividad anterior para que el programa indique al jugador 2 cuántas letras coinciden (son iguales y están en la misma posición) entre el texto introducido por él y el original.

```

<terminated> Main [Java Application] C:\Program Files\Java\
Jugador 1. Introduzca una palabra: pamplona
A qué palabra correspondo el anagrama: amaplpno
Jugador 2. ¿Cuál es el original?
amapolan
Letras correctas : 1
Jugador 2. ¿Cuál es el original?
pamplona
Letras correctas : 8
Muy bien ...

```

## 8. Clases

74. Diseñar la clase CuentaCorriente, que almacena los datos: DNI y nombre del titular, así como el saldo. Las operaciones típicas con una cuenta corriente son:

- Crear una cuenta: se necesita el DNI y nombre del titular. El saldo inicial será 0.
- Sacar dinero: el método debe indicar si ha sido posible llevar a cabo la operación, si existe saldo suficiente.
- Ingresar dinero: se incrementa el saldo.
- Mostrar información: muestra la información disponible de la cuenta corriente.

```

<terminated> Main [Java Application] C:
Nombre: Pepe
Dni: 12345678A
Saldo: 700.0 euros
Puedo sacar 700 euros: true
No hay dinero suficiente
Puedo sacar 500 euros: false

```

75. En la clase cuentaCorriente sobrecargar los constructores para poder crear objetos.

- Con el DNI del titular de la cuenta y un saldo inicial.
- Con el DNI, nombre y el saldo inicial.

Escribir un programa que compruebe el funcionamiento de los métodos.

```

Saldo: 700.0 euros
Puedo sacar 700 euros: true
No hay dinero suficiente
Puedo sacar 500 euros: false
Nombre-: Sin asignar
Dni: 98765432-Z
Saldo: 2000.0 euros

```

76. Modificar la visibilidad de la clase CuentaCorriente para que sea visible desde clases externas y la visibilidad de sus atributos para que:

- saldo no sea visible para otras clases.
- nombre sea público para cualquier clase.
- dni solo sea visible por clases vecinas.

Realizar un programa para comprobar la visibilidad de los atributos.

77. Todas las cuentas corrientes con las que se va a trabajar pertenecen al mismo banco. Añadir un atributo que almacene el nombre del banco (que es único) en la clase Cuentacorrente. Diseñar un método que permita recuperar y modificar el nombre del banco que pertenecen todas las cuentas corrientes).

```

<terminated> main [Java Application] C:\Progran
Nombre-: Pepe
Dni: 12345678-A
Saldo: 0.0 euros
Banco: International Java Bank
Nombre-: Pepe
Dni: 12345678-A
Saldo: 0.0 euros
Banco: Banco Central
Nombre-: Pepe
Dni: 12345678-A
Saldo: 0.0 euros
Banco: Caja de Ahorros de Do-While
Nombre-: Ana
Dni: 999999999-E
Saldo: 0.0 euros
Banco: Caja de Ahorros de Do-While

```

78. Existen gestores que administran las cuentas bancarias y atienden a sus propietarios.

Cada cuenta, en caso de tenerlo, cuenta con un único gestor. Diseñar la clase Gestor de la que interesa guardar su nombre, teléfono y el importe máximo autorizado con el que pueden operar. Con respecto a los gestores, existen las siguientes restricciones:

- Un gestor tendrá siempre un nombre y un teléfono.
- Si no se asigna, el importe máximo autorizado por operación será de 10000 euros.
- Un gestor, una vez asignado, no podrá cambiar su número de teléfono. Y todo el mundo podrá consultarlo.

El nombre será público y el importe máximo solo será visible por clases vecinas. Modificar la clase CuentaCorriente para que pueda disponer de un objeto Gestor. Escribir los métodos necesarios.

```
<terminated> main [Java Application] C:\
Información del gestor
Nombre: Antonio González
Teléfono: 666 555 444
Importe máximo: 10000.0euros
Información de la cuenta
Nombre: Pepita
Dni: 1234567 8-A
Saldo: 0.0
Información del gestor
Nombre: Antonio González
Teléfono: 666 555 444
Importe máximo: 10000.0euros
Información de la cuenta
Nombre: Ana
Dni: 98765432-Z
Saldo: 0.0
Cuenta sin gestor
Información de la cuenta
Nombre: Sancho
Dni: 11222333-B
Saldo: 0.0
Información del gestor
Nombre: Bea Rodríguez
Teléfono: 987 543 210
Importe máximo: 12000.0euros
Información de la cuenta
Nombre: Pepita
Dni: 1234567 8-A
Saldo: 0.0
```

79. Escribir un programa que lea por teclado una hora cualquiera y un número  $n$  que represente una cantidad en segundos. El programa mostrara la hora introducida y las  $n$  siguientes, que se diferencian en un segundo. Para ello hemos de diseñar previamente la clase **Hora** que dispone de los atributos hora, minuto y segundo. Los valores de los atributos se controlarán mediante métodos set/get.

```
<terminated> main [Java Application] C:\Prog
Hora :
13
Minuto:
32
segundo:
15
Cuántos segundos quiere mostrar:
42
13:32:15
13:32:16
13:32:17
13:32:18
13:32:19
13:32:20
13:32:21
13:32:22
13:32:23
13:32:24
13:32:25
13:32:26
13:32:27
13:32:28
13:32:29
```

80. Diseñar la clase **Texto** que gestiona una cadena de caracteres con algunas características:

- La cadena de caracteres tendrá una longitud máxima que se especifica en el constructor.
- Permite añadir un carácter al principio o al final, siempre y cuando no se exceda la longitud máxima, es decir, exista espacio disponible.
- Igualmente, permite añadir una cuenta, al principio o al final del texto, siempre y cuando no se rebase el tamaño máximo establecido.
- Es necesario saber cuántas vocales (mayúsculas y minúsculas) hay en el texto.
- Cada objeto de tipo **Texto** tiene que conocer la fecha en la que se creó, así como la fecha y hora de la última modificación efectuada.
- Deberá existir un método que muestre la información que gestiona cada texto.

```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\java.exe
Texto creado el 2022-02-10
Última modificación: 2022-02-10T13:40:33.790123200
Ho;La
Número de vocales: 2
```

81. Definir una clase que permita controlar un sintonizador digital de emisoras FM; concretamente, se desea dotar al controlador de una interfaz que permita subir (up) o bajar (down) la frecuencia (en saltos de 0.5 MHz) y mostrar la frecuencia sintonizada en un momento dado (display). Supondremos que el rango de frecuencias para manejar oscila entre los 80 MHz y los 108 MHz y que, al inicio, el controlador sintonice la frecuencia indicada en el constructor o 80 MHz por defecto. Si durante una operación de subida o bajada se sobrepasa uno de los dos límites, la frecuencia sintonizada debe pasar a ser la del extremo contrario. Escribir un pequeño programa principal para probar su funcionamiento.

```
Problems @ Javadoc Dec
<terminated> Main [Java Application]
Sintonizando : 80.5 MHz
Sintonizando : 107.5 MHz
Sintonizando : 108.0 MHz
```

82. Modelar una casa con muchas bombillas, de forma que cada bombilla se pueda encender o apagar individualmente. Para ello, hacer una clase **Bombilla** con una variable privada que indique si está encendida o apagada, así como un método que nos diga el estado de bombilla concreta. Además, queremos poner un interruptor general, de forma que si este se apaga, todas las bombillas quedan apagadas. Cuando el interruptor general se activa, las bombillas vuelven a estar encendidas o apagadas, según estuvieran antes. Cada bombilla se enciende y se apaga individualmente, pero solo responde que está encendida si su interruptor particular está activado y además hay luz general.

```
terminator: main (2) Java Appli
b1: Encendida
b2: Apagada

Cortamos la luz general
b1: Apagada
b2: Apagada

Activamos la luz general
b1: Encendida
b2: Apagada
```

83. Hemos recibido el encargo de un cliente para definir los paquetes y las clases necesarias (**solo implementar los atributos y los constructores**) para gestionar una empresa ferroviaria, en la que se distinguen dos grandes grupos: el personal y la maquinaria. En el primero se ubican todos los empleados de la empresa, que se clasifican en tres grupos: los mecánicos y los jefes de estación. De cada uno de ellos es necesario guardar:

- Maquinistas: su nombre, DNI, sueldo y el rango que tienen adquirido.
- Mecánicos: su nombre, teléfono (para contactar en caso de urgencia) y en qué especialidad desarrollan su trabajo (esta puede ser: frenos, hidráulica, electricidad o motor).
- Jefes de estación: su nombre, DNI y la fecha en la que fue nombrado jefe de estación.

En la parte de maquinaria podemos encontrar trenes, locomotoras y vagones. De cada uno de ellos hay que considerar:

- Vagones: tienen un número que los identifica, una carga máxima (en kilos), la carga actual y el tipo de mercancía con el que están cargados.
- Locomotoras: disponen de una matrícula (que las identifica), la potencia de sus motores y una antigüedad (año de fabricación). Además, cada locomotora tiene asignado un mecánico que se encarga de su mantenimiento.
- Trenes: están formados por una locomotora y un máximo de 5 vagones. Cada tren tiene asignado un maquinista que es responsable de él.

Todas las clases correspondientes al personal (**Maquinista, Mecánico y JefeEstacion**) serán de uso público. Entre las clases relativas a la maquinaria solo será posible construir, desde clases externas, objetos de tipo **Tren** y de tipo **Locomotora**. La clase **Vagón** será solo visible por sus clases vecinas.



84. Las listas son estructuras dinámicas de datos donde se pueden insertar o eliminar elementos de un determinado tipo sin limitación de espacio.

Implementar la clase **Lista** correspondiente a una lista de números de la clase **Integer**.

Los números se guardarán en una tabla que se redimensionará con las inserciones y eliminaciones, aumentando o disminuyendo la capacidad de la lista según el caso.

Entre los métodos de la clase, se incluirán las siguientes tareas:

- Un constructor que inicialice la tabla con un tamaño 0.
- Obtener el número de elementos insertados en la lista.
- Insertar un número al final de la lista.
- Insertar un número al principio de la lista.
- Insertar un número en un lugar de la lista cuyo índice, que es el de la tabla, se pasa como parámetro.
- Añadir al final de la lista los elementos de otra lista que se pasa como parámetro.
- Eliminar un elemento cuyo índice en la lista se pasa como parámetro.
- Obtener el elemento cuyo índice se pasa como parámetro.
- Buscar un número en la lista, devolviendo el índice del primer lugar donde se encuentre. Si no está, devolverá -1.
- Mostrar los elementos de la lista por consola.

```
<terminated> main [Java Application] C:\Progr
Lista : [4, 5, 6]
Lista : [1, 2, 3, 4, 5, 6]
Lista : [1, 2, 99, 3, 4, 5, 6]
Lista : [1, 2, 3, 4, 5, 6]
3
Lista : [10, 20, 30, 40, 50]
Lista : [1, 2, 3, 4, 5, 6, 12]
```

85. Añadir a la clase **Lista** el método estático:

`Lista concatena(Lista l1, Lista l2)`

que construye y devuelve una lista que contiene, en el mismo orden, una copia de todos los elementos de **l1** y **l2**.

```
<terminated> main [Java Application] C:\Progr
Lista : [1, 1, 2, 3, 10, 20, 30]
```

86. Una pila es una estructura dinámica de datos donde los elementos se insertan (se apilan) y se retiran (se desapilan) siguiendo la norma de que el último que se apila será el primero en desapilarse, como ocurre con una pila de platos. Cuando vamos a retirar un plato de una pila a nadie se le ocurrirá tirar de uno de los de abajo; retiramos (desapilamos) el que está encima de todos, que fue el último en ser apilado. Se llama cima de la pila al último elemento apilado (o al primer

elemento para desapilar). Los metodos fundamentales de una pila son **apilar()** y **desapilar()**.

Implementar la clase **Pila** para números **Integer**, donde se usa una lista (un objeto de la clase Lista implementada en la Actividad 84) para guardar los elementos apilados.

```
<terminated> main (3) [Java A
null
9 8 7 6 5 4 3 2 1 0
```

87. Implementar el método no estático

```
void insertarFinal(int nuevo)
```

que inserta un número entero al final de **tablaEnteros[]**, que es un atributo no estático de la clase **Main**. Escribir un programa que inicialice la tabla con los números del 1 al 10 y después la muestre por consola.

```
<terminated> Main (4) [Java Application] C:\Program
tabla: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

## 9. Herencia

88. Diseñar la clase **Hora**, que representa un instante de tiempo compuesto por la hora (de 0 a 23) y los minutos. Dispone de los métodos:

- **Hora (hora, minuto )**, que construye un objeto con los datos pasados como parámetros.
- **void inc ()**, que incrementa la hora en un minuto.
- **boolean setMinutos (valor)**, que asigna un valor, si es válido, a los minutos.  
Devuelve **true** o **false** según haya sido posible modificar los minutos o no.
- **boolean setHora (valor)**, que asigna un valor, si está comprendido entre 0 y 23, a la hora. Devuelve true o false según haya sido posible cambiar la hora o no.
- **String toString()**, que devuelve un **String** con la representación de la hora.

```
<terminated> main (7) [Java Application]
11:30
12:31
Escriba una hora:
25
La hora no se pudo cambiar
```

```
<terminated> main (2) [Java #
11:30
12:31
Escriba una hora:
23
23:31
```

89. A partir de la clase **Hora** implementar la clase **HoraExacta**, que incluye en la hora los segundos. Además de los métodos heredados de **Hora**, dispondrá de:

- **HoraExacta(hora, minuto, segundo)**, que construye un objeto con los datos pasados como parámetros.
- **setSegundo(valor)**, que asigna, si está comprendido entre 0 y 59, el valor indicado a los segundos.
- **inc()**, que incrementa la hora en un segundo.

```
<terminated> main (2) [Java #
11:15:23
11:15:29
Escriba los segundos:
35
11:15:35
```

90. Añadir a la clase **HoraExacta** un método que compare si dos horas (la invocante y otra pasada como parámetro de entrada al método) son iguales o distintas.

```
<terminate
true
false
```

91. Crear la clase abstracta **Instrumento**, que almacena una tabla las notas musicales de una melodía (dentro de una misma octava). El método **add()** añade nuevas notas musicales. La clase también dispone del método abstracto **interpretar()** que, en cada subclase que herede de **Instrumento**, mostrara por consola las notas musicales según las interprete. Utilizar enumerados para definir las notas musicales.