FUNDAMENTOS DE PROGRAMACIÓN
Ejercicios del TEMA 3. Funciones.

1. Desarrollar un programa que contenga un método de clase (estático) que recibe como argumentos dos números enteros y muestra todos los números enteros comprendidos entre ellos, ambos incluidos.

Se deberán realizar dos llamadas en el programa principal, una para el rango [-10,10] y otra para un rango cuyos límites se solicitarán por por teclado.

a. Iterativo.

b. Recursivo:

2. Realizar un método estático en java que, a partir de un número entero positivo mayor que 1, devuelva el mayor divisor del número distinto de él mismo. Verificar su funcionalidad en el método *main*.

a. Iterativo
b. Recursivo

3. Realizar un programa en java que lea un número entero positivo y, si el número tiene cuatro cifras como máximo escriba su inverso. Para ello se deberá utilizar un método estático que, a partir de un número devuelva su inverso. Por ejemplo, si el número es 345, la función devolverá el valor 543

a. Iterativo

4. Realizar un programa en java que solicite dos números por teclado, y si ambos son positivos devuelva la suma de sus factoriales

Por ejemplo, si se recogen los valores dato1 = 5 y dato2 = 3, el resultado deberá ser:

$$5! + 3! = 5*4*3*2*1 + 3*2*1 = 120 + 6 = 126$$

a. Factorial iterativo:
b. Factorial recursivo:
c. Método *main*

5. Desarrollar los siguientes apartados:

a. Realizar un método estático en java que reciba como argumento un número entero positivo y devuelva la suma de sus cifras.

- Iterativo
- Recursivo

b. Escribir un **programa** que:

- Calcule e imprima los **10 primeros números enteros positivos** que cumplan la condición de que la suma de sus cifras sea mayor que 10, utilizando la función del apartado a.
- Calcule e imprima los números **comprendidos entre 200 y 300** que cumplan que la suma de sus cifras sea 9, utilizando la función del apartado a.

6. Se introducen por teclado una serie de números enteros positivos (el final de la secuencia se marca con un cero). El programa imprimirá los **números perfectos** de la secuencia de entrada. Para resolver el problema se deberán utilizar los métodos estáticos siguientes:

a. Método entero **sumaDivisores** que calcule y devuelva la suma de los divisores de un número excluido él mismo. Por ejemplo:

- Si introducimos: 30 devolverá 42 (42 = 1 + 2 + 3 + 5 + 6 + 10 + 15)
- Si introducimos: 6 devolverá 6 (6 = 1 + 2 + 3)

a. Recursivo:

b. Método booleano **numeroPerfecto** que indique si un número es perfecto o no lo es. Un número es perfecto si es igual a la suma de sus divisores (no se incluye él mismo). Se deberá utilizar la función sumaDivisores del apartado anterior.

Por ejemplo: el número 6 es perfecto porque 6 = 1 + 2 + 3

c. Método main:

7. Escribe un **programa** que imprima **todos los números perfectos** que hay en un intervalo. El intervalo está formado por dos números enteros positivos introducidos por teclado. Se deberán utilizar los métodos desarrollados en el ejercicio anterior.

8. Primer parcial curso 2021-22

Dada una secuencia de puntos (x,y) del plano cartesiano, donde el ultimo valor se indica con (0,0) (centinela), elabore un programa que procese la secuencia de puntos leídos por teclado e imprima cuántos puntos se encuentran a una distancia del centro de coordenadas superior a 3.14 (usando la función anterior), y además imprima cuántos puntos se encuentran en el primer cuadrante (x>0 e y>0).

Además del main, es necesario desarrollar y utilizar las siguientes funciones:

- Método estático *leerDatos* de tipo *double* que, recibiendo como argumento una variable de tipo *Scanner* y el mensaje que se desea mostrar (de tipo *String*), devuelva como resultado un dato de tipo *double*.
- Método estático *distanciaPunto* de tipo *double* que, recibiendo como argumento las coordenadas de un punto (x e y de tipo *double*), determine la distancia de dicho punto al centro de coordenadas. La distancia de un punto (x,y) al centro se determina por la expresión: $\sqrt{x^2 + y^2}$

Para calcular la raíz cuadrada se utiliza la función *Math.sqrt()*.

Por ejemplo, si los puntos leídos por teclado son:

1,4	2,6
4,2	0,9
-3,6	2,5
2,4	1,7
4,2	-3,3
-3,6	2,5
0	0

La salida será:

```
Puntos en Primer Cuadrante =   3
Puntos con distancia al origen mayor a 3.14 =   4
```

9. Primer parcial curso 2021-22

Hacer un programa que recoja de teclado dos números enteros positivos y escriba la lista de los divisores mayores o iguales a 1 que sean comunes a los dos números.

Además del main, es necesario desarrollar y utilizar las siguientes funciones:

- Función estática *leerNúmero*, de tipo *int*, que recibe como argumento un objeto de tipo Scanner, y pide al usuario un número entero positivo. Si se introduce un número negativo lo vuelve a pedir indicando que el número debe ser positivo. Cuando se introduzca un número correcto se devolverá como resultado.
- Función estática *escribirDivisoresComunes* que recibe como argumento dos números enteros positivos, y escribe por pantalla los divisores comunes de ambos números.

10. Examen final curso 2021-22

Se desea calcular de manera aproximada el logaritmo en base B de un número X. Si ambos son números naturales, se puede calcular el entero Z más cercano a dicho logaritmo ($Z \cong \log_B X$) teniendo en cuenta que:

$$B^Z \leq X < B^{Z+1}$$

Si además se cumple que $B^Z = X$ entonces el logaritmo entero Z obtenido es exacto ($Z = \log_B X$).

SE PIDE: Codificar un programa en Java para calcular logaritmos enteros aproximados de números naturales y verificar cuántos son logaritmos exactos. Para ello es necesario desarrollar las siguientes funciones:

- Función *logaritmoEntero* que calcula de manera aproximada el logaritmo entero de un número X en base B por medio de un algoritmo iterativo.
- Función main, que pedirá por teclado un número n, y a continuación una serie de n números naturales. Para cada uno de ellos calculará sus logaritmos enteros en base 2 usando la función *logaritmoEntero*, y al final indicará cuántos logaritmos son exactos.

Ejemplo de funcionamiento:

```
Cantidad de números: 3
Teclee un número: 2
Logaritmo Entero en base 2 de 2 es 1
Teclee un número: 6
Logaritmo Entero en base 2 de 6 es 2
Teclee un número: 32
Logaritmo Entero en base 2 de 32 es 5
Cantidad de Logaritmos Enteros Exactos: 2
```

En la función *main* se puede usar la función *Math.pow* para verificar si el logaritmo entero calculado es exacto.

11. Examen final curso 2021-22

Se dice que dos números enteros positivos A y B son números amigos si la suma de los divisores propios de A es igual a B, y la suma de los divisores propios de B es igual a A. Los divisores propios de un número incluyen el 1 pero no el mismo número.

Por ejemplo, el 1184 y el 1210 son números amigos, dado que:

- Los divisores propios de 1184 son: 1, 2, 4, 8, 16, 32, 37, 74, 148, 296, 592, y la suma de estos da como resultado 1210.
- Los divisores propios de 1210 son: 1, 2, 5, 10, 11, 22, 55, 110, 121, 242, 605, y la suma de estos da como resultado 1184.

Hacer un programa que lea de teclado dos números enteros positivos y compruebe si son amigos o no, mostrando un mensaje indicándolo. Es obligatorio realizar y utilizar las siguientes funciones:

- Función entera *leerNumero*, que recibe como argumento un mensaje a mostrar y un Scanner, y pide un número entero. Si es mayor que 0 lo devuelve como resultado, si es negativo o 0 lo sigue pidiendo reiteradamente.
- Función entera *sumaDivisoresPropios*, que recibe como argumento un número y el primer posible divisor, y de forma recursiva calcula la suma de todos los divisores propios.

Ejemplo de funcionamiento:

```
Introduce el primer número mayor que cero: 1184
Introduce el segundo número mayor que cero: -4
Introduce el segundo número mayor que cero: 1210

1184 y 1210 son amigos
```

12. Examen extraordinario curso 2021-22

Un número natural es un cuadrado perfecto si el mismo se obtiene al elevar al cuadrado cualquier número natural, en otras palabras, podemos verificar que un número natural es cuadrado perfecto si su raíz cuadrada es también un número natural. Considere que dispone de la función *parteDecimal* que dado un valor de tipo *double* devuelve su parte decimal:

```
public static double parteDecimal (double valor)
```

Por ejemplo, si invocamos *double k=parteDecimal (34.56)* devuelve en k el valor 0.56.

Se pide:

1- Desarrollar la función esPerfecto que dado un número *valor* y mediante las invocaciones necesarias de la función *parteDecimal* devuelva si el mismo es un cuadrado perfecto.

2- Desarrollar la función *main*, que permita introducir dos valores naturales, que corresponden a los limites inferior y superior de un rango sobre el cual se desea encontrar todos los números cuadrados perfectos en dicho rango. Los números se deben imprimir a tres por fila.

Un ejemplo de la consola en la ejecución de la función main es:

```
Introduzca el limite Inferior: 15
Introduzca el limite Superior: 70
Numeros Perfectos en el Rango:
   16   25   36
   49   64
Cantidad Total: 5
```

Apartado 1. Función *esPerfecto*.

Apartado 2. Función *main*.

13. Examen extraordinario curso 2021-22

Dados dos enteros $x \geq 1$ y $n \geq 0$, se quiere determinar el valor de x^n empleando el método de cálculo de potencias por cuadrados. Este método es más eficiente que las típicas alternativas basadas en bucles con incremento constante, por tanto, las soluciones basadas en ese tipo de ciclos no serán válidas. El método se resume en la siguiente fórmula:

$$x^n = \begin{cases} (x^2)^{n/2} & \text{si } n \text{ es par,} \\ x \cdot (x^2)^{(n-1)/2} & \text{si } n \text{ es impar.} \end{cases}$$

a) Escribir la función potencia en java que, recibiendo como argumentos los valores de x (de tipo *double*) y n (de tipo *int*), implemente este método de forma recursiva siguiendo la fórmula anterior (no se tendrán en cuenta las soluciones iterativas, o realizadas siguiendo el método de cálculo clásico).

public static double potencia (double x, int n)

b) Realizar una función main que solicite los datos x y n, invoque al método anterior, y a continuación imprima el resultado. Se seguirán solicitando datos hasta que se introduzca como exponente 0. Se deberán usar los siguientes formatos para los datos de tipo *double*:
   - *x* se escribirá en notación decimal usando dos decimales como máximo.
   - el resultado calculado por la función se escribirá con cuatro decimales en notación científica

Ejemplo de ejecución del programa:

```
Introduzca la base: 2
Introduzca el exponente: 10
El valor 2.0 elevado a 10 es:
2,00 ^ 10 = 1,0240e+03
Introduzca la base: 34,879
Introduzca el exponente: 23
El valor 34.879 elevado a 23 es:
34,88 ^ 23 = 3,0128e+35
Introduzca la base: 45,98765
Introduzca el exponente: 3
El valor 45.98765 elevado a 3 es:
45,99 ^ 3 = 9,7258e+04
Introduzca la base: 23,9
Introduzca el exponente: 0
El valor 23.9 elevado a 0 es:
23,90 ^ 0 = 1,0000e+00
Adiós
```

Apartado a. Función *potencia*.
Apartado b. Función *main*.

14. Examen primer parcial curso 2022-23

Ejercicio 2. (3 puntos).

Se desea calcular la suma de los n primeros términos de la sucesión geométrica:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^{n-1}}$$

Hacer un programa que genere aleatoriamente números enteros positivos (n, con valores posibles entre 0 y 20) y escriba la suma de los n primeros términos de la sucesión anterior. Se seguirán mostrando los resultados correspondientes a cada número hasta que n tome el valor 0. Los resultados se deberán mostrar en formato de punto decimal con seis decimales, de acuerdo con el siguiente ejemplo de ejecución:

Además del main, es necesario desarrollar y utilizar las siguientes funciones:

- Función *generarNúmero*, de tipo *int*, que recibe como argumento dos números enteros n1 y n2, y genera un valor aleatorio comprendido entre ambos valores (mayor o igual que n1 y menor que n2 usando la función *Math.random*. La cabecera del método estático *random* de la clase *Math* es:

```
public static double random ();
```

```
// Genera un número aleatorio mayor o igual que 0 y menor que 1
```

- Función *sumaSucesión*, de tipo *double* que recibe como argumento un número entero positivo (n), y devuelve como resultado la suma de los n primeros términos de la sucesión. En esta función se puede usar la función *Math.pow*. La cabecera del método estático *pow* de la clase *Math* es:

```
public static double pow (double a, double b);
// Donde a es la base y b el exponente
```

15. Examen primer parcial curso 2022-23

Ejercicio 3 (3 puntos)
Se define un número Armstrong como aquel en el que la suma de los cubos de sus dígitos es igual al número en sí. Por ejemplo, 153 es un número Armstrong, ya que 153 = (1*1*1) + (5*5*5) + (3*3*3).

Hacer una función esNumeroArmstrong, de tipo boolean, que recibiendo un número entero como argumento devuelva true si ese número es un número Armstrong y false en caso contrario. En esta función se puede usar la función Math.pow descrita en el ejercicio anterior.
Además de esta función, se debe desarrollar un main que pruebe su correcto funcionamiento. Para ello, el programa recogerá de teclado un número entero y escribirá si el número es o no un número Armstrong. Se seguirán solicitando números hasta que se introduzca un número menor o igual que 0. Ejemplo de ejecución:

```
Introduzca un número positivo: 153
153 es un número Armstrong
Introduzca un número positivo: 155
155 no es un número Armstrong
Introduzca un número positivo: 378
378 no es un número Armstrong
Introduzca un número positivo: 371
371 es un número Armstrong
Introduzca un número positivo: 0
```

16. Examen primer parcial curso 2022-23

Ejercicio 4 (3 puntos)

La definición recursiva de la multiplicación de dos números no negativos a y b se deriva de la definición de la multiplicación como una suma abreviada y la propiedad asociativa de la suma. Es decir:
a * b = a + a+ … (b veces) … + a
De modo que si se asocian los sumandos se obtiene que:
a * b = a + [a * (b - 1)].
Así pues, la definición recursiva de la multiplicación es:

$$a * b = \begin{cases} a + (a * (b-1)) \; si \;\; b > 0 \\ \\ 0 \qquad\qquad si \; b = 0 \end{cases}$$

Hacer un programa que recoja de teclado un número positivo y escriba su tabla de multiplicar (del 0 al 10). Se seguirán pidiendo números y mostrando la tabla de multiplicar correspondiente a cada número hasta que se introduzca el valor 0
Además del main, es necesario desarrollar y utilizar las siguientes funciones:

- Función leerNúmero, de tipo int, que recibe como argumento un objeto de tipo Scanner, y pide al usuario un número entero positivo. Si se introduce un número negativo lo vuelve a pedir indicando que el número debe ser positivo. Cuando se introduzca un número correcto se devolverá como resultado.
- Función multiplicarRecursivo, de tipo int que recibe como argumento dos números enteros positivos, y devuelve como resultado la multiplicación recursiva de ambos números. No se considerarán las soluciones no recursivas.

17. Examen primer parcial Global curso 2022-23

Ejercicio 4 (3 puntos)
La definición recursiva del método ruso de multiplicación es:

$$a * b = \begin{cases} b \;\; si \;\; a = 1 \\ b + \left(\frac{a}{2} * (2 * b)\right) \;\; si\ a\ es\ impar\ distinto\ de\ 1 \\ \frac{a}{2} * (2 * b) \qquad\qquad si\ a\ es\ par \end{cases}$$

Realizar la función *multiplicacionRusa*, de tipo *int,* que recibe como argumentos dos números enteros positivos, y devuelve como resultado la multiplicación recursiva de ambos números utilizando el método ruso de multiplicación. No se considerarán las soluciones no recursivas o utilizando métodos diferentes del planteado.