EJERCICIOS PROPUESTOS

Ejercicio 1

Dibujar el **diagrama de flujo** del siguiente código y calcular su **complejidad ciclomática**. Realizar la **prueba de cubrimiento**, la **prueba de condiciones** y la **prueba de bucles**, suponiendo que los valores iniciales no pueden ser menores de 0.

```
i = i_ini;
while (i <= 6) {
  System.out.print("Ficha: ");
  j = j_ini;
  while (j <= i) {
    System.out.print("|" + i + ":" + j + "| ");
    j++;
  }
  System.out.print("\n");
  i++;
}
```

REPASO: ¿Qué muestra por pantalla este código si **i_ini=1** y **j_ini=1**?

Ejercicio 2

Considérese una aplicación bancaria, donde el usuario puede conectarse al banco por Internet y realizar una serie de operaciones bancarias. Una vez accedido al banco con las consiguientes medidas de seguridad (clave de acceso y demás), la información de entrada del procedimiento que gestiona las operaciones concretas a realizar por el usuario requiere la siguiente entrada:

Especificación

- **Código del banco**. En blanco o número de tres dígitos. En este último caso, el primero de los números tiene que ser mayor que 1.
- **Código de sucursal**. Un número de cuatro dígitos. El primero de ellos mayor de 0.
- **Número de cuenta**. Número de cinco dígitos.
- **Clave personal**. Valor alfanumérico de cinco posiciones.
- **Orden**. Este valor se introducirá según la orden que se desee realizar. Puede estar en blanco o ser una de las dos cadenas siguientes:
    a. “Talonario”. El usuario recibirá un talonario de cheques
    b. “Movimientos”. El usuario recibirá los movimientos del mes en curso.
    c. En blanco. El usuario recibirá los dos documentos.

Realizar la **prueba de clases de equivalencia de datos** y la **prueba de valores límite**.

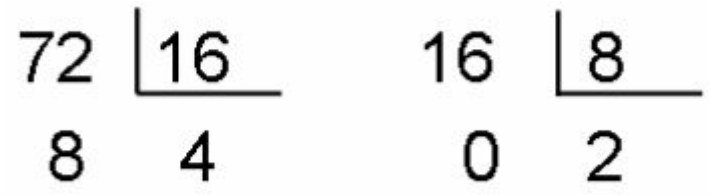
Ejercicio 3

Realizar las pruebas mediante JUnit de la siguiente clase **Mcd**:

- La clase **Mcd.java** tiene dos atributos de tipo int, dos setters para dichos atributos y un método llamado **calculoMcd** que devuelve el Máximo Común Divisor de los dos atributos por el algoritmo de Euclides.
- La clase **TestMcd.java** instancia un objeto de la clase **Mcd** y comprueba mediante un método llamado **testMcd** el funcionamiento correcto del método **calculoMcd**.
- Se debe probar mediante los *assertEquals* adecuados que funciona, es decir una prueba válida (por ejemplo que el Mcd de 72 y 16 vale 8) y otra no válida (por ejemplo que el Mcd de 72 y 16 vale 4).

Algoritmo de Euclides:

a. Se divide el número mayor por el menor.
b. Si la división es exacta, el MCD es el divisor.
c. Si la división no es exacta, se divide el divisor por el resto hasta obtener una división exacta
d. El último divisor es el MCD.



MCD(72, 16) = 8

Ejercicios prácticos 6 y 7 de libro (página 69)

6. Crea una clase de test en JUnit.

Dimas quiere realizar una clase que convierta grados Fahrenheit a Celsius, y viceversa. Conoce la fórmula y quiere implementar dos métodos en la clase fahrenheittocelsius() y celsiustofahrenheit() que conviertan grados de una unidad a otra, y viceversa.

Además, quiere testear que convierten los métodos correctamente los valores –5, 0, 15 y 32.

¿Puedes ayudarlo a crear la clase y el test con JUnit 4?

7. Crea un conjunto de test o suite de test en JUnit.

Dimas se ha animado y, dentro del mismo proyecto, quiere realizar una clase que convierta números romanos a decimales, y viceversa (métodos roman2dec y dec2roman). Quiere realizar un par de test con JUnit para que testee los números XXI y 2016.

Además, cree que puede ser útil una segunda clase que convierta dólares a euros, y viceversa (métodos dollar2euro y euro2dollar), también con sus test en JUnit que testeen 10,5 dólares y 20,30 euros.

Por último, para no realizar los test por separado, lo que se propone es crear una *suite* de pruebas que testee las tres clases.