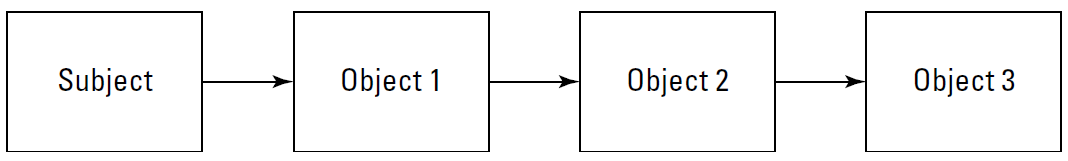
**CHAIN OF REPONSIBILITY PATTERN**

Mô hình hoạt động của pattern này là hoạt động chạy theo 1 dãy. Tức là A kéo theo B rồi kéo theo C … đến khi nào tới vị trí cần dừng lại hoặc tới điểm cuối cùng



Ví dụ như khi ta click chuột, sự kiện click chuột xảy ra, sau đó hàm thực thi sự kiện hoạt động rồi gửi cho window, rồi thực thi hàm chỉnh sửa window …. Ta thử xem xét cách thực thi pattern như thế nào

Nguyên tắc ở đây là khi 1 đối tượng được thực thi, nếu nó chưa phải đối tượng cuối và chưa phải đối tượng cần phải dừng, thì nó sẽ gọi tiếp đối tượng phía sau nó. Như vậy những việc ta cần làm trong 1 đối tượng là tạo một flag kiểm tra việc kết thúc hay chưa và 1 con trỏ (gọi vậy cho dễ hiểu) chỉ tới đối tượng tiếp theo cần thực hiện sau đối tượng hiện tại. Tất cả các công việc này sẽ được đặt trong 1 interface. Ta xem cách xây dựng interface này

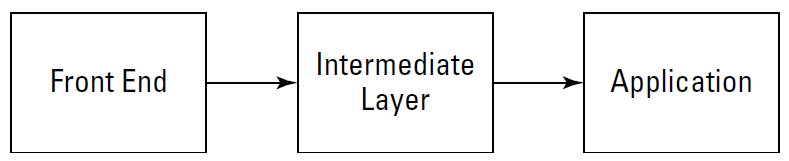
interface HelpInterface

{

public void getHelp(int helpConstant);

}

interface chỉ có 1 hàm getHelp, hàm getHelp có vai trò như khởi động tiến trình chạy qua các đối tượng (tức là khi gọi hàm này thì bắt đầu việc chạy qua các đối tượng), ta sẽ xem xét tiếp các class thực thi nó sẽ như thế nào. Để ví dụ rõ ràng ta sẽ cho 3 đối tượng có kết nối như sau



Bắt đầu với đối tượng Front End.

public class FrontEnd implements HelpInterface

{

**final int FRONT\_END\_HELP = 1;**

HelpInterface successor;

public FrontEnd(HelpInterface s)

{

successor = s;

}

**public void getHelp(int helpConstant)**

**{**

**if(helpConstant != FRONT\_END\_HELP){**

**successor.getHelp(helpConstant);**

**} else {**

**System.out.println(“This is the front end. Don’t you like it?”);**

**}**

**}**

}

Một vài đặc điểm quan trọng ở đây ta cần chú ý là trong hàm khởi tạo ta phải đưa đối tượng phía sau nó vào, tức là báo cho biết đối tượng nào sẽ được gọi sau đối tượng này, trong ví dụ ta sẽ đưa đối tượng Intermediate Layer vào. Ở đây HelpInterface đóng vai trò như một con trỏ trỏ tới đối tượng sau. Chúng ta có thêm giá trị int để kiểm tra xem đối tượng này có phải đối tượng dừng hay ko, nếu không thì chuyển cho đối tượng tiếp theo. Chúng ta sẽ làm tương tự như vậy cho các đối tượng sau

public class IntermediateLayer implements HelpInterface

{

**final int INTERMEDIATE\_LAYER\_HELP = 2;**

HelpInterface successor;

public IntermediateLayer(HelpInterface s)

{

successor = s;

}

**public void getHelp(int helpConstant)**

**{**

**if(helpConstant != INTERMEDIATE\_LAYER\_HELP){**

**successor.getHelp(helpConstant);**

**} else {**

**System.out.println(“This is the intermediate layer. Nice, eh?”);**

**}**

**}**

}

Đối tượng cuối cùng là application không cần phải để cờ chặn vì tới đây thì tiến trình cũng phải chấm dứt

public class Application implements HelpInterface

{

public Application()

{

}

**public void getHelp(int helpConstant)**

**{**

**System.out.println(“This is the MegaGigaCo application.”);**

**}**

}

Bây giờ chúng ta dựng chương trình để test pattern này

public class TestHelp

{

public static void main(String args[])

{

final int FRONT\_END\_HELP = 1;

final int INTERMEDIATE\_LAYER\_HELP = 2;

final int GENERAL\_HELP = 3;

**Application app = new Application();**

**IntermediateLayer intermediateLayer = new IntermediateLayer(app);**

**FrontEnd frontEnd = new FrontEnd(intermediateLayer);**

**frontEnd.getHelp(GENERAL\_HELP);**

}

}

Như ta thấy khi khởi tạo các đối tượng thì ta phải đưa giá trị phía sau nó vào. Sau khi gọi hàm getHelp và đưa cho nó giá trị dừng, ta thấy là nó sẽ không dừng ở Front hay Intermediate do đó sẽ chạy tới đối tượng Application, do đó giá trị xuất ra là

This is the MegaGigaCo application.