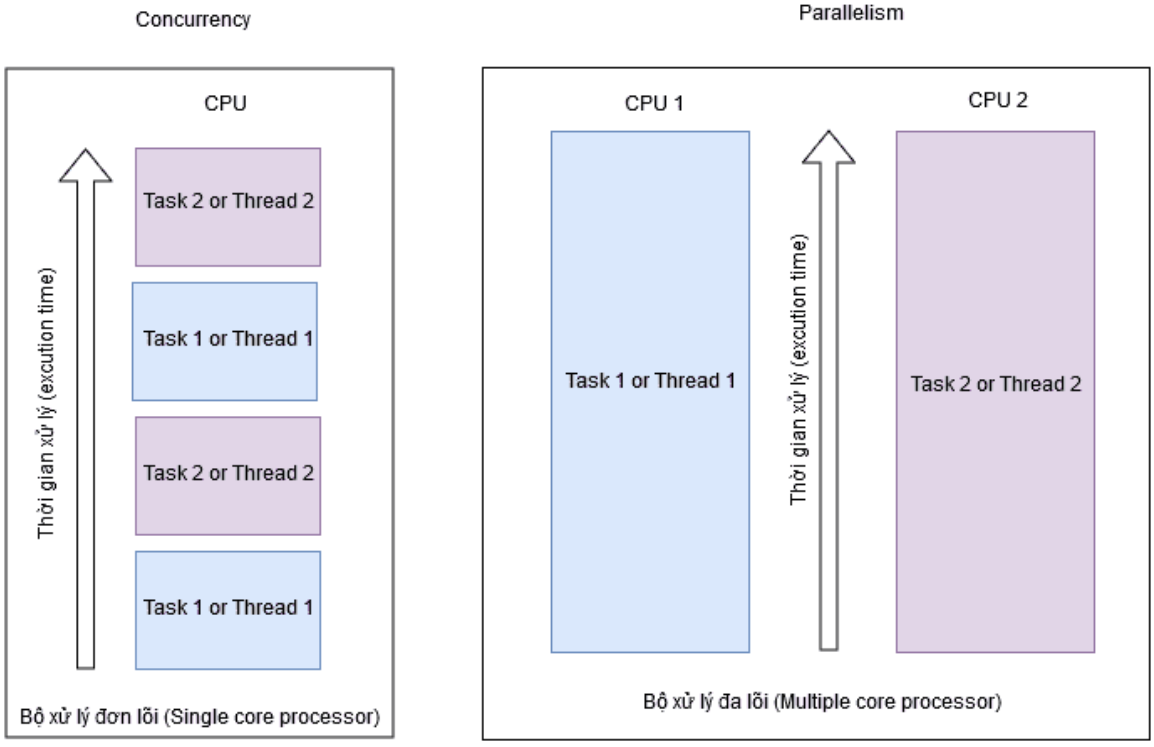
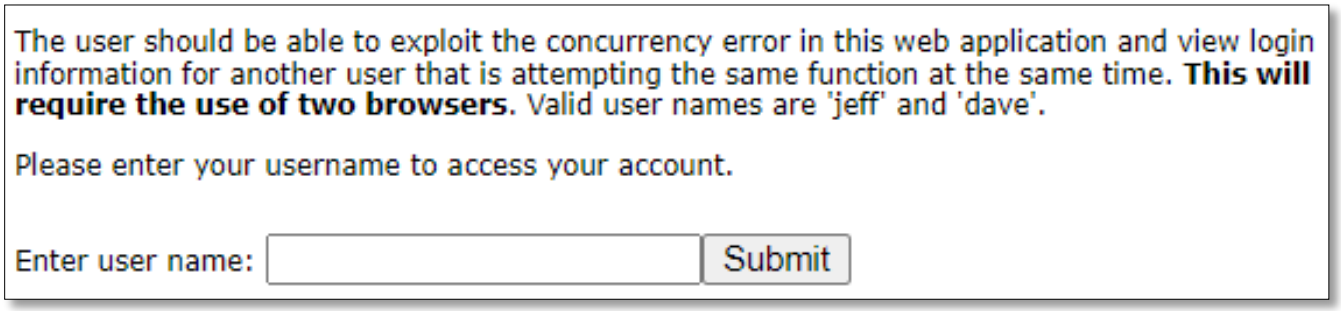
Concurrency

"Concurrency errors" nói về việc các lỗi bất đồng bộ xảy ra khi xử lý đồng thời trong việc lập trình đa luồng. Nếu đã học về lập trình, các bạn sẽ nghe qua khái niệm lập trình đa luồng (multi threads), không còn mới mẻ nữa mà nó là 1 phần của nhiều hệ thống lớn bây giờ. Và cái vấn đề xảy ra khi lập trình đa luồng như thế là việc ta phải kiểm soát được các chức năng, các luồng không đụng độ nhau trong quá trình hoạt động khi sử dụng cùng 1 tài nguyên dữ liệu.

"Concurrency" khác với "Parallelism", các bạn có thể đọc thêm về chúng trên mạng. Nôm na là "Parallelism" trong lập trình đa luồng sẽ xử lý song song, và cách thức để chống đụng độ xung đột, tương tranh tài nguyên cũng khác với "Concurrency". "Concurrency" xử lý mọi thứ theo 1 ngăn xếp, luân phiên thay đổi giữa các tiến trình để làm cho ta cảm thấy như mọi việc được làm 1 cách song song.



Thread Safety Problems



Bài mẫu để hiểu hơn về cách vận hành lỗi "Concurrency". Yêu cầu rằng hãy nhập tên người dùng và website sẽ hiển thị thông tin của 2 người dùng ấy lên cho chúng ta, chỉ 2 người dùng hợp lệ được chấp nhận là **jeff** và **dave.**

Okay, bài này khuyên ta nên làm trên 2 trình duyệt, cứ làm theo và mình sẽ giải thích lý do tại sao.

```
1  POST /WebGoat/attack?Screen=69&menu=800 HTTP/1.1
2  Host: 192.168.48.131
3  Content-Length: 27
4  Cache-Control: max-age=0
5  Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
6  Upgrade-Insecure-Requests: 1
7  Origin: http://192.168.48.131
8  Content-Type: application/x-www-form-urlencoded
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
   Safari/537.36
10 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
   d-exchange;v=b3;q=0.9
11 Referer: http://192.168.48.131/WebGoat/attack?Screen=69&menu=800
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: JSESSIONID=D7939D2C49C846E577EB886D12C27D48
15 Connection: close
16
17 username=jeff&SUBMIT=Submit
```

```
POST /WebGoat/attack?Screen=69&menu=800 HTTP/1.1
Host: 192.168.48.131
Content-Length: 27
Cache-Control: max-age=0
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.48.131
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
d-exchange;v=b3;q=0.9
Referer: http://192.168.48.131/WebGoat/attack?Screen=69&menu=800
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=D7939D2C49C846E577EB886D12C27D48
Connection: close

username=dave&SUBMIT=Submit
```

Ở đây mình dùng BurpSuite chặn lại request của cả 2 trang web, chỉ để nhấn forward 1 lúc 2 tiến trình trên cho nhanh thôi. nếu tự tin nhanh tay, các bạn có thể điền tên và submit thật nhanh trên tại giao diện 2 trang web cũng được. Cách nào cũng như nhau.

Và kiểm tra kết quả mà nó trả về, trang web thứ nhất mình submit với **jeff**, nó trả về **dave**, còn trang web thứ 2 mình submit với **dave** thì kết quả cũng là **dave**. Vậy chuyện quái gì đang xảy ra với người thứ nhất vậy ?

Okay, ở đây ta nhìn lại 1 chút, **jeff** chính là người đầu tiên mình gửi request, thì hiển nhiên đây sẽ là người được xử lý trước, còn yêu cầu cho **dave** sẽ được xử lý sau.

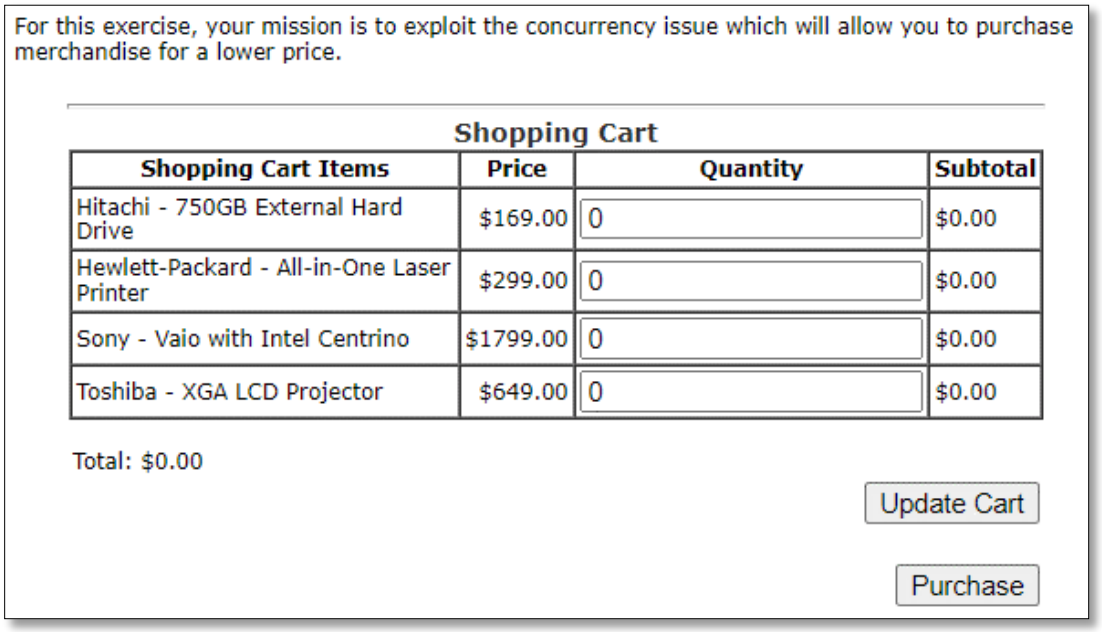
```
current_user = None

def user_login(new_user):
        if current_user is None:
                current_user = new_user
                thread.sleep(1.5)
                return current_user.info
        else:
                current_user = new_user
                return current_user.info
```

Mình đã cố gắng tìm kiếm thử đoạn code mà Server dùng để xử lý request cho bài lần này nhưng k thấy. Đoạn code trên là 1 đoạn mã giả mình lấy trên mạng, có thể đem nó mô phỏng lại cách mà Server xử lý yêu cầu của người dùng và trả về kết quả. Cùng phân tích 1 chút nào.

Từ đầu, giả định không hề có thông tin người dùng nào đang đăng nhập. Người dùng đầu tiên gửi yêu cầu và đặt biến toàn cục của **current_user** là chính họ và chuyển sang trạng thái ngủ của luồng trong 1500 ms (1,5 giây). Sau khi hết 1,5s, hàm trả về thông tin của current_user mà nó giả định là họ. Tuy nhiên, trong khi người dùng đầu tiên đang ngủ thì người dùng thứ hai đã đăng nhập. Sau đó, hàm user_login đã gán giá trị của current_user cho giá trị của người dùng thứ hai và trả lại dữ liệu đó cho người dùng thứ hai. Khi người dùng đầu tiên thức dậy từ chế độ ngủ của luồng, nó sẽ trả về thông tin cho người dùng 1, tuy nhiên, người dùng thứ hai đã thay đổi giá trị biến toàn cầu trong khi người dùng thứ nhất đang ngủ, khiến cả hai đều trả về giá trị là người dùng thứ hai đã đăng nhập.

Quay lại với lý thuyết ban nãy, đây là vấn đề xung đột về 1 dữ liệu chung khi có nhiều tiến trình cùng truy suất và sử dụng nó.

Concurrency: Shopping Cart Concurrency Flaw


For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.

Shopping Cart

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	$169.00	0	$0.00
Hewlett-Packard - All-in-One Laser Printer	$299.00	0	$0.00
Sony - Vaio with Intel Centrino	$1799.00	0	$0.00
Toshiba - XGA LCD Projector	$649.00	0	$0.00

Total: $0.00

Update Cart

Purchase

Bài tập đưa ra 1 trang mua hàng, cho phép ta nhập số lượng và có thể update lại giỏ hàng để xem tổng giá tiền phải thanh toán.


Shopping Cart

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	$169.00	0	$0.00
Hewlett-Packard - All-in-One Laser Printer	$299.00	0	$0.00
Sony - Vaio with Intel Centrino	$1799.00	1	$0.00
Toshiba - XGA LCD Projector	$649.00	0	$0.00

Total: $0.00

Update Cart

Purchase

Shopping Cart

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	$169.00	0	$0.00
Hewlett-Packard - All-in-One Laser Printer	$299.00	0	$0.00
Sony - Vaio with Intel Centrino	$1799.00	1	$1,799.00
Toshiba - XGA LCD Projector	$649.00	0	$0.00

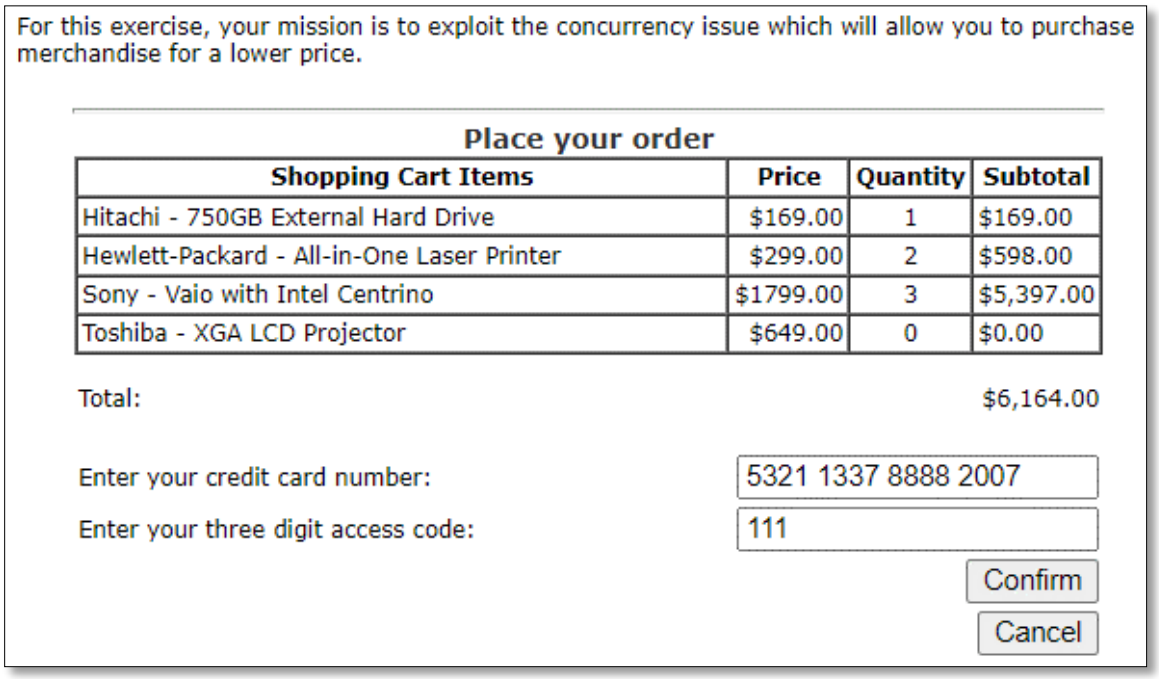
Total: $1,799.00

Update Cart

Purchase

Không update thì không có gì xảy ra đâu nhé.

Và sau khi update giỏ hàng thì ta có thể tiến hành thanh toán số hàng hóa đó.



For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.

Place your order

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	$169.00	1	$169.00
Hewlett-Packard - All-in-One Laser Printer	$299.00	2	$598.00
Sony - Vaio with Intel Centrino	$1799.00	3	$5,397.00
Toshiba - XGA LCD Projector	$649.00	0	$0.00

Total: $6,164.00

Enter your credit card number: 5321 1337 8888 2007

Enter your three digit access code: 111

Confirm

Cancel

N Okay, yêu cầu của bài đưa ra là ta hãy tìm cách thanh toán giỏ hàng của ta với món hàng đắt nhất nhưng số tiền lại là giá của sản phẩm rẻ nhất.

Sử dụng kỹ thuật, lối suy nghĩ và luồng hoạt động trên ví dụ của bài đầu tiên, ta có thể nôm na phân tích thế này:

Đầu tiên, ta thử bằng cách: người dùng 1 chọn món hàng đắt nhất ➔ update lại giỏ hàng để số tiền hiển thị cần thanh toán là số tiền của món hàng đắt nhất. ➔ Sau đó, người dùng thứ 2 chọn món hàng rẻ nhất và cũng update lại giỏ hàng. Mục đích của việc này là để kiểm tra xem, liệu ta có khả năng ghi đè giá trị hiển thị tổng số tiền của người thứ nhất (người chọn món hàng đắt) bằng số tiền của món hàng rẻ mà người thứ 2 chọn hay không.

Shopping Cart

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	$169.00	0	$0.00
Hewlett-Packard - All-in-One Laser Printer	$299.00	0	$0.00
Sony - Vaio with Intel Centrino	$1799.00	1	$1,799.00
Toshiba - XGA LCD Projector	$649.00	0	$0.00

Total: $1,799.00

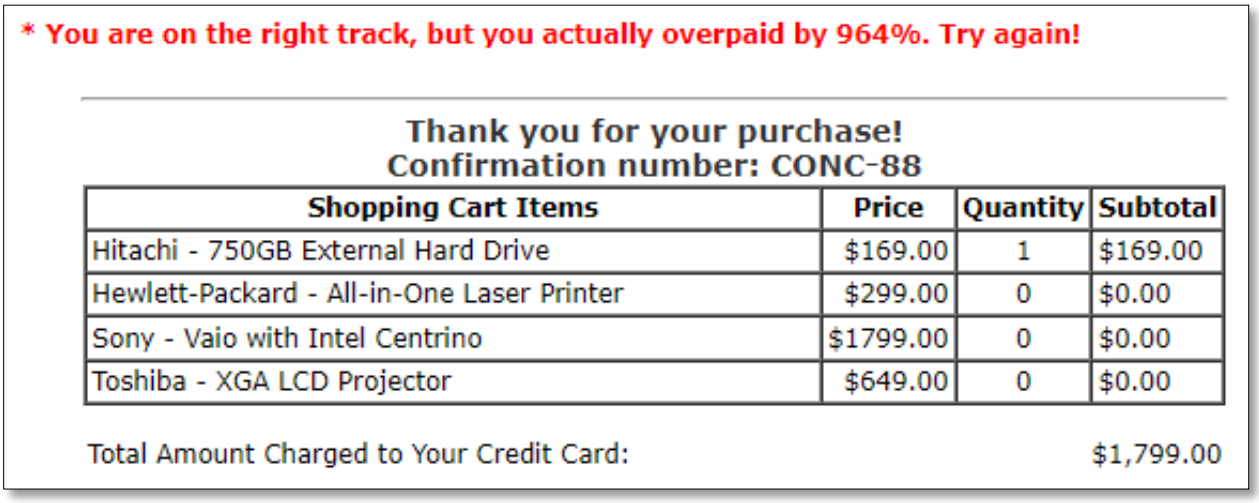
Update Cart

Purchase

Ôi không, không khả thi rồi. Thử cách khác vậy.

Lần này, chúng ta sẽ tìm cách mua món hàng đắt nhất với giá rẻ nhất, bằng việc thử luồng hoạt động như sau: người dùng đầu tiên sẽ chọn 1 món hàng đắt nhất và cập nhật lại giỏ hàng ➔ sau đó bấm thanh toán để số tiền hiển thị cần thanh toán là số tiền của món hàng đắt nhất ➔ người dùng thứ 2 sẽ chọn món hàng rẻ nhất, cũng update lại giỏ hàng ➔ và cuối cùng là người dùng 1 tiến hành thanh toán để kiểm tra hóa đơn

Bằng việc này ta có thể kiểm tra được việc, liệu cái biến total kia có phải biến dùng chung hay không, và nếu là dùng chung thì ta có thể sử dụng người thứ 2 để ghi đè giá trị total của người thứ nhất (cách hoạt động như bài 1).



* You are on the right track, but you actually overpaid by 964%. Try again!

Thank you for your purchase!
Confirmation number: CONC-88

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	$169.00	1	$169.00
Hewlett-Packard - All-in-One Laser Printer	$299.00	0	$0.00
Sony - Vaio with Intel Centrino	$1799.00	0	$0.00
Toshiba - XGA LCD Projector	$649.00	0	$0.00

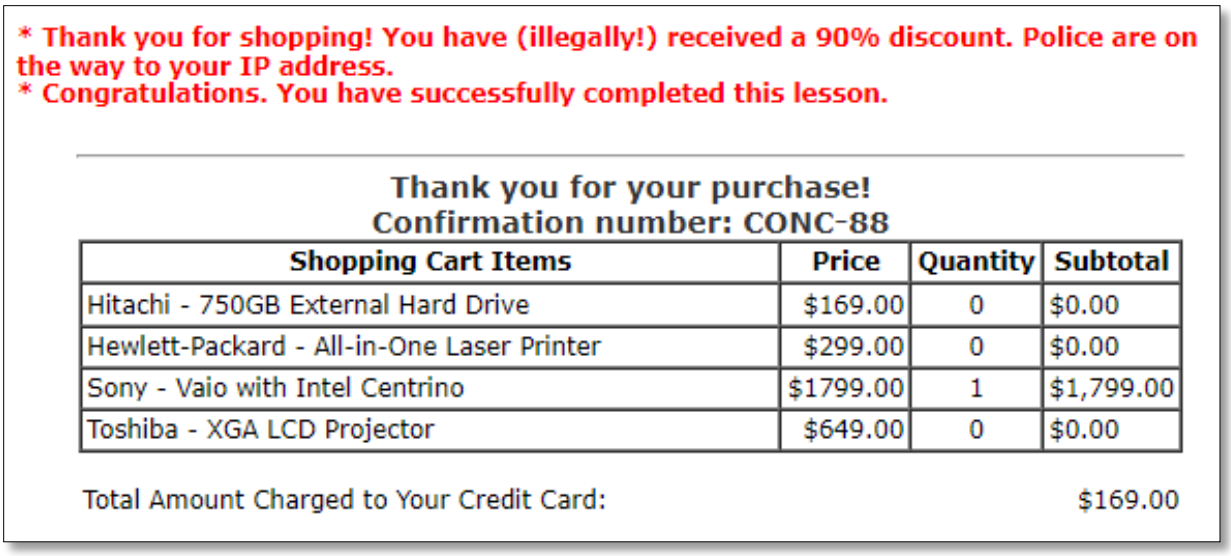
Total Amount Charged to Your Credit Card: $1,799.00

Không được rồi. Số tiền ta thanh toán vẫn là số tiền của món hàng đắt nhất. Nhưng nhìn kỹ lại thì, món hàng hiển thị khi thanh toán lại là món hàng có số tiền nhỏ nhất. Có vẻ ta đang đi đúng hướng.

Vậy là sao nhỉ, ta đang cố ghi đè số tiền của món hàng đắt nhất, thành số tiền của món hàng rẻ nhất, nhưng thứ bị ghi đè lại là nội dung của giỏ hàng.

Gotcha, mình nghĩ thứ mình cần ghi đè ở đây là nội dung hóa đơn, chứ không phải số tiền.

Tiến hành thử nghiệm lại, ta sẽ tiến hành chọn mua món hàng rẻ nhất trước, đứng chờ ở màn hình thanh toán ➔ và dùng người dùng thứ 2 chọn món hàng đắt nhất, xong rồi update lại giỏ hàng. Mục đích là để nó ghi đè thông tin giỏ hàng của người đầu tiên, số tiền của người đầu tiên cần trả vẫn là số tiền của món hàng rẻ nhất, nhưng nội dung xuất ra lại ghi nội dung của món hàng đắt nhất.



Woala, đúng rồi.