Ejercicio 3 - Preguntas para hacer después del ejercicio

Aporte de los mensajes de DD

En un double dispatch (DD), ¿qué información aporta cada uno de los dos llamados?

Nos aporta un resultado distinto para cada subclase. En nuestro caso, para cada operación en la que hagamos un double dispatch vamos a tener un método para la subclase Fraccion y otro método con el mismo nombre para la subclase Entero. Por ejemplo: Para la operación suma, no es lo mismo sumar un entero con un entero que un entero con una fracción, por lo que tendremos implementaciones distintas para cada caso.

Lógica de instanciado

Con lo que vieron y saben hasta ahora, ¿donde les parece mejor tener la lógica de cómo instanciar un objeto? ¿por qué? ¿Y si se crea ese objeto desde diferentes lugares y de diferentes formas? ¿cómo lo resuelven?

Conviene tener la lógica de como instanciar objetos en la categoría de métodos de instancia, esto se debe a que las instancias (colaboradores internos) que creemos, serán utilizadas en los respectivos métodos que se encuentran en esa categoría. En la categoría de métodos de clase tendremos los métodos correspondientes a la inicialización y a chequeos.

Si se crea ese objeto en diferentes lugares y de diferentes formas, lo instanciamos en la categoría de instance y chequeamos que se pueda pertenecer a la clase (inicializarse correctamente) en la categoría de class.

Nombres de las categorías de métodos

Con lo que vieron y trabajaron hasta ahora, ¿qué criterio están usando para categorizar métodos?

Categorizamos los métodos teniendo en cuenta el propósito del método (el qué), poniendo buenos nombres que permitan clasificarlos de una forma comprensible para

el usuario y según la similitud que tienen los métodos entre sí.

Subclass Responsibility

Si todas las subclases saben responder un mismo mensaje, ¿por qué ponemos ese mensaje sólo con un “self subclassResponsibility” en la superclase? ¿para qué sirve?

Self subclassResponsibility le permite a la clase delegar una tarea o rol en sus subclases mediante la implementación de mensajes abstractos, cumpliendo con las buenas prácticas y atacando el problema del polimorfismo con un mensaje claro que entiende la superclase en cuestión.

No rompas

¿Por qué está mal/qué problemas trae romper encapsulamiento?

El problema que trae romper encapsulamiento es que estamos permitiendo que ciertos objetos se conozcan más de lo que deberían conocerse, lo que nos impide tener una asignación correcta de responsabilidades y rompe con las buenas prácticas de programación orientada a objetos.