TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN MẪU THIẾT KẾ**

**QUẢN LÝ ĐIỂM THI**

*Giảng viên*: **ThS Vũ Đình Hồng**

*Người thực hiện*: **Nguyễn Trường Khương Duy - 51800371**

**Đoàn Nguyễn Văn Hậu – 51800747**

Lớp **: 18050281**

Khoá  **: 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**ĐỒ ÁN CUỐI KÌ MÔN MẪU THIẾT KẾ**

**QUẢN LÝ ĐIỂM THI**

*Giảng viên*: **ThS Vũ Đình Hồng**

*Người thực hiện*: **Nguyễn Trường Khương Duy - 51800371**

**Trịnh Thị Mai Hân – 51802081**

**Đoàn Nguyễn Văn Hậu – 51800747**

**Lê Duy Khanh - 51802084**

Lớp **: 18050281**

Khoá  **: 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

LỜI CẢM ƠN

Nhóm chúng em xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc đến Thầy **Vũ Đình Hồng**. Cảm ơn thầy vì đã hỗ trợ chúng em trong suốt thời gian học tập trên lớp và thời gian làm báo cáo cuối kì. Kính mong sự chỉ dẫn và đóng góp của thầy để báo cáo của chúng em được hoàn thiện hơn. Những góp ý từ thầy, chúng em sẽ tiếp nhận để sau này hoàn thành tốt hơn ở những bài đồ án trong tương lai. Chúng em xin chúc quý thầy nhiều sức khỏe để làm việc thật tốt. Chúng em xin chân thành cảm ơn thầy

TÓM TẮT

Mở đầu, giới thiệu về hệ thống quản lý thi trắc nghiệm. Là một hệ thống quản lý việc học tập thi cử của sinh viên. Quản lý thông tin giảng viên, sinh viên, thông tin bài thi,.. Trong quá trình thực hiện hệ thống còn tồn tại một số vấn đề về việc tối ưu hóa hệ thống. Và từ đây sử dụng các mẫu thiết kế để áp dụng vào hệ thống. Nhằm tăng khả năng tối ưu, dễ dàng kế thừa, đọc hiểu và sửa chữa.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc104911031)

[TÓM TẮT ii](#_Toc104911032)

[MỤC LỤC 1](#_Toc104911033)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 2](#_Toc104911034)

[I. Giới thiệu hệ thống: 3](#_Toc104911035)

[II. Áp dụng pattern vào hệ thống: 3](#_Toc104911036)

[2.1. Singleton pattern: 3](#_Toc104911037)

[2.2. Factory pattern: 5](#_Toc104911038)

[2.3. Abstract Factory pattern: 7](#_Toc104911039)

[2.4. Builder pattern: 9](#_Toc104911040)

[2.5. Command pattern: 13](#_Toc104911041)

[2.6. Facade pattern: 15](#_Toc104911042)

[2.7. Adapter pattern: 19](#_Toc104911043)

[2.8. Bridge pattern: 23](#_Toc104911044)

[III. Kết luận: 25](#_Toc104911045)

[3.1. Ưu điểm: 25](#_Toc104911046)

[3.2. Khuyết điểm: 25](#_Toc104911047)

[3.3. Hướng phát triển: 25](#_Toc104911048)

[TÀI LIỆU THAM KHẢO 26](#_Toc104911049)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 2.1: Class diagram Singleton pattern 4](#_Toc104911051)

[Hình 2.2: Class SqlSingleton áp dụng Singleton pattern. 5](#_Toc104911052)

[Hình 2.3: Code áp dụng Singleton pattern 5](#_Toc104911053)

[Hình 2.4: Class diagram Factory pattern 7](#_Toc104911054)

[Hình 2.5: Class diagram Abstract Factory pattern 9](#_Toc104911055)

[Hình 2. 6: Khởi tạo thông tin với nhiều tham số trong class Sinhvien 10](#_Toc104911056)

[Hình 2.7: Class diagram Builder pattern 11](#_Toc104911057)

[Hình 2. 8: Code áp dụng Builder pattern 11](#_Toc104911058)

[Hình 2.9: Interface chứa các hàm trả về giá trị QLUserBuilder 12](#_Toc104911059)

[Hình 2. 10: Khởi tạo Sinhvien sử dụng Builer pattern. 12](#_Toc104911060)

[Hình 2. 11: Khởi tạo Object Giaovien khi sử dụng Builder pattern. 13](#_Toc104911061)

[Hình 2. 12: Sơ đồ lớp Command pattern 14](#_Toc104911062)

[Hình 2. 13: Code áp dụng Command pattern. 15](#_Toc104911063)

[Hình 2. 14: Sơ đồ lớp Façade pattern 17](#_Toc104911064)

[Hình 2. 15: Interface QLSV 20](#_Toc104911065)

[Hình 2. 16: Interface In thông tin 20](#_Toc104911066)

[Hình 2. 17: Class diagram Adapter pattern 21](#_Toc104911067)

[Hình 2. 18: Áp dụng Apdater pattern vào hệ thống 22](#_Toc104911068)

[Hình 2. 19: Class Adaptee xử lý dữ liệu truyền vào của Adapter pattern. 23](#_Toc104911069)

[Hình 2. 20: Sơ đồ lớp Bridge pattern 24](#_Toc104911070)

[Hình 2. 21: Code áp dụng Bridge pattern 25](#_Toc104911071)

1. Giới thiệu hệ thống:
2. Áp dụng pattern vào hệ thống:
   1. Singleton pattern:
      1. Định nghĩa:

* Singleton pattern là một trong những pattern thuộc nhóm khởi tạo (**Creational pattern).** Được áp dụng trong trường hợp đảm bảo rằng một class chỉ có duy nhất một instance (khởi tạo).
  + 1. Lý do áp dụng:

Text

Description automatically generated

* Để sử dụng kết nối database Sql trong project thì mỗi lần sử dụng đều khởi tạo kết nối điều đó khiến chậm trễ hệ thống. Ngoài ra, thì kết nối chỉ cần một lần kết nối là có thể sử dụng cho toàn bộ hệ thống thay vì phải kết nối nhiều lần.
* Thay vào đó áp dụng Singleton pattern sẽ tạo class dùng để kết nối để database một cách toàn cục. Và nếu khởi tạo rồi thì class chỉ trả về instance khởi tạo trước đó.
* Điều đó tránh lãng phí thời gian trong việc xử lý kết nối và tránh lãng phí khởi tạo nhiều lần.
  + 1. Sơ đồ lớp:

Diagram

Description automatically generated

Hình 2.1: Class diagram Singleton pattern

* + 1. Áp dụng code vào hệ thống:
* Class SqlSingleton chứa phương thức áp dụng **Singleton pattern** dùng khởi tạo duy nhất một kết nối đến SQL Server.

Text

Description automatically generated

Hình 2.2: Class SqlSingleton áp dụng Singleton pattern.

* Code áp dụng vào để khởi tạo kết nối Sql Server.

Text

Description automatically generated

Hình 2.3: Code áp dụng Singleton pattern

* 1. Factory pattern:
     1. Định nghĩa:
* Factory Method Design Pattern hay gọi ngắn là: Factory Pattern là một trong những Pattern thuộc nhóm khởi tạo **(Creational Pattern).** Nhiệm vụ của Factory Pattern là quản lý và trả về các đối tượng theo yêu cầu, giúp cho việc khởi tạo đổi tượng một cách linh hoạt hơn.
* Factory Pattern đúng nghĩa là một nhà máy, và nhà máy này sẽ “sản xuất” các đối tượng theo yêu cầu của chúng ta.
* Trong Factory Pattern, chúng ta tạo đối tượng mà không để lộ logic tạo đối tượng ở phía người dùng và tham chiếu đến đối tượng mới được tạo ra bằng cách sử dụng một interface chung.
  + 1. Lý do áp dụng:
* **Theo như form menu ta sẽ có:**
  + 6 chức năng thuộc về quản lý: quản lý sinh viên, quản lý giảng viên, quản lý khoa, quản lý lớp, quản lý môn học và quản lý đề thi.
  + 4 chức năng thuộc về hệ thống: lịch sử thi, chuẩn bị thi, làm bài thi, in phiếu điểm.
  + 1 chức năng ngoài: đăng xuất.
* Để dễ quản lý các chức năng khác nhau và khởi tạo các đối tượng 1 cách linh hoạt hơn thì ta sẽ áp dụng factory method. Ở đây chúng ta sẽ có 2 factory là: **Quản lý** và **Hệ thống.**
  + 1. Sơ đồ lớp:

Diagram

Description automatically generated

Hình 2.4: Class diagram Factory pattern

* + 1. Code áp dụng:
* Vì project được nâng cấp từ factory method thành abstract factory nên sẽ không có code áp dụng factory method mà sẽ có ở bên abstract factory
  1. Abstract Factory pattern:
     1. Định nghĩa:
* Abstract Factory pattern là một trong những nhóm khởi tạo **(Creational pattern).** Nó là phương pháp tạo ra một Super-factory dùng để tạo ra các Factory khác. Hay còn được gọi là Factory của các Factory. Abstract Factory Pattern là một Pattern cấp cao hơn so với Factory Method Pattern.
* Trong Abstract Factory pattern, một interface có nhiệm vụ tạo ra một Factory của các object có liên quan tới nhau mà không cần phải chỉ ra trực tiếp các class của object. Mỗi Factory được tạo ra có thể tạo ra các object bằng phương pháp giống như Factory pattern.

**Ví dụ:** Abstract factory là một nhà máy lớn chứa nhiều nhà máy nhỏ, trong các nhà máy đó có những xưởng sản xuất, các xưởng đó tạo ra những sản phẩm khác nhau.

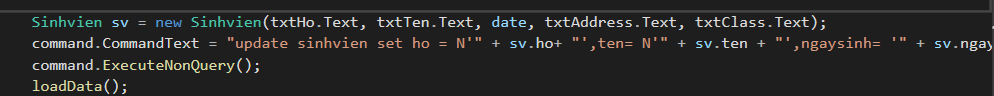
* + 1. Lý do áp dụng:
* Vì project của chúng ta có nhiều factory và ta không muốn khởi tạo trực tiếp các factory mà ta muốn khởi tạo 1 factory đại diện có liên quan đến nó.
* Nếu có sự thay đổi gì thì chúng ta dễ chỉnh sửa, cập nhật mà không ảnh hưởng đến hệ thống bên trong.
  + 1. Sơ đồ lớp:

Graphical user interface

Description automatically generated with low confidence

Hình 2.5: Class diagram Abstract Factory pattern

* + 1. Code áp dụng:
  1. Builder pattern:
     1. Định nghĩa:
* Builder pattern thuộc trong một những nhóm khởi tạo **(Creational pattern).** Là mẫu thiết kế được tạo ra để xây dựng cho đối tượng phức tạp với nhiều tham số, phương thức. Bằng cách sử dụng đối tượng đơn giản, tiếp cận từng bước vào đối tượng phức tạp thông qua việc xây dựng đối tượng độc lập để cụ thể hóa đối tượng phức tạp.
  + 1. Lý do áp dụng:
* Có nhiều tham số được truyền vào constructor của lớp **Sinhvien, giáo viên.**
* Việc khởi tạo với nhiều tham số như vậy khiến cả người lập trình trực tiếp hệ thống khó khăn trong việc thêm parameters vào constructor. Huống chi nếu để phần mềm để người khác kế thừa.



Hình 2. 6: Khởi tạo thông tin với nhiều tham số trong class Sinhvien

* Khi một đối tượng có thể được tạo ra với nhiều parameter một số có thể là bắt buộc và một số khác có thể là tùy chọn tuỳ theo từng yêu cầu của người dùng, tuỳ vào hoàn cảnh của ứng dụng. Chúng ta, có thể tạo ra nhiều **constructor** theo từng nhu cầu hoặc gán giá trị **null** cho các parameter không cần thiết. Tuy nhiên, code rất khó đọc, khó bảo trì, người sử dụng có thể gán nhầm giá trị nếu một loạt các tham số có cùng kiểu dữ liệu.
* Như trong project áp dụng Builder pattern là hợp lý vì một sinh viên gồm nhiều parameters: **họ, tên, lớp, ngày sinh, địa chỉ**,... Builder sử dụng tiếp cận từng bước, việc xây dựng các đối tượng độc lập với các đối tượng khác.
  + 1. Class diagram:

A screenshot of a computer

Description automatically generated with medium confidence

Hình 2.7: Class diagram Builder pattern

* + 1. Code áp dụng:
* Object Sinh viên, giáo viên được builder pattern khởi tạo.

Text

Description automatically generated

Hình 2. 8: Code áp dụng Builder pattern

Text

Description automatically generated

Hình 2.9: Interface chứa các hàm trả về giá trị QLUserBuilder

* Khởi tạo **Object Sinh viên** khi sử dụng Builder pattern.

Text

Description automatically generated

Hình 2. 10: Khởi tạo Sinhvien sử dụng Builer pattern.

* Khởi tạo **Object Giáo viên** khi sử dụng Builder pattern.

Text

Description automatically generated

Hình 2. 11: Khởi tạo Object Giaovien khi sử dụng Builder pattern.

* 1. Command pattern:
     1. Định nghĩa:
* Command Pattern là một trong những Pattern thuộc nhóm hành vi **(Behavior Pattern)**. Nó cho phép chuyển yêu cầu thành đối tượng độc lập, có thể được sử dụng để tham số hóa các đối tượng với các yêu cầu khác nhau như log, queue (undo/redo), transtraction.
* Command Object giống như một class trung gian được tạo ra để lưu trữ các câu lệnh và trạng thái của object tại một thời điểm nào đó.
  + 1. Lý do áp dụng:
* Các thao tác thêm, sửa , xoá và tìm kiếm được tạo và thực thi ở các thời điểm khác nhau
* Có thể thêm dễ dàng các thao tác khác nhau trong tương lai mà không cần thay đổi các lớp hiện có
  + 1. Sơ đồ lớp:

Graphical user interface, application, Teams

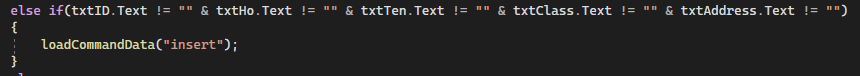
Description automatically generated

Hình 2. 12: Sơ đồ lớp Command pattern

* + 1. Code áp dụng:

Text

Description automatically generated



Hình 2. 13: Code áp dụng Command pattern.

* Trong này có áp dụng phần builder để lấy tham số đầu vào và khởi tạo các lệnh cần thiết.
* Sử dụng hàm if để lọc các lệnh cần chạy.
  1. Facade pattern:
     1. Định nghĩa:
* Façade pattern là mẫu thiết kế thuộc nhóm cấu trúc **(Structural pattern).** Façade cung cấp cho một giao diện đơn giản như một thư viện, một framework hoặc những thứ phức tạp được lưu trữ trong class. Tại đây chúng ta chỉ việc sử dụng thông qua Facade để thực hiện.
* Giúp chúng ta dễ hiểu, dễ chỉnh sửa các phương thức sử dụng.
  + 1. Lý do áp dụng:
* Trong hệ thống thì hàm thực hiện khi ấn nút Add (Ví dụ) có quá nhiều feature được thêm vào. Điều đó khiến một chức năng trở nên quá tải. Vừa phải kiểm tra, phải thực hiện thêm vào,… Khiến cho việc maintain, thay đổi sẽ gặp nhiều khó khăn.
* Thay vào đó ta sẽ chỉnh sửa sao cho mỗi class chỉ có một trách nhiệm duy nhất.

Text

Description automatically generated

Text

Description automatically generated

* + 1. Sơ đồ lớp:

Diagram

Description automatically generated

Hình 2. 14: Sơ đồ lớp Façade pattern

* + 1. Code áp dụng:
* Đối với sinh viên có các chức năng thêm, xóa, sửa thông tin sinh viên. Thì chúng ta thực hiện bấm nút thêm tại class **QLSV** thì tại đây chỉ biết thực hiện thêm còn quy trình ở sau thì các class phụ trách.

Text

Description automatically generated

* Khi ấn nút **“Thêm”** thì việc kiểm tra và thực hiện sẽ được chuyển qua **InfoSvFacade** thực hiện.

Text

Description automatically generated

* Tại lúc thực hiện thì kiểm tra **Facade** sẽ gọi class **InfoSvChecker** để kiểm tra thông tin của Sinh viên nhập vào có hợp lệ không.

Text

Description automatically generated

* Và đây là cách mà class kiểm tra thông tin sinh viên có hợp lệ hay không để thêm vào hệ thống.
* Tương tự với **QLGV** thì cũng áp dụng giống như **QLSV**.
  1. Adapter pattern:
     1. Định nghĩa:
* Adapter pattern (mẫu chuyển đổi) là một trong những pattern thuộc nhóm cấu trúc **(Structural pattern)**. Với adapter pattern được sử dụng như một cổng chuyển đổi. Cho phép các giao diện (interface) khác nhau có thể làm việc trực tiếp với nhau thông qua adapter.
* Đảm nhiệm vai trò trung gian giữa 2 lớp, chuyển đổi dữ liệu hiển thị giữa các interface với nhau. Điều này giúp việc giao tiếp giữa các lớp khác nhau trở nên dễ dàng hơn.
  + 1. Lý do áp dụng:
* Trong hệ thống, có 2 interface **QLSV** và interface **in thông tin.** Với 2 interface khác nhau về dữ liệu truyền vào.

Graphical user interface, table

Description automatically generated with medium confidence

Hình 2. 15: Interface QLSV

Graphical user interface, application, Word

Description automatically generated

Hình 2. 16: Interface In thông tin

* Vấn đề gặp phải là làm sao hiển thị dữ liệu sinh viên từ interface **QLSV** sang interface **In thông tin** theo mỗi cơ sở.
* Vì thế là áp dụng Adapter pattern làm trung gian để xử lý dữ liệu từ **QLSV**  và để hiển thị theo interface **In thông tin.**
  + 1. Sơ đồ lớp:

Graphical user interface

Description automatically generated

Hình 2. 17: Class diagram Adapter pattern

* + 1. Code áp dụng:
* Gọi interface mới PrintSV với 2 tham số: **mã cơ sở, thông tin sinh viên.**

Text

Description automatically generated

* Với các dữ liệu được truyền vào interface thì ta sử dụng adapter để đổi dữ liệu từ **QLSV** sang **PrintSV** (dữ liệu DataSet).

Text

Description automatically generated

Hình 2. 18: Áp dụng Apdater pattern vào hệ thống

* Class Adaptee sử lý dữ liệu truyền vào của adapter.

Text

Description automatically generated

Hình 2. 19: Class Adaptee xử lý dữ liệu truyền vào của Adapter pattern.

* 1. Bridge pattern:
     1. Định nghĩa:
* Bridge Pattern là một trong những Pattern thuộc nhóm cấu trúc **(Structural Pattern).** Ý tưởng của nó là tách tính trừu tượng (**abstraction**) ra khỏi tính hiện thực (**implementation**) của nó. Từ đó có thể dễ dàng chỉnh sửa hoặc thay thế mà không làm ảnh hưởng đến những nơi có sử dụng lớp ban đầu.
* Điều đó có nghĩa là, ban đầu chúng ta thiết kế một class với rất nhiều xử lý, bây giờ chúng ta không muốn để những xử lý đó trong class đó nữa. Vì thế, chúng ta sẽ tạo ra một class khác và di chuyển các xử lý đó qua class mới. Khi đó, trong lớp cũ sẽ giữ một đối tượng thuộc về lớp mới, và đối tượng này sẽ chịu trách nhiệm xử lý thay cho lớp ban đầu.
  + 1. Lý do áp dụng:
* Hiện tại dự án có hơn 5 phần quản lý thông tin mà mỗi phần quản lý gồm có 4 chức năng khác nhau là: **tạo, đọc, xoá, sửa**. Và tổng cộng hơn 20 hàm khác nhau nhưng về cơ bản thì chức năng của các quản lý đều giống nhau. Không những thế, mỗi khi thêm 1 phần quản lý khác là phải thêm 4 chức năng **CRUD**
* Giảm số lượng lớp con phải khởi tạo
* Dễ bảo trì hơn và kích thước ứng dụng sẽ nhỏ hơn
  + 1. Sơ đồ lớp:

Graphical user interface

Description automatically generated with low confidence

Hình 2. 20: Sơ đồ lớp Bridge pattern

* + 1. Code áp dụng:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Hình 2. 21: Code áp dụng Bridge pattern

1. Kết luận:
   1. Ưu điểm:

* Sau khi áp dụng các mẫu thiết kế vào hệ thống thì hệ thống trở nên dễ dàng thực hiện, sửa chữa và kế thừa.
* Ngoài ra, hệ thống được cải thiện hiệu suất, tối ưu hóa các đoạn code sử dụng
* Từ đó, đem lại trải nghiệm cho người sử dụng hệ thống hoặc kế thừa.
  1. Khuyết điểm:
* Còn một số hàm, class chưa được tối ưu về mặt thực hiện, chưa thực sự hướng đối tượng theo tên gọi.
* Ngoài ra, về nội dung hệ thống hiển thị còn một số trục trặc (Chưa cân đối hiển thị, font chữ bị lỗi, chưa thống nhất vị trí hiển thị,..).
* Chính những khuyết điểm làm ảnh hưởng đi phần nào về trải nghiệm người dùng và người kế thừa hệ thống.
  1. Hướng phát triển:
* Hi vọng việc cải thiện hệ thống của phần mềm sẽ đem lại cơ hội sử dụng ngoài thực tiễn đời sống.
* Có một số hiển thị về giao diện, cần được cải thiện, bổ sung thêm để hoàn thiện phần mềm. Sau này, sẽ cải tiến thêm một số chức năng có thể giúp người dùng trực tiếp liên kết phần mềm thông website.
* Mục đích tạo ra phần mềm là đem lại lợi ích cho người sử dụng. Đó cũng là mục đích cuối cùng của việc thực hiện phần mềm.

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

1. Coder, G., 2022. *Hướng dẫn Java Design Pattern – Bridge - GP Coder (Lập trình Java)*. [online] GP Coder. Available at: [https://gpcoder.com/4520-huong-dan-java-design-pattern-bridge>](https://gpcoder.com/4520-huong-dan-java-design-pattern-bridge%3e) [Accessed 31 May 2022].
2. Coder, G., 2022. *Hướng dẫn Java Design Pattern – Bridge - GP Coder (Lập trình Java)*. [online] GP Coder. Available at: [https://gpcoder.com/4520-huong-dan-java-design-pattern-bridge](https://gpcoder.com/4520-huong-dan-java-design-pattern-bridge%20) [Accessed 31 May 2022].
3. Coder, G., 2022. *Hướng dẫn Java Design Pattern – Command - GP Coder (Lập trình Java)*. [online] GP Coder. Available at: [https://gpcoder.com/4686-huong-dan-java-design-pattern-command](https://gpcoder.com/4686-huong-dan-java-design-pattern-command%20) [Accessed 31 May 2022].
4. Coder, G., 2022. *Hướng dẫn Java Design Pattern - Factory Method - GP Coder (Lập trình Java)*. [online] GP Coder. Available at: [https://gpcoder.com/4352-huong-dan-java-design-pattern-factory-method](https://gpcoder.com/4352-huong-dan-java-design-pattern-factory-method%20) [Accessed 31 May 2022].
5. Coder, G., 2022. *Hướng dẫn Java Design Pattern - Abstract Factory - GP Coder (Lập trình Java)*. [online] GP Coder. Available at: [https://gpcoder.com/4365-huong-dan-java-design-pattern-abstract-factory](https://gpcoder.com/4365-huong-dan-java-design-pattern-abstract-factory%20) [Accessed 31 May 2022]

**Tiếng Anh**

1. Refactoring.guru. 2022. Bridge. [online] Available at: [https://refactoring.guru/design-patterns/bridge](https://refactoring.guru/design-patterns/bridge%20) [Accessed 31 May 2022].
2. Sourcemaking.com. 2022. *Design Patterns and Refactoring*. [online] Available at: [https://sourcemaking.com/design\_patterns](https://sourcemaking.com/design_patterns%20) [Accessed 31 May 2022].
3. Sourcemaking.com. 2022. *Design Patterns and Refactoring*. [online] Available at: [https://sourcemaking.com/design\_patterns/builder](https://sourcemaking.com/design_patterns/builder%20) [Accessed 31 May 2022].
4. Sourcemaking.com. 2022. *Design Patterns and Refactoring*. [online] Available at: [https://sourcemaking.com/design\_patterns/singleton](https://sourcemaking.com/design_patterns/singleton%20) [Accessed 31 May 2022].
5. Sourcemaking.com. 2022. *Design Patterns and Refactoring*. [online] Available at: [https://sourcemaking.com/design\_patterns/facade](https://sourcemaking.com/design_patterns/facade%20) [Accessed 31 May 2022]
6. Sourcemaking.com. 2022. *Design Patterns and Refactoring*. [online] Available at: [https://sourcemaking.com/design\_patterns/adapter](https://sourcemaking.com/design_patterns/adapter%20) [Accessed 31 May 2022].