Mediator

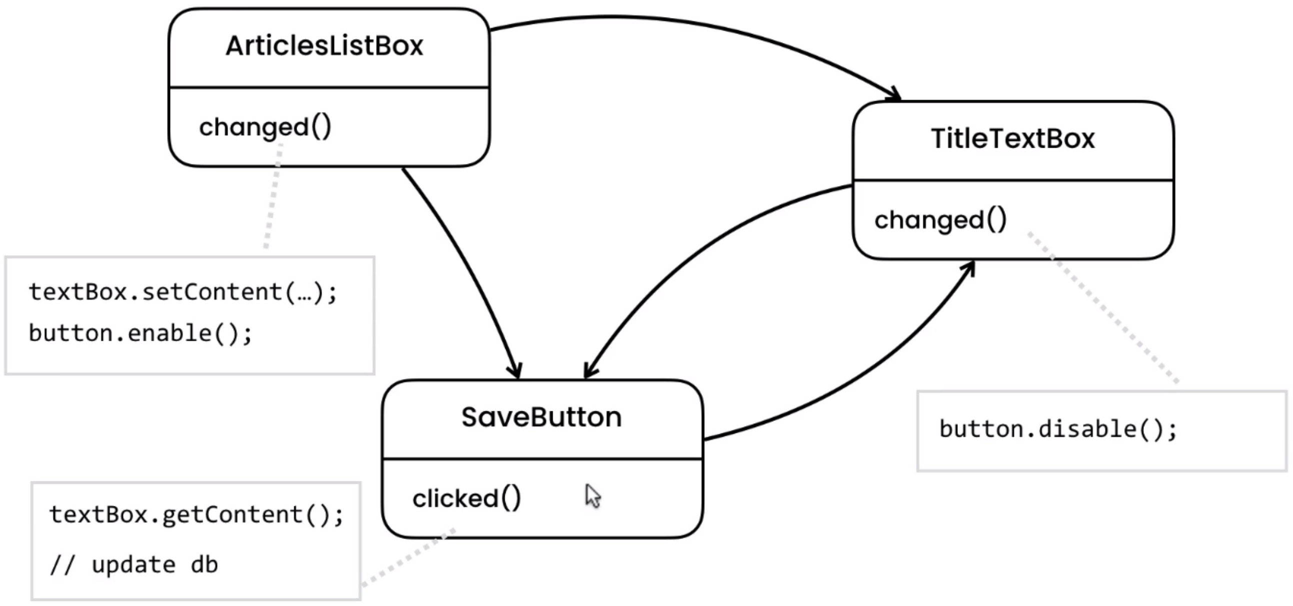
Example Problem

You are developing a form for a mobile app and the save button is disabled if no Articles are selected and enabled if an Article is selected:

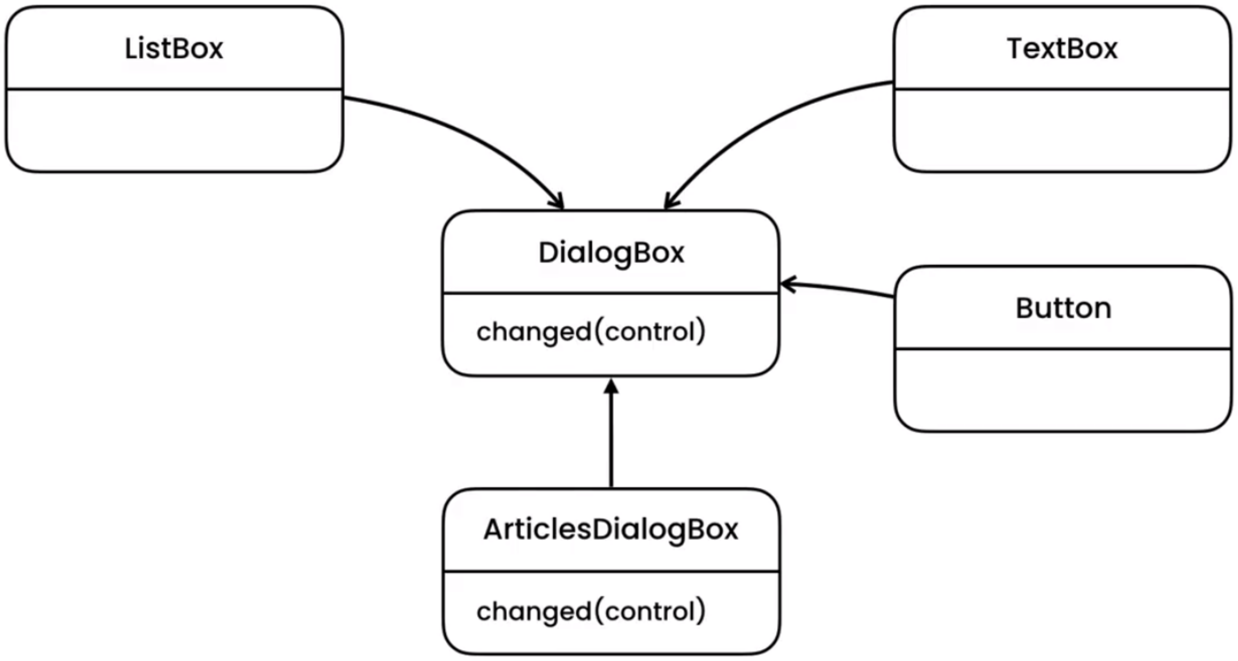
Article 1
Article 2
Article 3
Save

Article 1
Article 2
Article 3
Article 1
Save

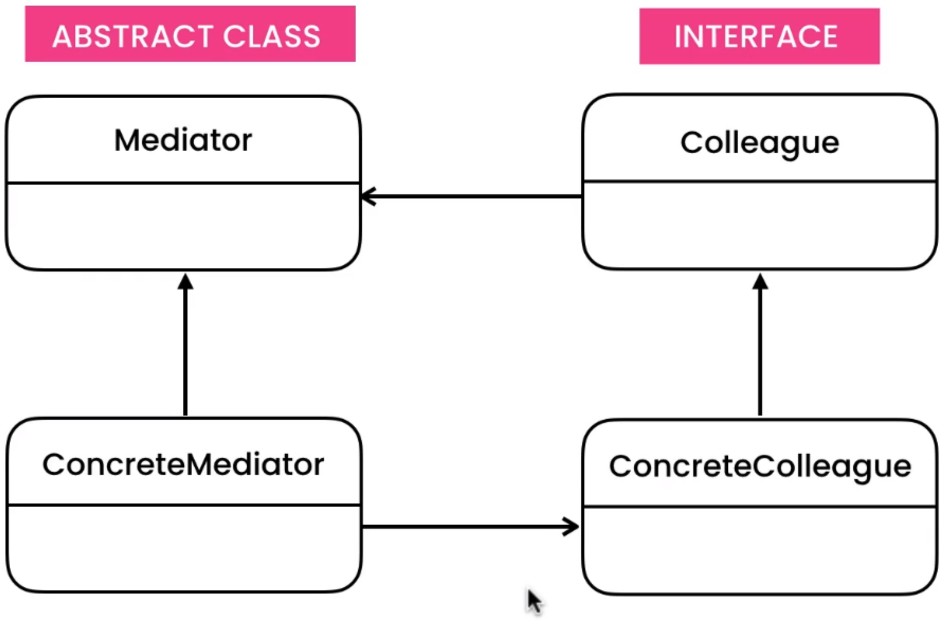
In order to enable and disable the button several classes need to speak to one another and if they speak directly with one another you can end up with several dependencies and tight coupling as shown below :



In order to reduce this coupling you can implement the mediator pattern as below :

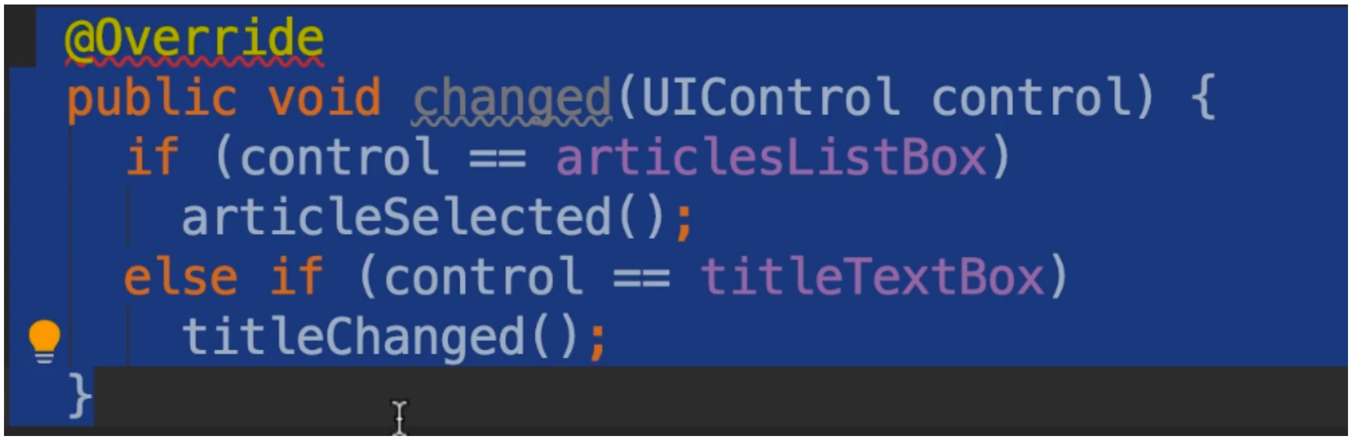


The GOF outline the pattern as follows :



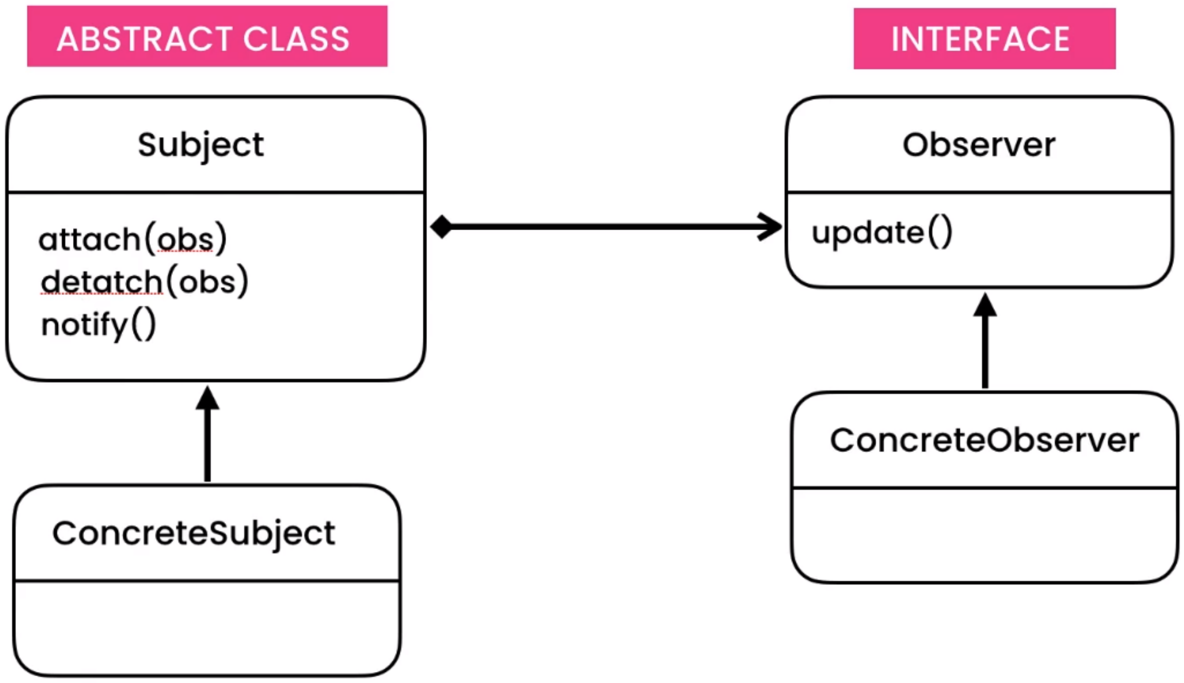
In our example A UIControl class (which can be considered as part of a GUI framework) acts as the colleague and the GUI Elements (ListBox, textBox and Buttton) extend the Colleague. the Dialog Box is set up as the Meadiator and the ArticlesDialogBox is the ConcreteMediator

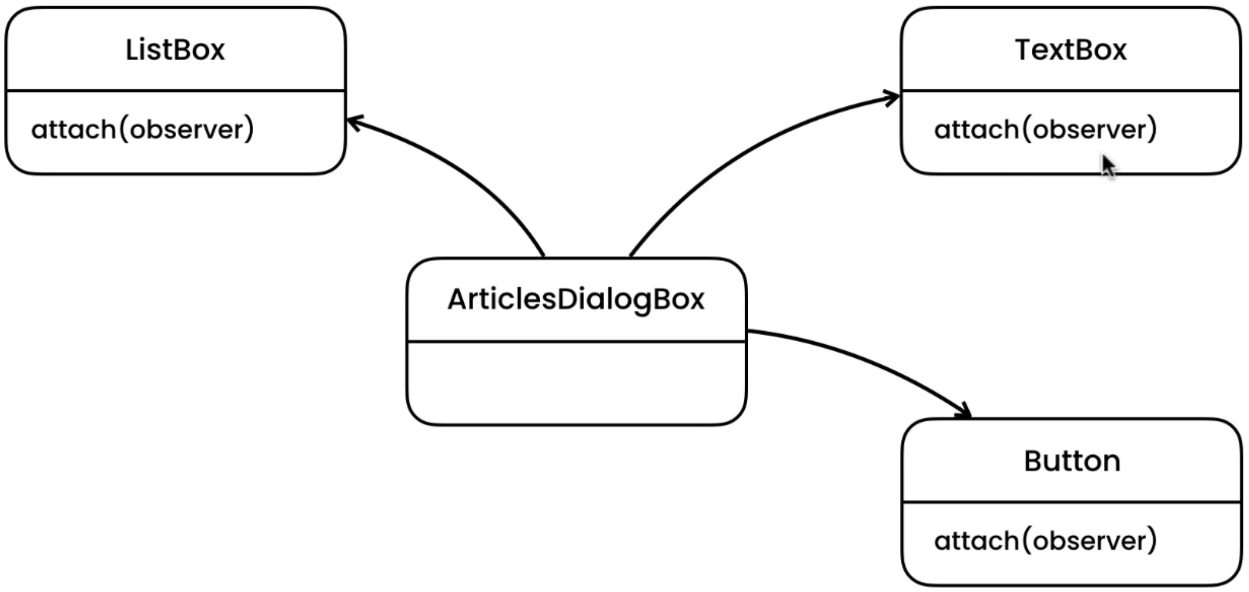
One issue with the above implementation s the need to handle logc in the change method of the concrete mediatior which if there are a lot of GUI elements will become complex and violate the OCP.



```
@Override
public void changed(UIControl control) {
  if (control == articlesListBox)
    articleSelected();
  else if (control == titleTextBox)
    titleChanged();
}
```

To correct this we can implment the mediator pattern with the Observer pattern as below :





The ArticlesDialog Box acts as the Observer removing the need for a DialogBox :


```java
public class ArticlesDialogBox {
  private ListBox articlesListBox = new ListBox();
 private TextBox titleTextBox = new TextBox();
 private Button saveButton = new Button();

 public ArticlesDialogBox() {
    articlesListBox.addEventHandler(this::
articleSelected);
 titleTextBox.addEventHandler(this::titleChanged);
 }

  public void simulateUserInteraction() {
    articlesListBox.setSelection("Article 1");
 titleTextBox.setContent("");
 titleTextBox.setContent("Article 2");
 System.out.println("TextBox: " + titleTextBox.
getContent());
 System.out.println("Button: " + saveButton.
isEnabled());
 }

  private void titleChanged() {
    var content = titleTextBox.getContent();
 var isEmpty = (content == null || content.
isEmpty());
 saveButton.setEnabled(!isEmpty);
 }

  private void articleSelected() {
    titleTextBox.setContent(articlesListBox.
getSelection());
 saveButton.setEnabled(true);
 }
}
```

```java
public abstract class UIControl {
  private List<EventHandler>
eventHandlers = new ArrayList<>();

 public void addEventHandler
(EventHandler observer) {
    eventHandlers.add(observer);
 }

  protected void notifyEventHandle
rs() {
    for (var observer :
eventHandlers)
      observer.handle();
 }
}
```

```java
public
interface Even
tHandler {
  void handle()
;
}
```

```java
public class Button
extends UIControl {
  private boolean i
sEnabled;

 public boolean isE
nabled() {
    return isEnabled
;
 }

  public void setEn
abled(boolean enabl
ed) {
    isEnabled =
enabled;
 notifyEventHandler
s();
 }
}
```

```java
public class ListBox extends UIC
ontrol {
  private String selection;

 public String getSelection() {
    return selection;
 }

  public void setSelection
(String selection) {
    this.selection = selection;
 notifyEventHandlers();
 }
}
```

```java
public class TextBox extends
UIControl {
  private String content;

 public String getContent()
{
    return content;
 }

  public void setContent
(String content) {
    this.content = content;
 notifyEventHandlers();
 }

}
```