**Câu hỏi Strategy**

Nhóm 1:

**Mẫu thiết kế này có được sử dụng để thay thế cho sự kế thừa không? Vì sao?**

Có, Strategy Pattern có thể được sử dụng để thay thế cho sự kế thừa trong một số trường hợp.

Khi sử dụng kế thừa, các lớp con kế thừa các thuộc tính và phương thức của lớp cha. Điều này có nghĩa là các lớp con sẽ chịu ảnh hưởng bởi các thay đổi trong lớp cha và các lớp con sẽ kế thừa tất cả các phương thức dù chúng có cần sử dụng hay không. Điều này có thể gây ra các vấn đề trong quản lý và bảo trì code.

Nhóm 2:

**Strategy hoạt động như thế nào, ví dụ cụ thể hoạt động**

Nhóm 3:

**Behavior Pattern là như thế nào?**

Behavior Pattern là một nhóm các mẫu thiết kế phần mềm (design patterns) trong lập trình hướng đối tượng (OOP) nhằm giải quyết các vấn đề liên quan đến cách các đối tượng tương tác và thực hiện hành vi của chúng.

Nhóm 4:

**Những bài toán cụ thể nào sử dụng Strategy Pattern tốt nhất**

Ví dụ, khi viết một ứng dụng thương mại điện tử, chúng ta có thể sử dụng Strategy Pattern để thực hiện các chiến lược khác nhau cho việc giảm giá sản phẩm dựa trên các yếu tố như giá trị đơn hàng, loại sản phẩm, độ ưu tiên của khách hàng, v.v.

Nhóm 6:

**Có nhiều class giống nhau nhưng khác nhau về vài xử lý thì có nên sử dụng Strategy không?**

khi có nhiều class giống nhau nhưng khác nhau về một số phương thức hoặc xử lý, sử dụng Strategy Pattern là một lựa chọn tốt để giải quyết vấn đề này, giúp tách các phương thức hoặc xử lý khác nhau thành các class riêng biệt và giúp quản lý code dễ dàng hơn.

Nhóm 7:

Nhóm 8:

**Khi nào thì cẩn sử dụng Strategy?**

Nó được sử dụng khi bạn muốn có thể thay đổi hành vi của một đối tượng trong thời gian chạy, hoặc khi bạn muốn phân tách logic xử lý riêng biệt ra khỏi đối tượng chính.

Khi bạn muốn thay đổi hành vi của một đối tượng tại thời điểm chạy (runtime).

Khi bạn muốn loại bỏ sự phụ thuộc giữa đối tượng và các thuật toán mà nó sử dụng để xử lý dữ liệu.

Khi bạn muốn phân tách các thuật toán khác nhau và đưa chúng vào các class riêng biệt để quản lý dễ dàng hơn.

Khi bạn muốn sử dụng nhiều thuật toán khác nhau để xử lý một dữ liệu nhưng không muốn phải thực hiện nhiều điều kiện nhánh trong code.

Nhóm 9:

**Làm thế nào để đảm bảo tính linh hoạt và dễ dàng bảo trì của Strategy pattern?**

Xác định đúng các đối tượng thuộc về Strategy: Chọn đúng các đối tượng và chức năng nên được phân tách ra thành các thuật toán khác nhau. Tránh tình trạng phân tách quá nhiều hoặc quá ít.

Đảm bảo tính chuyển đổi giữa các thuật toán: Chúng ta cần đảm bảo rằng các Strategy có thể dễ dàng thay thế lẫn nhau mà không ảnh hưởng đến hành vi của đối tượng sử dụng nó. Các Strategy cần có cùng một interface hoặc lớp cơ sở để đảm bảo tính chuyển đổi.

Tách riêng các thuật toán khác nhau: Các thuật toán khác nhau nên được phân tách riêng lẻ để dễ dàng quản lý và bảo trì. Việc phân tách này cũng giúp các lớp con dễ dàng mở rộng các thuật toán hoặc thêm các thuật toán mới.

Nhóm 10:

**Ở Strategy Pattern được áp dụng trong ứng dụng thực tế nào? Vì sao? Và khi nào dùng chúng nó?**

Xử lý nghiệp vụ trong hệ thống giao dịch tài chính: Hệ thống giao dịch tài chính thường phải xử lý nhiều loại định dạng tài khoản, loại tiền tệ và các thủ tục xác thực khác nhau. Strategy Pattern có thể được sử dụng để tách các thuật toán xử lý này thành các lớp riêng biệt và có thể dễ dàng mở rộng hoặc thêm mới các thuật toán khác.

Xử lý định dạng file trong các ứng dụng đọc và ghi file: Các ứng dụng đọc và ghi file thường phải xử lý nhiều loại định dạng file khác nhau, ví dụ như CSV, JSON, XML, v.v. Strategy Pattern có thể được sử dụng để tách các thuật toán xử lý này thành các lớp riêng biệt và có thể dễ dàng mở rộng hoặc thêm mới các thuật toán khác.

Xử lý định dạng hình ảnh trong các ứng dụng xử lý ảnh: Các ứng dụng xử lý ảnh thường phải xử lý nhiều loại định dạng hình ảnh khác nhau, ví dụ như JPEG, PNG, BMP, v.v. Strategy Pattern có thể được sử dụng để tách các thuật toán xử lý này thành các lớp riêng biệt và có thể dễ dàng mở rộng hoặc thêm mới các thuật toán khác.

Nhóm 11:

**Strategy và Template Pattern có thể kết hợp với nhau không? Và ứng dụng ra sao?**

Có thể kết hợp Strategy và Template Pattern với nhau để xây dựng một giải pháp linh hoạt và tái sử dụng được. Khi kết hợp hai mẫu thiết kế này, chúng ta sẽ tạo ra một lớp cơ sở với các phương thức mẫu (template method) để xử lý các bước cơ bản trong một thuật toán và sử dụng Strategy Pattern để thay đổi các phần xử lý cụ thể của thuật toán.

Cụ thể, lớp cơ sở sẽ định nghĩa một số phương thức mẫu cho việc xử lý các bước trong thuật toán. Các phương thức này có thể được triển khai trong lớp con hoặc được thay đổi bởi một Strategy tương ứng để thực hiện các xử lý cụ thể trong bước đó của thuật toán.

Nhóm 12:

**Có nên áp dụng Strategy Parttern cho các dự án vừa và nhỏ không?**

Với các dự án vừa và nhỏ, việc sử dụng Strategy Pattern có thể giúp tách riêng các thuật toán khác nhau thành các đối tượng riêng biệt, giúp cho mã nguồn trở nên dễ bảo trì và mở rộng hơn. Nó cũng cho phép thay đổi thuật toán một cách linh hoạt, mà không ảnh hưởng đến các đối tượng khác trong hệ thống.

**Câu hỏi TemplateMethod**

**Nhóm 1: Giải thích mối quan hệ giữa template method và factory method?**

- Trong lập trình hướng đối tượng, Factory Method và Template Method là hai mẫu thiết kế phần mềm quan trọng. Factory Method là một phương thức tạo đối tượng, trong đó lớp con quyết định loại đối tượng được tạo. Template Method là một mẫu thiết kế phần mềm cho phép ta xác định một bộ khung cho một thuật toán và cho phép các lớp con thay đổi một số bước trong thuật toán đó.

- Factory Method là một trường hợp đặc biệt của Template Method, nghĩa là nó thực hiện một phương thức tạo đối tượng trong một bộ khung được xác định trước. Đồng thời, một Factory Method cũng có thể được sử dụng như một bước trong một Template Method lớn hơn, nghĩa là phương thức tạo đối tượng của nó được sử dụng như một bước trong một thuật toán được định nghĩa trong Template Method.

**Nhóm 2: Khi tạo một template method, làm thế nào để biết khi nào nên sử dụng abstract method và khi nào sử dụng hook?**

Abstract method được sử dụng khi bạn muốn đảm bảo rằng tất cả các lớp con phải cài đặt phương thức này để đảm bảo tính đúng đắn của hệ thống. Điều này đặc biệt hữu ích khi bạn muốn đảm bảo rằng một số phần của phương thức không được định nghĩa trong lớp cha và phải được triển khai trong lớp con.

Hook được sử dụng khi bạn muốn cho phép lớp con tự do lựa chọn cách triển khai một phần của phương thức. Hook có thể được triển khai bởi lớp con nhưng nó không được yêu cầu để đảm bảo tính đúng đắn của hệ thống.

**Nhóm 3: Template method có càng nhiều bước để override thì có dễ báo trì hay không?**

Không dễ bảo trì , Template method là một mẫu thiết kế hướng đối tượng trong lập trình, trong đó một lớp cha định nghĩa một số bước trong một thuật toán và cho phép các lớp con ghi đè lên một số bước này để thay đổi hành vi của thuật toán.

Tùy thuộc vào số lượng bước trong thuật toán mà cần ghi đè, việc bảo trì có thể trở nên phức tạp hơn hoặc dễ dàng hơn. Nếu có quá nhiều bước cần phải ghi đè, việc bảo trì có thể trở nên khó khăn hơn vì các lớp con có thể có sự khác biệt lớn trong các bước được ghi đè và các lớp con này có thể phải được sửa đổi khi có thay đổi trong thuật toán.

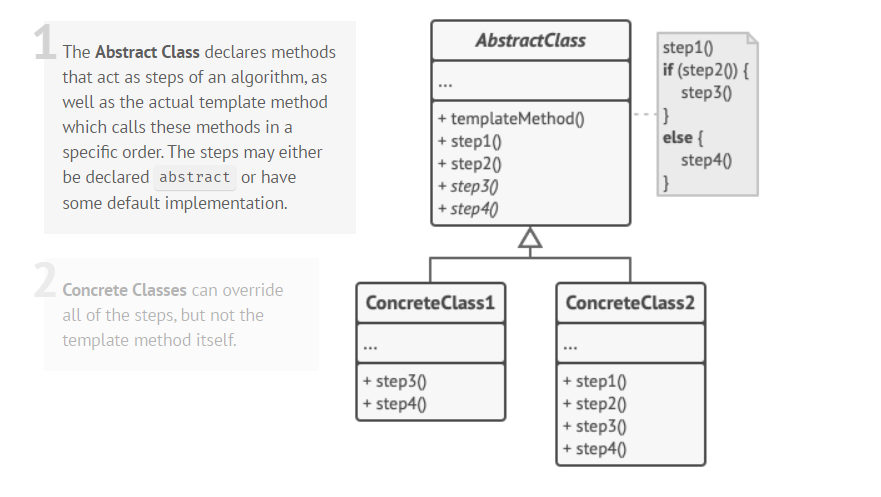
Tuy nhiên, nếu các bước được ghi đè ít và rõ ràng, việc bảo trì có thể trở nên dễ dàng hơn. Các lớp con chỉ cần ghi đè những bước cần thiết để đạt được mục đích

**Nhóm 4: Điểm yếu của Template Method?**

-Template method có càng nhiều bước để override càng khó bảo trì

- có thể bị giới hạn bởi khung thuật toán được cung cấp.

**Nhóm 6: Những thành phần chính nào cấu tạo thành một Template Method?**



Các thành phần trong mô hình:

* AbstractClass khai báo các phương thức hoạt động như các bước của một thuật toán, cũng như template method thực tế gọi các phương thức này theo một thứ tự cụ thể. Các bước có thể được khai báo là trừu tượng hoặc có một số triển khai mặc định.
* Các lớp Concrete có thể ghi đè tất cả các bước, nhưng không phải chính template method.

Nhóm 7: (Nhóm này không đặt câu hỏi)

**Nhóm 8: Khi nào cần sử dụng Template Method? Giải thích lại ví dụ demo?**

* Khi có một thuật toán với nhiều bước và mong muốn cho phép tùy chỉnh chúng trong lớp con.
* Mong muốn chỉ có một triển khai phương thức trừu tượng duy nhất của một thuật toán.
* Mong muốn hành vi chung giữa các lớp con nên được đặt ở một lớp chung.
* Các lớp cha có thể gọi các hành vi trong các lớp con của chúng một cách thống nhất (step by step).

**Nhóm 9: Làm thế nào để sử dụng Template Method để giảm thiểu việc lặp lại mã trong thiết kế phần mềm?**

* Ví dụ ta có các bước để pha chế đồ uống ,là ( chuẩn bị Ly,pha chế,rót ra ly)

Nếu ta pha cà phê thì chuẩn bị ly ->(pha chế) lọc qua phin -> đổ ra ly

Nếu ta pha trà thì chuẩn bị ly ->(pha chế) ngâm túi trà -> đổ ra ly

Ta thấy các bước đổ ra ly và chuẩn bị ly nó giống nhau ở giữa 2 loại đồ uống thay vì phải viết lại như vậy vậy ở các lớp thì ta sẽ tạo đưa các bước này vào lớp cha (superclass) và sử dụng tính đa hình , ta sẽ định nghĩa 1 lớp abstract lớp này sẽ định nghĩa khung của việc pha chế đồ uống ,trong class này ta sẽ định nghĩa 1 phương thức là templatethod trong phương thức này ta sẽ khai báo các bước và gọi các bước này thông qua nó , còn phần loại code trùng lặp thì t đưa các bước đổ ra ly và chuẩn bị ly vào trong lớp này thì các lớp con sẽ chỉ cần định triển khai các bước theo qui định mà ko cần định nghĩa lại các bước đổ ra ly và chuẩn bị ly

**Nhóm 10: Ở Template Method có được thuận lợi và khó khăn khi sử dụng ứng dụng thực tế nào? Tại sao? Và có thể dùng ứng dụng nào trong thiết mẫu này?**

**Ưu điểm**

Tái sử dụng code (reuse), tránh trùng lặp code (duplicate): đưa những phần trùng lặp vào lớp cha (abstract class).

Cho phép người dùng override chỉ một số phần nhất định của thuật toán lớn, làm cho chúng ít bị ảnh hưởng hơn bởi những thay đổi xảy ra với các phần khác của thuật toán.

**Nhược điểm**

Template method có càng nhiều bước để override càng khó bảo trì.

Có thể bị giới hạn bởi khung thuật toán được cung cấp.

**Và có thể dùng ứng dụng nào trong thiết mẫu này ?**

* Khi có một thuật toán với nhiều bước và mong muốn cho phép tùy chỉnh chúng trong lớp con.
* Mong muốn chỉ có một triển khai phương thức trừu tượng duy nhất của một thuật toán.
* Mong muốn hành vi chung giữa các lớp con nên được đặt ở một lớp chung.
* Các lớp cha có thể gọi các hành vi trong các lớp con của chúng một cách thống nhất (step by step).

**Nhóm 11: Trong lớp cha và lớp con, cái nào giảm thiểu mã lệnh (code) trùng lặp sẽ tốt hơn?**

-theo template method thì việc đưa các đoạn mã trùng lặp vào lớp Cha .

**Nhóm 12: Template method có phải là tách hàm nhỏ ra không?**

Không