**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN**

**MEDIATOR DESIGN PATTERN**

**Giảng viên:** Hồ Tuấn Thanh

**Lớp:** 19/4

**Mã nhóm**: 7

**Tên nhóm:** DHKTech

**Thành viên:**

Đặng Thái Duy - 19120491

Nguyễn Thế Hải - 19120069

Trần Nhật Hào - 19120070

Nguyễn Trọng Kha - 19120083

*Thành phố Hồ Chí Minh – Tháng 5/2021*

Mục lục

1. [Đặt vấn đề 3](#_Toc72182939)
2. [Giới thiệu Mediator Design Pattern 4](#_Toc72182941)
3. [Cấu trúc 5](#_Toc72182943)
4. [Cài đặt 6](#_Toc72182945)
5. [Ưu và nhược điểm 9](#_Toc72182947)
6. [Một vài ứng dụng trong thực tế 10](#_Toc72182949)
7. [Tài liệu tham khảo 11](#_Toc72182951)

# 1

# Đặt vấn đề

1. **Bối cảnh:**

Hiện nay chúng ta có thể giao tiếp online với nhau nhờ vào ứng dụng *ChatRoom*. Ứng dụng này được tạo ra nhằm mục đích làm cho cuộc sống chúng ta trở nên dễ dàng hơn. *Beau* và *Quincy* tham gia vào *ChatRoom* thì có thể gửi tin nhắn trực tiếp cho tất cả mọi người hoặc gửi cho một người cụ thể nào đó trong nhóm.

Hãy tưởng tượng nếu Beau và Quincy muốn thêm một vài quy tắc khi nhận tin nhắn như sau:

* Beau không muốn nhận tin nhắn vào lúc giữa trưa vì lúc đó anh ấy đang nghỉ ngơi.
* Quincy thì lại không muốn nhận tin nhắn vào cuối tuần và đặc biệt là tin nhắn có nội dung chứa icon 🙂.

Graphical user interface, application, Teams

Description automatically generated

1. **Tình trạng của ChatRoom:**

Khi một đối tượng muốn chỉnh sửa một vài quy tắc hay gửi nhận tin nhắn, ta bắt buộc phải chỉnh sửa từng mã nguồn của các đối tượng còn lại.

# 2

# Giới thiệu Mediator Design Pattern

*Mediator* là một Design Pattern cung cấp giao thức uỷ quyền trung tâm nơi mà các bộ phận khác của chương trình có thể giao tiếp với nhau.

Khi Mediatorđược thêm vào hệ thống, tất cả các đối tượng xung quanh nó sẽ được đơn giản hoá đi rất nhiều:

* Chúng nói cho Mediator biết khi trạng thái thay đổi.
* Chúng phản hồi các yêu cầu từ Mediator*.*

Xét lại ví dụ *ChatRoom*, trước khi thêm một *Admin* cho phòng như một Mediator, tất cả các thành viên phải nắm được các nguyên tắc nhận tin nhắn của nhau… điều này vô tình dẫn đến sự ràng buộc quá chặt chẽ với nhau. Với sự bổ sung một Admin, các thành viên sẽ được *nới lỏng một cách hoàn toàn* với các thành viên khác và có thể *hoạt động độc lập nhau* vì Admin sẽ là người nắm giữ toàn bộ các nguyên tắc trong phòng.

Graphical user interface, application

Description automatically generated

Mediator *chứa tất cả các phương thức điều khiển logic cho hệ thống*. Khi một đối tượng nào đó được thêm một quy tắc mới, hoặc một đối tượng mới được thêm vào hệ thống, tất cả các phương thức logic này sẽ chỉ được thêm vào Mediator.

# 3

# Cấu trúc

Graphical user interface, application

Description automatically generated

* **Mediator:**

Mediator để xác định các interface giao tiếp giữa các đối tượng tương đồng.

* **ConcreteMediator:**

Thực hiện interface giao tiếp Mediator, để điều chỉnh các thông tin liên lạc giữa các đối tượng. Thông báo cho các đối tượng tương đồng đang tồn tại, biết về mục đích giao tiếp.

* **ConcreteColleauge:**

Thông qua Mediator để giao tiếp với các đối tượng khác.

# 4

# Cài đặt

Arrow

Description automatically generated with medium confidence

1. **Tham gia phòng chat:**

* Đối tượng của lớp Participant gửi yêu cầu tham gia vào phòng chat có tên là room thông qua phương thức join.
* Yêu cầu tham gia này sẽ được xử lý bởi hàm registration của lớp ChatRoom.

|  |
| --- |
| bool Participant::join(ChatRoom\* room)  {  if (!room)  return false;    this->room = room;  room->registration(\*this);  return true;  } |

* Để đơn giản nhất, phòng chat sẽ không có những yêu cầu gì đối với người tham gia.

|  |
| --- |
| void ChatRoom::registration(Participant& participant)  {  participants[numParticipants] = participant;  numParticipants++;  } |

1. **Gửi và nhận tin nhắn:**

* Thành viên trong phòng chat sẽ gửi tin nhắn có nội dung là message đến các thành viên khác trong phòng thông qua phương thức send.
* Khi gửi tin nhắn, người gửi có thể gửi đến bất kì một người dùng cụ thể nào đó. Nếu người gửi không chỉ định cụ thể người nhận thì tin nhắn này sẽ được gửi đến mọi người trong phòng.
* Người dùng không gửi tin nhắn trực tiếp đến các người dùng khác. Việc chuyển tin nhắn sẽ được phòng chat xử lý.

|  |
| --- |
| void Participant::send(string message, Participant\* to = NULL)  {  if (to && \*this == \*to)  {  cout << "Self sending!" << endl;  return;  }  room->send(message, this, to);  } |

* Phương thức send của lớp ChatRoom chính là người trung gian của người gửi và nhận tin nhắn. Tại đây, tin nhắn sẽ được điều phối (và xử lý nếu cần) đến các người nhận .

|  |
| --- |
| void ChatRoom::send(string message, Participant\* from, Participant\* to = NULL)  {  if (to)  {  for (int i = 0; i < numParticipants; i++)  if (participants[i] == \*to)  {  participants[i].receive(message, from);  return;  }  cout << "This one is not in the room!" << endl;  }    else  for (int i = 0; i < numParticipants; i++)  if (participants[i] != \*from)  participants[i].receive(message, from);  } |

* Nhận tin nhắn là phương thức đơn giản nhất. Hàm receive của lớp Participant chỉ cần in tên người gửi cùng với nội dung tin nhắn.

|  |
| --- |
| void Participant::receive(string message, Participant\* from)  {  cout << from->name << " to " << name << ": " << message << endl;  } |

# 5

# Ưu và nhược điểm

1. **Ưu điểm:**

* Giảm bớt việc giao tiếp trực tiếp giữa các bộ phận nhờ việc giảm bớt các khớp nối. Từ đó làm tăng tính tái sử dụng của mã nguồn.
* Mediator thay thế mối quan hệ nhiều-nhiều (many-to-many) giữa các bộ phận bằng quan hệ một-nhiều (one-to-many) giữa Mediator với các bộ phận trong chương trình.

Graphical user interface, application

Description automatically generated

* Đảm bảo nguyên tắc Single Responsibility Principle (SRP) và Open/Closed Principle (OCP).

1. **Nhược điểm:**

* Điểm trừ lớn nhất của Mediator là nếu không được thiết kế đúng cách, các đối tượng của nó sẽ trở nên rất phức tạp và khó bảo trì và trở thành một God object.

# 6

# Một vài ứng dụng trong thực tế

1. **Smart Home:**

* **A picture containing application

  Description automatically generated**Một smart home gồm có rất nhiều thiết bị thông minh: đồng hồ báo thức, máy pha cà phê, TV,…
* Khi người chủ muốn xem TV vào một khung giờ đã định sẵn, tất cả mọi thứ phải được chuẩn bị một cách tự động. Ví dụ, quy trình là: chuông báo đến giờ 🡪 máy pha cà phê bắt đầu pha 🡪 màn tự động kéo xuống 🡪 TV được bật.
* Giải pháp: lớp SmartHome là một Mediator – đầu não trung tâm của căn nhà; lớp Alarm, CoffeePot, Curtain là các Colleauge – các thiết bị thông minh trong căn nhà. Khi chuông báo cho đối tượng của SmartHome là đã đến giờ thì đối tượng này sẽ ra lệnh cho máy pha cà phê, màn và TV làm nhiệm vụ.

1. **Quản lý việc cất hạ cánh trong sân bay:**

A picture containing computer, dark, night sky

Description automatically generated

* Tương tự như bài toán ChatRoom đã nêu. Để điều khiển việc cất hạ cánh của máy bay, các phi công của các máy bay đang cất cánh hoặc hạ cánh sẽ kết nối với tháp ATC chứ không phải giao tiếp một cách trực tiếp với nhau.
* Trong trường hợp này, ATC sẽ là Mediator và các máy bay sẽ là các Colleauge.

# 7

# Tài liệu tham khảo

* Head First Design Pattern
* [Mediator Pattern – Viblo](https://viblo.asia/p/design-patterns-mediator-pattern-gDVK2WrjZLj)
* [Mediator Design Pattern – Free Code Camp](https://www.youtube.com/watch?v=KOVc5o5kURE)
* [Mediator pattern - Wikipedia](https://en.wikipedia.org/wiki/Mediator_pattern)
* [Hướng dẫn Java Deisign Pattern – GP Coder](https://gpcoder.com/4740-huong-dan-java-design-pattern-mediator/)
* [Mediator Design Pattern, Smart Home case](https://cdmana.com/2021/02/20210208124517194D.html)