**Document Smart Contract SellLaptopProduct**

1. **Yêu cầu đặt ra**

Tạo ra một Smart Contract để bán các sản phẩm laptop. Yêu cầu tạo ra các chức năng có thể quản lý thông tin admin (thêm, sửa, xóa), tạo ra các chức năng (thêm, sửa xóa) để quản lý các sản phẩm laptop hiện có, và tạo ra các chức năng quản lý thông tin các khách hàng đã mua hàng. Và tạo ra một địa chỉ để nhận tiền từ các khách hàng.

Khách hàng có thể tự đăng ký vào smart contract và có thể tự mua các sản phẩm laptop mà họ muốn. Và khách hàng có thể xem lại các đơn hàng mà họ đã mua.

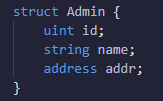
Admin có thể tự chuyển một số sản phẩm cho khách hàng.

Admin có thể chuyển trạng thái của đơn hàng về Complete.

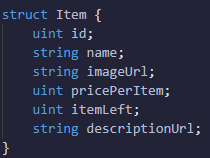
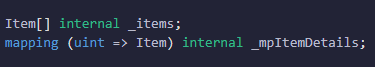
* Các ràng buộc cần thực hiện trong Smart Contract
* Chỉ có tài khoản thuộc admin mới có thể thêm, sửa, xóa các sản phẩm. Và có thể thực hiện việc chuyển sản phẩm cho khách hàng, có thể cập nhật trạng thái đơn hàng.
* Chỉ có các tài khoản thuộc nhóm khách hàng đã đăng ký thì mới có thể mua hàng.
* Kiểm tra số lượng sản phẩm hiện có phải lớn hơn số lượng khách hàng mua.
* Kiểm tra số tiền khách hàng có đủ để mua sản phẩm đó hay không trước khi tạo ra đơn hàng.

1. **Phân tích yêu cầu**

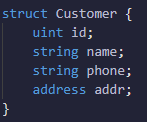
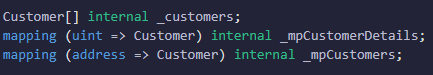
Tạo ra một cấu trúc để quản lý thông tin admin (id, name, addr), đồng thời tạo ra biến **\_admins** có kiểu dữ liệu là **Admin[]** để lưu danh sách thông tin admin, và tạo ra một mapping **\_mpAdminDetails** để ánh xạ từ một **adminId** ra thông tin **admin** một cách nhanh nhất. Phạm vi truy cập của 2 biến **\_admin, \_mpAdminDetails** đều là **internal.**

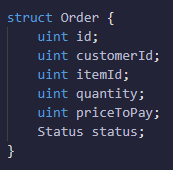
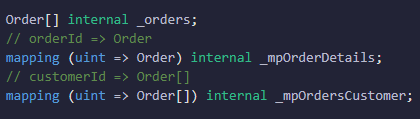
Tạo ra một cấu trúc để quản lý thông tin các sản phẩm hiện có (id, name, imageUrl, pricePerItem, itemLeft, descriptionUrl) đồng thời tạo ra biến \_**items** có kiểu dữ liệu là **Item[]** để lưu danh sách thông tin các sản phẩm hiện có, và tạo ra một mapping \_**mpItemDetails** để ánh xạ từ một **itemId** ra thông tin sản phẩm một cách nhanh nhất. Phạm vi truy cập của 2 biến \_**items, \_mpItemDetails** đều là **internal.**

**** ****

Tiếp theo để quản lý thông tin các khách hàng ta tạo ra biến **\_customers** có kiểu dữ liệu là **Customer[]** và tạo ra một mapping \_**mpCustomerDetails** để ánh xạ từ **customerId** và tạo ra một mapping **\_mpCustomers** để ánh xạ từ địa chỉ ví ra thông tin khách hàng một cách nhanh nhất. Phạm vi truy cập của 3 biến \_**customers, \_mpCustomerDetails, \_mpCustomers** đêu là **internal.**

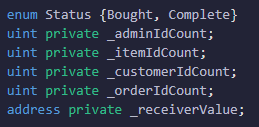
 

Để quản lý được các sản phẩm nào đã được bán ra và khách hàng nào đã mua ta tạo ra một cấu trúc **Order** (id, customerId, itemId, quantity, priceToPay, status) từ cấu trúc này ta sẽ biết được khách hàng nào đã mua, mua sản phẩm nào với số lượng là bao nhiêu, số tiền cần thanh toán và cuối cùng là trạng thái của đơn hàng. Để lấy nhanh thông tin một đơn hàng từ id ta tạo ra một mapping \_**mpOrderDetails** biến này sẽ ánh xạ từ id ra thông tin đơn hàng tương ứng. Ngoài ra để xem được những đơn đã đặt của khách hàng bất kỳ ta tạo ra một mapping \_**mpOrderCustomer** biến này sẽ ánh xạ từ id của khách hàng ra thông tin các đơn hàng mà họ đã đặt.

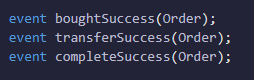
 

Ta tạo ra một **enum** để quản lý các trạng thái của đơn hàng, ta sẽ tạo ra 2 trạng thái là **bought** là đã mua hàng và **complete** là đơn hàng đã hoàn thành.

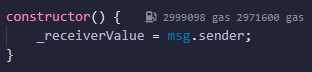
Tiếp theo ta tạo ra các biến có phạm vi truy cập là **private** để lưu giá trị Id cho các cấu trúc đã tạo ở trên và các biến này luôn tăng. Và để có thể nhận tiền từ khách hàng ta tạo ra một biến **\_receiverValue.**



Và ta cũng tạo ra 3 sự kiện để thông báo khi thực hiện xong một chức năng. Sự kiện **boughtSuccess** sẽ được gọi khi khách hàng thực hiện mua sản phẩm thành công, sự kiện **transferSuccess** sẽ được gọi khi admin thực hiện chuyển sản phẩm cho một khách hàng thành công, và sự kiện **completeSuccess** sẽ được gọi khi admin cập nhật lại trạng thái đơn hàng là **Complete.**



Trong hàm **constructor** ta thực hiện việc khởi tạo địa chỉ nhận tiền từ khách hàng là địa chỉ tài khoản **deploy** Smart Contract này.



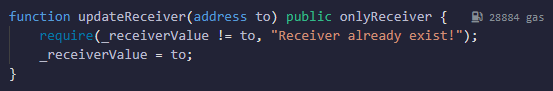
Dựa vào yêu cầu đã đặt ra ta tiến hành tạo ra các **modifier** cần thiết.

* **onlyReceiver:** sẽ kiểm tra địa chỉ đang thực hiện có phải là receiver không nếu không phải sẽ tiến hành revert và đưa ra thông báo.
* **isExistItem:** sẽ kiểm tra sản phẩm này có tồn tại dựa vào id truyền vào hay không nếu không tồn tại sản phẩm nào sẽ tiến hành revert và đưa ra thông báo.
* **isExistAddress:** modifier này được dùng để thực hiện kiểm tra địa chỉ ví truyền vào có hợp lệ khi thực hiện chức năng thêm, cập nhật thông tin khách hàng hay không, bời vì mỗi khách hàng chỉ có một địa chỉ ví duy nhất nên ta phải kiểm tra trước khi thêm, còn khi cập nhật ta còn phải kiểm tra thêm một điều kiện nữa là phải khác id khách hàng đang cập nhật. Nếu địa chỉ truyền vào đã tồn tại thì sẽ tiến hành revert và đưa ra thông báo.
* **isExistOrderId:** sẽ kiểm tra xem id truyền vào có tồn tại đơn hàng nào hay không nếu không tồn tại thì sẽ tiến hàng revert và đưa ra thông báo.
* **isExistCustomer:** sẽ kiểm tra địa chỉ truyền vào có phải là khách hàng đã đăng ký vào nhóm khách hàng thuộc smart contract này chưa. Nếu đia chỉ truyền vào không thuộc nhóm khách hiện có sẽ tiến hành revert và đưa ra thông báo.
* **onlyAdmin:** sẽ kiểm tra xem địa chỉ đang thực hiện có thuộc nhóm admin hiện tại không, nếu không tồn tại sẽ tiến hành revert và đưa ra thông báo.



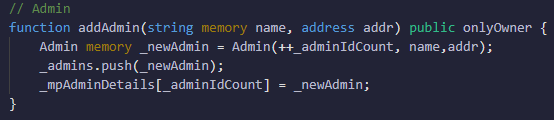
* 1. **Chức năng cập nhật địa chỉ người nhận tiền**

Hàm **updateReceiver** sẽ nhận vào 1 tham số là địa chỉ sẽ được cập nhật là receiverValue và trước khi thực hiện hàm này ta kiểm tra địa chỉ đang thực hiện có phải là **receiverValue** hiện tại hay không và khi thực thi hàm ta cần kiểm tra địa chỉ truyền vào có phải là địa chỉ **receiverValue** hiện tại nếu là **receiverValue** hiện tại thì kết thúc hàm và đưa ra thông báo ngược lại thì cập nhật lại địa chỉ **receiverValue**.



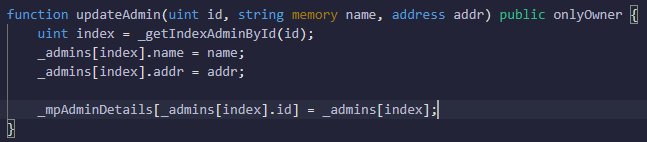
1. **Chức năng quản lý admin**
   1. **Chức năng thêm admin**

Để thực hiện hàm này ta cần truyền 2 tham số là **name, addr** và trước khi thực hiện hàm này ta cũng kiểm tra địa chỉ đang thực hiện có phải là chủ hợp đồng hay không. Nếu thỏa là chủ hợp đồng thì ta tiến hành tạo mới admin và lưu thông tin này vào 2 biến đã đề cập ở trên là **\_admins** và **\_mpAdminDetails.** Khi tạo mới admin thì biến **\_adminIdCount** cũng tự động cộng thêm 1.

****

* 1. **Chức năng cập nhật admin**

Để biết cập nhật thông tin cho admin nào thì ta cần truyền vào tham số **id** và các thông tin cần cập nhật cho admin là **name, addr** cũng tương tự như hàm thêm admin ta cũng kiểm tra địa chỉ đang thực hiện có phải là chủ hợp đồng trước khi vào hàm. Khi vào hàm ta kiểm tra id truyền vào có tồn tại hay không nếu không tồn tại thì kết thúc hàm. Ngược tại thì lấy ra thông tin admin theo **id** và cập nhật lại **name, addr**, đồng thời ta cũng cập nhật lại thông tin vào biến **\_mpAdminDetails.**

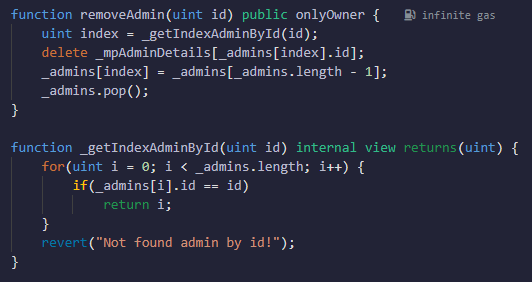


* 1. **Chức năng xóa admin**

Để biết xóa admin nào ta cần truyền vào tham số **id** và trước khi vào hàm ta kiểm tra xem địa chỉ đang thực hiện có phải là chủ hợp đồng hay không. Nếu thỏa ta lấy thông admin ra theo id và tiến hành xóa admin này trong 2 biến lưu trữ trước đó **\_mpAdminDetails, \_admins.**

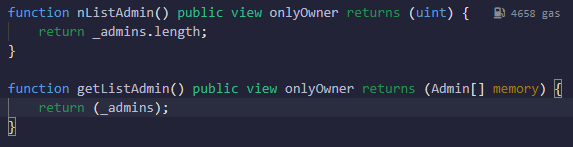
Để xóa admin trong mảng **Admin[]** ta không thể dùng **delete** như thường bởi vì khi xóa như vậy mảng vẫn không thay đổi kích thước và index của mảng không được liền mạch với nhau. Để giải quyết vấn đề này ta tiến hành làm như sau:

* Đầu tiên ta tìm index theo id admin cần xóa.
* Tiếp đến ta cập nhật lại thông tin admin này giống với thông tin admin ở cuối mảng.
* Cuối cùng dùng hàm **pop()** để xóa vị trí cuối của mảng.



* 1. **Chức năng xem số lượng và danh sách admin hiện tại**

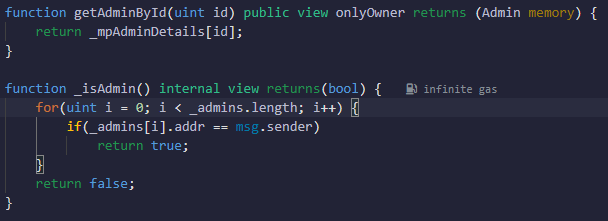
Ở trên ta sử dụng biến \_**admins** để lưu danh sách admin nên ta có thể biết được danh sách admin hiện tại và dể xem số lượng admin hiện có ta sử dụng **\_admins.length.**



* 1. **Chức năng xem chi tiết thông tin admin theo id**

Để xem thông tin admin dựa vào id ta sử dụng biến **\_**mpAdminDetails**, \_mpAdminDetails** sẽ ánh xạ từ id ra thông tin của admin một cách nhanh chóng.

Hàm **\_isAdmin** sẽ kiểm tra xem địa chỉ đang thực hiện có tồn tại trong danh sách \_admins không nếu tồn tại thì trả về **true** ngược lại thì trả về **false.** Hàm này được sử dụng trong modifier **onlyAdmin**



1. **Chức năng quản lý sản phẩm**

**4.1 Chức năng thêm sản phẩm**