

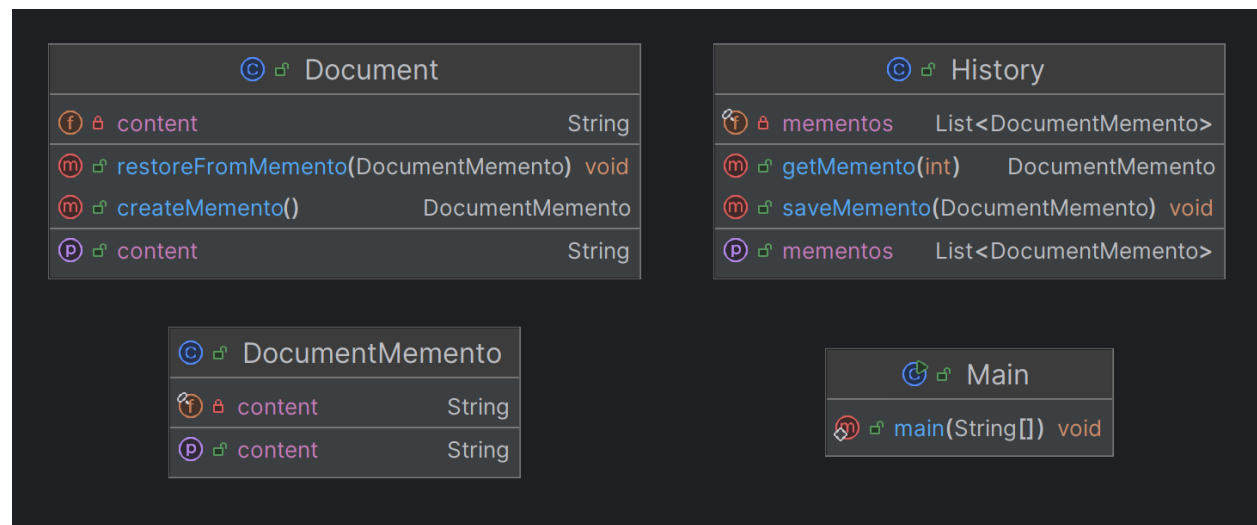
Họ và tên: Nguyễn Khánh Quy

MSSV: 21110282

Memento Pattern

Mẫu **Memento** trong lập trình hướng đối tượng là một mẫu thiết kế hành vi được sử dụng để quản lý và khôi phục trạng thái trước đó của một đối tượng. Mẫu này giúp lưu trữ trạng thái của một đối tượng mà không làm ảnh hưởng đến tính toàn vẹn của đối tượng đó. Điều này giúp tạo ra một cách cấu trúc linh hoạt và dễ dàng mở rộng để xử lý việc lưu trữ và khôi phục trạng thái của đối tượng trong các tình huống phức tạp.

Dưới đây là một ví dụ về mẫu **Memento** được triển khai bằng ngôn ngữ lập trình Java. Trong ví dụ này, chúng ta sẽ thiết kế một hệ thống quản lý tài liệu, cho phép người dùng lưu trữ và khôi phục các phiên bản trước đó của tài liệu một cách dễ dàng.



Bước 1: Tạo lớp Memento (DocumentMemento):

Lớp này đại diện cho đối tượng được sử dụng để lưu trữ trạng thái của tài liệu.

Trong ví dụ, **DocumentMemento** chứa một trường **content** để lưu trữ nội dung của tài liệu.

Lớp này cung cấp một phương thức **getContent()** để truy xuất nội dung của tài liệu.

```
public class DocumentMemento {  
  
    private final String content;  
  
    public DocumentMemento(String content) {  
        this.content = content;  
    }  
  
    public String getContent() {  
        return content;  
    }  
  
}
```

Bước 2: Tạo lớp Originator (Document):

Lớp này đại diện cho tài liệu mà chúng ta muốn lưu trữ và khôi phục trạng thái của nó.

Trong ví dụ, **Document** chứa một trường **content** để lưu trữ nội dung của tài liệu.

Lớp này cung cấp hai phương thức quan trọng: **createMemento()** để tạo một **memento** mới từ trạng thái hiện tại của tài liệu và **restoreFromMemento()** để khôi phục lại trạng thái của tài liệu từ một **memento** đã cho.

```
public class Document {  
  
    private String content;  
  
    public void setContent(String content) {  
        this.content = content;  
    }  
  
    public String getContent() {  
        return content;  
    }  
  
}
```

```

    }

    // Tạo memento từ trạng thái hiện tại của tài liệu
    public DocumentMemento createMemento() {
        return new DocumentMemento(content);
    }

    // Khôi phục trạng thái từ memento
    public void restoreFromMemento(DocumentMemento memento) {
        content = memento.getContent();
    }
}

```

Bước 3: Tạo lớp Caretaker (History):

Lớp này chịu trách nhiệm quản lý các **memento** của tài liệu.

Trong ví dụ, **History** duy trì một danh sách các **memento** được lưu trữ.

Lớp này cung cấp các phương thức quan trọng: **saveMemento()** để lưu trữ một **memento** mới vào danh sách, **getMemento()** để truy xuất một **memento** từ danh sách theo chỉ mục và **getMementos()** để truy xuất tất cả **memento**.

```

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

public class History {

    private final List<DocumentMemento> mementos = new ArrayList<>();

    public void saveMemento(DocumentMemento memento) {
        mementos.add(memento);
    }

    public DocumentMemento getMemento(int index) {
        return mementos.get(index);
    }

    public List<DocumentMemento> getMementos() {
        return mementos;
    }
}

```

Bước 4: Thử nghiệm:

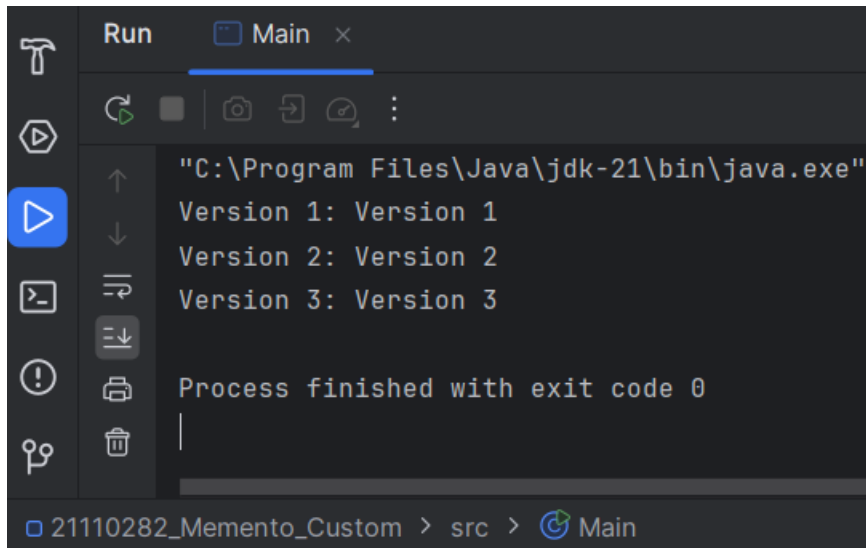
Trong hàm **main**, chúng ta tạo một đối tượng tài liệu và một đối tượng lịch sử.

Chúng ta thêm các phiên bản mới của tài liệu vào lịch sử bằng cách tạo một **memento** từ trạng thái hiện tại của tài liệu và lưu trữ nó trong danh sách **memento** của lịch sử.

Sau đó, chúng ta duyệt qua danh sách các **memento** trong lịch sử và sử dụng mỗi **memento** để khôi phục lại trạng thái của tài liệu từ mỗi phiên bản đã lưu trữ, in ra các phiên bản đã lưu trữ.

```
public class Main {  
  
    public static void main(String[] args) {  
        // Tạo một tài liệu  
        Document document = new Document();  
  
        // Lịch sử lưu trữ các phiên bản của tài liệu  
        History history = new History();  
  
        // Thêm các phiên bản mới của tài liệu và lưu trữ chúng trong lịch sử  
        document.setContent("Version 1");  
        history.saveMemento(document.createMemento());  
  
        document.setContent("Version 2");  
        history.saveMemento(document.createMemento());  
  
        document.setContent("Version 3");  
        history.saveMemento(document.createMemento());  
  
        // In ra tất cả các phiên bản của tài liệu từ lịch sử  
        for (int i = 0; i < history.getMementos().size(); i++) {  
            document.restoreFromMemento(history.getMemento(i));  
            System.out.println("Version " + (i + 1) + ": " +  
document.getContent());  
        }  
    }  
}
```

Kết quả:



```
Run Main x
"C:\Program Files\Java\jdk-21\bin\java.exe"
Version 1: Version 1
Version 2: Version 2
Version 3: Version 3
Process finished with exit code 0
21110282_Memento_Custom > src > Main
```

Trong ví dụ này, chúng ta sử dụng mẫu **Memento** để quản lý trạng thái của tài liệu. Khi tài liệu thay đổi, chúng ta lưu trữ trạng thái hiện tại của nó vào một đối tượng **memento** và lưu trữ **memento** đó trong lịch sử. Điều này giúp chúng ta có thể khôi phục lại trạng thái trước đó của tài liệu từ **memento** tương ứng, cung cấp tính năng hoàn tác hoặc điều chỉnh trong ứng dụng của chúng ta.