1) En un congreso de Aritmética se decidieron a organizar sus expresiones matemáticas. Las expresiones con las que trabajan son expresiones básicas, que incluyen suma, resta, producto, división, potencia y raíz. Por ejemplo:

- ((2+3)*5)
- $(5+5)^{(3-1)}$
- ((7+8) / (5-3))

Luego de muchas discusiones se pusieron de acuerdo en que es lo que querían resolver con la organización:

- Obtener el valor de la expresión: en los ejemplos serian 25; 100; 7,5 respectivamente
- Obtener sólo los números de la expresión: en los ejemplos serian [2,3,5]; [5,5,3,1]; [7,8,5,3]
- Obtener sólo los operadores de la expresión: en los ejemplos serian [+,*]; [+,pot, -]; [+,/,-]
- Imprimir fácilmente las expresiones, por ello quieren que el método toString() devuelva el String con la expresión propiamente dicha incluyendo los paréntesis “((2+3)*5)”; “(5+5)(3-1)”;“((7+8) / (5-3))”.

A uno de los matemáticos de la reunión le molestaba la posibilidad de que todo fallara al calcular la expresión, si la misma generaba un error matemático. Decidieron que **no se pueda dividir por cero** y que **nunca se le pida a un número negativo la raíz**. Cuando el cálculo de una expresión genera un error matemático, se debe devolver en el reemplazo del mismo el número “-23”. (Si bien este número puede variar, debe ser el mismo para todas las expresiones)

Lo revolucionario de la solución fue el cálculo inverso de la expresión, es decir si era una suma, el cálculo inverso hace la resta, si era un producto la división, si era una potencia la raíz, y así con cada operador. Por ejemplo en el primer caso el cálculo opuesto seria ((2-3)/5) es decir -0,2, en el segundo ejemplo seria $\sqrt[3+1]{(5-5)}$ es decir 0 y así sucesivamente.

Como en el congreso van a tener un listado con todas las expresiones aritméticas de los presentes, quieren poder devolver el listado ordenado por distintos aspectos (no pudieron ponerse de acuerdo); unos quieren ordenarlos por el **valor del cálculo** de la expresión de forma ascendente, otros por la **cantidad de operadores** de forma descendente, y otros por la **comparación del String** que representa la expresión.

Implementar la solución al problema del organizador de expresiones en JAVA, tener en cuenta todos los conceptos vistos en clase. Prestar especial atención a la abstracción de funcionalidad.

Implementar en un main la construcción de los tres ejemplos anteriores.

2) ¿Aplicó algún patrón de diseño en la solución propuesta? ¿Cuál o cuáles? En caso de responder afirmativamente, explique brevemente cada patrón.

3) Describa el Patron Decorator y de un ejemplo práctico del mismo