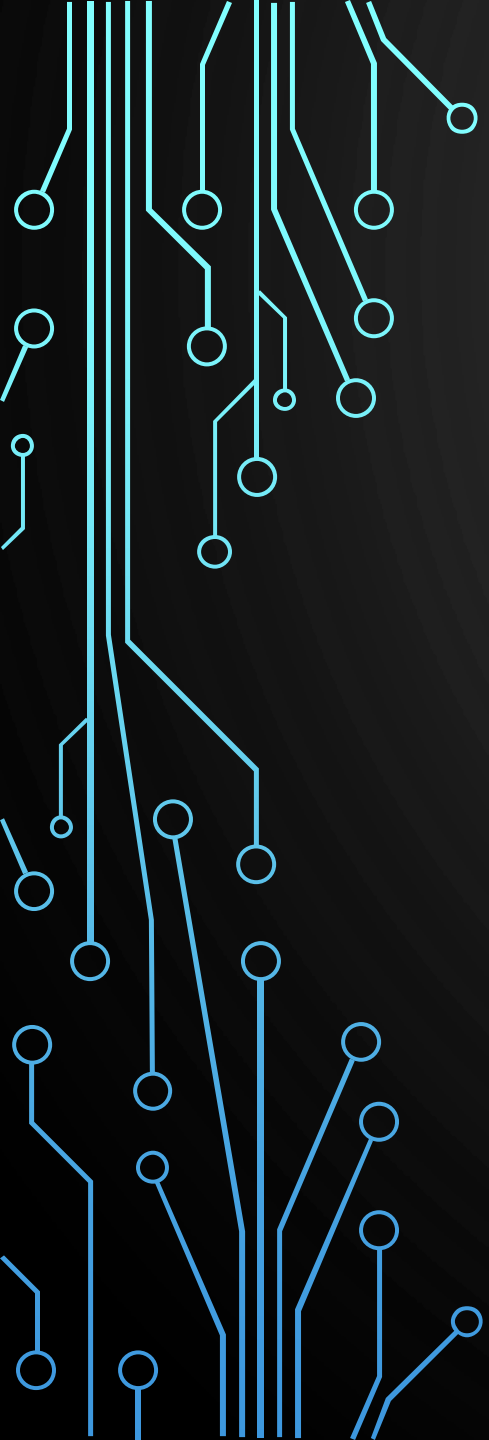


FACTORY METHOD

OOP - LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

NHÓM OOP FOR LIFE!

OOP FOR LIFE! – WHO WE ARE

Thành viên:

- Bùi Lê Tuấn Anh - 19120163 ***(Nhóm trưởng)***
- Ngô Nhật Tân – 19120128
- Phạm Tiến Khải – 19120250

NỘI DUNG

BÀI TOÁN ĐẶT RA

NHỮNG CÁCH GIẢI "ĐI VÀO LỊCH SỬ"

FACTORY METHOD

"ĐỔI MỚI BẮT ĐẦU" – ÁP DỤNG CHO BÀI TOÁN

MỘT SỐ VÍ DỤ KHÁC

BÀI TẬP VÀ CÂU HỎI TƯƠNG TÁC

BÀI TOÁN

- ✓ **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 việc sản xuất của công ty này**

PHÂN TÍCH BÀI TOÁN

YÊU CẦU ĐẶT RA LÚC NÀY LÀ GÌ?

- ✓ **Việc sản xuất của 1 dòng xe không gây ảnh hưởng đến các dòng xe khác (Tức là các dây chuyền sản xuất của các dòng xe là độc lập)**
- ✓ **Việc mở rộng phần mềm để sản xuất dòng xe mới không gây ảnh hưởng đến các dòng xe đã được sản xuất trước trước đó.**

NHỮNG CÁCH GIẢI "ĐI VÀO LỊCH SỬ"

CÁCH 1: KẾ THỪA THÔNG THƯỜNG

TestNoFactoryMethod.cpp M

src > srcFactoryMethod > TestNoFactoryMethod.cpp > main(void)

You, seconds ago | 2 authors (You and others)

```cpp
#include <iostream>
#include <vector>
using namespace std;

```

You, seconds ago | 1 author (You)

```cpp
class Cars{
public:
  virtual void output() = 0;
  virtual void moveToStore() = 0;
};

```

CHUYỆN GÌ ĐÃ XẢY RA?

TestNoFactoryMethod.cpp M

src > srcFactoryMethod > TestNoFactoryMethod.cpp > main(void)

```cpp
class PickUpTruck: public Cars{ //Xe bán tải
public:
  PickUpTruck(){
    cout << "A pickup truck will be created!!!" << endl;
  }
  void output(){
    cout << "A pickup truck is in production" << endl;
  }
  void moveToStore(){
    cout << "A pickup truck has been moved to store" << endl;
  }
};

```

Tương tự với các lớp SportsCars và ConvertibleCars…

NHỮNG CÁCH GIẢI "ĐI VÀO LỊCH SỬ" (2)

```
int main(void){
  PickUpTruck* a = new PickUpTruck();
  a->output();
  ConvertibleCar* c = new ConvertibleCar();
  c->output();      You, seconds ago • Uncommit
}
```

THƯƠNG HIỆU PHÁT TRIỂN, VIỄN CẢNH SẢN XUẤT 10 DÒNG Ô TÔ KHÁC NHAU…

10 LỚP KẾ THỪA, 10 HÀM KHỞI TẠO???

BẤT CẬP PHÁT SINH:

- ✖ Phải gọi đến từng constructor cụ thể của từng lớp để tạo được đối tượng mong muốn. Phải biết hết tên của các class có liên quan đến quá trình khởi tạo.
- ✖ Không che giấu quá trình khởi tạo một đối tượng.
- ✖ Khó khăn trong việc mở rộng thêm các đối tượng mới trong tương lai…

GIẢI PHÁP TỐT HƠN???


WHAT?
WHY?
HOW?

FACTORY METHOD – WHAT?

MẪU THIẾT KẾ TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

NHÓM KHỞI TẠO (CREATIONAL)

Định nghĩa một interface cho việc tạo đối tượng

Giao lại việc chọn đối tượng để tạo cho các lớp con kế thừa

VIRTUAL CONSTRUCTOR (HÀM DỰNG ẢO)

FACTORY METHOD – WHY?

- ✓ Phân quyền và tránh nhập nhằng giữa quá trình khởi tạo và các quá trình xử lý khác.
- ✓ Tận dụng được khả năng từ hàm dựng ảo (Virtual Constructor)
- ✓ Tuân thủ đồng thời hai nguyên lý trong chuỗi nguyên lý SOLID:
  - ✓ Nguyên lý Đơn trách nhiệm: Có thể tách riêng mã nguồn phần khởi tạo, giúp dễ theo dõi hơn.
  - ✓ Nguyên lý Đóng – mở: Có thể đưa vào thêm các lớp mới, không gây ảnh hưởng đến mã nguồn cũ.

SINGLE RESPONSIBLITY PRINCIPLE

OPEN – CLOSED PRINCIPLE

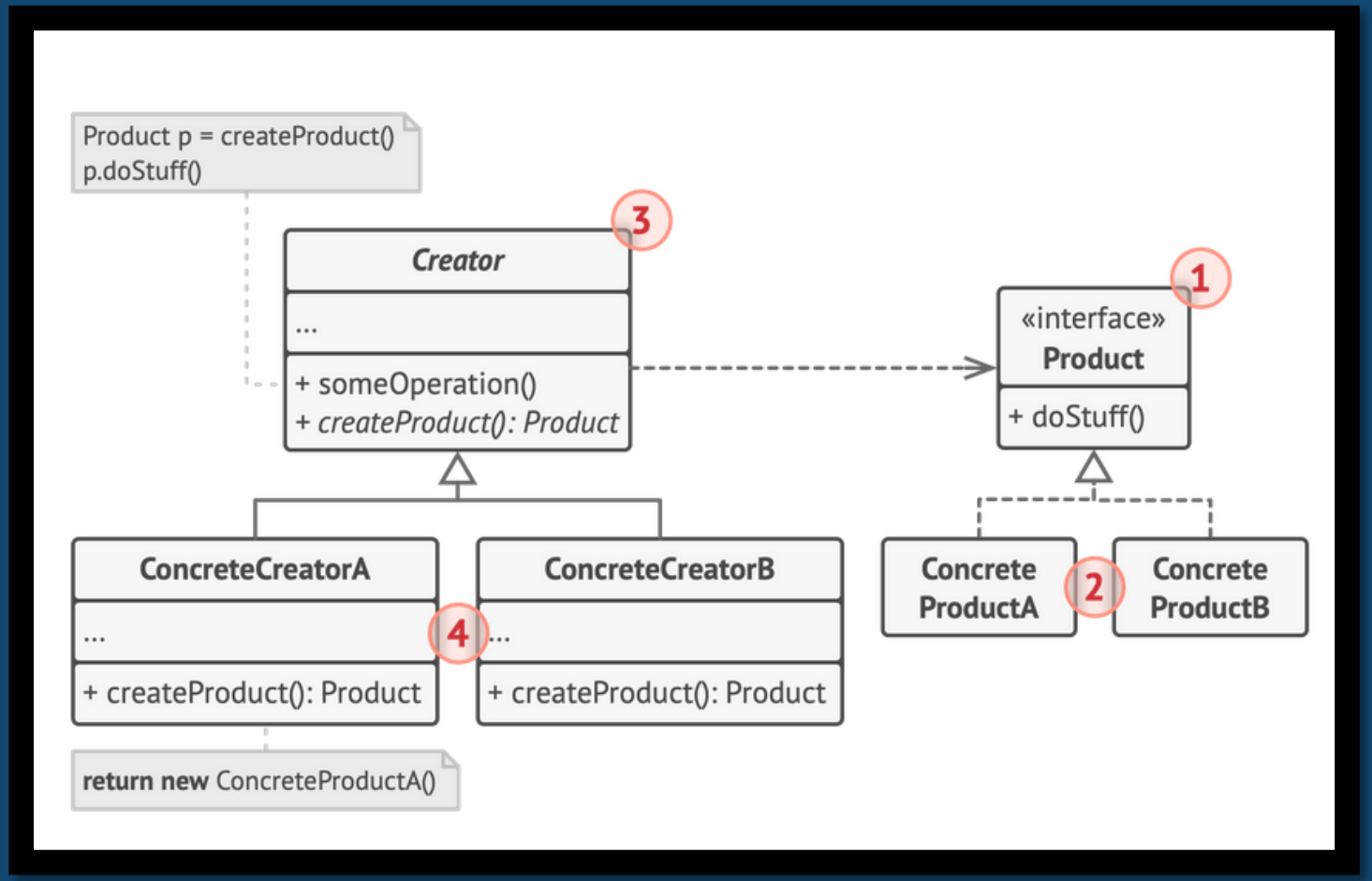
FACTORY METHOD – HOW?

Interface & Inheritance: Product

Abstract Class: Creator

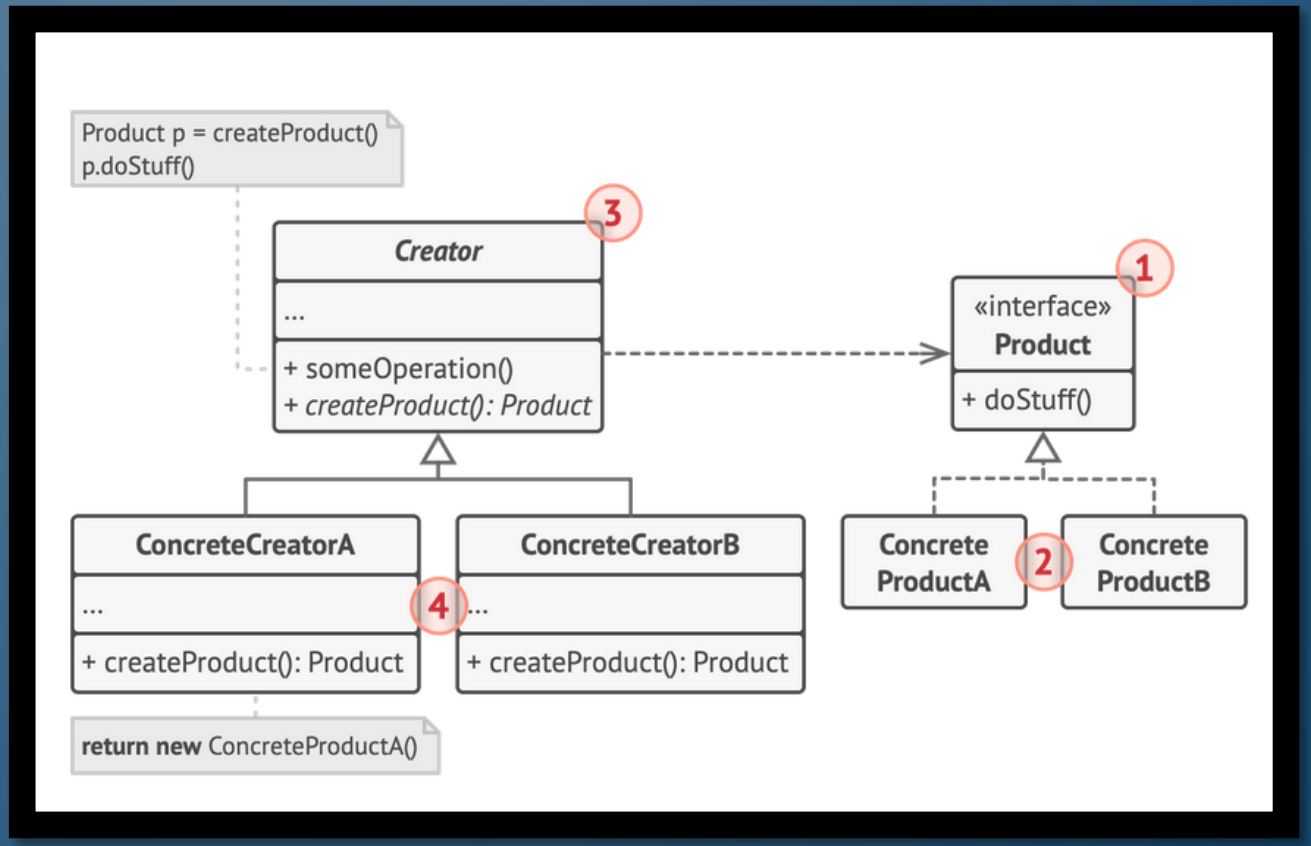
Concrete Creator Classes

FACTORY METHOD – SƠ ĐỒ LỚP



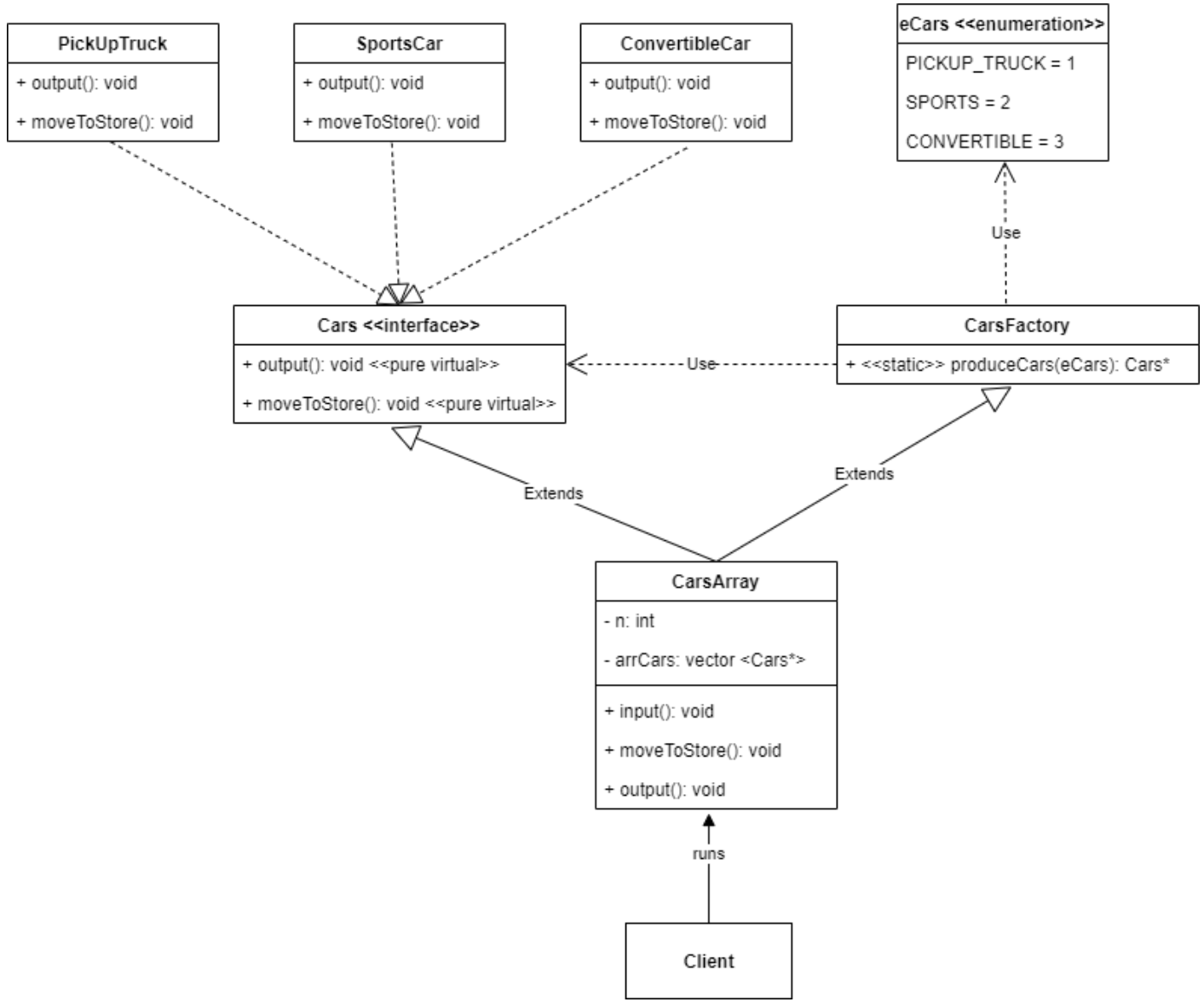
SƠ ĐỒ LỚP

1. **Lớp Product**: Lớp trừu tượng với các tính năng để các lớp con thừa hưởng và định nghĩa lại
2. **Các lớp Concrete Product**: Kế thừa và tùy biến các hàm của lớp Product, phù hợp theo từng đối tượng cụ thể
3. **Lớp Creator**: Lớp trừu tượng, phục vụ chính cho việc khởi tạo đối tượng mới từ Product
4. **Các lớp Concrete Creator**: Xác định đối tượng cụ thể sẽ được tạo.



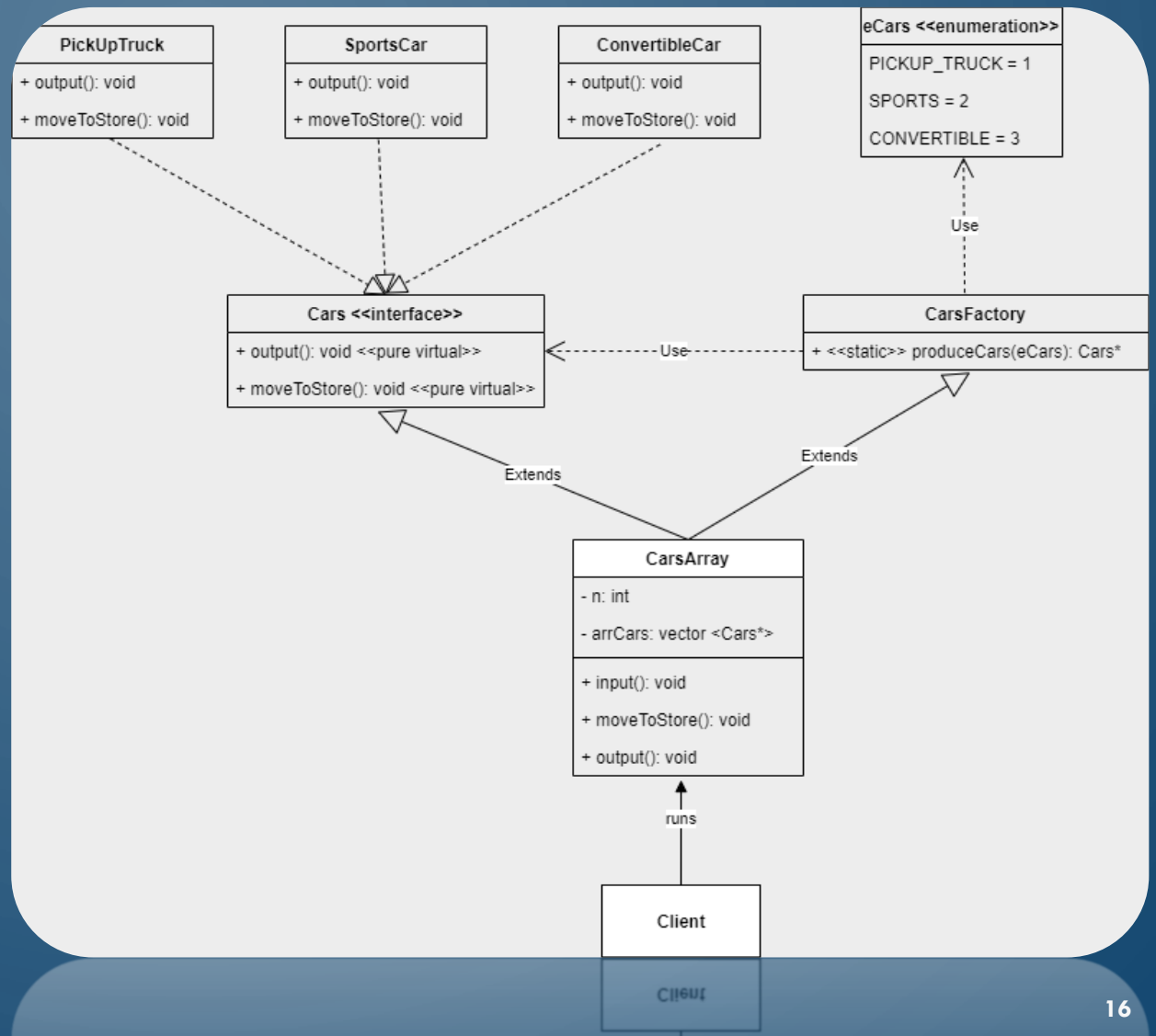
“ĐỔI MỚI BẮT ĐẦU” – ÁP DỤNG CHO BÀI TOÁN

SƠ ĐỒ LỚP CHO CHƯƠNG TRÌNH



“ĐỔI MỚI BẮT ĐẦU” – PHÂN TÍCH SƠ ĐỒ LỚP

1. **Lớp Cars:** Tương ứng với lớp Product
2. **Các lớp PickUpTruck, SportsCar và Convertible**: Tương ứng với các ConcreteProduct.
3. **Lớp CarsFactory**: Tương ứng với lớp Creator
4. **Enumeration eCars:** Dạng biến thể của các ConcreteCreator, đánh số các đối tượng nhằm giúp lớp CarsFactory xác định chính xác đối tượng cần tạo.
5. **Lớp CarsArray**: Khu vực lưu trữ các xe đã được khởi tạo, kế thừa lại toàn bộ hàm của Cars và CarsFactory.



“ĐỔI MỚI BẮT ĐẦU” – ÁP DỤNG CHO BÀI TOÁN (2)

TestFactoryMethod.cpp M | header.h M

src > srcFactoryMethod > header.h > ...

```
You, seconds ago | 1 author (You)
class Cars{
public:
  virtual void output() = 0;
  virtual void moveToStore() = 0;
};

You, seconds ago | 1 author (You)
class PickUpTruck: public Cars{ //Xe bán tải
public:
  void output(){
    cout << "A pickup truck is in production" << endl;
  }
  void moveToStore(){
    cout << "A pickup truck has been moved to store" << endl;
  }
};

You, seconds ago | 1 author (You)
class SportsCar: public Cars{ //Xe thể thao
public:
  void output(){
    cout << "A sports car is in production" << endl;
  }
  void moveToStore(){
    cout << "A sports car has been moved to store" << endl;
  }
};

You, seconds ago | 1 author (You)
class ConvertibleCar: public Cars{ //Xe mui trần
  void output(){
    cout << "A convertible car is in production" << endl;
  }
  void moveToStore(){
    cout << "A convertible car has been moved to store" << endl;
  }
};
```

GOM HÀM DỰNG BẰNG ENUM (BẢN SỐ)…

```
enum eCars{
  PICKUP_TRUCK = 1,
  SPORTS = 2,
  CONVERTIBLE = 3
};

You, 8 minutes ago | 1 author (You)
class CarsFactory{
public:
  static Cars* produceCars(eCars _carsID){
    switch (_carsID)
    {
      case 1: return new PickUpTruck();
      case 2: return new SportsCar();
      case 3: return new ConvertibleCar();
      default: return NULL;
    }
  }
};
```

“ĐỔI MỚI BẮT ĐẦU” – ÁP DỤNG CHO BÀI TOÁN (3)

DÙNG VECTOR ĐỂ QUẢN LÝ SỐ XE…

```
class CarsArray: public Cars, public CarsFactory{
private:
  vector<Cars*> arrCars;
  int n;
public:
  void input(){
```

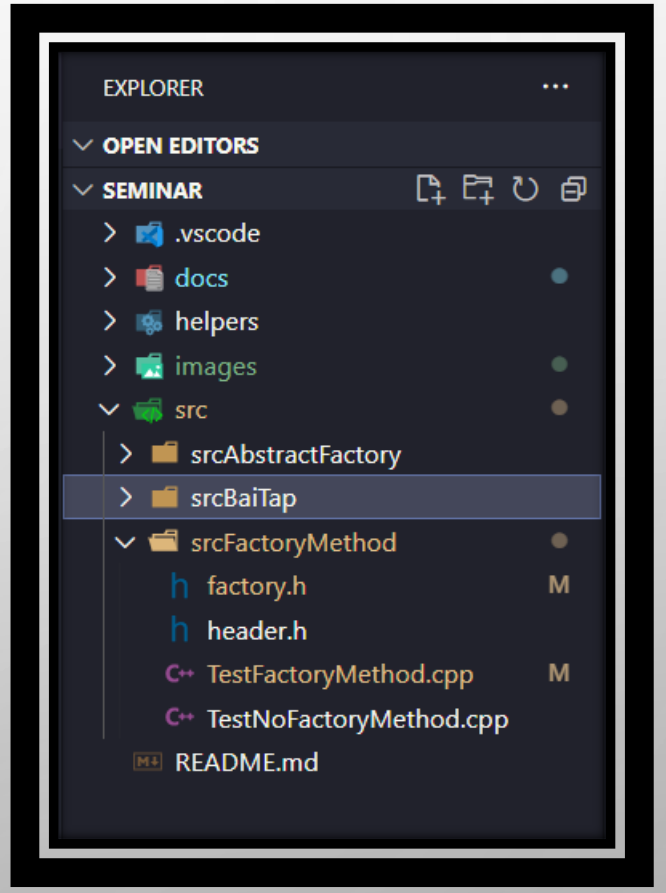
```
void output(){
  for (unsigned int i = 0; i < n; i++){
    arrCars.at(i)->output();
  }
}
void moveToStore(){
  for (unsigned int i = 0; i < n;i++){
    arrCars.at(i)->moveToStore();
  }
}
```

P/s: CarsArray CHỈ LÀ TÊN, CÒN VIỆC SỬ DỤNG MẢNG HAY VECTOR TÙY THUỘC VÀO LẬP TRÌNH VIÊN…

“ĐỔI MỚI BẮT ĐẦU” – ÁP DỤNG CHO BÀI TOÁN (4)

```
int main(void){
  CarsArray* a = new CarsArray();
  a->input();
  cout << "_____________" << endl;
  a->output();
  cout << "_____________" << endl;
  a->moveToStore();        You, seconds ag
  return 0;
}
```

- ✓ Chỉ cần quan tâm loại xe, quy trình sản xuất cứ để nhà máy lo…
- ✓ Mã nguồn dễ quản lý, dễ mở rộng…



MỘT SỐ BÀI TOÁN KHÁC

- Một hiệu sách có kinh doanh 3 thể loại: Truyện tranh, sách văn học và sách ngoại văn. Mỗi quyển sách có mã, tên, tác giả và năm xuất bản. Cần có phần mềm quản lý hiệu sách này.

✓ Product: Sách, ConcreteProduct: Truyện tranh, sách văn học, sách ngoại văn

✓ Creator: Hiệu sách, ConcreteCreator là các creators tương ứng với từng thể loại

MỘT SỐ BÀI TOÁN KHÁC (2)

- Một cửa hàng điện thoại có kinh doanh 3 thương hiệu: Samsung, Apple, Xiaomi. Mỗi điện thoại có mã, tên, nhãn hiệu và năm sản xuất. Cần có phần mềm quản lý cửa hàng này.

✓ Product: Điện thoại, ConcreteProduct: Samsung, Apple, Xiaomi

✓ Creator: Cửa hàng, ConcreteCreator là các creators tương ứng với từng thương hiệu.

MỘT SỐ BÀI TOÁN KHÁC (3)

- Kho chứa vaccine COVID-19 tại Việt Nam hiện đang lưu trữ 3 loại vaccine: AstraZeneca, Nanocovax và Pfizer-BioNTech. Mỗi vaccine đều có mã, quốc gia sản xuất và hạn sử dụng. Cần phần mềm quản lý vaccine trong kho.

✓ Product: Vaccine, ConcreteProduct: AstraZeneca, Nanocovax, Pfizer-BioNTech

✓ Creator: Kho chứa, ConcreteCreator là các creators tương ứng với từng loại vaccine.

CÂU HỎI TƯƠNG TÁC

- Sẽ có 5 câu hỏi, mỗi câu hỏi sẽ được hiển thị trong vòng 5 giây trên màn hình.
- Sau khi kết thúc 5 câu hỏi, bình luận đáp án theo thứ tự 1A, 2B, 3C…
- 5 bạn có đáp án nhanh và chính xác nhất sẽ được ghi nhận điểm.

CÂU HỎI TƯƠNG TÁC

❑ Câu 1: **Factory Method thuộc nhóm design pattern nào?**

A. Creational Pattern

B. Structural Pattern

C. Behavioral Pattern

D. Tất cả đều sai

CÂU HỎI TƯƠNG TÁC

❑ Câu 2: **Factory Method giải quyết vấn đề nào?**

A. Tạo ra một tập hợp các đối tượng liên quan hoặc phụ thuộc lẫn nhau mà không chỉ ra đó là những lớp cụ thể nào tại thời điểm thiết kế.

B. Tạo một đối tượng mà không cần thiết chỉ ra một cách chính xác lớp nào sẽ được tạo.

C. Xây dựng lớp đối tượng sao cho chỉ tồn tại nhiều nhất một đối tượng có kiểu dữ liệu là lớp này trong chương trình.

D. Xây dựng một đối tượng phức tạp bằng cách sử dụng các đối tượng đơn giản

CÂU HỎI TƯƠNG TÁC

❑ Câu 3: **Đâu là ưu điểm của Factory Method?**

A. Code ngắn gọn hơn.

B. Có sự phụ thuộc chặt chẽ giữa creator và concrete products.

C. Hạn chế sự phụ thuộc giữa creator và concrete products.

D. A và C đều đúng

CÂU HỎI TƯƠNG TÁC

- Câu 4: **Mối quan hệ giữa Creator và Product trong Factory Method là gì?**

A. Composition

B. Dependency

C. Aggregation

D. Inheritance

CÂU HỎI TƯƠNG TÁC

❑ Câu 5: **Lớp nào nhận nhiệm vụ khai báo hàm constructor trong Factory Method?**

- **A. ConcreteProduct**
- **B. ConcreteCreator**
- **C. Product**
- **D. Creator**

CÂU HỎI TƯƠNG TÁC

❑ **Đáp án:**

✓ **Câu 1**. A – Factory Method thuộc nhóm Creational

✓ **Câu 2**. B – Factory hỗ trợ tạo một đối tượng mà không cần thiết chỉ ra một cách chính xác lớp nào sẽ được tạo.

✓ **Câu 3**. C – Mã nguồn của Factory Method **DÀI HƠN**!!!

✓ **Câu 4**. B – Creator cần biết lớp nào sẽ được sử dụng từ Product – lớp cha của các ConcreteProduct, **PHỤ THUỘC.**

✓ **Câu 5**. D – Creator **KHAI BÁO** hàm dựng Constructor ảo – ConcreteCreator mới định nghĩa lại.


ANY QUESTIONS?
PLEASE NO HARD QUESTIONS...


THANKS FOR
WATCHING!!!!!!
makeameme.org