Sergio Martín Santana
Alu0100611625

PRÁCTICA 4: EL PATRÓN MÉTODO PLANTILLA

OBJETIVO

Se pretende desarrollar un framework para el desarrollo de algoritmos de la familia divide y vencerás. Nos apoyaremos para ello en el patrón Método Plantilla. El método plantilla para esta familia puede venir definido por dos tipos auxiliares, Problem y Solution que representan el problema y la solución y por un método plantilla del que se derivarán los algoritmos deseados.

PATRÓN PLANTILLA

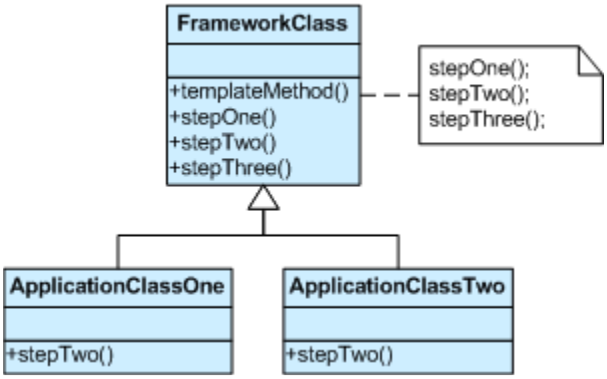
INTRODUCCIÓN

El objetivo de este patrón de diseño es definir en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos.

PROBLEMAS QUE SOLUCIONA

- Para implementar las partes de un algoritmo que no cambian y dejar que sean las subclases quienes implementen el comportamiento que puede variar
- Refactorizar para Generalizar. Cuando el comportamiento repetido de varias subclases debería factorizarse y ser localizado en una clase común para evitar el código duplicado
    - Identificar diferencias en el código existente
    - Separar las diferencias en nuevas operaciones
- Para controlar las extensiones de las subclases. Se puede definir el método de manera que llame a operaciones en determinados puntos, permitiendo así las extensiones sólo en esos puntos.

ESTRUCTURA



Sergio Martín Santana
Alu0100611625

PATRONES RELACIONADOS

- Factory Method: Se invocan frecuentemente desde métodos plantilla.
- Strategy: Los métodos plantilla utilizan la herencia para modificar una parte de un algoritmo. Las estrategias utilizan la delegación para variar el algoritmo completo.

DESARROLLO DEL EJEMPLO IMPLEMENTADO

ESTRUCTURA CREADA

Los ejemplos creados son homónimos entre sí en cuanto a estructura se refiere, ambos cuentan con dos clases que implementan las interfaces (Problem y Solution), y una clase que hereda de otra abstracta que contiene el método plantilla.

Esta clase principal, "FactDivConq" y "FiboDivConq", contiene todos los métodos abstractos redefinidos, en cada uno de ellos implementaremos las características de nuestro algoritmo para solucionar el problema, además deberemos de realizar casteos cada vez que utilicemos un problema o solución, a la clase problema/solución del problema en cuestión.

Esta estructura nos facilita la resolución de infinidad de nuevos problemas de divide y vencerás, únicamente implementando tres nuevas clases: <nombre>Problem.java, <nombre>Solution.java y <nombre>DivConq.java. Y modificando el fichero de Inicio, que contiene la GUI.

DIAGRAMA UML