**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.

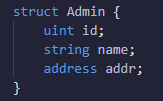
Admin có thể tự chuyển một số sản phẩm cho khách hàng.

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

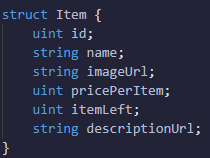
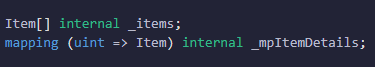
* Các ràng buộc cần thực hiện trong Smart Contract
* Chỉ có tài khoản thuộc admin mới có thể thêm, sửa, xóa các sản phẩm. Và có thể thực hiện việc chuyển sản phẩm cho khách hàng, có thể cập nhật trạng thái đơn hàng.
* Chỉ có các tài khoản thuộc nhóm khách hàng đã đăng ký thì mới có thể mua hàng.
* Kiểm 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.
* Mỗi tài khoản khách hàng chỉ có 1 địa chỉ ví duy nhất và không có khách hàng nào trùng địa chỉ ví với nhau.

1. **Phân tích yêu cầu**

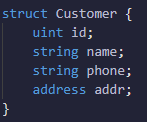
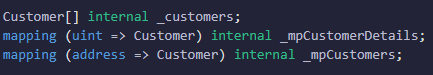
Tạo ra một cấu trúc để quản lý thông tin admin (id, name, addr), đồng thời tạo ra biến **\_admins** có kiểu dữ liệu là **Admin[]** để lưu danh sách thông tin admin, và tạo ra một mapping **\_mpAdminDetails** để ánh xạ từ một **adminId** ra thông tin **admin** một cách nhanh nhất. Phạm vi truy cập của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

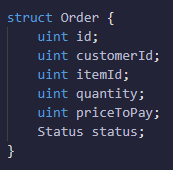
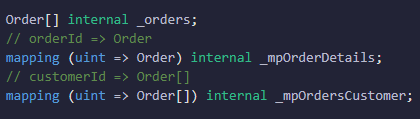
Tạo ra một cấu trúc để quản lý thông tin các sản phẩm hiện có (id, name, imageUrl, pricePerItem, itemLeft, descriptionUrl) đồng thời tạo ra biến \_**items** có kiểu dữ liệu là **Item[]** để lưu danh sách thông tin các sản phẩm hiện có, và tạo ra một mapping \_**mpItemDetails** để ánh xạ từ một **itemId**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

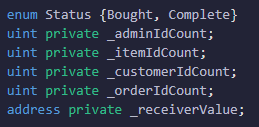
 

Để quản lý được các sản phẩm nào đã được bán ra và khách hàng nào đã mua ta tạo ra một cấu trúc **Order** (id, customerId, itemId, quantity, priceToPay, status) từ cấu trúc này ta sẽ biết được khách hàng nào đã mua, mua sản phẩm nào với số lượng là bao nhiêu, số tiền cần thanh toán và cuối cùng là trạng 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.

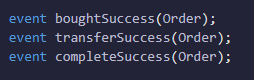
 

Ta tạo ra một **enum** để quản lý các trạng thái của đơn hàng, ta sẽ tạo ra 2 trạng thái là **bought** là đã mua hàng và **complete** là đơn hàng đã hoàn thành.

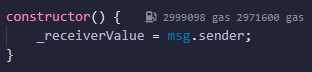
Tiếp theo ta tạo ra các biến có phạm vi truy cập là **private** để lưu giá trị Id cho các cấu trúc đã tạo ở trên và các biến này luôn tăng. Và để có thể nhận tiền từ khách hàng ta tạo ra một biến **\_receiverValue.**



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.**

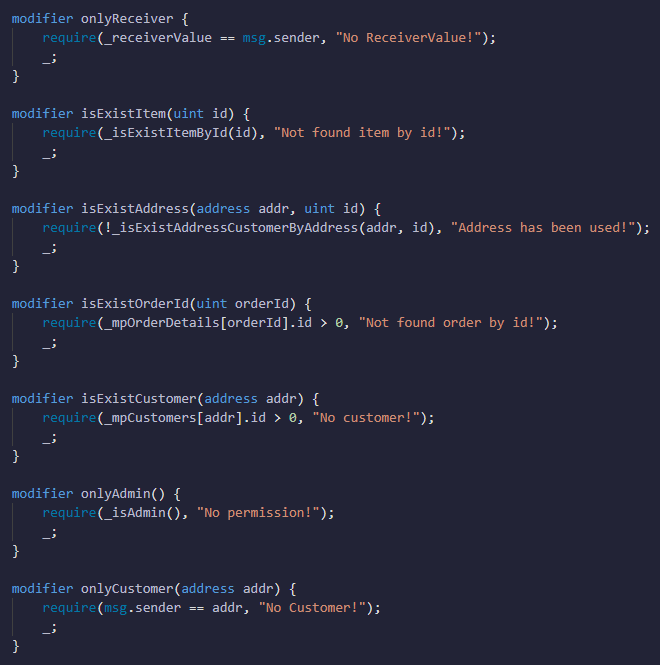


Trong hàm **constructor** ta thực hiện việc khởi tạo địa chỉ nhận tiền từ khách hàng là địa chỉ tài khoản **deploy** Smart Contract này.



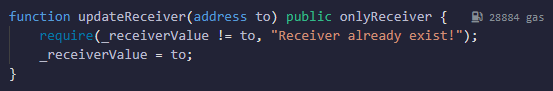
Dựa vào yêu cầu đã đặt ra ta tiến hành tạo ra các **modifier** cần thiết.

* **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 danh sách admin hiện tại không, nếu không tồn tại sẽ tiến hành revert và đưa ra thông báo.
* **onlyCustomer:** sẽ kiểm tra xem địa chỉ đang thực hiện có giống với dịa chỉ được truyền vào hay không nếu khác nhau thì sẽ tiến hành revert và đưa ra thông báo. Hàm này được dùng để kiểm tra khách hàng đang cập nhật có phải là chính khách hàng đó tự cập nhật hay không.



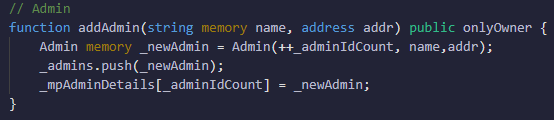
* 1. **Chức năng cập nhật địa chỉ người nhận tiền**

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**.



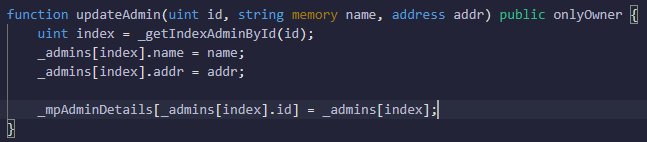
1. **Chức năng quản lý admin**
   1. **Chức năng thêm admin**

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.**

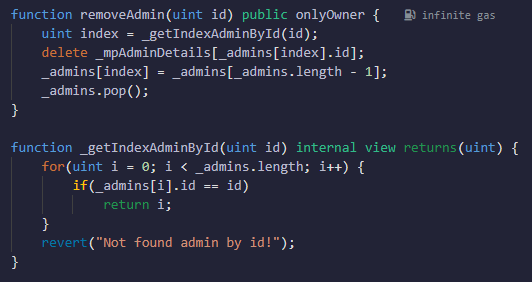


* 1. **Chức năng xóa admin**

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.**

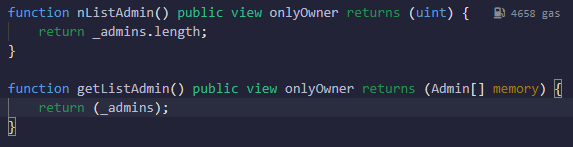
Để xóa admin trong mảng **Admin[]** ta không thể dùng **delete** như thường bởi vì khi xóa như vậy mảng vẫn không thay đổi kích thước và index của mảng không được liền mạch với nhau. Để giải quyết vấn đề này ta tiến hành làm như sau:

* 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**

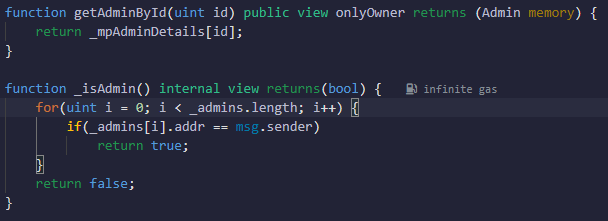
Ở trên ta sử dụng biến \_**admins** để lưu danh sách admin nên ta có thể biết được danh sách admin hiện tại và dể xem số lượng admin hiện có ta sử dụng **\_admins.length.**



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

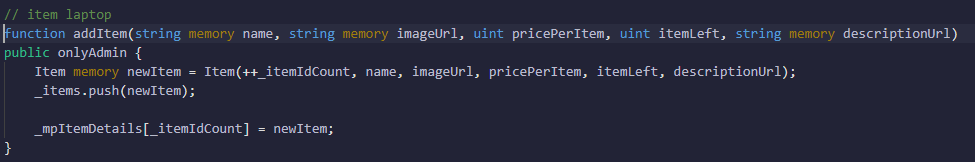
Để xem thông tin admin dựa vào id ta sử dụng biến **\_**mpAdminDetails**, \_mpAdminDetails** sẽ ánh xạ từ id ra thông tin của admin một cách nhanh chóng.

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**



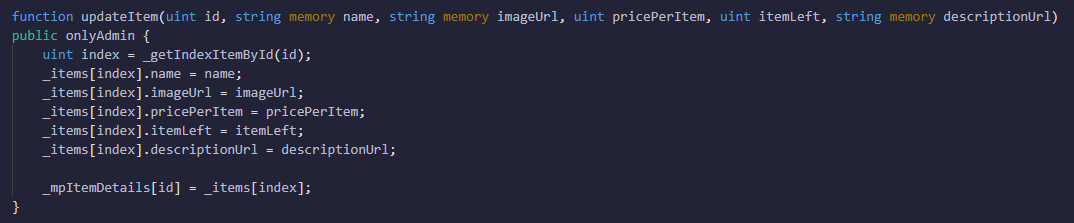
1. **Chức năng quản lý sản phẩm**
   1. **Chức năng thêm sản phẩm**

Để thực hiện hàm này ta cần truyền vào 5 tham số là **name, imageUrl, pricePerItem, itemLeft, descriptionUrl** và trước khi thực hiện hàm này ta sử dụng modifier **onlyAdmin** để kiểm tra địa chỉ đang thực hiện có phải là admin hay không. Nếu thỏa điều kiện của modifier ta thực thì hàm, đầu tiên ta tiến hành khởi tạo một đối tượng Item và thêm đối tượng này vào danh sách item hiện đang có và lưu đối tượng này vào biến mapping mà ta đã khai báo ở trên.



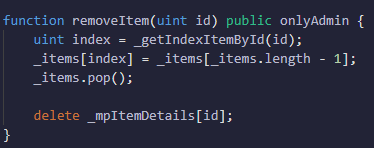
* 1. **Chức năng cập nhật sản phẩm**

Để thực hiện hàm này ta cũng truyền các tham số như ở hàm tạo item nhưng ta cần phải biết mình cần cập nhật thông tin cho sản phẩm nào nên ta phải truyền thêm một tham số là id, và trước khi thực hiện hàm ta cũng sử dụng modifier **onlyAdmin** như ở trên. Nếu đã thỏa điều kiện của modifier thì tiến hành thực thi hàm, đầu tiên ta tìm vị trí của item theo id, nếu không có item nào thì dưa ra thông báo và kết thúc hàm, ngược lại thì cập nhật lại thông tin item tại vị trí đã tìm thấy theo các tham số đã truyền vào và đồng thời ta cũng cập nhật lại thông tin trong biến mapping của item.

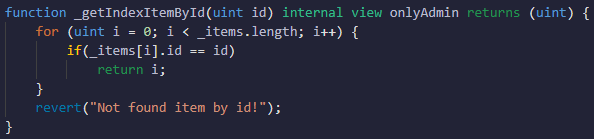


* 1. **Chức năng xóa sản phẩm**

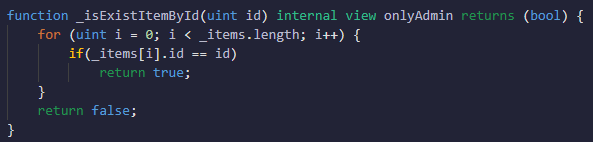
Để biết xóa sản phẩm nào ta cần truyền vào tham số id. Và trước khi thực hiện hàm này ta cũng sử dụng modifier **onlyAdmin** như ở trên. Nếu thỏa điều kiện của modifier thì thực thi hàm, đầu tiên ta tìm vị trí của item theo id cần xóa trong danh sách item hiện có, nếu tìm được item ta tiến hành cập nhật lại thông tin item tại vị trí cần xóa là thông tin của item tại vị trí cuối cùng của danh sách và tiếp theo ta sử dụng hàm **pop()** để xóa item tại vị trí cuối cùng và đồng thời ta cũng xóa luôn item trong biến mapping \_**mpItemDetails** đã tạo ở trên.



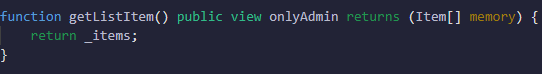
Hàm \_**getIndexItemById** được dùng để lấy vị trí của item theo tham số **id** được truyền vào, nếu tìm thấy item trong danh sách item hiện đang có ta kết thúc hàm và trả về vị trí hiện tại ngược lại thì ta revert và đưa ra thông báo. Hàm này được dùng để cập nhật thông tin hay xóa một item trong mảng hiện tại.



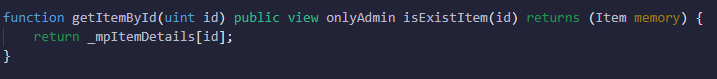
Hàm \_**isExistItemById** cũng thực hiện các bước như ở hàm **\_getIndexItemById** nhưng ở hàm này ta sẽ trả về kiểu dữ là bool, mục đích của hàm này là sử dụng để kiểm tra xem id truyền vào có tồn tại hay không nếu tồn thì ta mới tiến hành thực thi hàm.



Để xem được danh sách các sản phẩm hiện đang có ta có thể sử dụng biến \_**items** đã tạo ở trên.



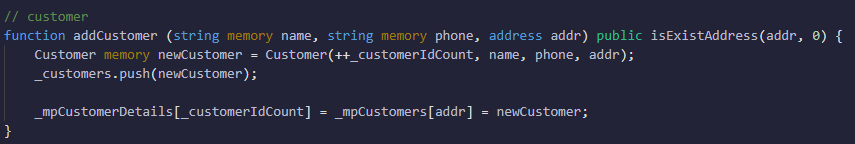
Hoặc để xem thông tin của một sản phẩm nào đó ta có thể sử dụng biến \_**mpItemDetails** biến này sẽ ánh xạ từ một id ra thông tin của một sản một cách nhanh chóng.



1. **Chức năng quản lý khách hàng**
   1. **Chức năng thêm khách hàng**

Để thao tác thêm mới một khách hàng ta cần truyền vào 3 tham số tương ứng với các trường thông tin mà ta cần quan tâm đối với khách hàng đó (name, phone, addr) và mỗi khách hàng chỉ có một địa chỉ ví nên trước khi thực hiện hàm này ta cần phải có một bước kiểm tra xem địa chỉ khách hàng này đã được sử dụng hay chưa và ở trên chúng ta đã tạo sẵn một modifier **isExistAddress** để làm việc này rồi nên ta chỉ cần gọi lại modifier này vào trước hàm này là được.

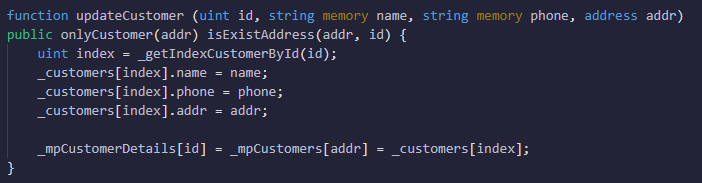
Khi thỏa điều kiện của modifier thì tiến hành thực thì hàm, đầu tiên ta khởi tạo một đối tượng Customer và thêm đối tượng này vào mảng \_**customers** hiện tại, đồng thời lưu đối tượng này vào một biến mapping \_**mpCustomerDetails** đã tạo ở trên, biến này sẽ giúp ta lấy ra thông tin một khách hàng một cách nhanh chóng dựa vào id.



* 1. **Chức năng cập nhật thông tin khách hàng**

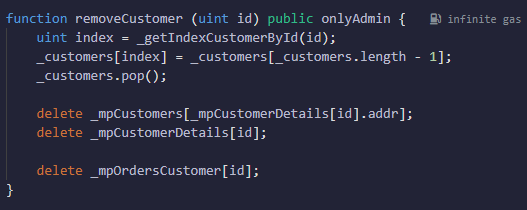
Để thực hiện hàm này ta cần truyền vào tham số id để ta biết được là đang cập nhật thông tin cho khách hàng nào. Và trước khi vào ta cũng thực hiện kiểm tra xem địa chỉ đang thực hiện có phải là khách hàng đang dược cập nhật hay không bằng modifier **onlyCustomer** đã tạo ở trên và kiểm tra địa chỉ ví cần cập nhật có hợp lệ hay không ta cũng sử modifier **isExistAddress** như ở hàm thêm khách hàng. Nếu thóa điều kiện của modifier thì thực thi hàm, đầu tiên ta sẽ lấy ra vị trí của khách hàng cần cập nhật dựa vào id truyền vào. Nếu tìm thấy vị trí khách hàng ta tiến hành cập nhật lại thông tin theo các tham số đã truyền vào và đồng thời cũng cập nhật lại thông tin này vào biến \_**mpCustomerDetails và \_mpCustomers.**

* **\_mpCustomerDetails** sẽ ánh xạ từ id ra thông tin khách hàng.
* \_**mpCustomers** sẽ ánh xạ từ địa chỉ address ra thông tin khách hàng.

****

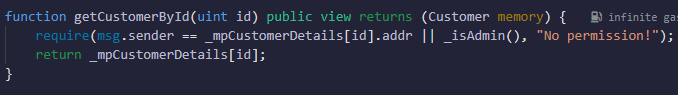
* 1. **Xóa thông tin khách hàng**

Để thực hiện hàm này ta truyền vào tham số id là id của khách hàng cần xóa. Và chỉ có admin mới có quyền thực hiện chức năng này nên ta sử dụng modifier **onlyAdmin**. Nếu thỏa điều kiện ta thực thi hàm, đầu tiên lấy vị trí của khách hàng cần xóa, tiếp theo ta cập nhật lại thông tin tại ví trí này là thông tin của khách hàng ở vị trí cuối cùng của mảng và dùng hàm **pop()** để xóa khách hàng ở vị trí cuối. Và ta cũng xóa thông tin khách hàng trên biến \_**mpCustomerDetails, \_mpCustomers.** Và xóa các đơn hàng mà khách hàng này đã đặt trước đó.



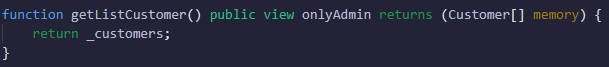
* 1. **Chức năng xem thông tin khách hàng**

Để xem được thông tin của khách hàng ta truyền tham số id là id của khách hàng cần xem. Đầu tiên hàm sẽ kiểm tra địa chỉ đang thực hiện có phải là khách hàng có id này không hoặc có phải là admin không nếu không thỏa điều kiện sẽ kết thúc hàm và đưa ra thông báo ngược lại thì trả về thông tin của khách hàng.



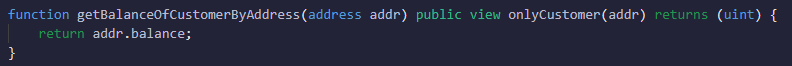
* 1. **Chức năng xem danh sách khách hàng**

Để xem danh sách khách hàng hiện có ta sử dụng hàm **getListCustomer** và hàm này sẽ kiểm tra địa chỉ đang thực hiện có phải là admin không bằng modifier **onlyAdmin** đã được định nghĩa ở trên. Nếu thỏa điều kiện của modifier thì thực hiện hàm và trả về danh sách khách hàng hiện có.



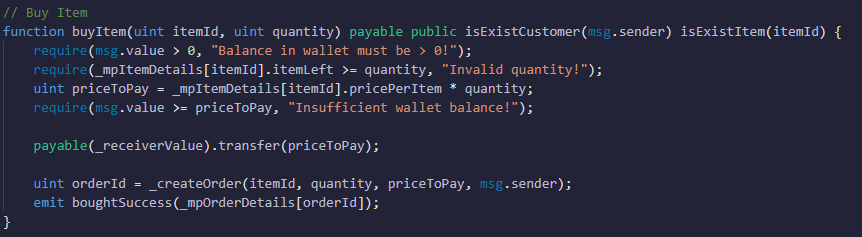
* 1. **Chức năng xem số dư hiện có của khách hàng**

Để khách hàng có thể xem dược số dư của mình, ta sử dụng hàm **getBalanceOfCustomerByAddress** và hàm này cần truyền vào tham số là địa chỉ của khách hàng cần xem, và trước khi thực hiện hàm sẽ phải kiểm tra địa chỉ đang thực hiện có phải là địa chỉ được truyền vào hay không bằng cách sử dụng modifier **onlyCustomer** đã được định nghĩa ở trên. Nếu thỏa mãn điều kiện của modifier ta thực thi hàm và trả về số dư hiện tại của địa chỉ truyền vào.

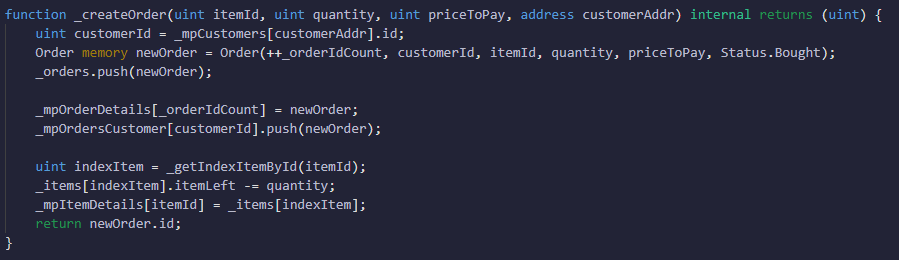


1. **Chức năng mua sản phẩm**

Để thực hiện chức năng này ta sử dụng hàm **buyItem** và hàm này cần truyền vào 2 tham số là id sản phẩm cần mua và số lượng là bao nhiêu (**itemId. quantity**) và trước khi thực hiện hàm này ta kiểm tra xem địa chỉ đang thực hiện có phải là khách hàng và kiểm tra xem id sản phẩm truyền vào có tồn tại hay không bằng 2 modifier đã định nghĩa ở trên là **isExistCustomer, isExistItem**. Và hàm này có sử dụng hàm **transfer** để chuyển tiền cho **receiverValue** nên ta cần thêm từ khóa **payable** phía sau hàm. Khi thỏa mãn các điều kiện của modifier ta thực thì hàm, đầu tiên ta kiểm tra số dư khách hàng có **> 0**, tiếp theo kiểm tra số lượng sản phẩm còn lại có >= số lượng mà khách hàng mua hay không, nếu thỏa mãn 2 điều kiện này ta tiến hành tính số tiền mà khách hàng sẽ phải trả và kiểm tra xem số dư của khách hàng có >= số tiền phải trả hay không. Nếu thỏa mãn điều kiện này ta thực hiện việc chuyển tiền cho người nhận, khi chuyển tiền thành công ta tiến hành tạo ra đơn hàng và cuối cùng ta kích hoạt event **boughtSuccess** để trả về thông tin đơn hàng vừa tạo.

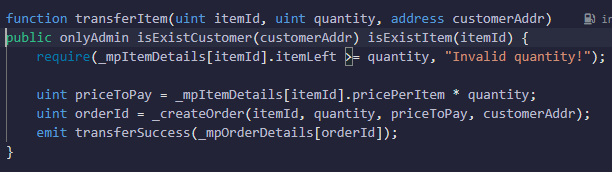


Hàm **\_createOrder** sẽ nhận vào các tham số (itemId, quantity, priceToPay, customerAddr), đầu tiên ta lấy id của customer và tiến hành khởi tạo đối tượng **Order** và thêm đối tượng này vào danh sách order hiện đang có và thêm order này vào danh sách order của customer đã đặt hàng. Cuối cùng ta trừ số lượng của item và trả về id của order vừa tạo.



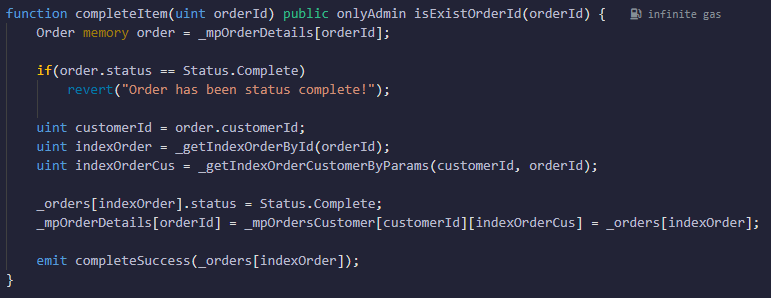
1. **Chức năng chuyển sản phẩm cho khách hàng**

Để thực hiện chuyển sản phẩm cho khách hàng ta sử dụng hàm **transferItem** hàm này cần truyền vào các tham số là id sản phẩm cần chuyển, số lượng, và địa chỉ khách hàng sẽ nhận sản phẩm (itemId, quantity, customerAddr). Trước khi thực hiện hàm, sẽ kiểm tra xem địa chỉ đang thực hiện có phải là admin không **(modifier onlyAdmin),** kiểm tra địa chỉ khách hàng truyền vào có tồn tại không (modifier isExistCustomer), kiểm tra xem id sản phẩm có tồn tại không **(modifier isExistItem),** nếu thỏa mãn các modifier này thì thực thi hàm, đầu tiên ta sẽ kiểm tra số lượng còn lại của sản phẩm có >= với số lượng được chuyển hay không nếu số lượng chuyển đi lớn hơn ta kết thúc hàm và đưa ra thông báo. Ngược lại ta tính số tiền cần thanh toán và tạo ra đơn hàng, cuối cùng ta kích hoạt event **transferSuccess** trá về thông tin đơn hàng vừa tạo.



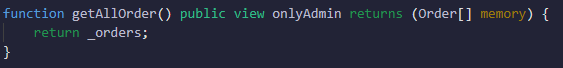
1. **Chức năng complete đơn hàng**

Để thực hiện chức năng này ta sử dụng hàm **completeItem** và truyền vào tham số là id của order cần thay đổi trạng thái là complete. Trước khi thực hiện hàm sẽ kiểm tra địa chỉ đang thực hiện có phải là admin **(modifier onlyAdmin)**, và kiểm tra xem id đơn hàng truyền vào có tồn tại **(isExistOrderId)**. Nếu thỏa mãn điều kiện của các modifier thì thực thi hàm. Đầu tiên lấy ra thông tin đơn hàng theo id, và kiểm tra xem trạng thái đơn hàng đã complete chưa nếu đã complete thì kết thúc hàm và đưa ra thông báo, ngược lại thì cập nhật lại trạng thái đơn hàng và kích hoạt event **completeSuccess** trả về thông tin đơn hàng.



1. **Chức năng xem danh sách đơn hàng**

Để xem danh sách đơn hàng hiện có ta sử dụng hàm **getAllOrder** và trước khi thực hiện ta kiểm tra địa chỉ đang thực hiện có phải là admin không **(modifier onlyAdmin)** nếu thỏa điều kiện ta thực thi hàm và trả về danh sách order đã được lưu ở biến \_**orders**.



1. **Chức năng xem danh sách đơn hàng của khách hàng**

Để xem danh sách đơn hàng của khách hàng ta sử dụng hàm **getAllOrderCustomerById** với tham số truyền vào là id của khách hàng. Trước tiên ta kiểm tra xem địa chỉ đang thực hiện có phải là địa chỉ của khách hàng đang cần tra cứu hay không hoặc kiểm tra xem địa chỉ có phải là admin không, nếu không thỏa mãn điều kiện thì kết thúc hàm và đưa ra thông báo, ngược lại thì lấy ra tất cả đơn hàng của khách hàng này. Biến \_**mpOrderCustomer** sẽ ánh xạ từ id của khách hàng ra danh sách các đơn hàng của khách hàng này.

