

# Mediator Design Pattern

Behavioral design pattern can change the way programs run. Mediator design pattern is one of the important and widely used behavioral design pattern. Mediator enables decoupling of objects by introducing a layer in between so that the interaction between objects happen via the layer.

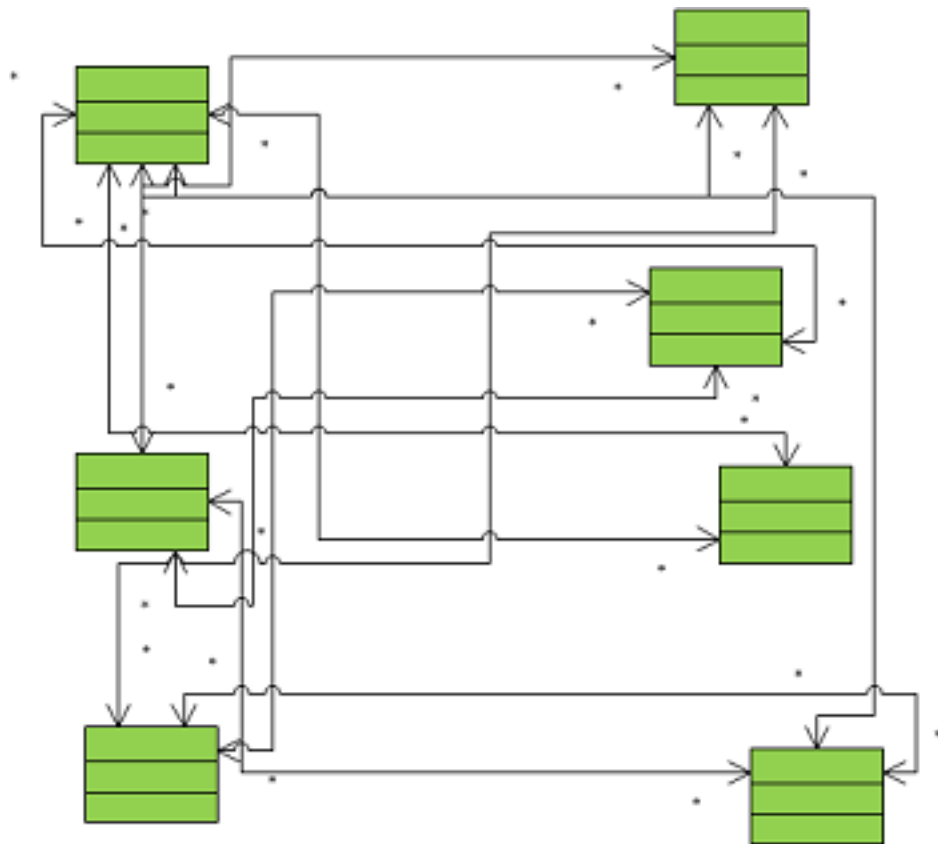
In an enterprise application where we have large number of classes the complexity also increases proportionately. Business logic might be spread across those multiple classes. There will be interaction between classes. If we draw line between classes where we have interaction, then it will end up like Idiyappam (Indian rice noodle).



# Before Mediator Design Pattern

---

So what is the problem we are trying to solve here? This complex interaction between objects creates dependency and tighter coupling. We stand for loose coupling, we want to reduce dependency as much as possible, we want more reuse.

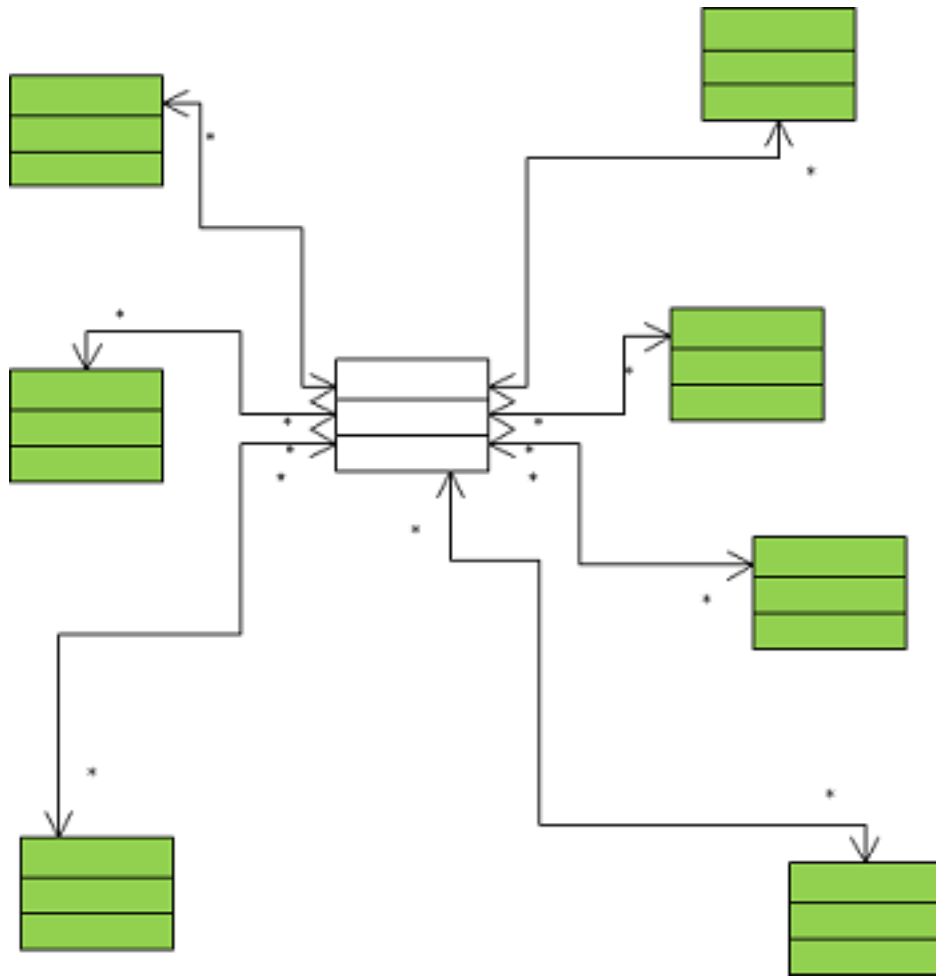


# After Mediator Design Pattern

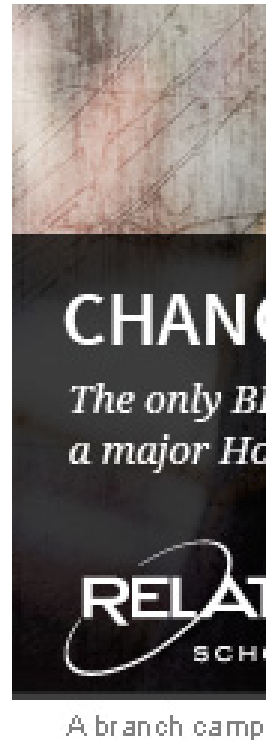
---

Define an object that acts as a mediator between communicating objects thus removing the direct dependency

between those communicating objects. This mediator object encapsulates the interaction information and uses it to enable communication between the objects.



The intermediate mediator object is not only to communicate between two objects, but it helps in interaction for a set of peer objects. This mediator object is like a router in networks, which is routes / controls or coordinates communication between systems.



## Real-time Example for Mediator Design Pattern

---

Air traffic controller (ATC) is a mediator between flights. It helps in communication between flights and co-ordinates/controls landing, take-off. Two flights need not interact directly and there is no dependency between them. This dependency is solved by the mediator ATC. If ATC is not there all the flights have to interact with one another and managing the show will be very difficult and things may go wrong.



## Mediator Pattern Implementation

---

In a mediator design pattern implementation we will have

- mediator interface – an interface that defines the communication rules between objects
- concrete mediator – a mediator object which will enables communication between participating objects
- colleague – objects communicating with each other through mediator object

## Java Example for Mediator Desing Pattern

---

```
package com.javapapers.designpattern.mediator;

public class ATCMediator implements IATCMediator {
    private Flight flight;
    private Runway runway;
    public boolean land;

    public void registerRunway(Runway runway) {
        this.runway = runway;
    }

    public void registerFlight(Flight flight) {
        this.flight = flight;
    }

    public boolean isLandingOk() {
        return land;
    }

    @Override
    public void setLandingStatus(boolean status) {
        land = status;
    }
}
```

```
package com.javapapers.designpattern.mediator;
```

```
public interface Command {  
    void land();  
}
```

```
package com.javapapers.designpattern.mediator;  
  
public interface IATCMediator {  
  
    public void registerRunway(Runway runway);  
  
    public void registerFlight(Flight flight);  
  
    public boolean isLandingOk();  
  
    public void setLandingStatus(boolean status);  
}
```

```
package com.javapapers.designpattern.mediator;  
  
public class Flight implements Command {  
    private IATCMediator atcMediator;  
  
    public Flight(IATCMediator atcMediator) {  
        this.atcMediator = atcMediator;  
    }  
}
```

```

public void land() {
    if (atcMediator.isLandingOk()) {
        System.out.println("Landing done....");
        atcMediator.setLandingStatus(true);
    } else
        System.out.println("Will wait to land....");
}

public void getReady() {
    System.out.println("Getting ready...");
}
}

```

```

package com.javapapers.designpattern.mediator;

public class Runway implements Command {
    private IATCMediator atcMediator;

    public Runway(IATCMediator atcMediator) {
        this.atcMediator = atcMediator;
        atcMediator.setLandingStatus(true);
    }

    @Override
    public void land() {
        System.out.println("Landing permission granted...");
        atcMediator.setLandingStatus(true);
    }
}

```



```
}
```

```
}
```

## MediatorDesignPattern.java

```
package com.javapapers.designpattern.mediator;

public class MediatorDesignPattern {
    public static void main(String args[]) {

        IATCMediator atcMediator = new ATCMediator();
        Flight sparrow101 = new Flight(atcMediator);
        Runway mainRunway = new Runway(atcMediator);
        atcMediator.registerFlight(sparrow101);
        atcMediator.registerRunway(mainRunway);
        sparrow101.getReady();
        mainRunway.land();
        sparrow101.land();

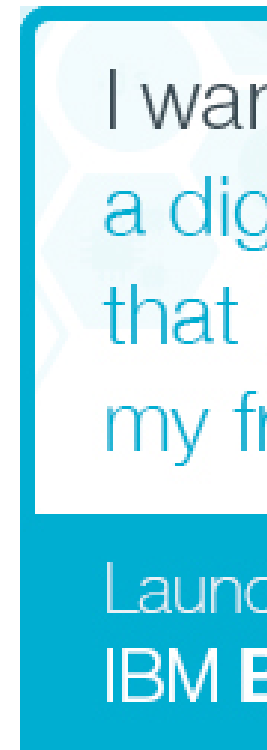
    }
}
```

## Mediator Design Pattern in Java API

---

[java.util.Timer](#) uses the mediator design pattern.

This Behavioral Design Pattern tutorial was added on 31/01/2013.



« [Screen Crack Android App](#)

[Java Weak Reference](#) »

## Comments on “Mediator Design Pattern”

---

*Jayani*

31/01/2013 at 12:05 pm

Hello Joe sir,

You have Nicely explained with good example!

Can we say JMS(Java Messaging System) is a combination of Observer and Mediator design patterns?

**Reply**

---

*Shailesh*

*31/01/2013 at 12:15 pm*

nice article ....

**Reply**

---

*C SATHIYARAJA*

*31/01/2013 at 12:23 pm*

Sir i want java mail concepts for received mail detail.. and reply mail edit before sent concepts.....file comparision for all received mail.....Are you free means contact immediately for my mail id.....

**Reply**

---

*Karesh*

*31/01/2013 at 1:36 pm*

Mr. C SATHIYARAJA,

What do you want Joe to implement everything for you?? He can give you hints and suggestions. Ask right questions here.

**Reply**

---

*Mohamed Galala*

*31/01/2013 at 1:14 pm*

You are really explaining the patterns in a very clear way.

Thank you for your effort.

I have only two comment:-

1- Your java example needs to be provided with comments.

For example, saying before the interface implementation that this is the Mediator Interface and so on.

2- Keep in mind the order of code snippets you are writing. For example, the mediator interface implementation must come before the implementation class.

Thank you one more time.

**Reply**

---

Hi Joe

I like all your post and the way how you are presenting to the users.

And also i like the your site theme, I just want to create website like this. Any help on this ?

**Reply**

---

GOOTAM

31/01/2013 at 5:04 pm

Good Work Joe. Keep it up.

**Reply**

---

Koray Tugay

31/01/2013 at 7:02 pm

Very good..

**Reply**

---

Thanks for the nice explanation...In ATCMediator you have declared land with public where as it could be declared as private. Correct me if i miss anything...

**Reply**

---

Bhimaray

01/02/2013 at 8:18 am

Hello sir, you explained very nicely

**Reply**

---

Muhammad Zahab Ahmed

01/02/2013 at 7:55 pm

You have explained very nicely...i enjoyed your blogs alot...Thanksss....

**Reply**

---

Anonymous

04/02/2013 at 1:05 am

very nice sir jee

**Reply**

---

*zeinab*

*05/02/2013 at 6:43 pm*

hi, i didnt find Sterategy design pattern in yr blog, can u explain that please?  
thanks in advance

**Reply**

---

*amjad*

*07/02/2013 at 11:17 am*

thanks, for your nice explanation

**Reply**

---

*Praveen*

*07/02/2013 at 6:20 pm*

example is very help-full, Good work Joe.

### **Reply**

---

*Nitin*

*13/02/2013 at 1:19 pm*

Great article....

### **Reply**

---

*sayras*

*15/02/2013 at 5:53 pm*

very nice explanation

thanks

### **Reply**

---

*aprajitha*

*16/02/2013 at 2:52 am*

Awesome real time example (Air Traffic Controller) to explain the pattern. Examples like these will last in memory forever than any set of code examples.



### **Reply**

---

*Shrikant Herwade*

*17/02/2013 at 12:53 am*

You really explain it in a better way..! simple n sober...Best Luck for future sharings...

### **Reply**

---

*Tamilarasu K*

*19/02/2013 at 11:19 pm*

Really Nice

### **Reply**

---

*mayur*

*06/03/2013 at 3:07 pm*

Can you explained the which interfece is mediator interface ,concrete mediator & colleague in above example.How they interact for multiple flight with single runway.

### **Reply**

---

*Anonymous*

*12/04/2013 at 8:04 pm*

nice..explanation

**Reply**

---

*Ankit*

*20/04/2013 at 8:47 pm*

very nice tutorials. Keep it up.

**Reply**

---

*Mahesh*

*22/04/2013 at 9:33 am*

Really helpful Joe

**Reply**

---

*Ravindra*

*23/04/2013 at 12:09 am*

Nicely presented with very good examples.

### **Reply**

---

*dhrumil*

*20/06/2013 at 1:02 pm*

nicely explain

### **Reply**

---

*yashwant*

*28/06/2013 at 1:11 pm*

Hey joe !! Very nice & simple explanation. Keep it up !!

### **Reply**

---

*Rahini Mariasingam*

*03/09/2013 at 8:54 pm*

Hi Joe

Thanks for your java blog . It is very helpful and simple and easy explanation for each design pattern.

Keep up your good work.

## **Reply**

---

*Anonymous*

*03/10/2013 at 3:19 am*

Best stuff here. Superb sir. Thanks for your support by explaining such a important concepts in so easy way

## **Reply**

---

*Ahmed*

*05/03/2014 at 3:39 am*

First of all thanks for the blog, Actually, I was recalling all the design patterns to solve a specific design problem in my current project, and Mediator was one of the patterns that stopped me, but the question how easy can we write a JUnit test for the Flight or Runway.

Do we have to create a mockup for the Mediator, because all of the classes became tightly coupled to the Mediator, may be it's an invalid assumption and may be not, your inputs are appreciated.

## **Reply**

---

*padmarao*

*14/04/2014 at 12:41 pm*

hi ,joe

please share the answer to my querys,

why spring bean is by default singleton?

and cache design pattern explain.....please

**Reply**

---

*Eswar1221*

*12/06/2014 at 3:24 am*

Hi Joe,

private Flight flight;

private Runway runway; are unused in this example. Why So ?

**Reply**

---

*Jay*

*05/08/2014 at 10:47 pm*

Good job... JOE.. nice explanation.. keep up the good work..

**Reply**

---

## Your Comment

Name

Email

Post Comment

### TUTORIAL MENU

---



Java



Android

 Design Patterns

 Creational Design Patterns

 Structural Design Patterns

 Behavioral Design Patterns

 Chain Of Responsibility Design Pattern

 Command Design Pattern

 Interpreter Design Pattern

 Iterator Design Pattern

 **Mediator Design Pattern**

 Memento Design Pattern

 Observer Design Pattern

 State Design Pattern

 Template Method Design Pattern


 Spring

 Web Services

 Servlet




 Java

 Android

 Design Patterns

 Spring

 Web Services

 Servlet

## *Site Map*

© 2008-2014 *javapapers.com*. The design and content are copyrighted to *Joe* and may not be reproduced in any form.