

MEDIATOR PATTERN

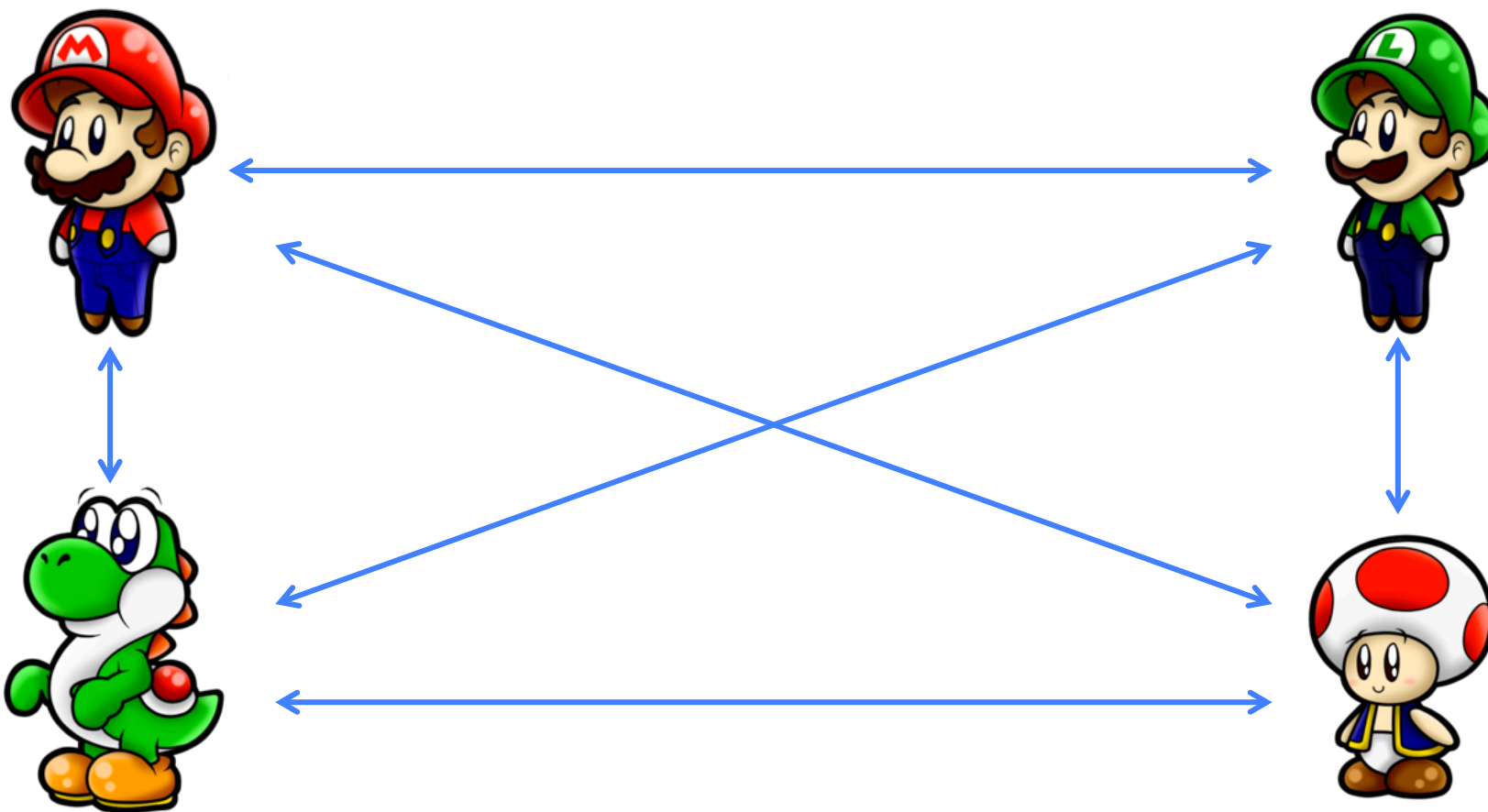
**“Luigi, can’t talk long. Bowser doesn’t
know my videophone still works for
calling out. I don’t know where he’s
keeping me. Sewage. He’s coming back.
Later!”**

-Princess Peach

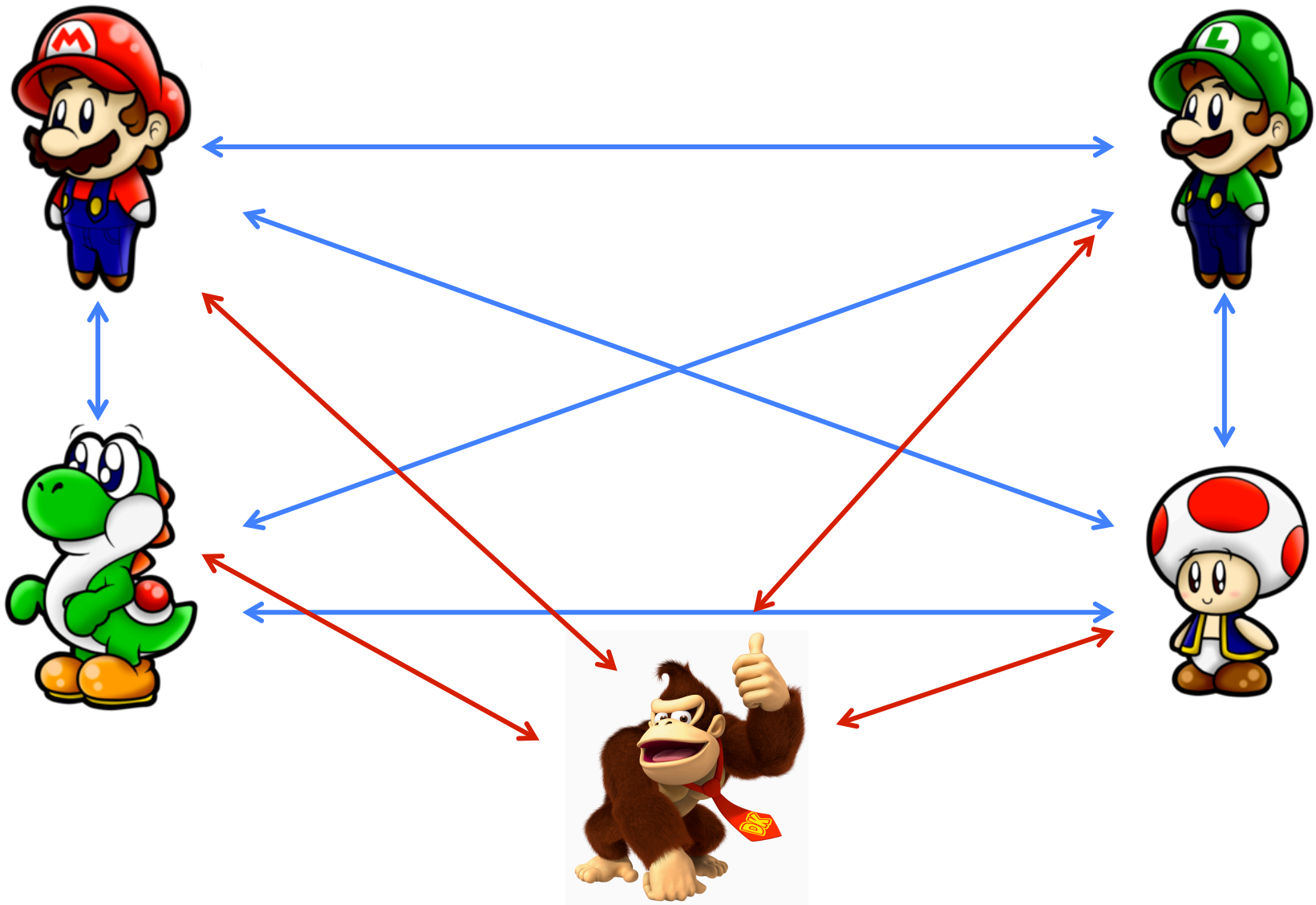
HERE'S THE DEAL...



Mario, Luigi, Yoshi, and the Mushroom need a system to communicate on their quest to save the Princess



Here's one way to connect our heroes...



What if we added a new character to the quest?

GAME OVER



This architecture is not all that scalable

(The number of connections grows at $O(n^2)$...)

WHAT IF INSTEAD...

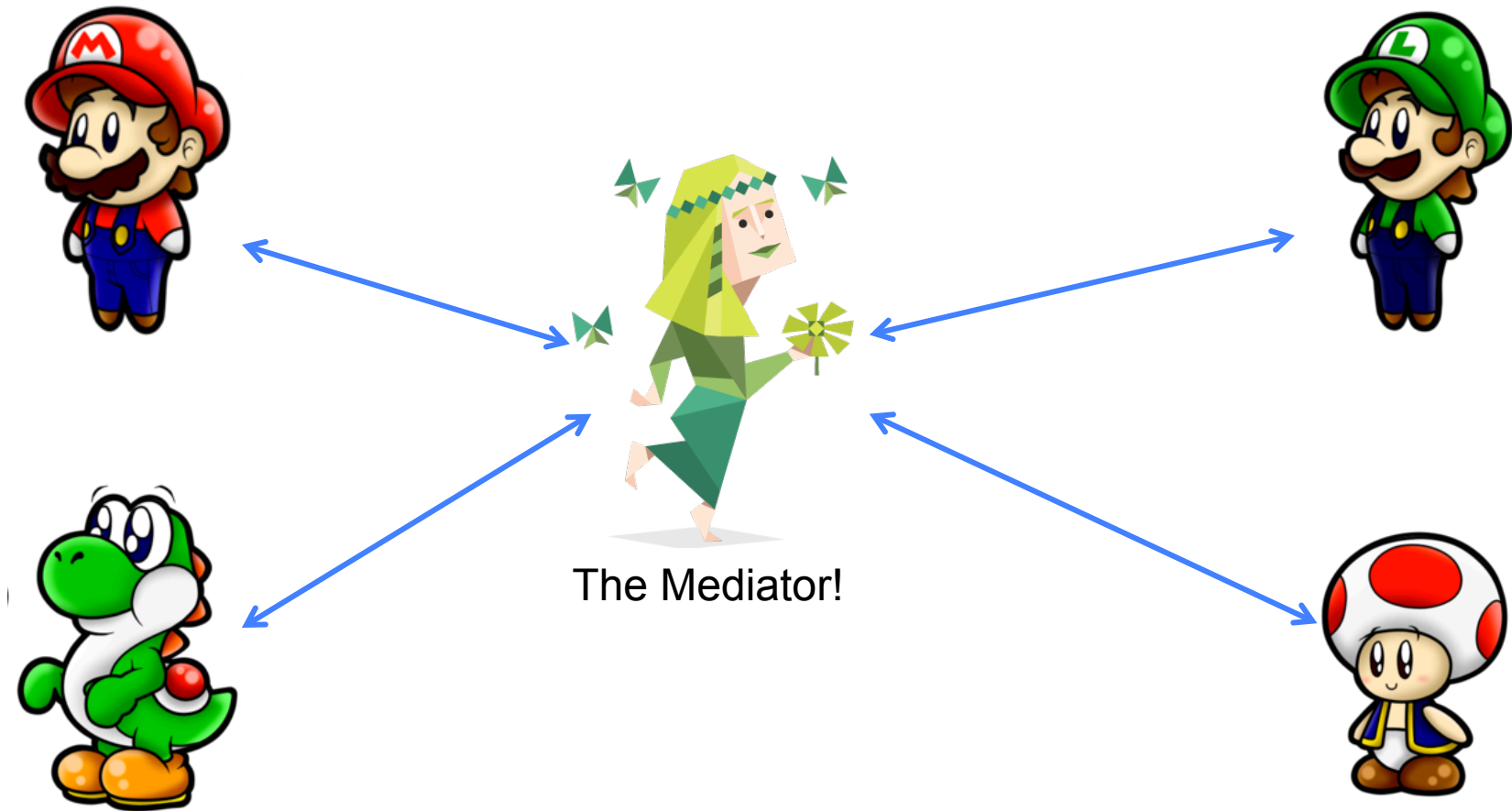


The Mediator!



We introduce a new role in the system – the mediator – responsible for relaying messages to our heroes

WHAT IF INSTEAD...



**Now our heroes do not have to know about each other.
They only interact with the mediator.**



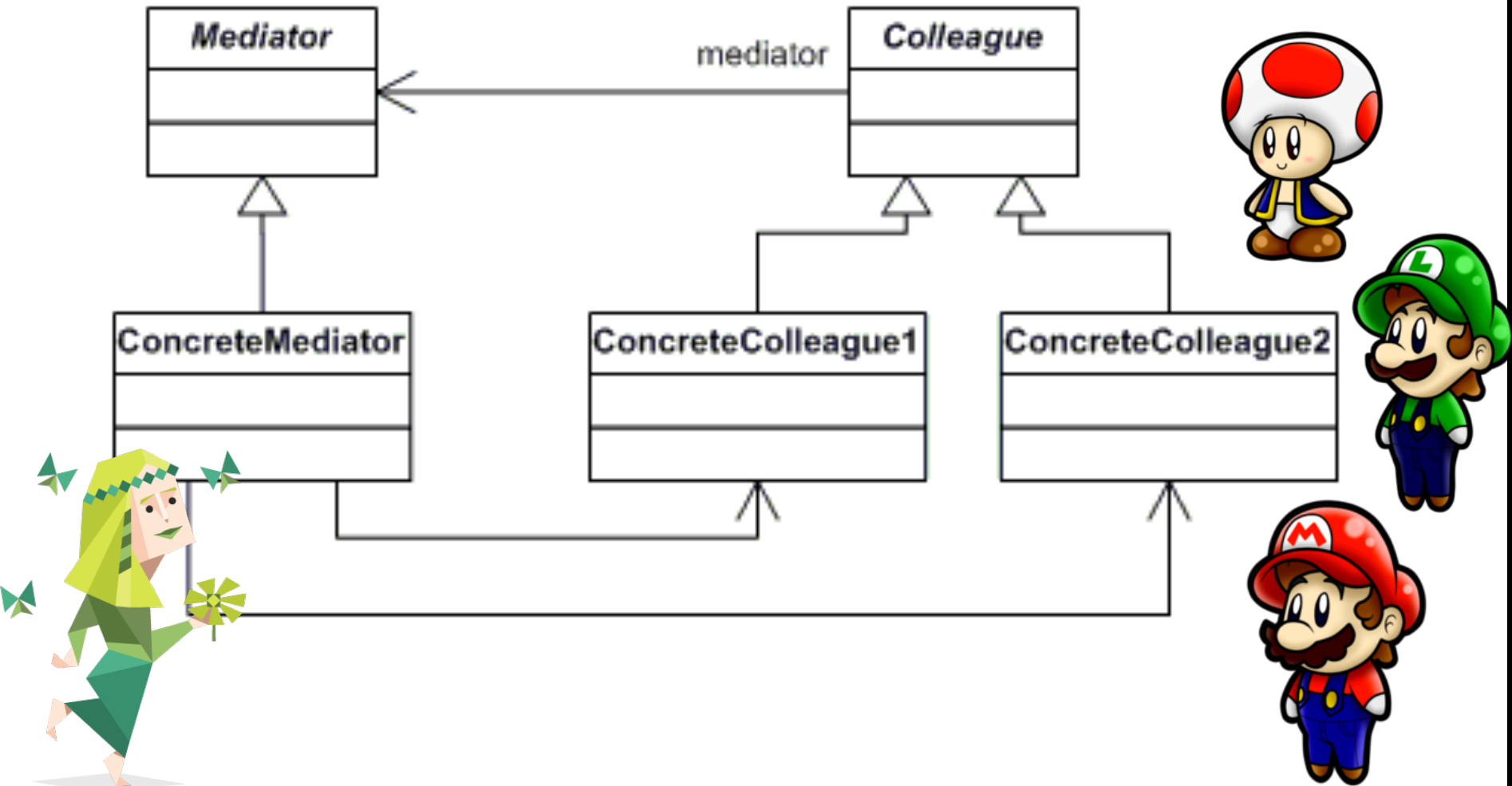
Now our architecture is way more scalable

The number of connections grows at $O(n)$

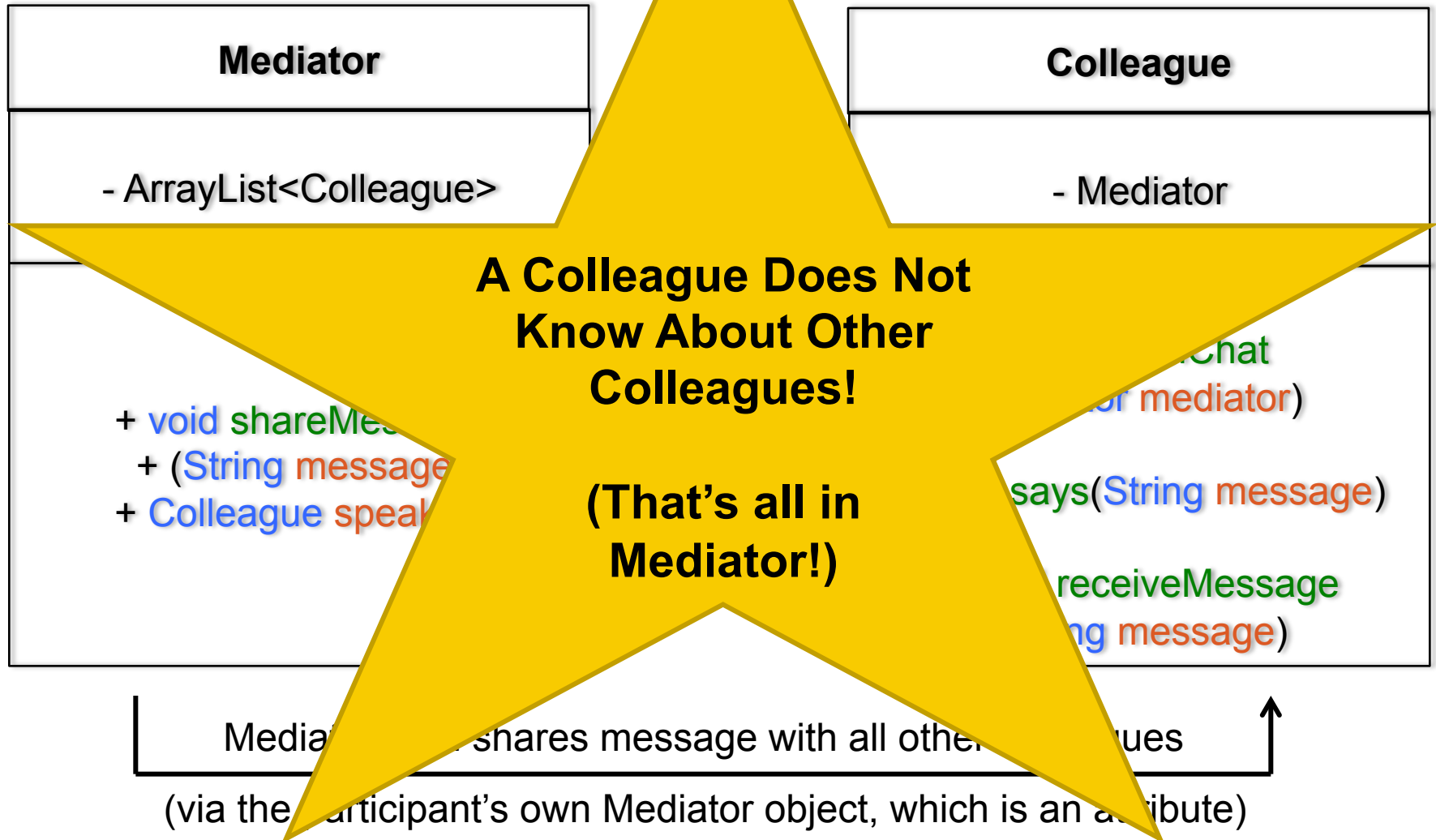


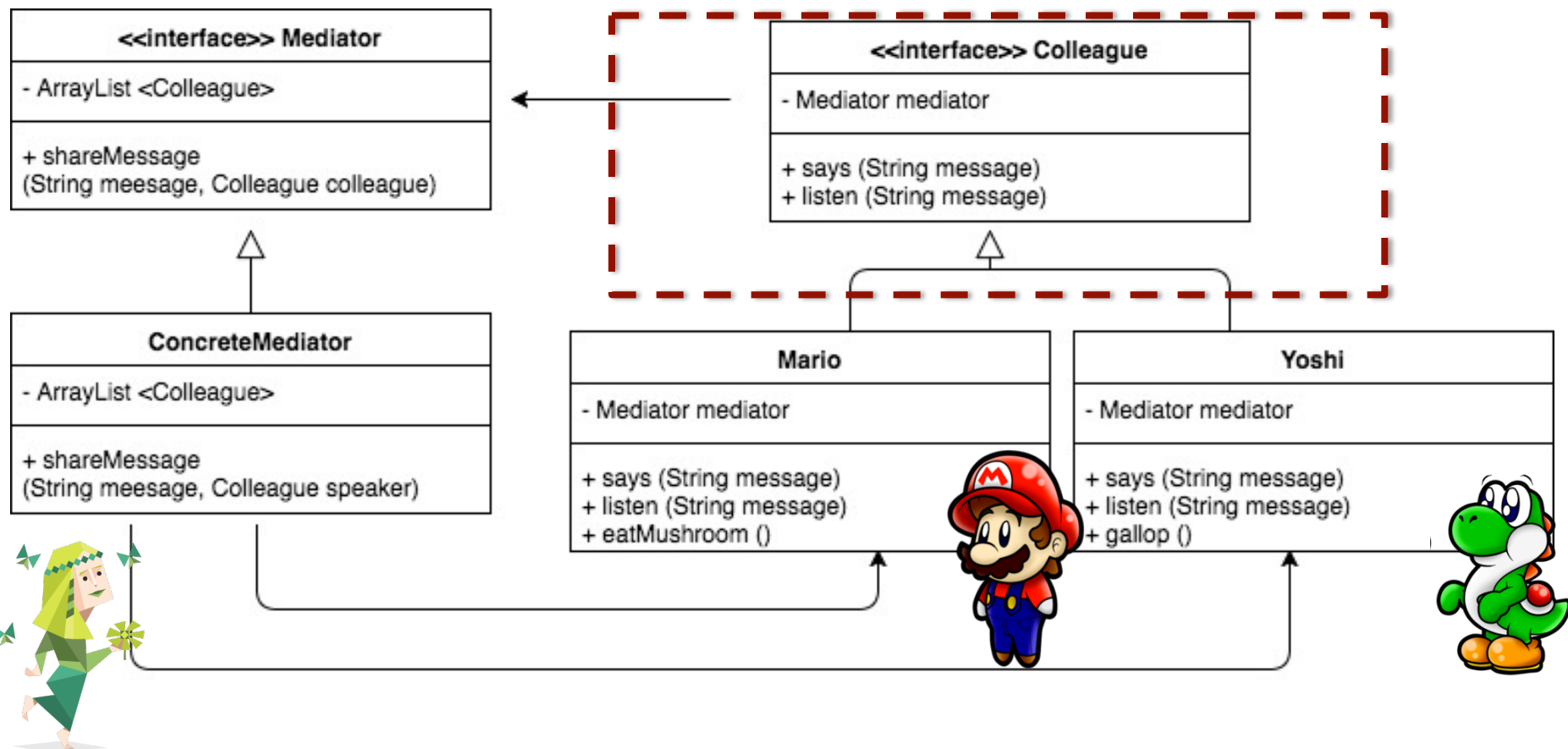
Ready for some UML?

ARCHITECTURE



MEDIATOR & COLLEAGUE





```
public class ConcreteColleage implements Colleage {
```

```
    String name;
```

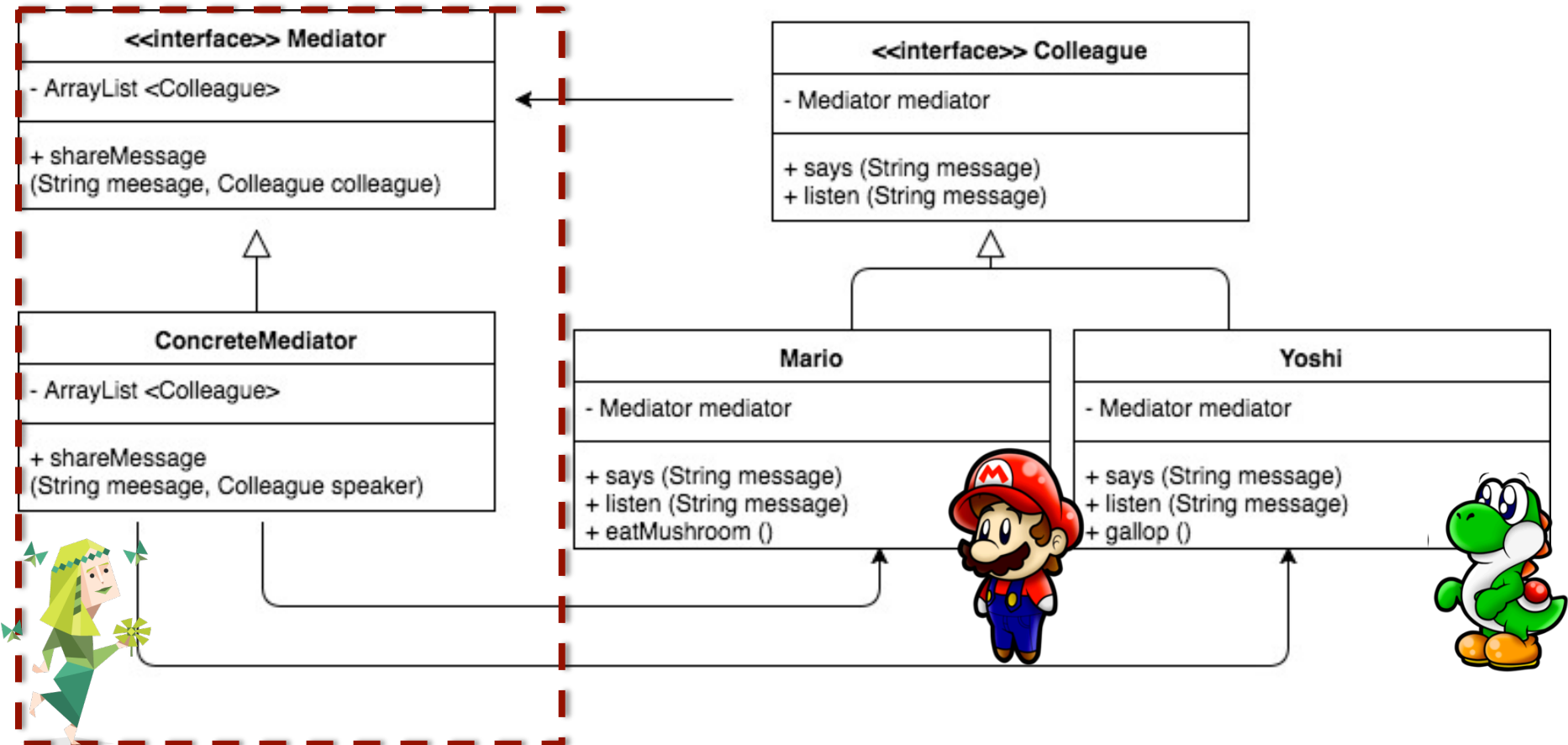
```
    Mediator mediator;
```

```
    public void says(String message){
```

```
        mediator.shareMessage (name + " says: " + message, this);
```

```
    }
```

```
}
```



```

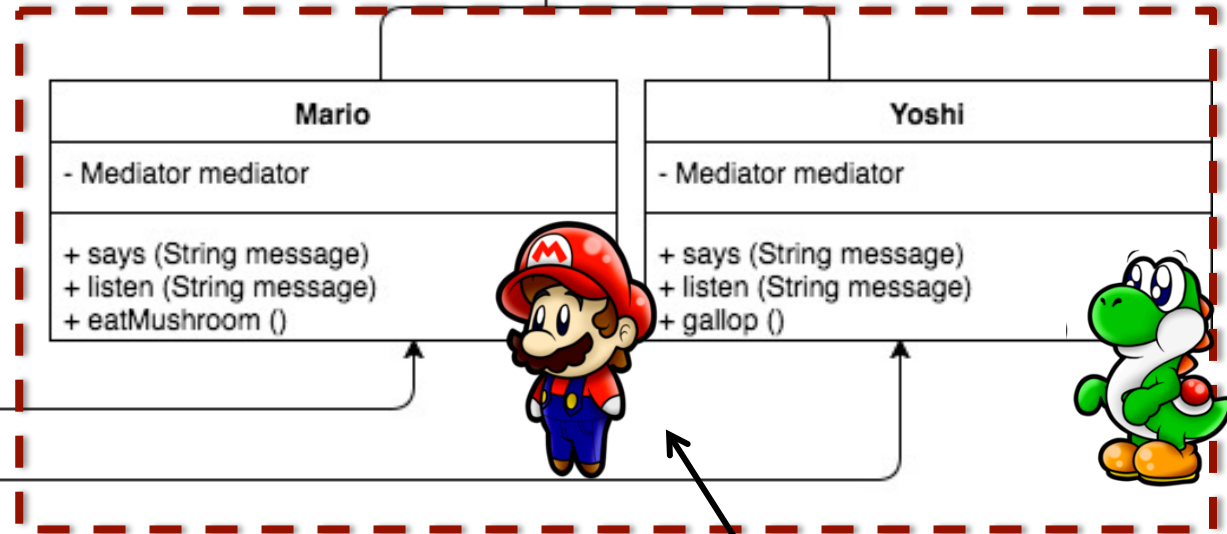
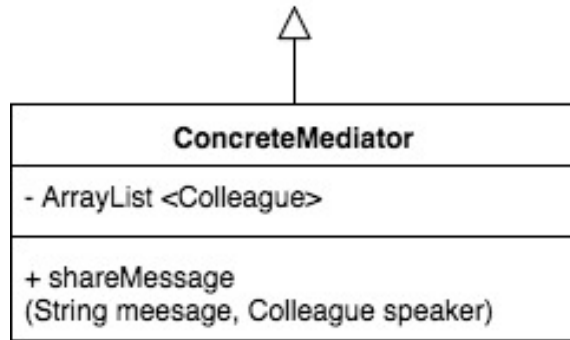
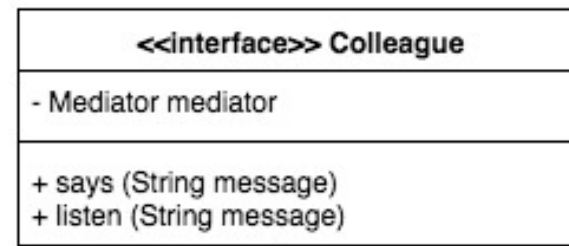
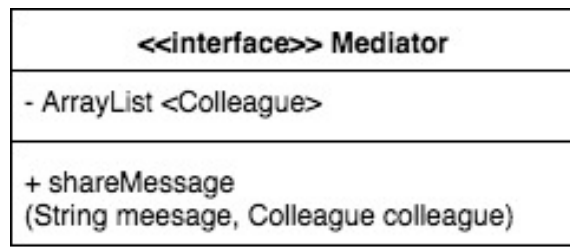
public class ConcreteMediator implements Mediator {
    ArrayList<Colleague> heroes;

```

```

    public void shareMessage(String message, Colleague speaker){
        for (Colleague hero : heroes){
            if (hero != speaker) hero.listen (message);
        }
    }
}

```



```
public class ConcreteColleague implements Colleague {
```

```
    String name;  
    Mediator mediator;
```

```
    public void listen (String message){  
        System.out.println (message);  
    }  
}
```

Psst... Isn't Mario a Singleton?

WHERE ELSE CAN WE FIND THE MEDIATOR?

(As if saving the Princess wasn't compelling enough... I get it, we can't all be into 90's video games!)

- GUI Window (Mediator) interacting with elements on the page (Colleagues)
- Radio Dispatch System
- Orchestra
- Air traffic control System
- Complex, High-stakes, Negotiation
- Dispute Resolutions

They're in more places than meets the eye!

SOME REAL WORLD EXAMPLES

The screenshot shows the Piazza Q&A interface for the course MPC5 51410. The top navigation bar includes the Piazza logo, the course name 'MPC5 51410' with a notification badge showing '11', and links for 'Q & A', 'Resources', and 'Statistics'. Below this is a breadcrumb trail: 'practicum1' (with a '2' badge), 'practicum2', 'practicum3', 'lectures', 'project', 'logistics', and 'other'. A secondary navigation bar shows filters: 'Unread', 'Updated', 'Unresolved', and 'Following'. The main content area on the left lists three posts:

- Test Cases** (Sat): "What is 'exactly' expected from the test scenarios and how many test cases/methods/functionalities are desired?" (1 unread, info icon)
- user story mapping--multiple or one** (Sat): "The users of the course registration system can be student, instructor, system administration, etc.... and their require" (info icon)
- A Clean Presentation of Use Case Mod...** (Fri): "Apologies for bombarding you with questions! It doesn't take too long to end up with a spaghetti-looking use case m" (info icon)

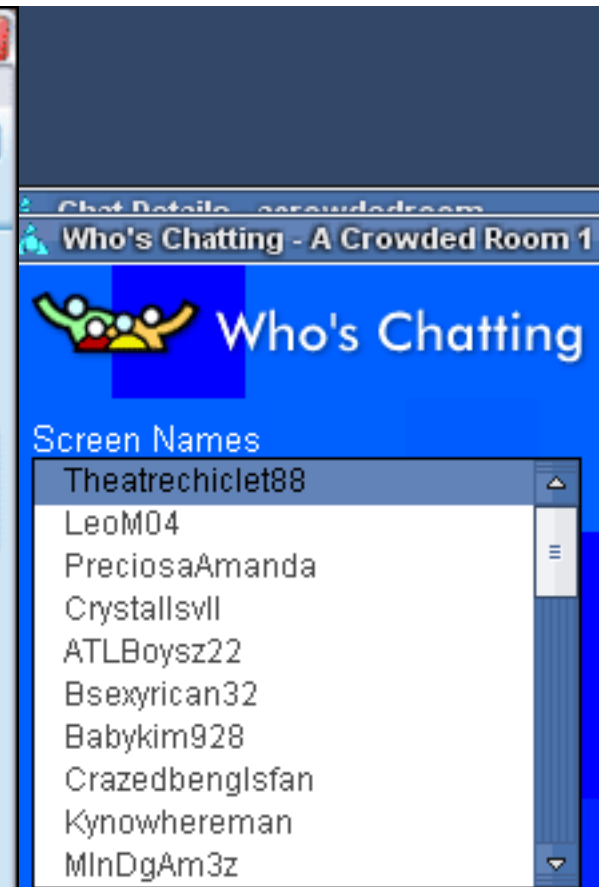
On the right, a 'Class at a Glance' summary box provides a quick overview:

- 2 unread posts**
- no unanswered questions**
- no unresolved followups**

Our Beloved Piazza

SOME REAL WORLD EXAMPLES

(Again, still not stuck in the 90's :p)



Who remembers these?

A WORD OF WARNING

Positives:

- Simplifies interaction between classes
- Allows for easier scalability
- From many-to-many interactions to one-to-many
- Allows easier reuse of Colleagues
- Easier to understand

Negatives:

- “Trade complexity of interaction for complexity of mediator”
- May end up with too complex of a mediator
- Risk is more concentrated in mediator
- Possibly leading to a more fragile system?



THAT'S IT FOLKS!

