**Dependency Injection ( tiêm phụ thuộc )**

* Định nghĩa từ các trạng thái wikipedia

rằng trong sự phụ thuộc của công nghệ phần mềm, tiêm phụ thuộc là một kỹ thuật trong đó một đối tượng nhận các đối tượng khác mà nó được gọi là sự phụ thuộc.

* Cái đó nghe có vẻ đúng, vậy nên hãy xem vấn đề và cách tiêm phụ thuộc giải

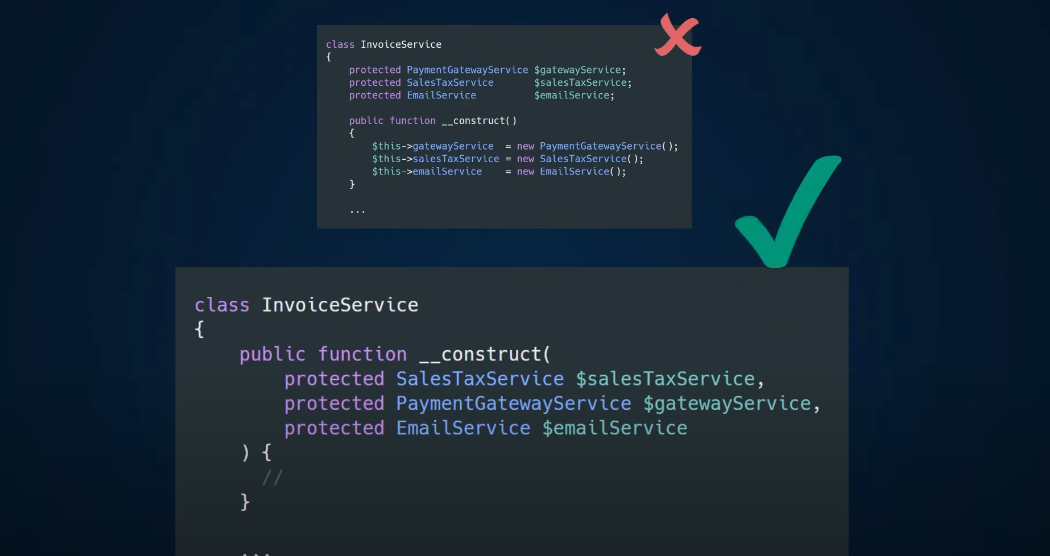
quyết vấn đề đó.

* Như bạn có thể thấy ở đây chúng tôi đang

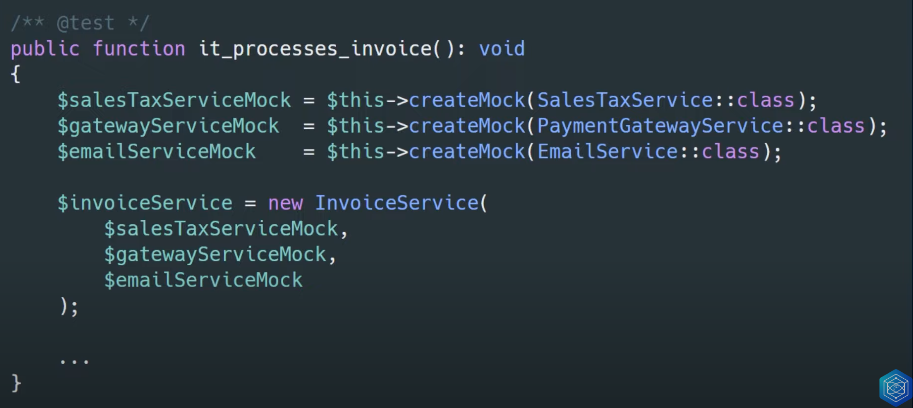
mã hóa các lớp này một cách chăm chỉ trong đơn vị xây dựng nơi mà lớp dịch vụ hóa đơn chịu trách nhiệm tạo ra. Sự phụ thuộc này tạo ra sự kết nối chặt chẽ làm cho mã khó bảo trì và kiểm tra hơn.



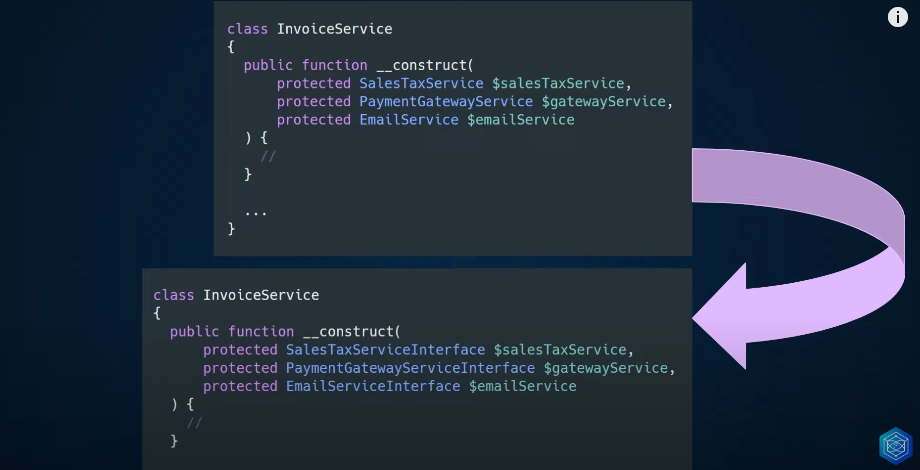
* Thay vì lớp dịch vụ hóa đơn tự tạo ra sự phụ thuộc của riêng nó, chúng ta có thể đảo ngược dòng chảy của kiểm soát và truyền các phụ thuộc bởi một hàm tạo.



* Dependency injection về cơ bản là một hình thức đảo ngược kiểm soát hay nói ngắn gọn là IOC
* IOC là một nguyên tắc chung và có thể được áp dụng cho mọi thứ và không chỉ tiêm phụ thuộc với IOC, dòng chảy của điều khiển về cơ bản bị đảo ngược, nơi thay vì lớp làm tê liệt các phụ thuộc của chính nó và tạo ra các đối tượng nó đảo ngược như vậy rằng mã bên ngoài có trách nhiệm cung cấp các phụ thuộc cho lớp.
* Có một bài viết tuyệt vời về đảo ngược kiểm soát mà tôi sẽ để lại liên kết trong phần mô tả . Vì vậy, hãy kiểm tra nếu bạn muốn đọc thêm về nó, có nhiều loại tiêm phụ thuộc khác nhau như tiêm phương pháp tiêm setter.
* Và vì vậy cái này được gọi là constructor injection và đó là những gì chúng ta sẽ tập trung trong bài học này
* Như tôi đã đề cập trước đây, dependency injection cùng với nhiều lợi ích khác giúp việc viết các bài kiểm tra dễ dàng hơn, ví dụ như trong bài học trước:
* Chúng tôi đã có thể tạo thử nghiệm tăng gấp đôi và vượt qua những điều đó dưới dạng phụ thuộc thay vì các lớp ban đầu mà Làm cho thử nghiệm dễ dàng hơn nhiều
* Chúng tôi cũng có thể sử dụng các giao diện và thay vì chấp nhận các lớp cụ thể là phụ thuộc



* Thay vì chấp nhận các lớp học cụ thể như phụ thuộc, Chúng tôi có thể chấp nhận một số triển khai giao diện cho phép chúng tôi trao đổi các triển khai dễ dàng khi cần thiết trong thời gian chạy .



* Được rồi, bây giờ chúng ta đã hiểu tiêm phụ thuộc, hãy xem các hộp chưa injections phụ thuộc đóng vai trò gì
* Hãy thu nhỏ lớp dịch vụ hóa đơn và di chuyển lên một cấp độ mà chúng ta cần chính lớp dịch vụ hóa đơn
* Đây có thể là một bộ điều khiển hoặc bất kỳ lớp nào khác cần một đối tượng của lớp dịch vụ hóa đơn
* Giả sử chúng ta có bộ điều khiển hóa đơn với phương thức lưu trữ xử lý hóa đơn sau khi người dùng Gửi biểu mẫu
* Vì vậy, về cơ bản người dùng điền vào một biểu mẫu với một số thông tin và chúng tôi chỉ giữ ví dụ này siêu đơn giản và chỉ sử dụng tên và trường số tiền
* 
* Nhưng tất nhiên, trong một ứng dụng thực tế, nó sẽ có nhiều lĩnh vực hơn như số thẻ tín dụng, địa chỉ thanh toán, v.v., nhưng đó là bên cạnh điểm mà phương thức store là nơi chúng ta cần đối tượng của lớp dịch vụ hóa đơn vì chúng ta cần xử lý hóa đơn
* Vì vậy, chúng ta đang tạo một đối tượng mới của Invoice Service Class ở đây Nhưng bởi vì chúng ta phải truyền lại các phụ thuộc theo cách thủ công, chúng ta cũng đang tạo ra các phụ thuộc này và đang truyền chúng xuống
* Vì vậy, bạn có thấy vấn đề ở đây trong khi lớp dịch vụ hóa đơn được kết hợp lỏng lẻo và sử dụng dependency injection, bộ điều khiển được kết hợp chặt chẽ và chúng tôi vẫn khó mã hóa các phụ thuộc
* 