

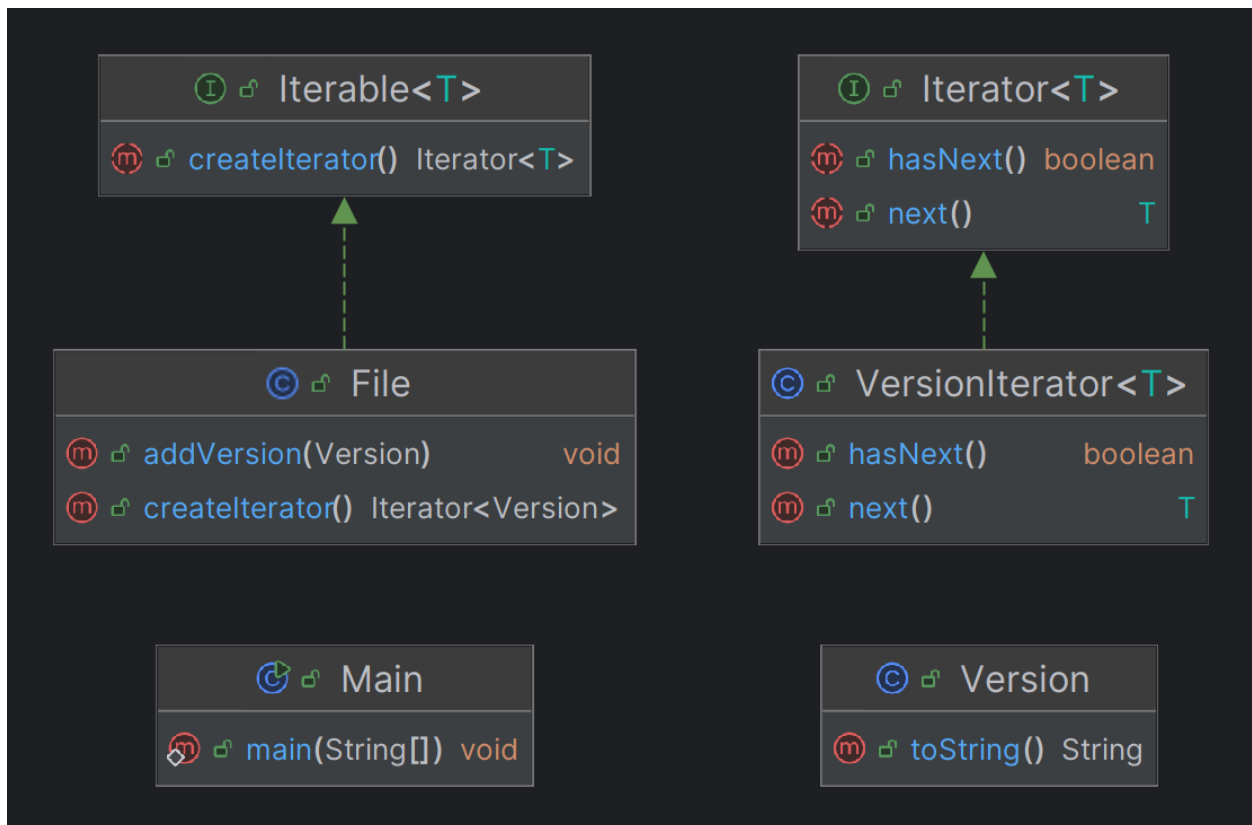
Họ và tên: Nguyễn Khánh Quy

MSSV: 21110282

## Iterator Pattern

Mẫu **Iterator** trong lập trình hướng đối tượng là một mẫu thiết kế hành vi được sử dụng để duyệt qua các phần tử của một tập hợp mà không cần biết cấu trúc nội bộ của tập hợp đó. **Iterator** cho phép chúng ta duyệt qua các phần tử một cách tuần tự mà không cần tiếp cận trực tiếp vào các phần tử hoặc biết về cách triển khai nội bộ của tập hợp.

Dưới đây là một ví dụ về mẫu **Iterator** bằng ngôn ngữ lập trình Java. Trong ví dụ này, chúng ta sẽ thiết kế một hệ thống quản lý tài liệu đơn giản. **Iterator** sẽ được sử dụng để duyệt qua các phiên bản của tài liệu một cách dễ dàng và linh hoạt.



## Bước 1: Tạo lớp (Version):

Lớp này đại diện cho mỗi phiên bản của tài liệu, lưu trữ nội dung và thời điểm tạo ra phiên bản.

```
public class Version {  
  
    String content;  
    String timestamp;  
  
    public Version(String content, String timestamp) {  
        this.content = content;  
        this.timestamp = timestamp;  
    }  
  
    @Override  
    public String toString() {  
        return "Version{" +  
            "content='" + content + '\'' +  
            ", timestamp='" + timestamp + '\'' +  
            '}';  
    }  
}
```

## Bước 2: Tạo Interface (Iterable, Iterator):

Interface **Iterable** định nghĩa phương thức **createIterator()** để tạo ra một **Iterator**.  
Interface **Iterator** định nghĩa các phương thức **hasNext()** để kiểm tra xem có phần tử tiếp theo không và **next()** để trả về phần tử tiếp theo.

```
public interface Iterable<T> {  
  
    Iterator<T> createIterator();  
  
}
```

```
public interface Iterator<T> {  
  
    boolean hasNext();  
    T next();  
  
}
```

### Bước 3: Triển khai lớp (VersionIterator):

Lớp **VersionIterator** triển khai interface **Iterator**. Nó duyệt qua danh sách các phiên bản của tài liệu và cung cấp các phương thức để kiểm tra xem còn phiên bản nào khác không và trả về phiên bản tiếp theo.

```
import java.util.List;

public class VersionIterator<T> implements Iterator<T> {

    private final List<Version> versions;
    private int position;

    public VersionIterator(List<Version> versions) {
        this.versions = versions;
        this.position = 0;
    }

    // Kiểm tra xem còn phiên bản tiếp theo hay không
    public boolean hasNext() {
        return position < versions.size();
    }

    // Trả về phiên bản tiếp theo
    public T next() {
        if (hasNext()) {
            T version = (T) versions.get(position);
            position++;
            return version;
        }
        throw new IllegalStateException("No more versions available.");
    }
}
```

### Bước 4: Tạo lớp (File):

Lớp **File** đại diện cho tài liệu và triển khai giao diện **Iterable**. Nó lưu trữ danh sách các phiên bản của tài liệu và cung cấp một phương thức để tạo ra một **Iterator** để duyệt qua các phiên bản này.

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

public class File implements Iterable<Version> {

    private final List<Version> versions;
```

```

public File() {
    versions = new ArrayList<>();
}

public void addVersion(Version version) {
    versions.add(version);
}

@Override
public Iterator<Version> createIterator() {
    return new VersionIterator<>(versions);
}
}

```

## Bước 5: Thử nghiệm

Trong hàm **main**, chúng ta tạo một tài liệu, thêm các phiên bản vào đó, sau đó tạo một **Iterator** và duyệt qua các phiên bản, in ra thông tin của từng phiên bản.

```

public class Main {

    public static void main(String[] args) {
        // Tạo một tài liệu
        File file = new File();

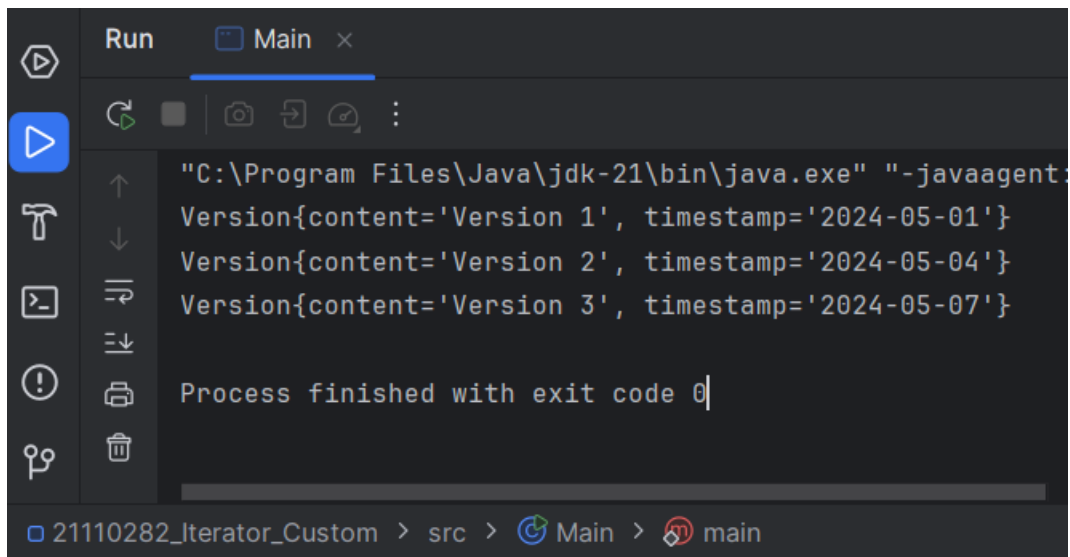
        // Thêm các phiên bản vào tài liệu
        file.addVersion(new Version("Version 1", "2024-05-01"));
        file.addVersion(new Version("Version 2", "2024-05-04"));
        file.addVersion(new Version("Version 3", "2024-05-07"));

        // Tạo Iterator
        Iterator<Version> iterator = file.createIterator();

        // Duyệt qua các phiên bản và in ra thông tin của từng phiên bản
        while (iterator.hasNext()) {
            Version version = iterator.next();
            System.out.println(version);
        }
    }
}

```

## Kết quả:



```
Run Main x
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:
Version{content='Version 1', timestamp='2024-05-01'}
Version{content='Version 2', timestamp='2024-05-04'}
Version{content='Version 3', timestamp='2024-05-07'}
Process finished with exit code 0
21110282_Iterator_Custom > src > Main > main
```

Trong ví dụ này, chúng ta sử dụng mẫu **Iterator** để duyệt qua danh sách các phiên bản của một tài liệu. Mỗi phiên bản của tài liệu được biểu diễn bằng một đối tượng **Version**, có chứa thông tin về nội dung và thời gian tạo. Mẫu **Iterator** giúp chúng ta truy cập các phiên bản này một cách dễ dàng và linh hoạt, cho phép chúng ta thực hiện các thao tác trên từng phiên bản một cách thuận tiện. Điều này tạo ra một cách cấu trúc linh hoạt cho việc quản lý các phiên bản và hỗ trợ việc xử lý các trạng thái và hành vi phức tạp của tài liệu.