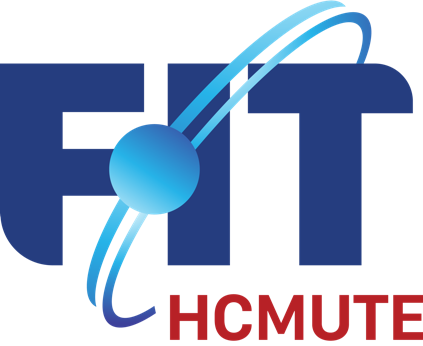
**A white rectangular frame with blue border

Description automatically generatedTRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

*Đề tài:*

**MEMENTO PATTERN**

**Môn:** Mẫu Thiết Kế Phần Mềm

**Mã lớp:** DEPA330879\_03CLC

**GVHD:** ThS. Nguyễn Minh Đạo

**SVTH:** Nguyễn Hà Quỳnh Giao

**MSSV:** 21110171

*TP. Hồ Chí Minh, tháng 05 năm 2024*

MỤC LỤC

[**MỞ ĐẦU** 1](#_Toc166111764)

[**1.** **Khái niệm:** 1](#_Toc166111765)

[**2.** **Cấu trúc:** 1](#_Toc166111766)

[**3.** **Trường hợp sử dụng:** 1](#_Toc166111767)

[**4.** **Ưu điểm:** 1](#_Toc166111768)

[**5.** **Nhược điểm:** 1](#_Toc166111769)

[**THIẾT KẾ** 2](#_Toc166111770)

[**1.** **Ý tưởng thiết kế:** 2](#_Toc166111771)

[**2.** **Giải pháp sử dụng mẫu Memento:** 2](#_Toc166111772)

[**3.** **Thiết kế các lớp:** 2](#_Toc166111773)

[**3.1.** **Lớp Orginator:** 2](#_Toc166111774)

[**3.2.** **Memento:** 3](#_Toc166111775)

[**3.3.** **Caretaker:** 3](#_Toc166111776)

[**4.** **Sử dụng:** 4](#_Toc166111777)

[**5.** **Phân tích:** 4](#_Toc166111778)

[**KẾT LUẬN** 5](#_Toc166111779)

[**TÀI LIỆU THAM KHẢO** 6](#_Toc166111780)

**MỞ ĐẦU**

1. **Khái niệm:**

Memento là một mẫu thiết kế hành vi (behavioral design pattern) cho phép bạn lưu và khôi phục trạng thái trước đó của một đối tượng mà không tiết lộ chi tiết về cách thức triển khai của nó.

1. **Cấu trúc:**

Một mẫu Mementor bao gồm:

* Orginator: Đối tượng hoạt động để lưu trữ và khôi phục.
* Mementor: chứa trạng thái của Orginator, cung cấp các phương thức trả về, không cho phép thay đổi ở bên trong.
* Caretaker: quản lý việc lưu trữ và khôi phục của Mementor nhưng không biết về cấu trúc bên trong của Mememtor.

1. **Trường hợp sử dụng:**

* Muốn tạo ra các bản ghi của trạng thái của đối tượng để có thể khôi phục trạng thái trước đó của đối tượng.
* Nếu truy cập trực tiếp vào các trường, phương thức getter/setter của đối tượng vi phạm tính đóng gói của nó.

1. **Ưu điểm:**

* Đơn giản hóa mã của originator bằng cách để caretaker duy trì lịch sử của trạng thái của người khởi tạo.
* Tạo bảng ghi của trạng thái của đối tượng mà không vi phạm tính đóng gói của nó.

1. **Nhược điểm:**

* Tiêu tốn nhiều RAM nếu tạo các memento quá thường xuyên.
* Theo thời gian tồn tại bản ghi lỗi thời cần loại bỏ.
* Hầu hết các ngôn ngữ lập trình động như PHP, Python và JavaScript không đảm bảo rằng trạng thái bên trong bản ghi (memento) sẽ không bị thay đổi.

**THIẾT KẾ**

1. **Ý tưởng thiết kế:**

Trong bài toán này, hệ thống xây dựng trình soạn thảo văn bản.

Ngôn ngữ lập trình: Java

1. **Giải pháp sử dụng mẫu Memento:**

A screenshot of a computer program

Description automatically generated

* Lớp Orginator:TextEditor – duy trì trạng thái cần lưu của memento
* Memento: TextEditorMemento – lưu trữ trạng thái của TextEditor tại một thời điểm.
* Caretaker: Caretaker – quản lý việc lưu trữ và khôi phục TextEditor.

1. **Thiết kế các lớp:**
   1. **Lớp Orginator:**

TextEditor.java

package org.example;  
package org.example;  
  
public class TextEditor {  
  
 private String text;  
  
 public String getText(){  
 return this.text;  
 }  
  
 public void setText(String text){  
 this.text = text;  
 }  
  
 public TextEditor(){  
 this.text = "";  
 }  
  
 public TextEditorMemento save(){  
 return new TextEditorMemento(text);  
 }  
  
 public void restore(TextEditorMemento textEditorMemento){  
 this.text = textEditorMemento.getState();  
 }  
}

* 1. **Memento:**

TextEditorMemento.java

package org.example;  
  
public class TextEditorMemento {  
 private String state;  
  
 public String getState() {  
 return state;  
 }  
  
 public TextEditorMemento(String state){  
 this.state = state;  
 }  
}

* 1. **Caretaker:**

Caretaker.java

package org.example;  
  
public class Caretaker {  
 private TextEditorMemento memento;  
 public Caretaker(){  
 this.memento = null;  
 }  
  
 public void save(TextEditor textEditor){  
 this.memento = textEditor.save();  
 }  
  
 public void undo(TextEditor textEditor){  
 if(memento != null){  
 textEditor.restore(memento);  
 this.memento = null;  
 } else {  
 System.*out*.println("No sate to restore");  
 }  
 }  
}

1. **Sử dụng:**

Main.java

package org.example;  
  
public class Main {  
 public static void main(String[] args) {  
 TextEditor textEditor = new TextEditor();  
 Caretaker caretaker = new Caretaker();  
  
 textEditor.setText("Text1");  
 System.*out*.println(textEditor.getText());  
 caretaker.save(textEditor);  
 textEditor.setText("Text2");  
 System.*out*.println(textEditor.getText());  
 caretaker.undo(textEditor);  
 System.*out*.println(textEditor.getText());  
 }  
}

1. **Phân tích:**

Mã nguồn triển khai mẫu thiết kế Memento pattern trong Java đưa ra một cách tổ chức rõ ràng và hiệu quả. Lớp TextEditor đóng vai trò là Originator, duy trì và quản lý trạng thái của nó. Lớp TextEditorMemento là Memento, chứa trạng thái đã lưu của TextEditor. Caretaker là người chăm sóc, quản lý việc lưu trữ và khôi phục Memento của TextEditor. Các phương thức save() và restore() trong TextEditor được sử dụng để lưu trạng thái vào Memento và khôi phục từ Memento đã lưu. Qua đó, mã nguồn này cung cấp một cách tiếp cận linh hoạt và dễ bảo trì để quản lý trạng thái của đối tượng.

**KẾT LUẬN**

Memento pattern là một mẫu thiết kế linh hoạt và mạnh mẽ trong lập trình hướng đối tượng, cho phép lưu trữ và khôi phục trạng thái của một đối tượng mà không làm giảm tính nhất quán của hệ thống. Bằng cách này, nó giúp bảo vệ tính nhất quán của trạng thái, dễ dàng mở rộng và giảm kết nối giữa các thành phần. Tuy nhiên, cần cẩn trọng về tăng dung lượng bộ nhớ và hiệu suất trong quá trình sử dụng. Tóm lại, Memento pattern là một công cụ mạnh mẽ để quản lý trạng thái của các đối tượng trong các hệ thống lớn và phức tạp, với điều kiện sử dụng cẩn thận để đảm bảo tính nhất quán và hiệu suất.

**TÀI LIỆU THAM KHẢO**

[1] Refactoring. Mementor. Refactoring. https://refactoring.guru/design-patterns/memento (ngày truy cập 09-05-2024)