Primera práctica

Diseño y Análisis de Algoritmos

Adrián Grassin Luis (alu01013494809@ull.edu.es)

30 de enero de 2024

Índice general

1. Principios Solid

Los principios SOLID son un conjunto de reglas que nos ayudan a diseñar una buena arquitectura de software. Estos principios fueron acuñados por Robert C. Martin (Uncle Bob) a principios de la década del 2000.

- **S**ingle Responsability Principle
- **O**pen/Closed Principle
- **L**iskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion Principle

Single Responsability Principle

El objetivo del principio de responsabilidad única se basa en que cada clase módulo sea responsable de una sola parte de la funcionalidad proporcionada por el software y esta responsabilidad debe estar encapsulada en su totalidad por la clase. Es decir, que si una clase tiene más de una responsabilidad, esta se debería subdividir en clases más pequeñas, cada una con su única responsabilidad haciendo que la lógica se encuentre separada y sea más fácil de mantener.

2. Patrón de estrategia

El patrón de estrategia es un patrón de diseño de software que permite seleccionar un algoritmo en tiempo de ejecución. Esto se logra encapsulando cada algoritmo (estrategia) en una clase separada normalmente siendo esta una clase hija de una interfaz o una clase abstracta de la cual se heredan los métodos que implementan el algoritmo.

Un ejemplo de este patrón podría verse en una práctica de la asignatura de *Algoritmos y Estructuras de Datos Avanzadas* de hace unos años en la que se implementaba un *juego de la vida de Conway* con distintas reglas de supervivencia, nacimiento y varios tipos de mundo. Al ejecutar el programa se podía elegir entre distintas reglas de supervivencia y nacimiento y distintos tipos de mundo. mediante el uso de punteros en el código a instancias hijas de la clase que implementaba la interfaz de reglas de supervivencia y nacimiento. Un ejemplo concreto podría ser el cálculo de las celulas adyacentes de cada célula, pudiendo elegir si el algoritmo de cálculo de adyacencia que funcionase con un mundo “sin bordes” o con un mundo finito, cambiando así la estrategia en la que se comprobaba el entorno de la celula.

Referencias

1. https://www.freecodecamp.org/espanol/news/los-principios-solid-explicados-en-espanol/
2. https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
3. https://www.freecodecamp.org/news/a-beginners-guide-to-the-strategy-design-pattern/
4. https://refactoring.guru/es/design-patterns/strategy