Behavioral Pattern: Mediator

Kevin Dockx
Architect

@KevinDockx | www.kevindockx.com

Coming Up

Describing the mediator pattern

- Chatroom scenario

Structure of the mediator pattern

Coming Up

Use cases for this pattern

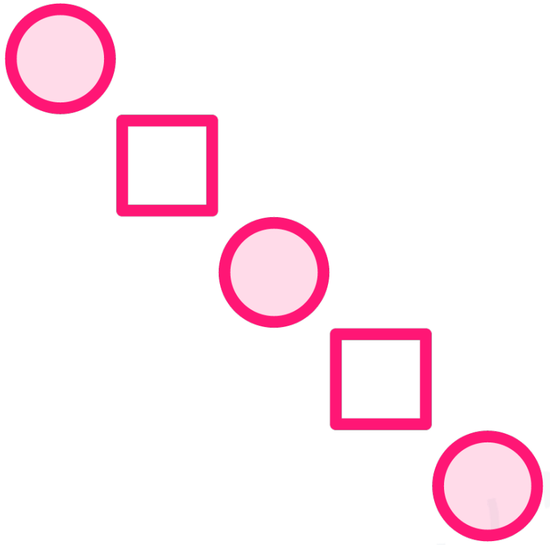
Pattern consequences

Related patterns

Describing the Mediator Pattern

Creational

Structural

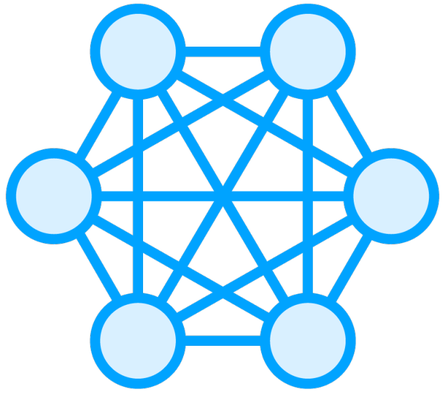


Behavioral

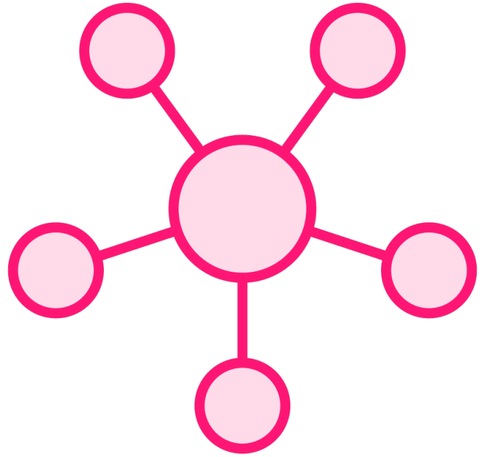
Mediator

The intent of this pattern is to define an object - the mediator - that encapsulates how a set of objects interact. It does that by forcing objects to communicate via that mediator.

Describing the Mediator Pattern



Objects hold references to each other. Management & keeping communication in sync is an issue.



A central object, the mediator, holds references of objects that want to communicate with each other. It handles communication between them.

```
public class Lawyer : TeamMember
{
    private List<TeamMember> _teamMembersInChat = new();
}

public class AccountManager : TeamMember
{
    private List<TeamMember> _teamMembersInChat = new();
}
```

Describing the Mediator Pattern

```
public class Lawyer : TeamMember
{
    private List<TeamMember> _teamMembersInChat = new();
}

public class AccountManager : TeamMember
{
    private List<TeamMember> _teamMembersInChat = new();
}
```

Describing the Mediator Pattern

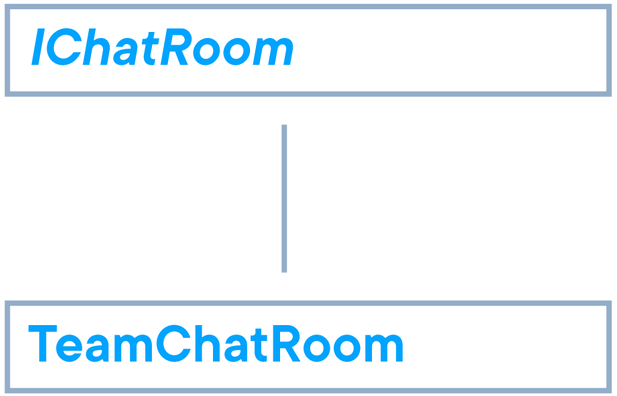
```
public class Lawyer : TeamMember
{
    private List<TeamMember> _teamMembersInChat = new();
}

public class AccountManager : TeamMember
{
    private List<TeamMember> _teamMembersInChat = new();
}
```

Describing the Mediator Pattern

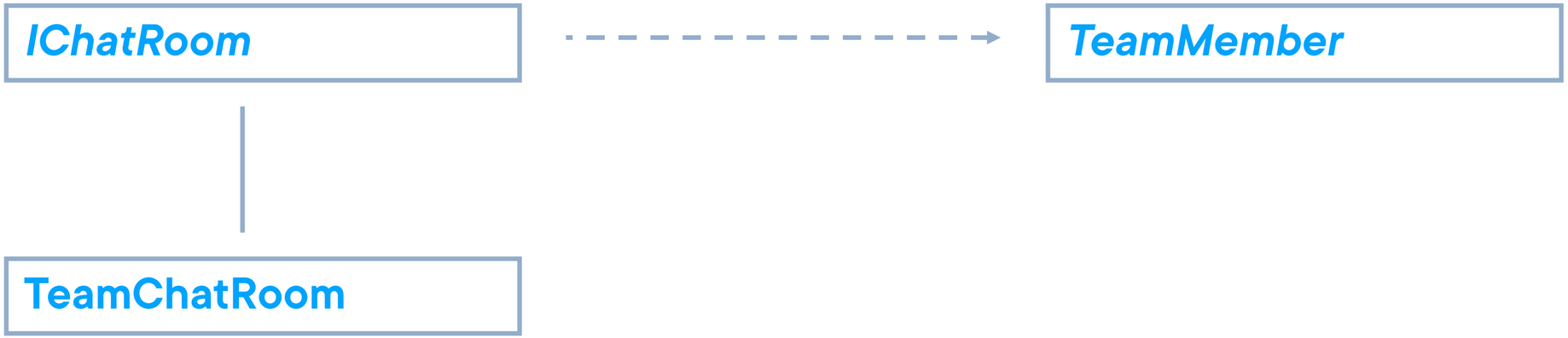
We want to avoid holding references between team member objects

Describing the Mediator Pattern

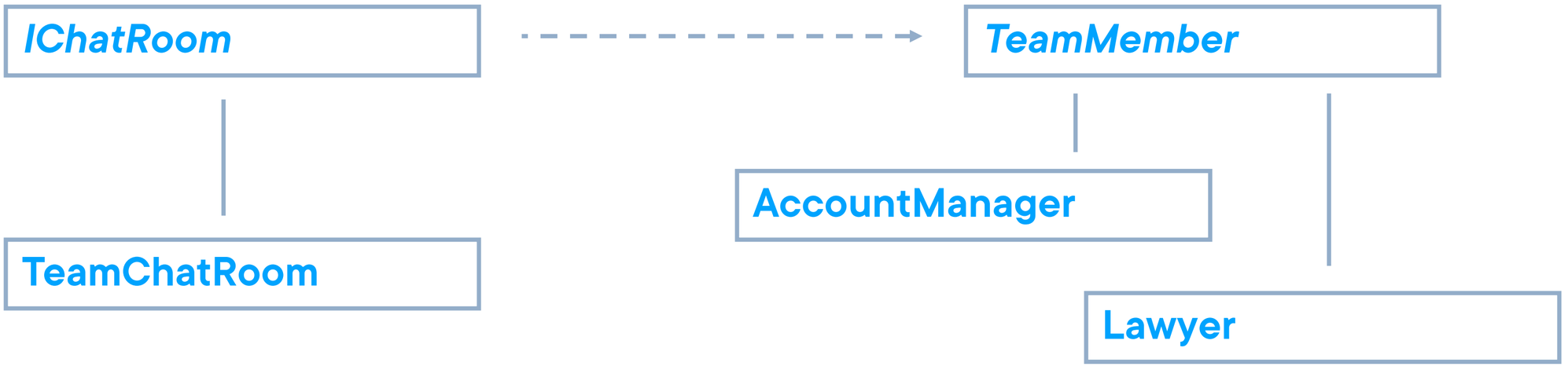
Describing the Mediator Pattern



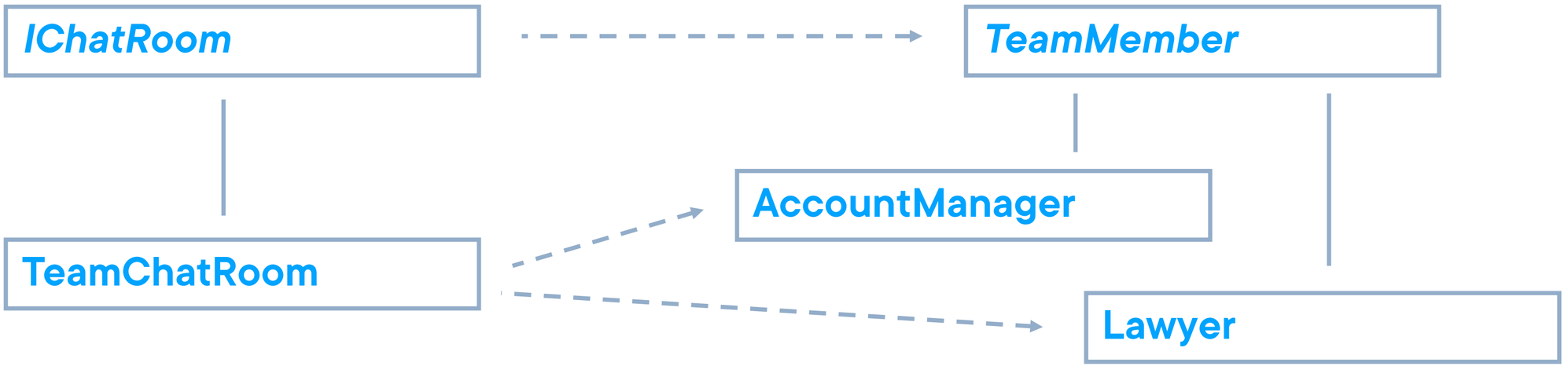
Describing the Mediator Pattern



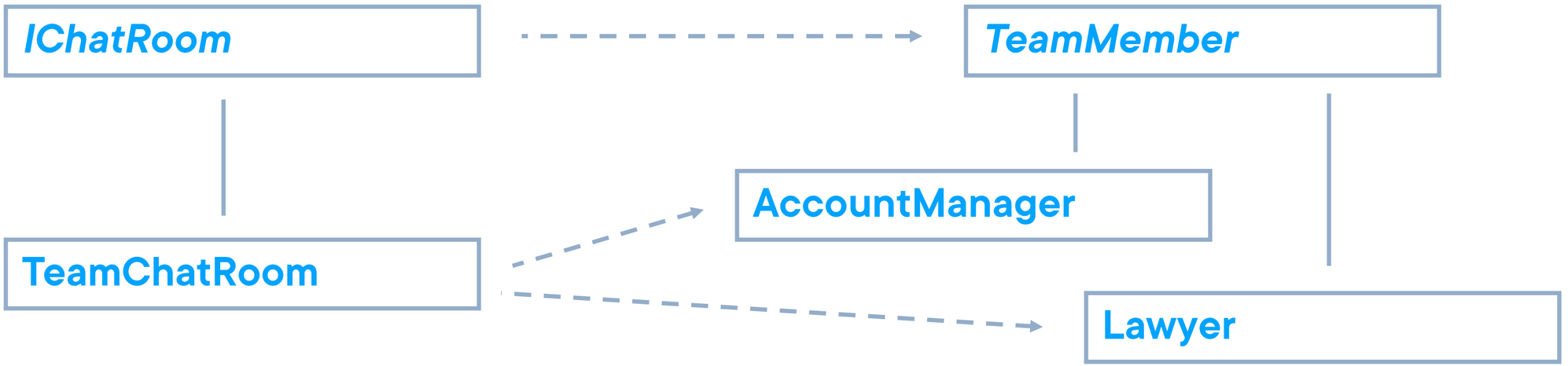
Describing the Mediator Pattern



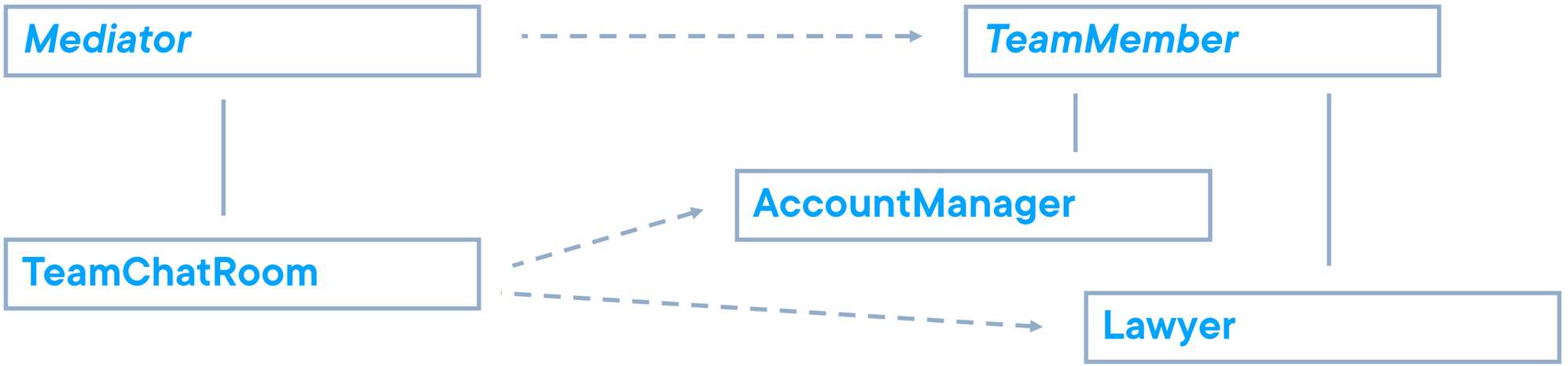
Describing the Mediator Pattern



Structure of the Mediator Pattern

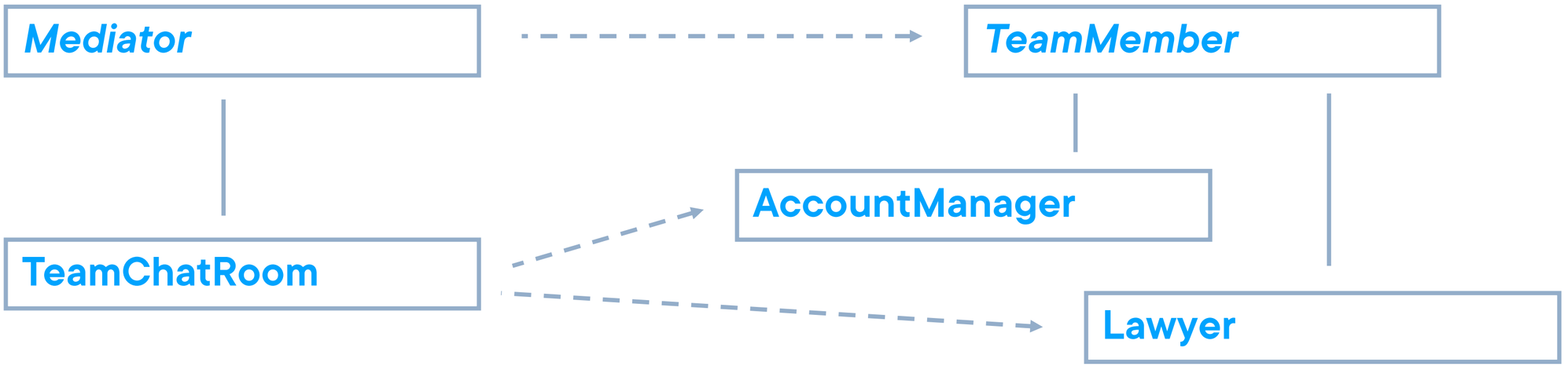


Structure of the Mediator Pattern

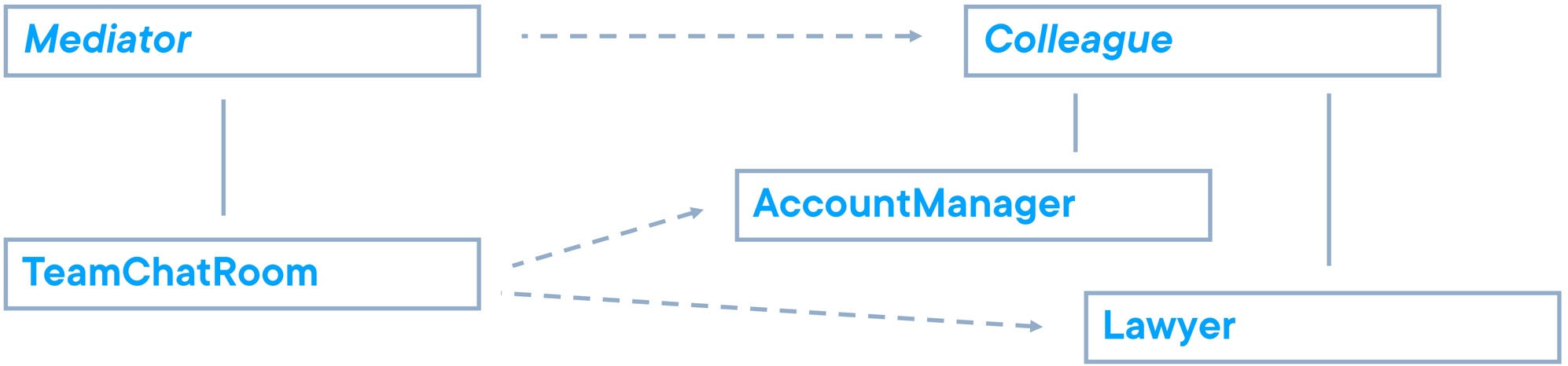


Mediator **defines an interface for communicating with** Colleague **objects**

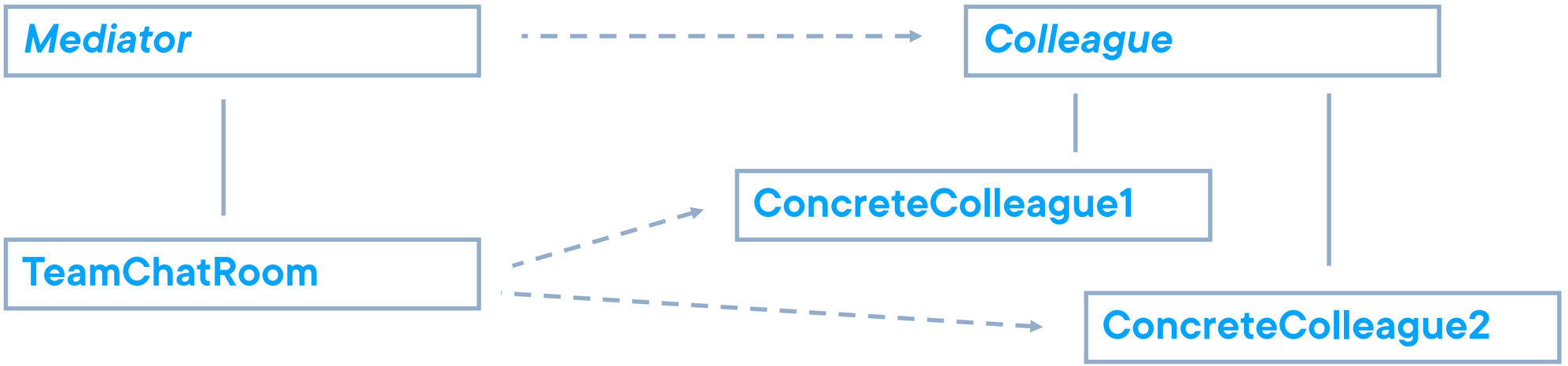
Structure of the Mediator Pattern



Structure of the Mediator Pattern

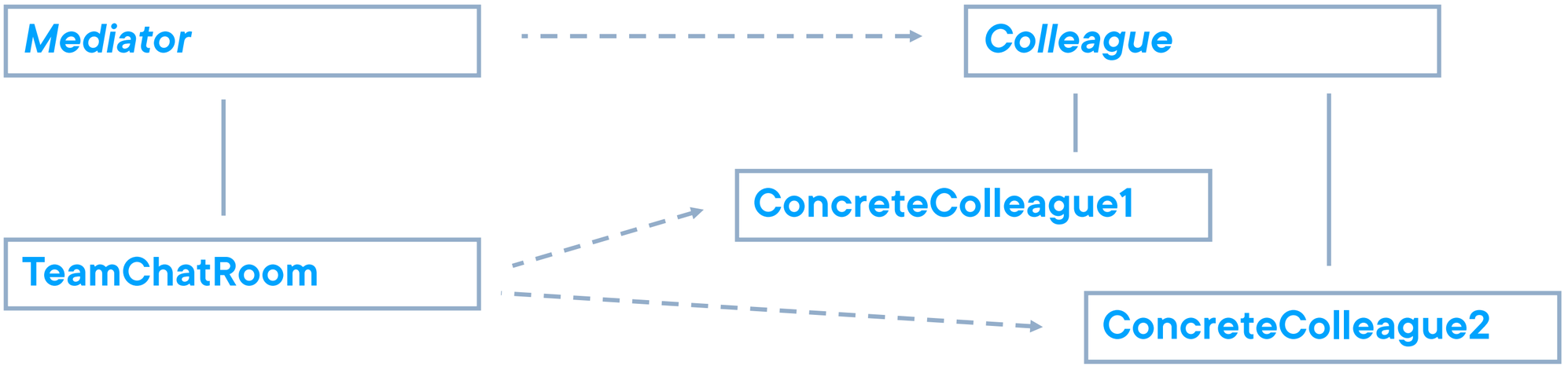


Structure of the Mediator Pattern

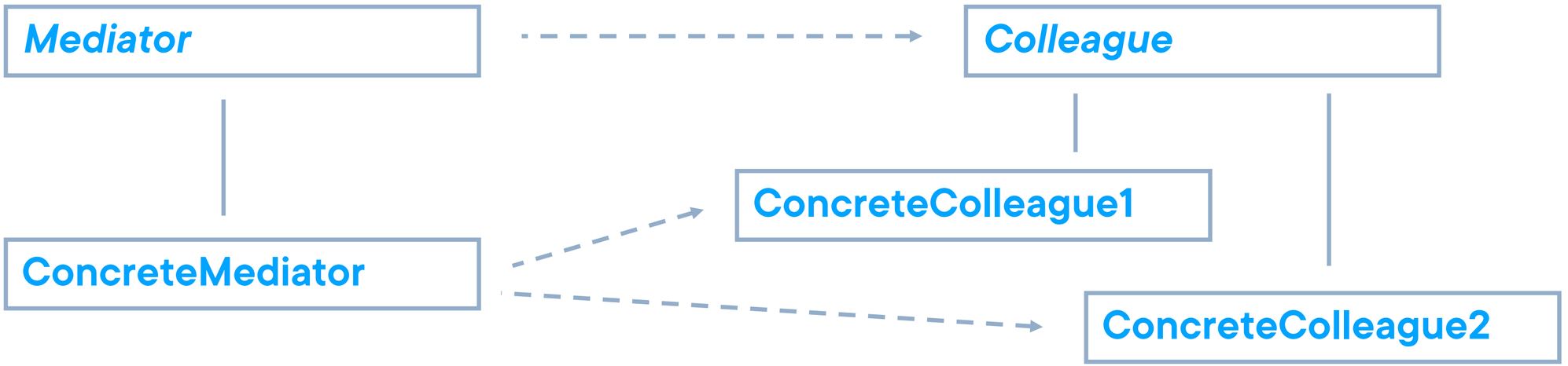


`Colleague` **knows its** `Mediator` **and communicates with it instead of with another** `Colleague`

Structure of the Mediator Pattern



Structure of the Mediator Pattern



ConcreteMediator **knows and maintains its** Colleagues, **and it implements cooperative behavior by coordinating** Colleague **objects**

Demo

Implementing the mediator pattern

Demo

Supporting communication between specific objects

Use Cases for the Mediator Pattern

When a set of object communicate in well-defined but complex ways

When, because an object refers to and communicates with many other objects, the object is difficult to reuse

When behavior that's distributed between several classes should be customizable without a lot of subclassing

Use Cases for the Mediator Pattern

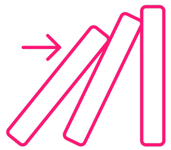
Hospital information system

Smart home automation

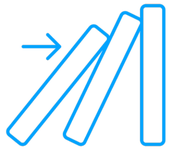
Stock trading system

Traffic management system

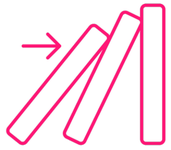
Pattern Consequences



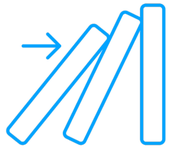
It limits subclassing



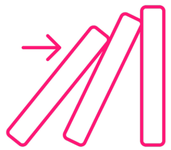
It decouples colleagues



It simplifies object protocols

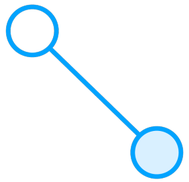


New mediators can be introduced without having to change the components: open/closed principle



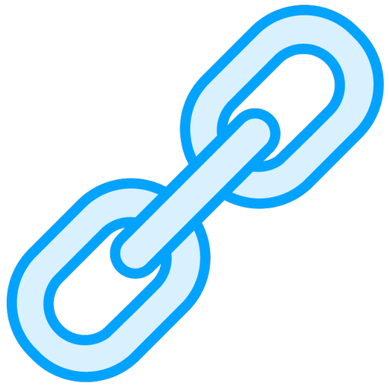
It centralizes control, which can make the mediator turn into a monolith

Related Patterns

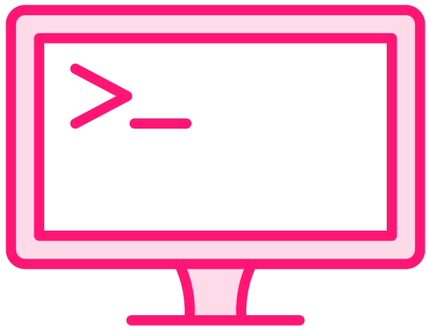


Facade
Mediator abstracts communication between objects. Facade abstracts the interface to the subsystem objects to promote ease of use.

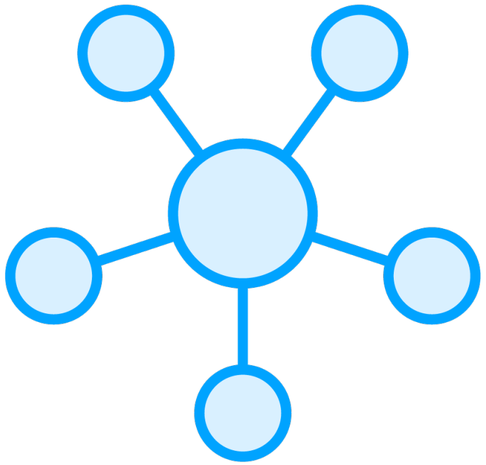
Patterns that Connect Senders and Receivers



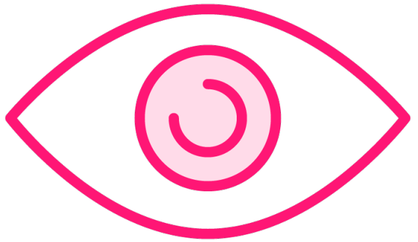
Chain of Responsibility
Passes a request along a chain of receivers



Command
Connects senders with receivers unidirectionally



Mediator
Eliminates direct connections altogether



Observer
Allows receivers of requests to (un)subscribe at runtime

Summary

Intent of the mediator pattern:

- To define an object that encapsulates how a set of objects interact

Promotes loose coupling

Object interaction can vary independently

Summary

Implementation:

- Use an `internal` method to ensure the mediator can't be set outside of an assembly
- Don't allow overriding methods when it's not needed

Up Next:

Behavioral Pattern: Chain of Responsibility