**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO MÔN HỌC**

**ABSTRACT FACTORY**

**Học phần: Lập trình hướng đối tượng**

**Lớp: 19CQ/4**

**Nhóm: OOP For Life!**

**Họ và tên các thành viên:**

1. **BÙI LÊ TUẤN ANH – 19120163**
2. **NGÔ NHẬT TÂN – 19120128**
3. **PHẠM TIẾN KHẢI – 19120250**

***Thành phố Hồ Chí Minh, tháng 05 năm 2021***

Nội dung

[Lời nói đầu iii](#_Toc73886940)

[I. BÀI TOÁN ĐẶT RA 1](#_Toc73886941)

[II. PHÂN TÍCH BÀI TOÁN 2](#_Toc73886942)

[III. MỘT SỐ HƯỚNG TIẾP CẬN TRƯỚC ĐÂY 3](#_Toc73886943)

[IV. GIỚI THIỆU VỀ MẪU THIẾT KẾ 4](#_Toc73886944)

[V. HƯỚNG TIẾP CẬN SỬ DỤNG MẪU THIẾT KẾ 6](#_Toc73886945)

[VI. MỘT SỐ BÀI TOÁN VÀ VÍ DỤ KHÁC 7](#_Toc73886946)

[KẾT LUẬN 8](#_Toc73886947)

[TÀI LIỆU THAM KHẢO 9](#_Toc73886948)

# Lời nói đầu

# BÀI TOÁN ĐẶT RA

Trước đó, trong phần nghiên cứu về Factory Method, chúng tôi đưa ra một bài toán như sau:

***Một công ty ô tô có 3 dòng xe: Xe bán tải, Xe du lịch/Thể thao, Xe mui trần (xe sang). Cần có phần mềm quản lý nhà máy sản xuất của công ty này.***

Bài toán này sẽ được nâng cấp với một **yêu cầu đặc biệt** (tình huống giả định thực tế) như sau:

***Ngay khi dịch COVID-19 bùng phát tại Việt Nam, để phục vụ tiến trình tiêm vaccine cho người dân tại các địa phương đang có dịch, Bộ Y tế đã giao công ty sản xuất dòng xe chuyên dụng cho Bộ. Trong đó, Bộ đã và đang phê duyệt cho việc sử dụng một số thương hiệu vaccine, mỗi thương hiệu vaccine có 1 loại xe vận chuyển khác nhau, cụ thể như sau:***

1. **Vaccine AstraZeneca, do Đại học Oxford (Anh) và công ty AstraZeneca (liên doanh Anh – Thụy Điển) sản xuất.**
2. **Vaccine Nano Covax, do công ty Nanogen (Việt Nam) sản xuất.**
3. **Vaccine Pfizer/BioNTech, do liên doanh hai công ty Pfizer (Mỹ) và BioNTech (Đức) sản xuất.**

# PHÂN TÍCH BÀI TOÁN

Yêu cầu đặt ra lúc này cho bài toán mới sẽ là:

1. Tiến hành mở rộng phần mềm sản xuất với tốc độ cao nhất, tức là ***không cần phải chỉnh sửa mã nguồn cũ mà vẫn có thể viết mã nguồn mới***, dựa vào mã nguồn cũ. Yêu cầu này được đặt ra nhằm đảm bảo việc sẽ có xe trong thời gian **sớm nhất**, để vaccine được bảo quản và vận chuyển **nhanh nhất** đến các địa phương đang có dịch.
2. ***Tách hoàn toàn*** mảng xe chuyên dụng với mảng xe kinh doanh. Đây là yêu cầu quan trọng, bởi không thể quản lý được việc sản xuất xe của cả hai mảng nếu mã nguồn được đặt chung **tại cùng một vị trí**, rất khó kiểm soát và không đảm bảo an toàn sản xuất.
3. Việc sản xuất của các loại xe này phải tuân thủ ***tính độc lập*** (tức là không ảnh hưởng lẫn nhau giữa các loại xe).
   1. Mỗi loại xe, mỗi dòng xe có các thuộc tính **khác nhau**.
   2. **Không thể** dùng cùng một loại phần mềm của mảng xe kinh doanh cho mảng chuyên dụng
   3. **Không thể** dùng cùng thông tin của dòng xe chở vaccine A cho dông xe chở vaccine B, vốn có điều kiện bảo quản trên xe khác dòng xe chở vaccine A.
4. Sau khi dịch bệnh được kiểm soát, các bộ ngành khác như **Bộ Công an, Bộ Quốc phòng, Bộ Nội vụ**… khi đặt hàng công ty, phần mềm này phải có khả năng ***tiếp tục mở rộng*** để đáp ứng nhu cầu **cấp bách** của quốc gia mà **không ảnh hưởng** đến dây chuyền sản xuất xe chở vaccine của Bộ Y tế.

Đây là những yêu cầu hết sức thực tế, đòi hỏi công ty phải có giải pháp để gia tăng khả năng sản xuất, đáp ứng yêu cầu của Chính phủ cũng như người dân.

*Ở phần sau của bản báo cáo này, chúng tôi xin giới thiệu một số hướng tiếp cận cho bài toán này cũng như các bài toán khác có tính chất tương tự.*

# MỘT SỐ HƯỚNG TIẾP CẬN TRƯỚC ĐÂY

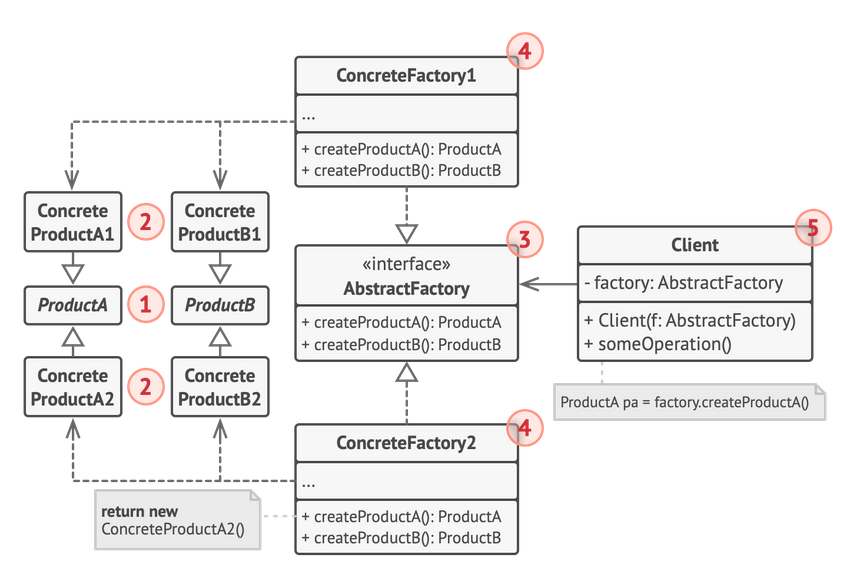
1. Hướng tiếp cận cơ bản - Kế thừa và Hàm dựng:
2. Hướng tiếp cận nâng cao – Factory Method:
3. Nghịch lý từ cả hai hướng tiếp cận:

# GIỚI THIỆU VỀ MẪU THIẾT KẾ

Để phục vụ tốt nhất cho bài toán này, chúng tôi giới thiệu mẫu thiết kế Abstract Factory (Nhà máy ảo/Nhà máy trừu tượng) – một mẫu thiết kế trong lập trình hướng đối tượng (OOP – Object-Oriented Programming) được nâng cấp từ chính mẫu thiết kế Factory Method (chính là hướng tiếp cận nâng cao trong bài toán, được giới thiệu ở mục III trước đó).

Mẫu thiết kế Abstract Factory được định nghĩa như sau:

***Abstract Factory là một mẫu thiết kế trong OOP thuộc nhóm Khởi tạo (Creational), cung cấp một lớp giao diện (interface) có khả năng tạo ra tập hợp các đối tượng liên quan với nhau hoặc phụ thuộc lẫn nhau mà không cần chỉ rõ lớp cụ thể sẽ được tạo ra.***

Mẫu thiết kế này được cụ thể hóa bằng sơ đồ lớp (UML) như sau:

Ta có thể cụ thể hóa sơ đồ này thành ba phân vùng: Phân vùng dành cho Sản phẩm (**Product**, ***số 1 và số 2***), phân vùng dành cho nhà máy (**AbstractFactory**, ***số 3 và số 4***) và phân vùng truy cập chính (**Client**, ***số 5***).

1. ***Các lớp Dòng sản phẩm (Product)****: sẽ khai báo và gom nhóm các sản phẩm thuộc cùng một dòng sản phẩm lại với nhau, nhằm thuận lợi hơn cho quá trình quản lý.*
2. ***Các lớp Loại sản phẩm (Concrete Product)****: Kế thừa theo từng dòng sản phẩm và tùy biến cho phù hợp với từng loại sản phẩm khác nhau.*
3. ***Lớp Nhà máy chính (AbstractFactory)****: Nhà máy này có chức năng khai báo các phương thức để khởi tạo từng dòng sản phẩm khác nhau từ các lớp Sản phẩm.*
4. ***Các lớp Nhà máy con (ConcreteFactory)****: Mỗi nhà máy con kế thừa dây chuyền của nhà máy cha (AbstractFactory) và định nghĩa lại sao cho phù hợp với từng dòng sản phẩm mà mình đang sản xuất.*
5. ***Lớp truy cập chính (Client)****: Có thể truy cập vào các nhà máy để xin cấp quyền sản xuất sản phẩm.*

Với cách bố trí lớp như vừa rồi, mẫu thiết kế Abstract Factory mang đến một số ưu và nhược điểm so với những hướng tiếp cận trước đây, cụ thể là:

* **Ưu điểm**:
  + *Đảm bảo tính tương thích cao giữa các sản phẩm trong cùng một dòng và giữa các sản phẩm với nhau.*
  + *Tách riêng phần mã nguồn của sản phẩm với mã nguồn chính, giúp dễ dàng quản lý và kiểm soát mã nguồn.*
  + *Tuân thủ đồng thời hai nguyên lý trong chùm nguyên lý* ***SOLID*** *trong OOP:*
    - ***Nguyên lý Đơn trách nhiệm (Single Responsiblity Principle)****: Tách phần mã nguồn tạo sản phẩm với các hàm thủ tục khác, giúp tăng khả năng bảo trì và kiểm soát mã nguồn.*
    - ***Nguyên lý Đóng – mở (Open/Closed Principle)****: Mã nguồn có thể mở rộng với nhiều dòng sản phẩm với nhiều loại sản phẩm khác nhau.*
* **Nhược điểm:**
* *Mã nguồn dài hơn, phức tạp hơn nên mất nhiều thời gian thiết kế hơn.*

***Ở phần V, chúng ta sẽ áp dụng mẫu thiết kế này vào bài toán đặt ra để đánh giá sự hiệu quả và tính ưu việt của mẫu thiết kế này...***

# HƯỚNG TIẾP CẬN SỬ DỤNG MẪU THIẾT KẾ

1. **Nhắc lại bài toán:**

Ở phần I, chúng ta xét đến bài toán sau:

***Một công ty ô tô có 3 loại xe thuộc dòng xe Kinh doanh: Xe bán tải, Xe du lịch/Thể thao, Xe mui trần (xe sang).***

***Ngay khi dịch COVID-19 bùng phát tại Việt Nam, để phục vụ tiến trình tiêm vaccine cho người dân tại các địa phương đang có dịch, Bộ Y tế đã giao công ty sản xuất dòng xe chuyên dụng cho Bộ. Trong đó, Bộ đã và đang phê duyệt cho việc sử dụng một số thương hiệu vaccine, mỗi thương hiệu vaccine có 1 loại xe vận chuyển khác nhau, cụ thể như sau:***

1. **Vaccine AstraZeneca, do Đại học Oxford (Anh) và công ty AstraZeneca (liên doanh Anh – Thụy Điển) sản xuất.**
2. **Vaccine Nano Covax, do công ty Nanogen (Việt Nam) sản xuất.**
3. **Vaccine Pfizer/BioNTech, do liên doanh hai công ty Pfizer (Mỹ) và BioNTech (Đức) sản xuất.**

***Cần có phần mềm quản lý nhà máy sản xuất của công ty này.***

*Bài toán này có đầy đủ yếu tố để sử dụng mẫu thiết kế Abstract Factory, gồm có:*

* + Hai dòng xe: ***Kinh doanh*** và ***Chuyên dụng***. Mỗi dòng xe sẽ tương ứng với một lớp **Dòng** **sản phẩm (Product)**.
  + Dòng xe **Kinh doanh** có 3 loại xe: ***Bán tải, Thể thao*** và ***Mui trần.*** Dòng xe **Chuyên dụng** có 3 loại xe chuyên chở 3 loại vaccine COVID-19: ***AstraZeneca, Nanocovax*** và **Pfizer/BioNTech**. Mỗi loại xe tương ứng với một lớp **Loại sản phẩm (Concrete Product)** ứng với từng dòng xe.
  + Cần có phần mềm chung quản lý nhà máy sản xuất **(Abstract Factory)**, nhưng tách riêng hai dòng xe để không bị xung đột giữa hai dòng **(Concrete Factory)**.

1. **Sơ đồ lớp với mẫu thiết kế mới – Phân tích sơ đồ**

Chúng ta sẽ vẽ sơ đồ lớp (UML) cho bài toán vừa rồi. Trong đó sẽ có một số điểm khác biệt so với mẫu thiết kế gốc.

Ta phân tích sơ đồ lớp này thành ba phần:

* A
* B
* C

1. **Một số đoạn mã nguồn đáng chú ý**
2. **Kiểm thử mã nguồn**

# MỘT SỐ BÀI TOÁN VÀ VÍ DỤ KHÁC

# KẾT LUẬN

# TÀI LIỆU THAM KHẢO