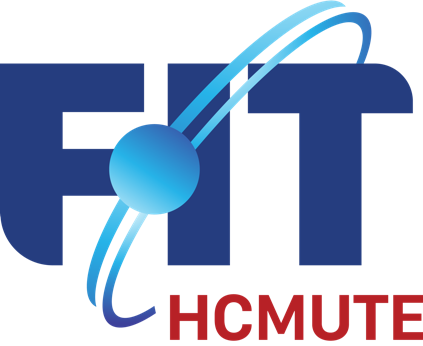
**A white rectangular frame with blue border

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

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

*Đề tài:*

**TEMPLATE METHOD 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 04 năm 2024*

MỤC LỤC

[**MỞ ĐẦU** 1](#_Toc166107648)

[**1.** **Khái niệm:** 1](#_Toc166107649)

[**2.** **Cấu trúc:** 1](#_Toc166107650)

[**3.** **Trường hợp sử dụng:** 1](#_Toc166107651)

[**4.** **Ưu điểm:** 1](#_Toc166107652)

[**5.** **Nhược điểm:** 1](#_Toc166107653)

[**THIẾT KẾ** 3](#_Toc166107654)

[**1.** **Ý tưởng thiết kế:** 3](#_Toc166107655)

[**2.** **Giải pháp sử dụng Template Method Pattern:** 3](#_Toc166107656)

[**3.** **Thiết kế các lớp:** 4](#_Toc166107657)

[**3.1.** **Lớp abstract:** 4](#_Toc166107658)

[**3.2.** **Lớp concrete:** 4](#_Toc166107659)

[**4.** **Sử dụng:** 5](#_Toc166107660)

[**5.** **Phân tích:** 5](#_Toc166107661)

[**KẾT LUẬN** 6](#_Toc166107662)

[**TÀI LIỆU THAM KHẢO** 7](#_Toc166107663)

# **MỞ ĐẦU**

1. **Khái niệm:**

Template Method là một mẫu thiết kế hành vi (behavioral design pattern) mà xác định cấu trúc của một thuật toán trong lớp cha, nhưng cho phép các lớp con ghi đè các bước cụ thể của thuật toán mà không thay đổi cấu trúc của nó.

1. **Cấu trúc:**

Một mẫu Template Method Pattern bao gồm:

* Lớp abstract: định nghĩa các phương thức mẫu, đại diện cho thuật toán chung. Chứa các bước cơ bản sử dụng thuật toán, sử dụng các phương thức trừu tượng hoặc định nghĩa những phương thức để thực hiện một thuật toán cụ thể. Ngoài ra còn có thể chứa một số phương thức mà lớp kế thừa không thể đè lên.
* Lớp Concrete: lớp kế thừa lớp abstract, định nghĩa cụ thể các phương thức một cách cụ thể.

1. **Trường hợp sử dụng:**

* Muốn mở rộng chỉ các bước cụ thể của một thuật toán, nhưng không phải toàn bộ thuật toán hoặc cấu trúc của nó.
* Một số lớp chứa các thuật toán gần như giống nhau với một số khác biệt nhỏ.

1. **Ưu điểm:**

* Việc ghi đè chỉ những phần cụ thể của một thuật toán lớn, giúp họ ít bị ảnh hưởng bởi những thay đổi xảy ra trong các phần khác của thuật toán.
* Có thể đưa mã trùng lặp vào một lớp cha.

1. **Nhược điểm:**

* Bị hạn chế bởi cấu trúc mẫu (skeleton) được cung cấp của một thuật toán.
* Việc ghi đè lên một phần triển khai mặc định của bước trong thuật toán thông qua một lớp con có thể vi phạm Nguyên tắc Thay thế Liskov. Nguyên tắc này đề cao tính tương thích và khả năng thay thế của các lớp con với lớp cha của chúng. Nếu một lớp con loại bỏ hoặc thay đổi một phần của triển khai mặc định mà không tuân theo cách cấu trúc tổng quát đã được định nghĩa trong lớp cha, có thể dẫn đến việc không thể thay thế lớp cha bằng lớp con một cách an toàn.
* Có xu hướng khó bảo trì hơn khi có nhiều bước hơn.

# **THIẾT KẾ**

1. **Ý tưởng thiết kế:**

Trong bài toán này, hệ thống xây dựng framework cho dịch vụ mạng sử dụng 2 loại giao thức HTTP và FTP.

Ngôn ngữ lập trình: Java

1. **Giải pháp sử dụng Template Method Pattern:**

A diagram of a server

Description automatically generated

* Lớp abstract: NetworkService – khai báo các phương thức trừu tượng và tạo cấu trúc của thuật toán.
* Lớp concrete: HttpService và FtpService – lớp con kế thừa, định nghĩa các cụ thể các phương thức trừu tượng từ lớp kế thừa.

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

NetworkService.java

package org.example;  
  
public abstract class NetworkService {  
 public void handlerRequest(String request){  
 establishConnect();  
 if(authenticate())  
 execute(request);  
 closeConnection();  
 };  
 protected abstract void establishConnect();  
 protected abstract boolean authenticate();  
 protected abstract void execute(String request);  
 protected abstract void closeConnection();  
}

* 1. **Lớp concrete:**

HttpService.java

package org.example;  
  
public class HttpService extends NetworkService{  
 @Override  
 protected void establishConnect() {  
 System.*out*.println("Establishing Http connect.");  
 }  
  
 @Override  
 protected boolean authenticate() {  
 System.*out*.println("Authenticating Http connect.");  
 return true;  
 }  
  
 @Override  
 protected void execute(String request) {  
 System.*out*.println("Executing Http connect.");  
 }  
  
 @Override  
 protected void closeConnection() {  
 System.*out*.println("Closing Http connect.");  
 }  
}

FtpService.java

package org.example;  
  
public class FtpService extends NetworkService{  
 @Override  
 protected void establishConnect() {  
 System.*out*.println("Establishing Ftp connect.");  
 }  
  
 @Override  
 protected boolean authenticate() {  
 System.*out*.println("Authenticating Ftp connect.");  
 return true;  
 }  
  
 @Override  
 protected void execute(String request) {  
 System.*out*.println("Executing Ftp connect.");  
 }  
  
 @Override  
 protected void closeConnection() {  
 System.*out*.println("Closing Ftp connect.");  
 }  
}

1. **Sử dụng:**

Main.java

package org.example;  
  
public class Main {  
 public static void main(String[] args) {  
 NetworkService http = new HttpService();  
 NetworkService ftp = new FtpService();  
  
 http.handlerRequest("Get /home");  
 ftp.handlerRequest("Get /file");  
 }  
}

1. **Phân tích:**

Với ví dụ trên, ta thấy rằng mẫu Template Method đã giúp tạo ra một cấu trúc tổ chức cho việc xử lý các yêu cầu mạng trong hệ thống. Bằng cách định nghĩa một khuôn mẫu chung trong lớp abstract NetworkService, mẫu này đã cho phép tái sử dụng mã nguồn và tạo ra tính linh hoạt trong việc triển khai các dịch vụ mạng cụ thể như HTTP và FTP. Mỗi lớp con như HttpService và FtpService chỉ cần triển khai các phương thức cụ thể liên quan đến việc thiết lập kết nối, xác thực, thực thi và đóng kết nối, trong khi các phần cốt lõi của quy trình được xử lý bởi phương thức template handlerRequest(). Nhờ vào mẫu này, việc thêm các dịch vụ mạng mới trong tương lai hoặc thay đổi cách thức xử lý của chúng trở nên dễ dàng hơn mà không làm ảnh hưởng đến cấu trúc chung của hệ thống, tạo ra sự linh hoạt và mở rộng trong quản lý và phát triển mã nguồn.

# **KẾT LUẬN**

Mẫu thiết kế Template Method là một công cụ quan trọng trong lập trình hướng đối tượng, giúp tạo ra một khuôn mẫu chung cho việc triển khai các thuật toán và quy trình trong hệ thống. Bằng cách định nghĩa các phần cốt lõi của quy trình trong lớp cha và để cho các phần cụ thể của thuật toán được triển khai bởi các lớp con, mẫu Template Method giúp tái sử dụng mã nguồn, giảm sự phức tạp và tăng tính linh hoạt trong mã nguồn. Thông qua việc áp dụng mẫu này, ta có thể dễ dàng thêm mới các biến thể của quy trình hoặc thay đổi cách thức xử lý mà không làm ảnh hưởng đến cấu trúc chung của hệ thống, giảm thiểu việc lặp lại mã nguồn và dễ bảo trì trong quá trình phát triển hệ thống.

# **TÀI LIỆU THAM KHẢO**

**[1]** Refactoring. Template Method. Refactoring ruru.[**https://refactoring.guru/design-patterns/strategy/java/example**](https://refactoring.guru/design-patterns/strategy/java/example) **.** Ngày truy cập: 09-05-2024.