"HEAVENS LIGHT IS OUR GUIDE"

RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Lab Report

Course Code: CSE 3206

Course Title: Software Engineering Sessional.

SUBMITTED BY:	**SUBMITTED TO:**
Arafat Hossain (Roll: **2003169**) Fahmida Rahman (Roll: **2003170**) RIFAT HOSEN FAHIM (Roll: **200371**) **Department** : CSE **Section** : C	Farjana Parvin Lecturer, Department of CSE, RUET

Mediator

Mediator is a behavioral design pattern that lets you reduce chaotic dependencies between objects. The pattern restricts direct communications between the objects and forces them to collaborate only via a mediator object.

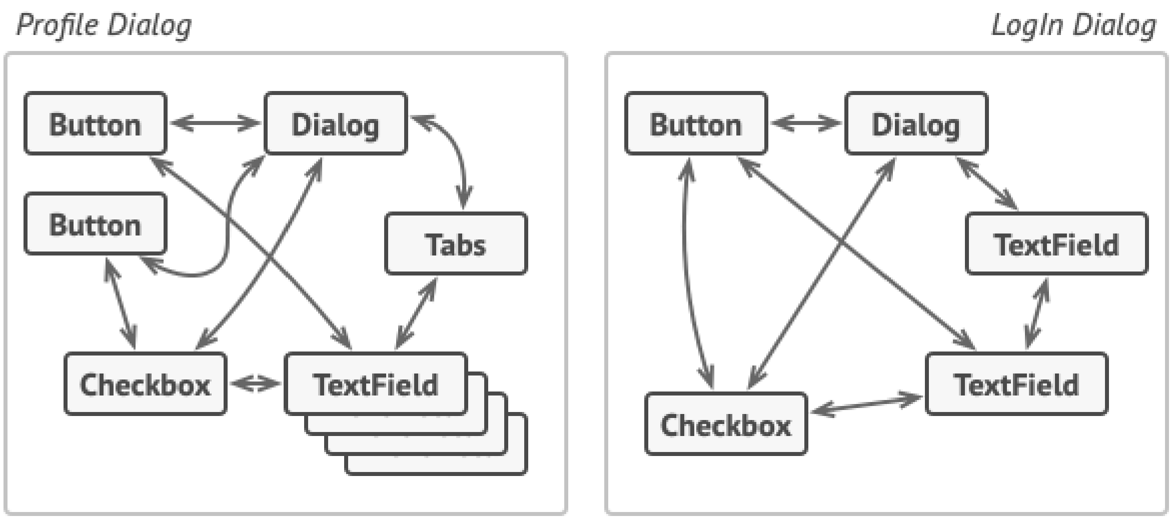


Fig: Chaotic dependency among components.

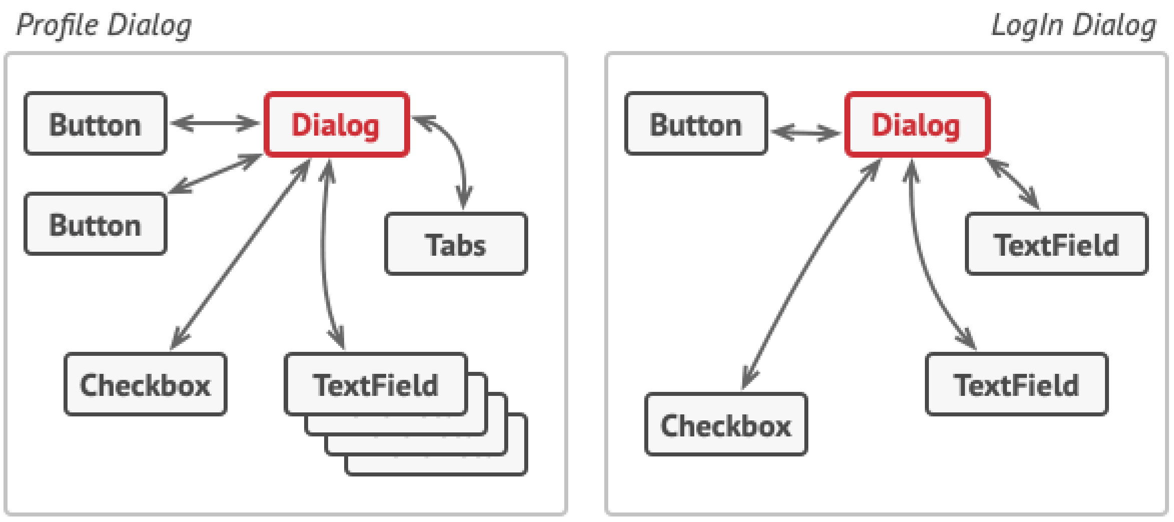


Fig: Objects communicating through the mediator component.

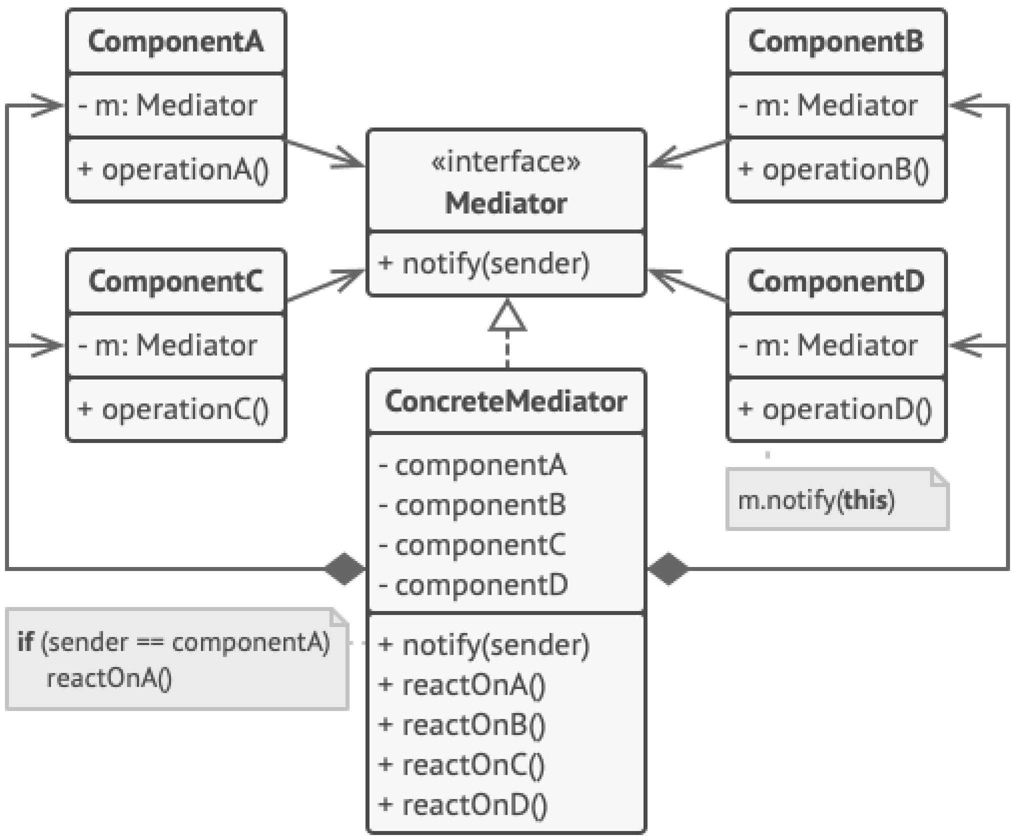


Fig: Example design of a mediator class.

Code:

```
// Mediator.java
public interface Mediator {
    void notify(Component sender, String event);
}

// Component.java
public abstract class Component {
    protected Mediator mediator;

    public Component(Mediator mediator) {
        this.mediator = mediator;
    }
}

// Button.java
public class Button extends Component {
```

```java
    public Button(Mediator mediator) {
        super(mediator);
    }

    public void click() {
        System.out.println("Button clicked.");
        mediator.notify(this, "click");
    }
}
// Textbox.java
public class Textbox extends Component {
    private String text = "";

    public Textbox(Mediator mediator) {
        super(mediator);
    }

    public void setText(String text) {
        this.text = text;
        System.out.println("Textbox set to: " + text);
    }

    public String getText() {
        return text;
    }
}
// Checkbox.java
public class Checkbox extends Component {
    private boolean checked = false;
    public Checkbox(Mediator mediator) {
        super(mediator);
    }

    public void toggle() {
        checked = !checked;
        System.out.println("Checkbox toggled: " + (checked ? "Checked" : "Unchecked"));
        mediator.notify(this, "toggle");
    }

    public boolean isChecked() {
        return checked;
    }
}
// AuthenticationDialog.java
public class AuthenticationDialog implements Mediator {
    private String title;
    private Checkbox loginOrRegisterCheckbox;
    private Textbox loginUsernameTextbox;
    private Textbox loginPasswordTextbox;
    private Textbox registrationUsernameTextbox;
    private Textbox registrationPasswordTextbox;
    private Textbox registrationEmailTextbox;
    private Button okButton;
    private Button cancelButton;

    public AuthenticationDialog() {
        loginOrRegisterCheckbox = new Checkbox(this);
        loginUsernameTextbox = new Textbox(this);
        loginPasswordTextbox = new Textbox(this);
        registrationUsernameTextbox = new Textbox(this);
        registrationPasswordTextbox = new Textbox(this);
        registrationEmailTextbox = new Textbox(this);
        okButton = new Button(this);
        cancelButton = new Button(this);
    }

    @Override
    public void notify(Component sender, String event) {
        if (sender == loginOrRegisterCheckbox && event.equals("toggle")) {
            if (loginOrRegisterCheckbox.isChecked()) {
                title = "Log in";
                System.out.println("Switching to Login Mode.");
                loginUsernameTextbox.setText("Username");
                loginPasswordTextbox.setText("Password");
                registrationUsernameTextbox.setText("");
                registrationPasswordTextbox.setText("");
                registrationEmailTextbox.setText("");
            } else {
                title = "Register";
                System.out.println("Switching to Registration Mode.");
                loginUsernameTextbox.setText("");
                loginPasswordTextbox.setText("");
                registrationUsernameTextbox.setText("New Username");
                registrationPasswordTextbox.setText("New Password");
                registrationEmailTextbox.setText("Email");
            }
        }

        if (sender == okButton && event.equals("click")) {
            if (loginOrRegisterCheckbox.isChecked()) {
                System.out.println("Logging in with: " + loginUsernameTextbox.getText());
                // Add login validation logic here
            } else {
                System.out.println("Registering with: " +
registrationUsernameTextbox.getText());
                // Add registration logic here
            }
        }
    }

    // Public getters for testing
    public Checkbox getLoginOrRegisterCheckbox() {
        return loginOrRegisterCheckbox;
    }

    public Textbox getLoginUsernameTextbox() {
        return loginUsernameTextbox;
    }

    public Textbox getLoginPasswordTextbox() {
        return loginPasswordTextbox;
    }

    public Button getOkButton() {
        return okButton;
    }

    public Button getCancelButton() {
        return cancelButton;
    }
}
```

```
// Main.java
public class Main {
    public static void main(String[] args) {
        AuthenticationDialog dialog = new AuthenticationDialog();
        // Simulate toggling the checkbox
        System.out.println("User toggles 'Login or Register' checkbox:");
        dialog.getLoginOrRegisterCheckbox().toggle();

        // Simulate setting text in the textboxes
        System.out.println("\nUser enters login credentials:");
        dialog.getLoginUsernameTextbox().setText("JohnDoe");
        dialog.getLoginPasswordTextbox().setText("1234");

        // Simulate clicking the OK button
        System.out.println("\nUser clicks OK:");
        dialog.getOkButton().click();

        // Toggle back to Registration Mode
        System.out.println("\nUser toggles back to 'Register' checkbox:");
        dialog.getLoginOrRegisterCheckbox().toggle();

        // Simulate entering registration data
        System.out.println("\nUser enters registration data:");
        dialog.getLoginUsernameTextbox().setText("");
        dialog.getLoginPasswordTextbox().setText("");
        dialog.getOkButton().click();
    }
}
```

Memento

Memento is a behavioral design pattern that saves and restores the previous state of an object without revealing the details of its implementation.Also known as snapshot design pattern.

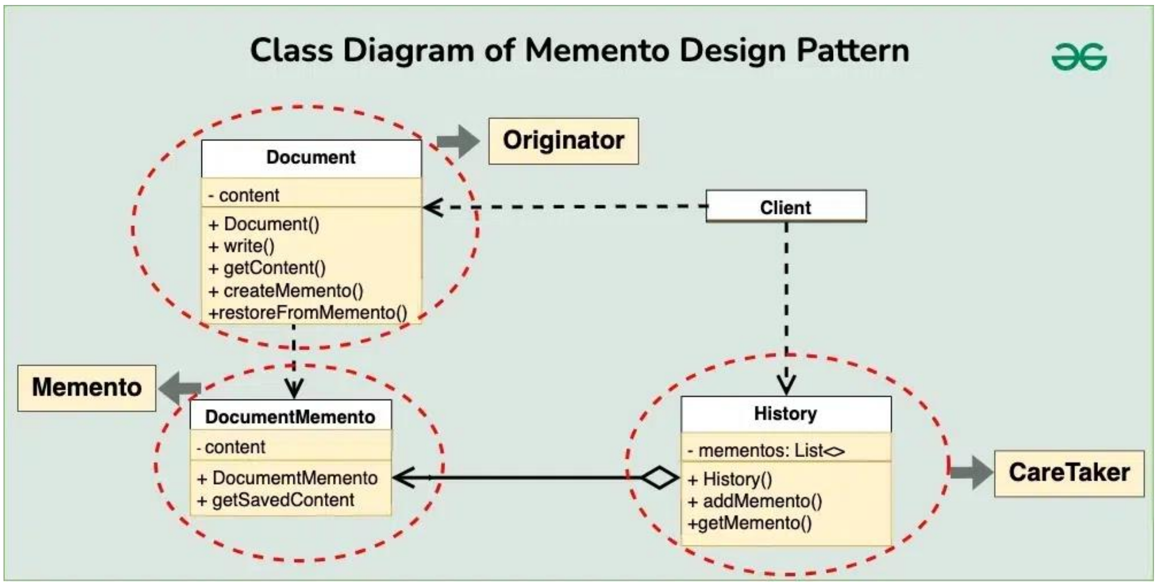


Fig: UML of Memento design pattern.

It is mainly consists of three components :

1. Originator:Mainly the object for which state need to be saved
2. Memento: Holds the state of the originator.
3. Caretaker :Holds the list of the memento.

Advantages :

1. Undo/Redo Support: Makes it easy to implement undo/redo functionality.
2. Saves and restores the previous state of an object .
3. Allows for tracking and restoring multiple states over time.
4. Encapsulation: Keeps the internal state hidden from the outside.

Disadvantages:

1. Increased Complexity: Managing multiple mementos can complicate the system.
2. High Memory Consumption: Storing many mementos increases memory usage.
3. Not Suitable for Complex Objects: Can be impractical for highly dynamic or complex objects.

4. Risk of Data Loss: If mementos are not saved or restored correctly, data can be lost.

Code:

memento/Originator.java

```java
class Originator {
    private int height;
    private int width;

    // Constructor
    public Originator(int height, int width) {
        this.height = height;
        this.width = width;
    }

    // Setters
    public void setHeight(int height) {
        this.height = height;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    // Getters
        public int getHeight() {
        return height;
    }

    public int getWidth() {
        return width;
    }

    // Create a new Memento
    public Memento createMemento() {
        return new Memento(this.height, this.width);
    }

    // Restore state from a Memento
    public void restoreMemento(Memento mementoToRestore) {
        this.height = mementoToRestore.getHeight();
        this.width = mementoToRestore.getWidth();
    }
}
```

memento

memento/Memento.java

```java
class Memento {
    private  int height;
    private  int width;

    // Constructor
    public Memento(int height, int width) {
        this.height = height;
        this.width = width;
    }
```

```java
    // Getters
    public int getHeight() {
        return height;
    }
```

```java
    public int getWidth() {
        return width;
    }
}
```

memento/CareTaker.java

```java
import java.util.ArrayList;
import java.util.List;

class Caretaker {
    private final List<Memento> history = new ArrayList<>();
```

```java
    // Add a Memento to history
    public void addMemento(Memento memento) {
        history.add(memento);
    }
```

```java
    // Undo and return the last saved Memento
    public Memento undo() {
        if (!history.isEmpty()) {
            int lastMementoIndex = history.size() - 1;
            Memento lastMemento = history.get(lastMementoIndex);
            history.remove(lastMementoIndex);
            return lastMemento;
        }
        return null;
    }
}
```

memento/MementoApplication.java

```java


import java.util.ArrayList;
import java.util.List;
```

```java
public class MementoApplication {
    public static void main(String[] args) {
        Caretaker caretaker = new Caretaker();
}

```

```java
        // Create first snapshot and add to caretaker
        Memento snapshot1 = originator.createMemento();
        caretaker.addMemento(snapshot1);
```

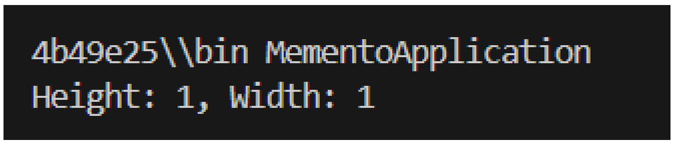
```java
        // Modify state
        originator.setHeight(1);
        originator.setWidth(1);
```

```java
        // Create second snapshot and add to caretaker
        Memento snapshot2 = originator.createMemento();
        caretaker.addMemento(snapshot2);
```

```java
        // Modify state again
        originator.setHeight(2);
        originator.setWidth(2);
```

```java
        // Undo: Restore the last saved state
        Memento restoreMemento = caretaker.undo();
        if (restoreMemento != null) {
            originator.restoreMemento(restoreMemento);
        }
        // Print restored state
        System.out.println("Height: " + originator.getHeight() + ", Width: " +
originator.getWidth());
    }
}
```

Output :



```
4b49e25\\bin MementoApplication
Height: 1, Width: 1
```