Experta ART

Prueba técnica

Para completar la prueba técnica, lea completamente las pautas establecidas en este documento.

Objetivo

Esta prueba es una excusa para que muestre lo que sabe. No se trata de resolver un problema, sino de cómo lo resuelve, y las decisiones que tome para resolverlo.

Preguntas

Responda y envíe en un documento pdf, sus respuestas a las siguientes preguntas.

1. ¿En qué consiste el principio de buen diseño *Dependency Inversion*?
2. ¿Qué características tiene un buen código o código limpio?

Codificación

El enunciado del problema a resolver, contempla un problema que deberá analizar y resolver de forma incremental. Deberá crear un repositorio git (por ejemplo en github) al que podamos acceder y crear un tag para formalizar la versión de la entrega. No se aceptarán entregas en otros formatos. Para resolver la prueba, se puede usar cualquiera de los siguientes lenguajes como base: Java, Python, Php o Ruby.
Puede incorporar en la entrega cualquier documentación que considere relevante en adición al código, y deberá dejar asentadas las hipótesis que tome para resolver el problema.

Enunciado

Idea General

Dada una matriz de 3 dimensiones inicializada con todas sus celdas en 0, crear una clase que la maneje, que soporte dos operaciones sobre dicha matriz:

- La actualización de una celda: **UPDATE**

```
UPDATE x y z W
```

Actualiza el valor de la celda (x,y,z) al valor W.

- La consulta del total acumulado de los valores comprendidos dentro de un rango: **QUERY**

```
QUERY x1 y1 z1 x2 y2 z2
```

Calcula la suma de todos los valores de las celdas comprendidas entre los puntos (x1, y1, z1) y (x2, y2, z2) inclusive.

Interfaz

La matriz cúbica a trabajar, se deberá tomar un parámetro para su construcción que indicará el tamaño del lado de dicha matriz.
La clase a desarrollar, debe contar con un método *execute* que recibirá una cadena como las antes descriptas:

- ❏ UPDATE x y z W
- ❏ QUERY x1 y1 z1 x2 y2 z2

Por ejemplo

```
UPDATE 2 1 4 25
```

```
UPDATE 1 1 3 14
```

```
QUERY 1 3 2 4 3 3
```

```
QUERY 2 1 1 4 6 2
```

Output

El resultado a producir por el método *execute* deberá incluir un estado de la ejecución, uno de los siguientes:

- SUCCESS
- ERROR

Y en caso de corresponder, un valor del mismo tipo que el W antes mencionado. Cómo lo devuelve, queda a su criterio.

Constraints

Siendo N el tamaño de la matriz

- 1 <= N <= 100
- 1 <= x1 <= x2 <= N
- 1 <= y1 <= y2 <= N
- 1 <= z1 <= z2 <= N
- 1 <= x,y,z <= N

Siendo W el valor de una celda

- -126 <= W <= 126

Ejemplo

```
UPDATE 2 2 2 4     -> SUCCESS
QUERY 1 1 1 3 3 3  -> SUCCESS / 4
UPDATE 1 1 1 23    -> SUCCESS
QUERY 2 2 2 4 4 4  -> SUCCESS / 4
QUERY 1 1 1 3 3 3  -> SUCCESS / 4
UPDATE 2 2 2 1     -> SUCCESS
QUERY 1 1 1 1 1 1  -> SUCCESS / 23
QUERY 1 1 1 2 2 2  -> SUCCESS / 24
QUERY 2 2 2 2 2 2  -> SUCCESS / 1
```

Ejecución

No es necesario que implemente una interfaz de usuario para probar el código, pero es obligatorio la creación de tests unitarios que validen el comportamiento de la clase objetivo de esta prueba.
Indique qué dependencias se requieren y cómo se deben ejecutar los tests para probar el código, según la *make-tool* (maven, ant, gradle, etc) que elija. Si decide incluir una interfaz de usuario, indique cómo ejecutarla.

Recomendaciones generales

Se espera que use un diseño orientado a objetos de acuerdo a un modelado del problema marco, que respete las buenas prácticas de programación y los principios de diseño de código.