**NHÓM 2: FACTORY METHOD - SINGLETON PATTERN**

**Câu hỏi các nhóm**

**FACTORY METHOD**

**Nhóm 1:** Tại sao chúng ta lại phải dùng Factory pattern thay cho việc khởi tạo đối tượng thông thường? Dùng Factory pattern có lợi ích gì?

* Factory Method Pattern giúp gom các đoạn code tạo ra product vào một nơi trong chương trình, nhờ đó giúp dễ theo dõi và thao tác. Với Pattern này, chúng ta có thể dễ dàng mở rộng, thêm những đoạn code mới vào chương trình mà không cần phá vỡ các đối tượng ban đầu
* Tạo ra 1 cách mới trong việc khởi tạo cá Object thông qua 1 interface chung. Khởi tạo các Objects mà che giấu đi xử lý logic của việc khởi tạo đấy. Giảm sự phụ thuộc giữa các module, các logic với các class cụ thể, mà chỉ phụ thuộc vào interface hoặc abstract class.

**Nhóm 4:** Theo mình biết thì, khi sử dụng Factory Method, Code sẽ trở nên phức tạp khi có quá nhiều class con để triển khai pattern, vậy thì mình có giải pháp nào cho cái nhược điểm này không?

* Không, đây cũng là một nhược điểm của Factory Method

**Nhóm 5**: Factory Pattern có thể được sử dụng trong các ứng dụng web không?

* Mình thấy factory method vs abstract factory tên hơi giống nhau lại cùng nằm trong nhóm creational pattern, không biết có khác nhau cái gì không?

**Nhóm 6:** Tại sao nói khi sử dụng factory method hạn chế sự phụ thuộc giữa creator và concrete products

**Nhóm 8:**

**Câu 1:** Nhóm trình bày có thể giải thích lại sơ đồ lớp UML của Factory Method design pattern được không?

**Câu 2:** Factory method pattern có kế thừa được không?

**Câu 3:** Khi nào thì nên sử dụng pattern này và lợi ích của pattern này là gì?

**Factory Pattern được sử dụng khi:**

* Chúng ta có một superclass với nhiều class con và dựa trên đầu vào, chúng ta cần trả về một class con. Mô hình này giúp chúng ta đưa trách nhiệm của việc khởi tạo một lớp từ phía người dùng (client) sang lớp Factory.
* Chúng ta không biết sau này sẽ cần đến những lớp con nào nữa. Khi cần mở rộng, hãy tạo ra subclass và implement thêm vào factory method cho việc khởi tạo sub class này.

**Lợi ích:**

* Giúp giảm sự phụ thuộc giữa các module (loose coupling): cung cấp 1 hướng tiếp cận với Interface thay thì các implement
* Giúp chương trình độc lập với những lớp cụ thể mà chúng ta cần tạo 1 đối tượng, code ở phía client không bị ảnh hưởng khi thay đổi logic ở factory hay subclass.
* Mở rộng code dễ dàng hơn: khi cần mở rộng, chỉ việc tạo ra sub class và implement thêm vào factory method.
* Khởi tạo các Objects mà che giấu đi xử lý logic của việc khởi tạo đấy. Người dùng không biết logic thực sự được khởi tạo bên dưới phương thức factory.
* Dễ dạng quản lý life cycle của các Object được tạo bởi Factory Pattern.
* Thống nhất về naming convention: giúp cho các developer có thể hiểu về cấu trúc source code

**Nhóm 10:**

**Câu 1:** Trong Factory Method ngoài quản lý ngân hàng ra còn có quản lý về những gì?

**Câu 2:** Trong Factory Method ngoài cái demo bạn đã làm đó ra còn có quản lý nào liên quan đến Factory Method không?

* Có thể quản lý xe, thức ăn, động vật, hình học,

**Nhóm 11:** Factory Method có liên quan gì đến phương thức tĩnh không?

* Factory Method sử dụng một hàm thành viên tĩnh (phương thức nhà máy tĩnh) để tạo và trả về các thể hiện, ẩn chi tiết của các mô-đun lớp khỏi người dùng.

**Nhóm 12:** Factory method có những ưu điểm gì so với các phương pháp khác?

* Dễ dàng bảo trì code với các lớp tách biệt
* chúng ta có thể dễ dàng thêm một kiểu sản phẩm mới mà không làm ảnh hưởng đến code hiện tại.

**Nhóm 13:** Factory pattern có sử dụng trong game development được không?

**SINGLETON**

**Nhóm 1**: "Mô hình Singleton là một mẫu thiết kế phần mềm để hạn chế sự khởi tạo của lớp đối tượng" bạn có thể giải thích rõ ý này được không ?

**Nhóm 3:** Những dự án nhỏ có nên xài Singleton hay không? Vì sao phải sử dụng keyword trung gian .instance? Bạn có trình bày là Singleton có thể kế thừa, vậy khi nào thì sử dụng kế thừa và tác dụng của nó là gì?

**Nhóm 4:**

**Câu 1:** Giải thích việc Singleton tạo ra quá nhiều phụ thuộc, không thể sử dụng đa hình và dễ tạo ra các bug.

**Câu 2:** Khi sử dụng singleton, sẽ làm tăng kết nối giữa các script, điều này khiến các script phụ thuộc vào singleton, vậy nếu mà mình thay đổi singleton, thì có sinh ra lỗi gì không?

**Nhóm 5:**

**Câu 1:** Trong OOP, Singleton pattern được sử dụng để giải quyết vấn đề gì và làm thế nào để triển khai nó trong các ứng dụng thực tế?

**Câu 2:** Trong OOP, Singleton pattern được sử dụng để giải quyết vấn đề gì và làm thế nào để triển khai nó trong các ứng dụng thực tế?

Giải quyết các vấn đề:

1. Đảm bảo rằng một lớp chỉ có một cá thể duy nhất
2. Cung cấp điểm truy cập toàn cầu cho cá thể đó.

**Nhóm 6:** Tại sao Singleton là anti pattern?

* Với ngày càng nhiều lớp gọi phương thức ‘get Instance()’, mã ngày càng được kết hợp chặt chẽ hơn, nguyên khối, không thể kiểm tra và khó thay đổi cũng như khó sử dụng lại vì các phụ thuộc không thể cấu hình và ẩn. Ngoài ra, sẽ không cần [Double-Checked Locking](https://www.baeldung.com/java-singleton-double-checked-locking) vụng về này nếu bạn gọi phương thức ‘get Instance()’ ít thường xuyên hơn (tức là một lần một lần).

**Nhóm 8:** Làm sao để nhận biết được class đó nên dùng Singleton? Chỉ sử dụng 1 lần hay thuyết phục?

**Nhóm 10:** Điểm giống và khác giữa Factory Method và Singleton là như thế nào?

* Singleton đảm bảo rằng bạn luôn nhận được cùng một phiên bản của bất kỳ loại nào bạn đang truy xuất, trong khi Factory method thường cung cấp cho bạn một phiên bản khác nhau của từng loại. Mục đích của singleton là nơi bạn muốn tất cả các cuộc gọi đi qua cùng một phiên bản

**Nhóm 11:** tại sao singleton chỉ tạo ra 1 lớp instance duy nhất

**Nhóm 12:**

**Câu 1:**Làm thế nào để đảm bảo rằng một class chỉ có một instance duy nhất khi sử dụng Singleton?

* Private constructor của class đó để đảm bảo rằng class lớp khác không thể truy cập vào constructor và tạo ra instance mới
* Tạo một biến private static là thể hiện của class đó để đảm bảo rằng nó là duy nhất và chỉ được tạo ra trong class đó thôi.
* Tại 1 thời điểm tạo đt sử dụng, phù hợp cho lớp **kết nối làm việc với DB**, trong 1 db làm việc thì mở kết nối, … Có nhiều đối tượng kết nối cùng lúc thì xung đột

**Câu 2:** Singleton - Cách xử lý đa luồng của nhóm là "Lock Object" khi dừng task sau lại, vậy cách này có làm chậm hiệu năng chương trình hay không? Và làm sao khi dùng Singleton đảm bảo chỉ sinh ra 1 đối tượng, cảm mơn nhóm!

**Câu 3:** Singleton có ảnh hưởng đến hiệu suất của ứng dụng hay không? Nếu có, thì làm thế nào để giảm thiểu tác động này?

* Không, mẫu Singleton không làm chậm ứng dụng. Về cơ bản, nó tạo một đối tượng duy nhất và làm cho nó có sẵn mỗi khi được yêu cầu

**Nhóm 13:** Nhược điểm của cách khởi tạo "Eager initialization" trong singleton là gì?

* Cách này có một nhược điểm là nếu lớp này không được sử dụng sau đó thì việc khởi tạo sẽ trở thành phí phạm vì biến instance sẽ được khởi tạo ngay lúc chương trình chạy bất kể là có sử dụng đến lớp Singleton hay không.