PRÁCTICA PRUEBAS IDM

Ejercicio 1 - Cálculo de años bisiestos

- **Apartado 1.** Se analizará la clase `Bisiestos`, donde uno de sus métodos llamado `esBisiesto`, devuelve true o false si el año que le proporcionamos es o no bisiesto.

- **Apartado 2.** Tenemos 1 parámetro:

    - Tenemos un único Parámetro `año`, de tipo *int*. Este será el año que queremos comprobar si es bisiesto o no.

- **Apartado 3.** Tendremos 3 caracterizaciones:

    - C1: año respecto del 0.
        - b1: año negativo
        - b2: año 0
        - b3: año positivo
    - C2: año múltiplo de 4.
        - b1: True
        - b2: False
    - C3: año múltiplo de 100.
        - b1: True
        - b2: False
    - C4: año múltiplo de 400.
        - b1: True
        - b2: False

- **Apartado 4.** Valores adecuados a cada bloque según criterios de cobertura:

    - C1: -200 | 0 | 2018
    - C2: 4 | 5
    - C3: 100 | 101
    - C4: 400 | 401

- **Práctica 2.a)** Enumerad los requisitos de prueba/test de acuerdo a PPC:

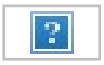
    - RT: [1, 2, 3], [1, 2, 4, 5, 8], [1, 2, 4, 6, 7, 8], [1, 2, 4, 6, 8]

- **Práctica 2.b)** Enumerad caminos de prueba/test de PPC:

- Caminos de prueba: [1, 2, 3], [1, 2, 4, 5, 8], [1, 2, 4, 6, 7, 8], [1, 2, 4, 6, 8]

- **Práctica 2.c)** Diagrama + comentarios en tests:

  - `testForNegativeYear` : Caminos recorridos: [1,2,3].
  - `testForZeroYear` : Caminos recorridos: [1,2,4,5,8].
  - `testMult4` : Caminos recorridos: [1,2,4,6,7,8].
  - `testMult100` : Caminos recorridos: [1,2,4,6,8].
  - `testMult400` : Caminos recorridos: [1,2,4,6,7,8].
  - Añadimos `testForNoBisiesto` : Caminos recorridos: [1,2,4,6,8].



Ejercicio 2 - Conversión de números romanos a base 10

- **Apartado 1.** Se analizará la clase `RomanNumeral` , la cual contiene un método `convierte` que se encarga de transformar el *String* proporcionado en un número en base 10 (int).

- **Apartado 2.** Tenemos un único parámetro `s` de tipo *String*, que será el número romano que queremos convertir a base diez.

- **Apartado 3.** Tendremos 3 caracterizaciones:

  - C1: null (string vacío).
  - C2: string que sea romano.
  - C3: string que NO sea romano.

- **Apartado 4.** Valores adecuados a cada bloque según criterios de cobertura:

  - C1: string vacío | XVII | HJK | MMJ | IIIII

- **Práctica 2.a)** Enumerad los requisitos de prueba/test de acuerdo a PPC:

  - Por simplicidad, solo se hace el grafo del método `convierte` :
    - [1,2]
    - [1,3,4,6]
    - [1,3,4,5,7], [1,3,4,5,8], [1,3,4,5,9], [1,3,4,5,10],[1,3,4,5,11],[1,3,4,5,12], [1,3,4,5,13], [1,3,4,5,14]
    - [4,5,7,4], [4,5,8,4], [4,5,9,4], [4,5,10,4], [4,5,11,4], [4,5,12,4], [4,5,13,4]
    - [5,7,4,5], [5,8,4,5], [5,9,4,5], [5,10,4,5], [5,11,4,5], [5,12,4,5], [5,13,4,5]

        - [5,7,4,6], [5,8,4,6], [5,9,4,6], [5,10,4,6], [5,11,4,6], [5,12,4,6], [5,13,4,6]
        - [7,4,5,7], [7,4,5,8], [7,4,5,9], [7,4,5,10], [7,4,5,11], [7,4,5,12], [7,4,5,13], [7,4,5,14]
        - [8,4,5,7], [8,4,5,8], [8,4,5,9], [8,4,5,10], [8,4,5,11], [8,4,5,12], [8,4,5,13], [8,4,5,14]
        - [9,4,5,7], [9,4,5,8], [9,4,5,9], [9,4,5,10], [9,4,5,11], [9,4,5,12], [9,4,5,13], [9,4,5,14]
        - [10,4,5,7], [10,4,5,8], [10,4,5,9], [10,4,5,10], [10,4,5,11], [10,4,5,12], [10,4,5,13], [10,4,5,14]
        - [11,4,5,7], [11,4,5,8], [11,4,5,9], [11,4,5,10], [11,4,5,11], [11,4,5,12],[11,4,5,13], [11,4,5,14]
        - [12,4,5,7], [12,4,5,8], [12,4,5,9], [12,4,5,10], [12,4,5,11], [12,4,5,12],[12,4,5,13], [12,4,5,14]
        - [13,4,5,7], [13,4,5,8], [13,4,5,9], [13,4,5,10], [13,4,5,11], [13,4,5,12],[13,4,5,13], [13,4,5,14]

- **Práctica 2.b)** Enumerad caminos de prueba/test de PPC:

    - Caminos de prueba:
        - [1,2]
        - [1,3,4,6]
        - [1,3,4,5,14]
        - [1,3,4,5,7,4,6], [1,3,4,5,8,4,6], [1,3,4,5,9,4,6], [1,3,4,5,10,4,6], [1,3,4,5,11,4,6], [1,3,4,5,12,4,6], [1,3,4,5,13,4,6] (No hacen falta evaluarlos. REDUNDANTES, aparecen posteriormente).
        - [1,3,4,5,7,4,5,7,4,6], [1,3,4,5,7,4,5,8,4,6], [1,3,4,5,7,4,5,9,4,6], [1,3,4,5,7,4,5,10,4,6], [1,3,4,5,7,4,5,11,4,6], [1,3,4,5,7,4,5,12,4,6], [1,3,4,5,7,4,5,13,4,6], [1,3,4,5,7,4,5,14]
        - [1,3,4,5,8,4,5,7,4,6], [1,3,4,5,8,4,5,8,4,6], [1,3,4,5,8,4,5,9,4,6], [1,3,4,5,8,4,5,10,4,6], [1,3,4,5,8,4,5,11,4,6], [1,3,4,5,8,4,5,12,4,6], [1,3,4,5,8,4,5,13,4,6], [1,3,4,5,8,4,5,14]
        - [1,3,4,5,9,4,5,7,4,6], [1,3,4,5,9,4,5,8,4,6], [1,3,4,5,9,4,5,9,4,6], [1,3,4,5,9,4,5,10,4,6], [1,3,4,5,9,4,5,11,4,6], [1,3,4,5,9,4,5,12,4,6], [1,3,4,5,9,4,5,13,4,6], [1,3,4,5,9,4,5,14]
        - [1,3,4,5,10,4,5,7,4,6], [1,3,4,5,10,4,5,8,4,6], [1,3,4,5,10,4,5,9,4,6], [1,3,4,5,10,4,5,10,4,6], [1,3,4,5,10,4,5,11,4,6], [1,3,4,5,10,4,5,12,4,6], [1,3,4,5,10,4,5,13,4,6], [1,3,4,5,10,4,5,14]
        - [1,3,4,5,11,4,5,7,4,6], [1,3,4,5,11,4,5,8,4,6], [1,3,4,5,11,4,5,9,4,6], [1,3,4,5,11,4,5,10,4,6], [1,3,4,5,11,4,5,11,4,6], [1,3,4,5,11,4,5,12,4,6], [1,3,4,5,11,4,5,13,4,6], [1,3,4,5,11,4,5,14]
        - [1,3,4,5,12,4,5,7,4,6], [1,3,4,5,12,4,5,8,4,6], [1,3,4,5,12,4,5,9,4,6], [1,3,4,5,12,4,5,10,4,6], [1,3,4,5,12,4,5,11,4,6], [1,3,4,5,12,4,5,12,4,6], [1,3,4,5,12,4,5,13,4,6], [1,3,4,5,12,4,5,14]
        - [1,3,4,5,13,4,5,7,4,6], [1,3,4,5,13,4,5,8,4,6], [1,3,4,5,13,4,5,9,4,6], [1,3,4,5,13,4,5,10,4,6], [1,3,4,5,13,4,5,11,4,6], [1,3,4,5,13,4,5,12,4,6],

[1,3,4,5,13,4,5,13,4,6], [1,3,4,5,13,4,5,14]

- **Práctica 2.c)** Diagrama + comentarios en tests:

  - `testForNullString` : Caminos recorridos: [1,2].
  - `testForNoRoman` : Caminos recorridos: [1,3,4,5,14].
  - `testForRoman` : Caminos recorridos: [1,3,4,5,13,4,5,13,4,5,12,4,5,11,4,6].
    - El camino [1,3,4,6] es inviable que sea recorrido, por lo que no habrá test para él. (Es inviable porque en la primera pasada del bucle for, la variable x no puede tomar valor negativo, como mínimo tomará valor 0 si el argumento de entrada al método es un String de un solo caracter. Recordemos que de ser el argumento de entrada un String vacío, el camino que seguiría sería el [1,2]).
    - Añadimos `testPrueba1` : Caminos recorridos: [1,3,4,5,7,4,5,7,4,6] ***(I)***
    - Añadimos `testPrueba2` : Caminos recorridos: [1,3,4,5,7,4,5,14] ***(II)***
    - Hemos añadido solo dos casos particulares del `testForRoman` (uno que ejecuta dos veces el bucle switch-case y hace el `return` ***(I)*** ; y otro que ejecuta el bucle dos veces pero acabando entrando por el `default` del switch, lanzando la excepción `IllegalArgumentException` ***(II)***), ya que entendemos que, al ser implementados por el mismo switch-case, si funcionan estos dos, han de funcionar todos.

Ejercicio 3 - Embotelladora

- **Apartado 1.** Se analizará la clase `Embotelladora` , la cual contiene un método llamado `calculaBotellasPequenas` , que debe devolver el número de botellas pequeñas, medianas y grandes necesarias para embotellar un número total de litros, minimizando el número de botellas grandes.

- **Apartado 2.** Hay 3 parámetros:

  - `pequeñas` : tipo int. Será el número de botellas de 1 litro disponibles en el almacén.
  - `grande` : tipo int. Será el número de botellas de 5 litros disponibles en el almacén.
  - `total` : tipo int. Número total de litros que queremos embotellar.

- **Apartado 3.** Tendremos en esta ocasión 3 caracterizaciones:

  - C1: Cantidad de botellas.
    - b1: ninguna (pequeñas y grandes son 0).

        - b2: solo pequeñas disponibles.
        - b3: solo grandes disponibles.
        - b4: ambas disponibles.
        - b5: pequeñas negativas.
        - b6: grandes negativas.
    - C2: Mayor cantidad de botellas.
        - b1: misma cantidad pequeñas que grandes
        - b2: mayor cantidad de pequeñas que de grandes
        - b3: mayor cantidad de grandes que de pequeñas
    - C3: Litros totales con respecto al 0.
        - b1: litros negativos
        - b2: litros 0
        - b3: litros positivos
    - C4: Abastecimiento dado litros totales y botellas
        - b1: abastecemos con botellas justas
        - b2: no abastecemos
        - b3: abastecemos y nos sobran botellas

- **Apartado 4.**

    - C1: 0,0,10 | 10,0,10 | 0,10,10 | 1,2,10 | -1,2,10 | 1,-2,10

    - C2: 3,3,8 | 3,1,8 | 1,3,8

    - C3: 10,0,-10 | 10,0,0 | 10,10,10

    - C4: 0,2,10 | 2,0,10 | 5,2,13

    - **Práctica 2.a)** Enumerad los requisitos de prueba/test de acuerdo a PPC:
      RT: [1, 2], [1, 3, 4, 6], [1, 3, 4, 7], [1, 3, 5, 8, 10], [1, 3, 5, 8, 11], [1, 3, 5, 9, 12], [1, 3, 5, 9, 13, 14], [1, 3, 5, 9, 13, 15]

    - **Práctica 2.b)** Enumerad caminos de prueba/test de PPC:

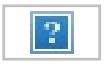
    - Caminos de prueba: [1, 2], [1, 3, 4, 6], [1, 3, 4, 7], [1, 3, 5, 8, 10], [1, 3, 5, 8, 11], [1, 3, 5, 9, 12], [1, 3, 5, 9, 13, 14], [1, 3, 5, 9, 13, 15]

    - **Práctica 2.c)** Diagrama + comentarios en tests:

    - `testForNegativePequenas` : Caminos recorridos: [1,2].

    - `testForNegativeGrandes` : Caminos recorridos: [1,2].

- `testForNegativeTotal` : Caminos recorridos: [1,2].
- `testForZeroBotellas` : Caminos recorridos: [1,2].
- `testForZeroTotal` : Caminos recorridos: [1,2].
- `testForSoloPequeñas` : Caminos recorridos: [1,3,5,8,10].
- `testForSoloGrandes` : Caminos recorridos: [1,3,4,5].
- `testForAmbasBotellas` : Caminos recorridos: [1,3,5,9,13,14].
- `testForIgualCantidad` : Caminos recorridos: [1,3,5,9,13,14].
- `testForMasPequenas` : Caminos recorridos: [1,3,5,9,13,14].
- `testForAbastecemosJustas` : Caminos recorridos: [1,3,5,9,13,14].
- `testForNoAbastecemos` : Caminos recorridos: [1,3,5,9,13,15].
- `testForAbastacemosSobra` : Caminos recorridos: [1,3,5,9,13,14].
- Con los tests que teníamos diseñados, falta por recorrer caminos: [1, 3, 4, 6], [1, 3, 4, 7], [1, 3, 5, 8, 11] y [1, 3, 5, 9, 12], por lo que:
- Añadimos `testPrueba1` : Caminos recorridos: [1,3,4,6].
- Añadimos `testPrueba2` : Caminos recorridos: [1,3,4,7].
- Añadimos `testPrueba3` : Caminos recorridos: [1,3,5,8,11].
- Añadimos `testPrueba4` : Caminos recorridos: [1,3,5,9,12].



Ejercicio 4 - Descuento Black Friday

- **Apartado 1.** Se analizará la clase `DescuentoBlackFriday` , la cual contiene un método llamado `PrecioFinal` , que devuelve un precio final un 30% menor que el precio original proporcionado, en el caso de que el día actual sea 23 de noviembre.

- **Apartado 2.** Un único parámetro `precioOriginal` de tipo *double*. Será el precio original de un producto marcado en su etiqueta, sin descuento.

- **Apartado 3.** Hay 1 caracterización:

  - C1: precio original respecto del 0:
    - b1: precio negativo
    - b2: precio 0
    - b3: precio positivo

- **Apartado 4.** Da valores adecuados a cada bloque según criterios de cobertura:

  - C1: -2.0 | 0.0 | 25.50

- **Práctica 2.a)** Enumerad los requisitos de prueba/test de acuerdo a PPC:
  RT: [1, 2], [1, 3, 4], [1, 3, 5]

- **Práctica 2.b)** Enumerad caminos de prueba/test de PPC:

  - Caminos de prueba: [1, 2], [1, 3, 4], [1, 3, 5]

- **Práctica 2.c)** Diagrama + comentarios en tests:

  - `testForNegativePrice` : Caminos recorridos: [1,2].
  - `testForZeroPrice` : Caminos recorridos: [1,2].
  - `testForValidPrice` : Caminos recorridos: [1,3,4].
  - Añadimos `testForValidPriceDesc` : Caminos recorridos: [1,3,5].