Le pattern **Médiateur** est un **pattern** **Comportementale** (Behevioral pattern)

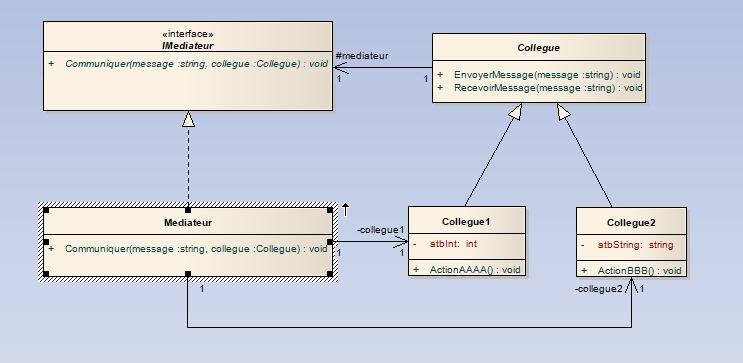
Il permet de **déléguer les interactions** des **classes « Collegue »** devant communiquer entres elles à une **classe abstraite « Mediateur » ou une interface « IMediateur ».**

Le passage d’information se fait via le médiateur, C’est le médiateur qui connait les classes « Collegue » et leur comportement et il coordonne leur interactions sans que les collègues n’ai consciences de leur existence mutuelle.

Une classe concrète « **Mediateur** » ou « **ConcreteMediateur** » implémentera le code nécessaire à la réalisation des **interactions entre « Collegue »**

Les classes **collegues** (ex : Collegue1 et Collegue2) **hériteront d’une superclasse** abstract **« Collegue »** qui permettra au passage d’implémenter les comportements commun.

Voici ce que cela donne en Diagramme



La **superclasse «Collegue» possède une instance de médiateur** qui sera accessible par ses enfants   
(Je l’ai mis en protected, mais il peut être mis en private et accessible via des méthodes ou Getteur)

Collègue pourra posséder des Méthodes communes à tous les collègues, (ou des méthodes abstraites à redéfinir ça reste un héritage classique de ce côté).

Les classes filles auront accès aux méthodes et attribut de la superclasse «Collegue» mais auront également des méthodes propres à chacune.

**C’est ici qu’entre en jeux le médiateur.** **La classe Concrète «Mediateur» ou «ConcreteMediateur» implémentera l’interface «IMediateur» ou héritera de la classe abstraite «Mediateur»**, et c’est cette classe qui implémentera le code, connaissant les différentes classes «Collegue» (Collegue1 et Collegue2 dans l’exemple) elle peut appeler les méthodes de chacune d’elle et passer les informations d’une classe à l’autre.