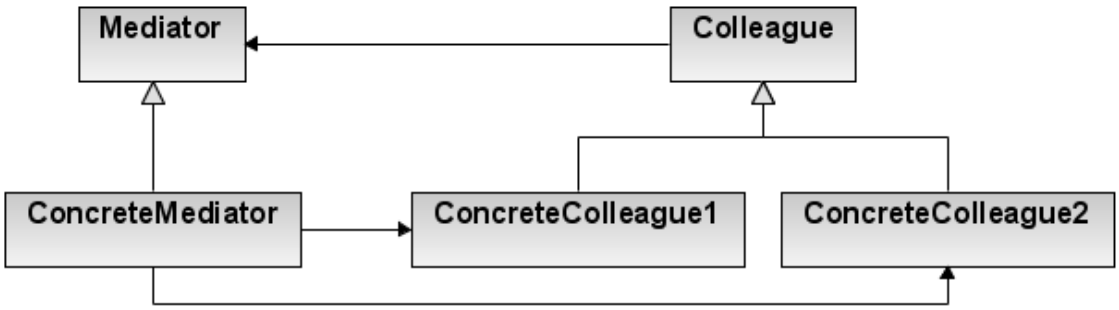
Samuel Santa – 202123685
Juana Morales – 202021591
Miguel Gomez – 202122562

Taller 5 – Patrones

Patrón seleccionado: Behavioral – Mediator



El patrón que elegimos para estudiar fue el patrón mediador, este es un patrón de comportamiento y es expuesto en el libro proporcionado como referencia para el curso "Design Patterns Elements of Reusable Object-Oriented Software." Principalmente este patrón pertenece a la categoría de patrones de comportamiento y es un patrón cuya implementación consiste en introducir una nueva clase (capa) que sirva como mediador entre otras dos haciendo que toda la interacción entre ambos objetos sea manejada directamente por el objeto mediador. El problema que soluciona este patrón es que ayuda con la reducción de acoplamiento en la mayor medida posible ya que cuando los objetos tienen un mediador entre ellos se facilita la comunicación. Un ejemplo del uso de este patrón se podría ver en un programa de pistas de carreras de formula 1 en los cuales todos los carros se comunican con la torre de control y ella con los carros en cuestión, situación en la cual se presenta un menor acoplamiento que si todos los carros se comunicaran entre sí.

Caso de estudio: Laniax's Event Dispatcher - https://github.com/Laniax/EventDispatcher

Laniax's Event Dispatcher es un framework realizado en java con el patrón de diseño mediador. Es un framework que permite declarar eventos y asimismo declarar listeners para estos mismos eventos. Este framework se inspiró en el generador de eventos de Symfony. El código fuente consiste en 3 clases principalmente las cuales incluyen Dispatcher, Event y EventListener. El patrón mediador es utilizado para disminuir el acoplamiento que pueda llegar a haber entre las clases Event y EventListener, teniendo una clase Dispatcher adicional para manejar todas las interacciones que pueda haber entre estas dos, simplificando los procesos de comunicación que puedan llegar a haber entre estas dos clases, minimizando la dependencia entre clases facilitando cualquier necesidad de escalabilidad en algún futuro.