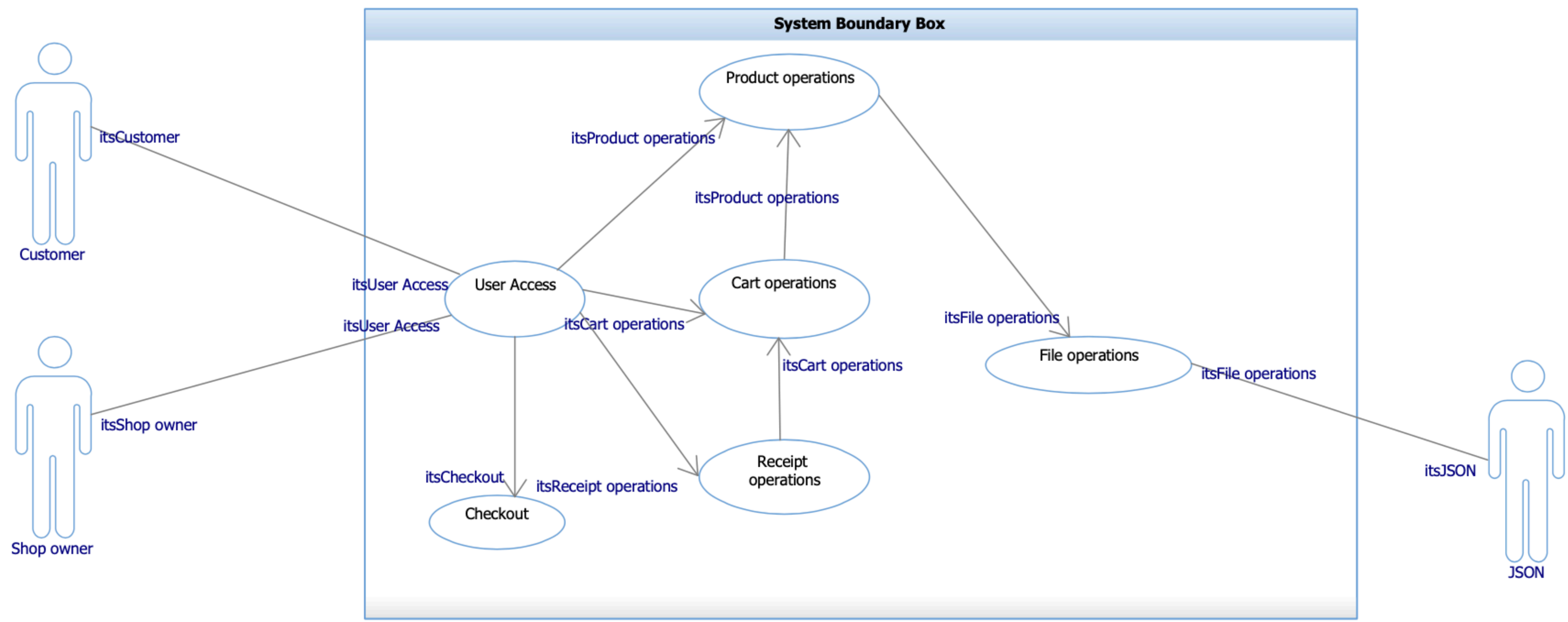


Walkthrough

Shopping application

Use case diagram



Features

This terminal application can be used by a customer through a command line menu and would be able to do the following actions.

- Show available products, prices and quantities available
 - Able to select and add to cart
 - Should be able to delete from cart
 - Should be able to view the cart
 - Should be able to view the receipt in screen
 - Should be able to do payment.
 - Updating JSON file for quantities and new products
 - Two modes should be there one for shop owner and another for customers
 - Errors like wrong input and product not in stock should be displayed to the user
-

The planning

Shopping

Software project

PLANNING

NEW

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / Shopping

Backlog

RV

Epic

Epic

Issues without epic

> Shopping

+ Create Epic

Backlog (15 issues)

SHOP-12 JSON file creation SHOPPING

SHOP-11 Shopping menu SHOPPING

SHOP-13 Display products upon user selection SHOPPING

SHOP-14 Add to cart SHOPPING

SHOP-15 Delete a product from cart SHOPPING

SHOP-16 View cart SHOPPING

SHOP-17 Payment SHOPPING

SHOP-18 Receipt in screen SHOPPING

SHOP-19 Bash scripting

SHOP-20 Select and add Ruby gems SHOPPING

SHOP-25 Install gems SHOPPING

SHOP-31 Add a new product in JSON SHOPPING

Shopping / SHOP-13

Display products upon user selection

To Do

Description

Add a description...

Child issues

Order by

0% Done

SHOP-24 Json file integrati...

TO DO

SHOP-39 Coding


TO DO

RV

Add a comment...

Pro tip: press M to comment

The planning



Shopping

Software project

PLANNING

NEW

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / Shopping

SHOP Sprint 2

Pseudocode Slide deck

⚡

☆

🕒 0 days remaining

Complete sprint

⋮

RV

+

Epic

GROUP BY

Subtask

Insights

TO DO 2 ISSUES

IN PROGRESS 1 ISSUE

BLOCKED 1 ISSUE

DONE 3 ISSUES

Issues without Subtask 7 issues

Sequence diagram

SHOPPING

SHOP-35

RV

Update README.MD- Sprint 2

SHOPPING

SHOP-28

RV

Write pseudocode

SHOPPING

SHOP-37

Software development plan

SHOPPING

SHOP-7

RV

Make and check in codebase

SHOPPING

SHOP-36

✓

Update slide deck - Sprint 2

SHOPPING

SHOP-29

✓

RV

Update slide deck - Sprint 1

SHOPPING

SHOP-9

✓

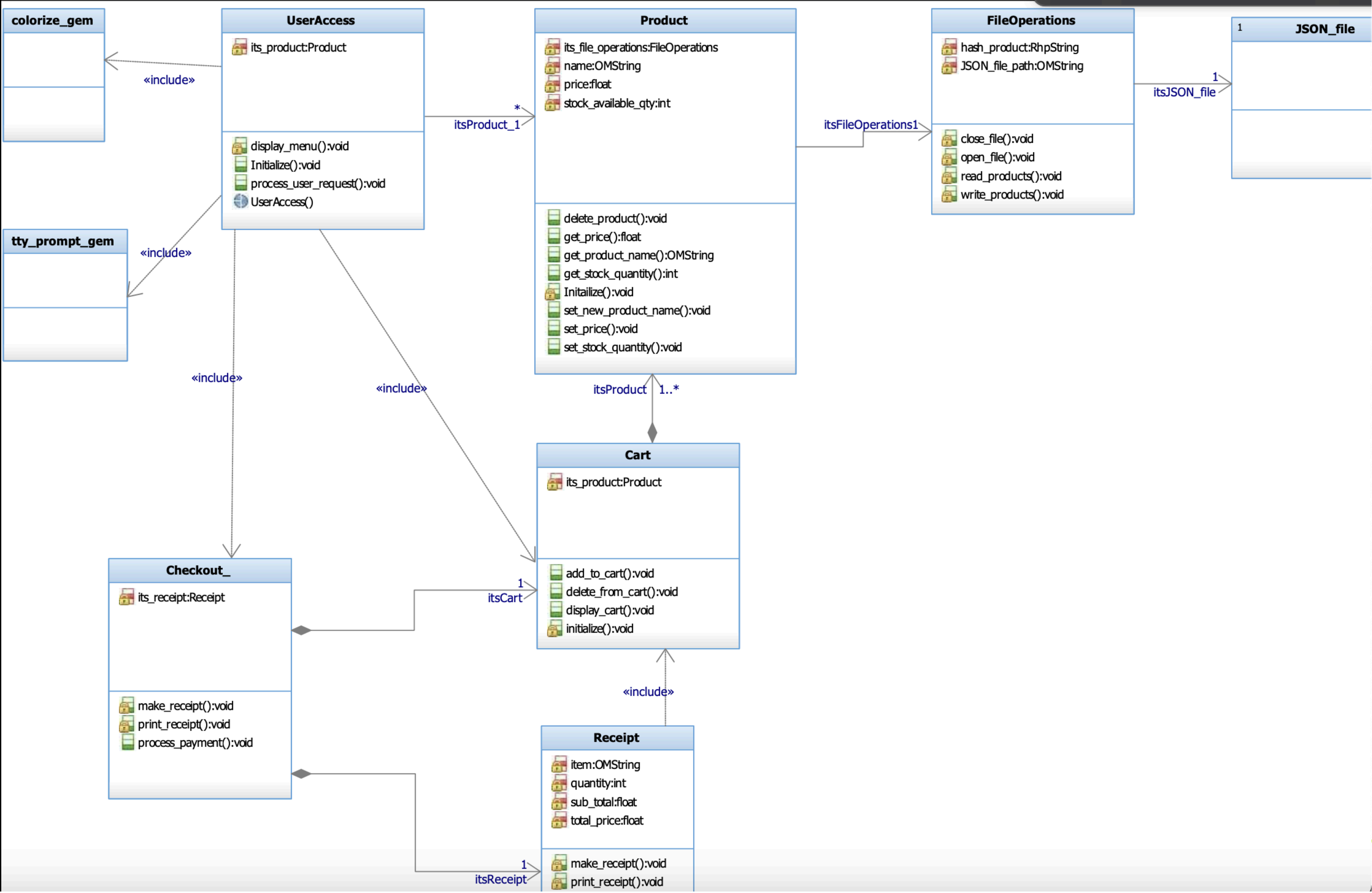
RV

Class analysis

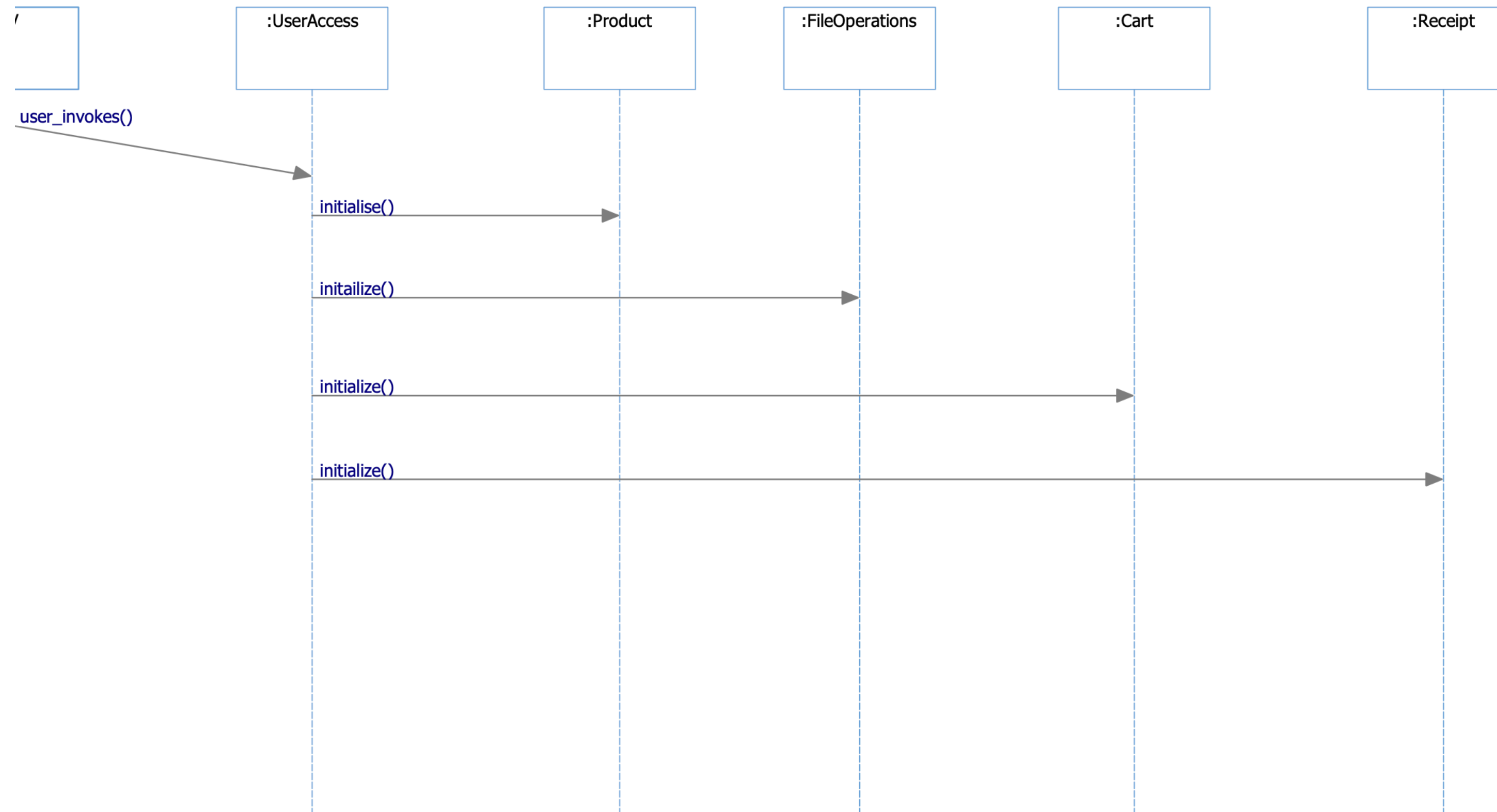
Identified several Classes

- UserAccess
 - Product
 - Cart
 - Checkout
 - Receipt
 - FileOperations
-

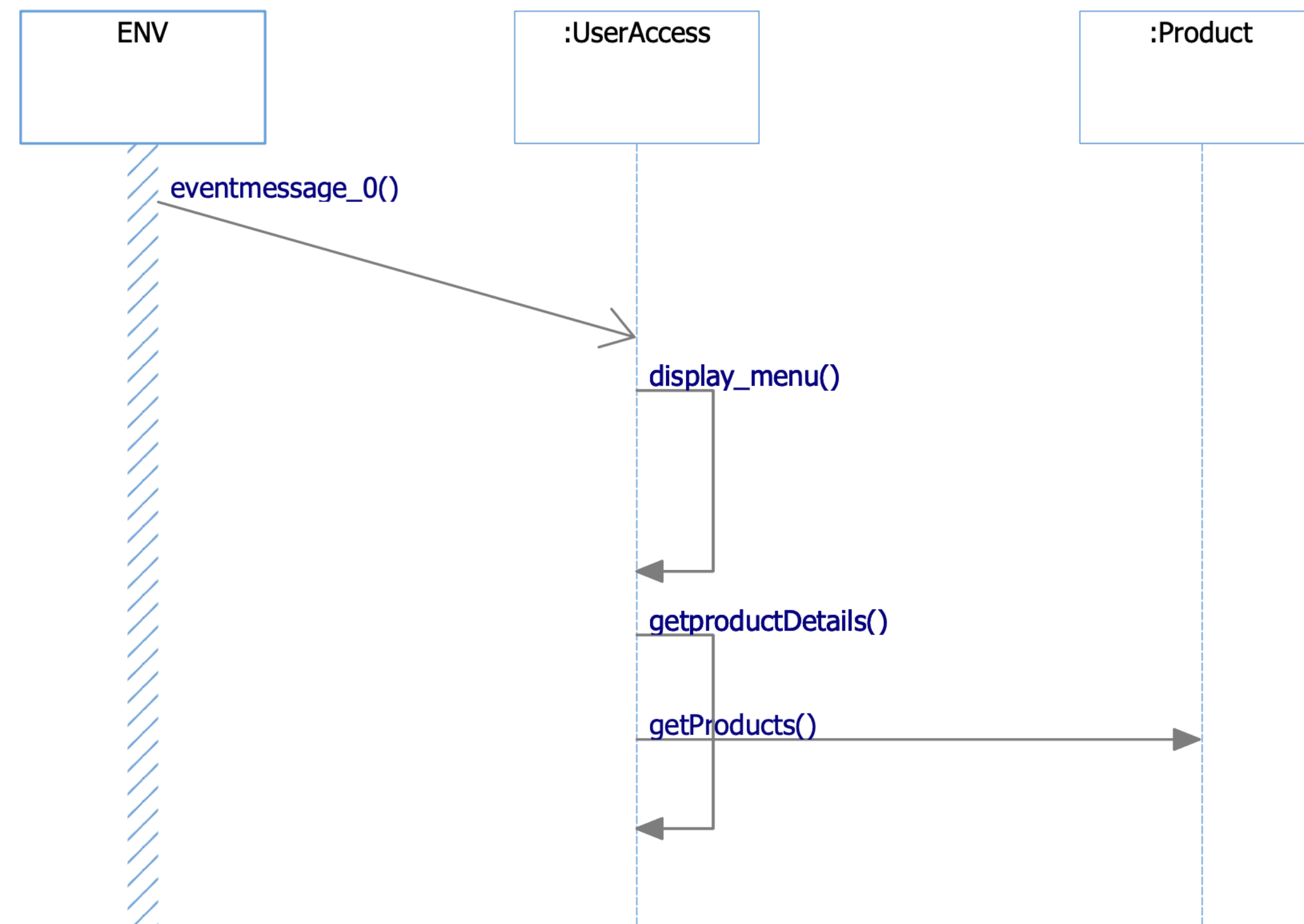
Class diagram



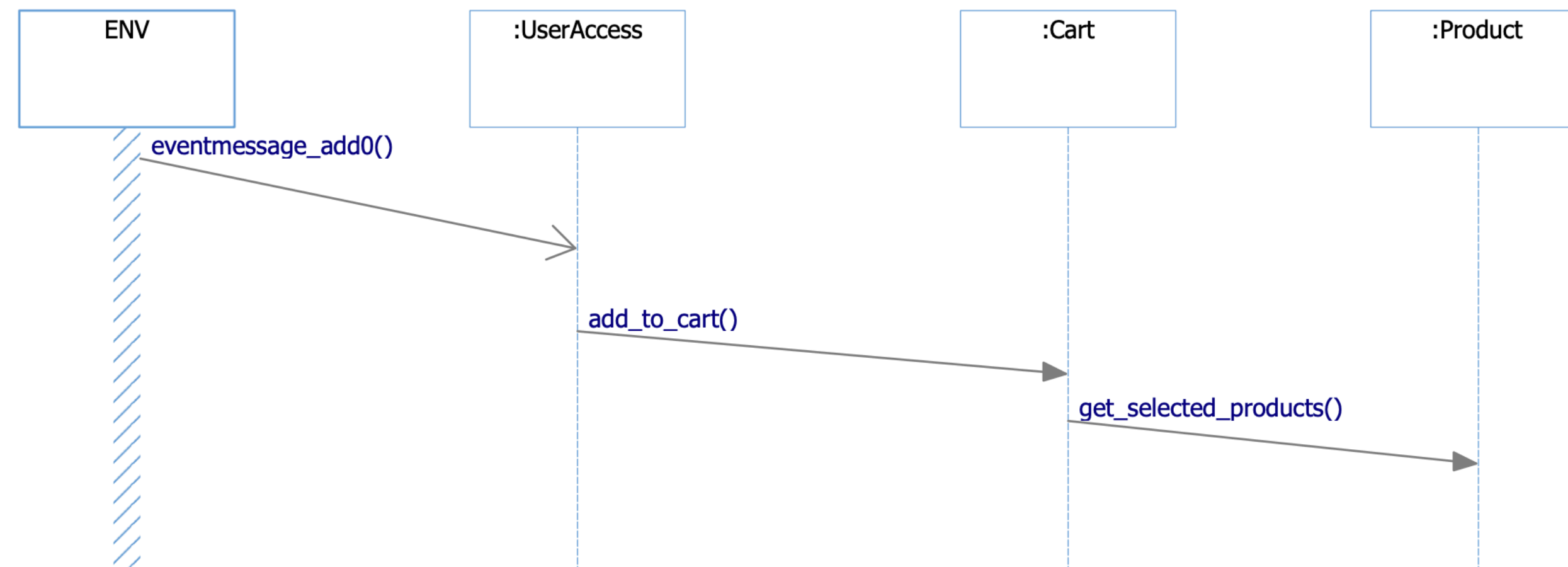
Sequence diagram - initialise



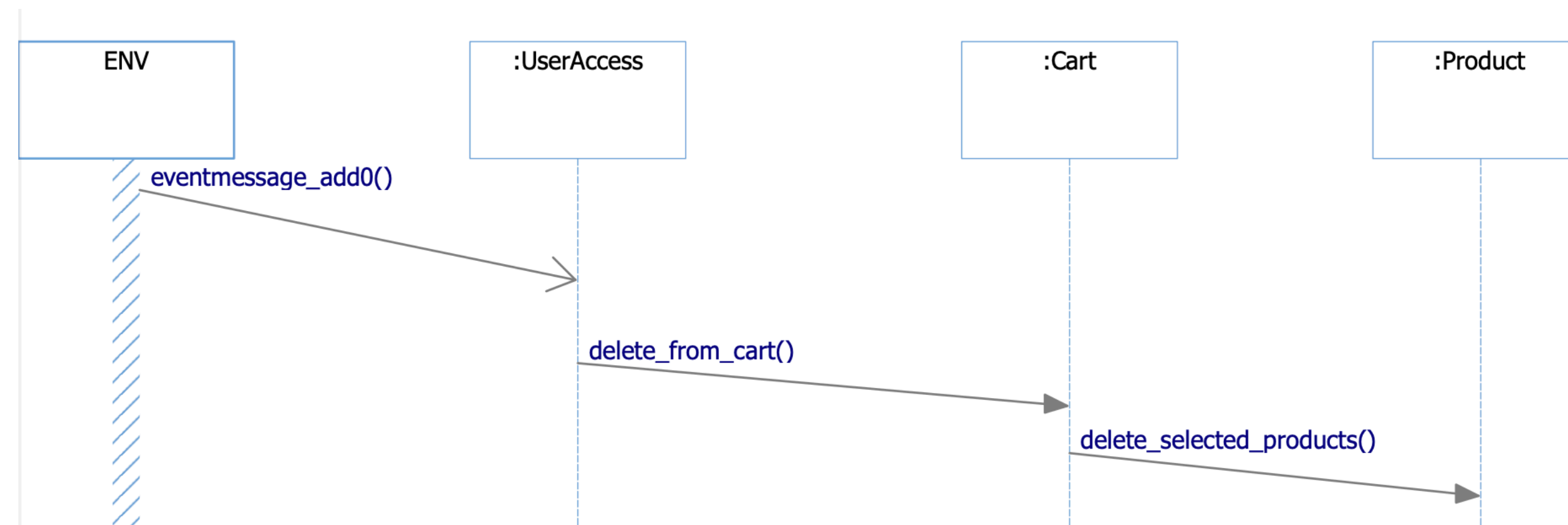
Sequence -look up products



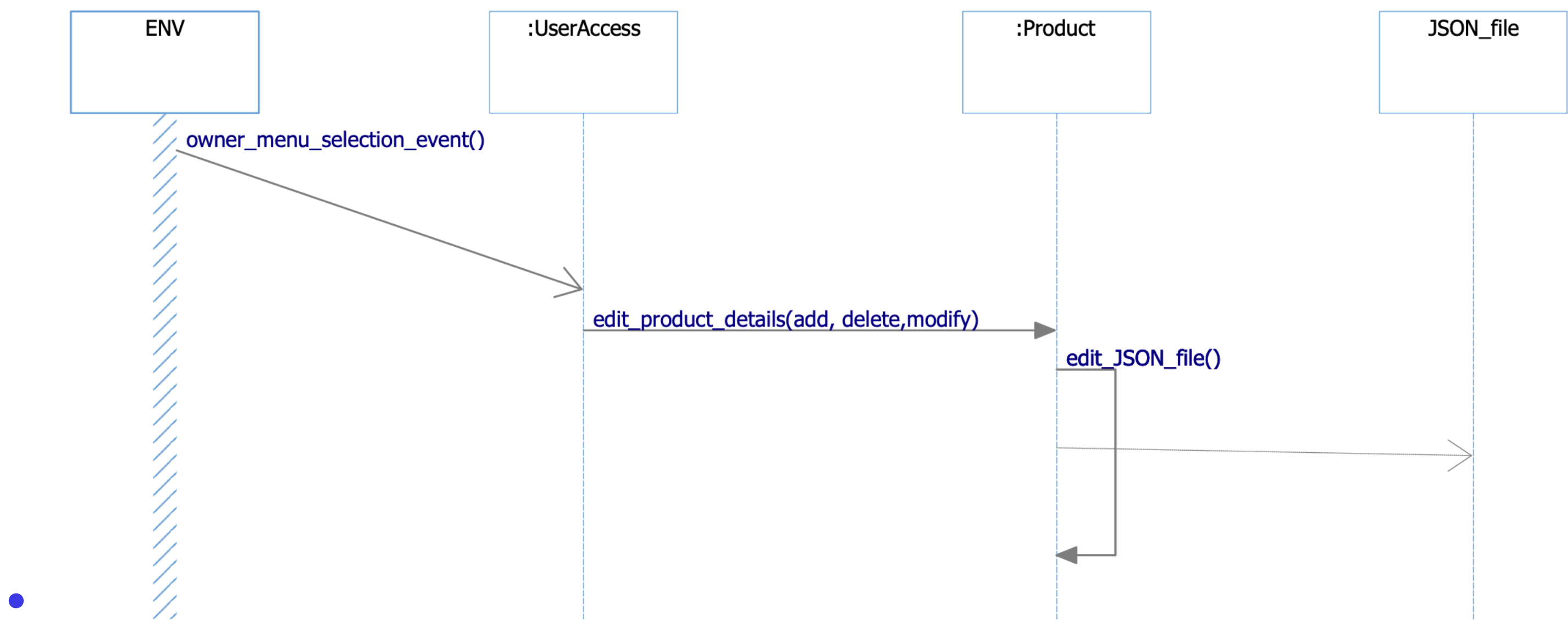
Add a product to cart



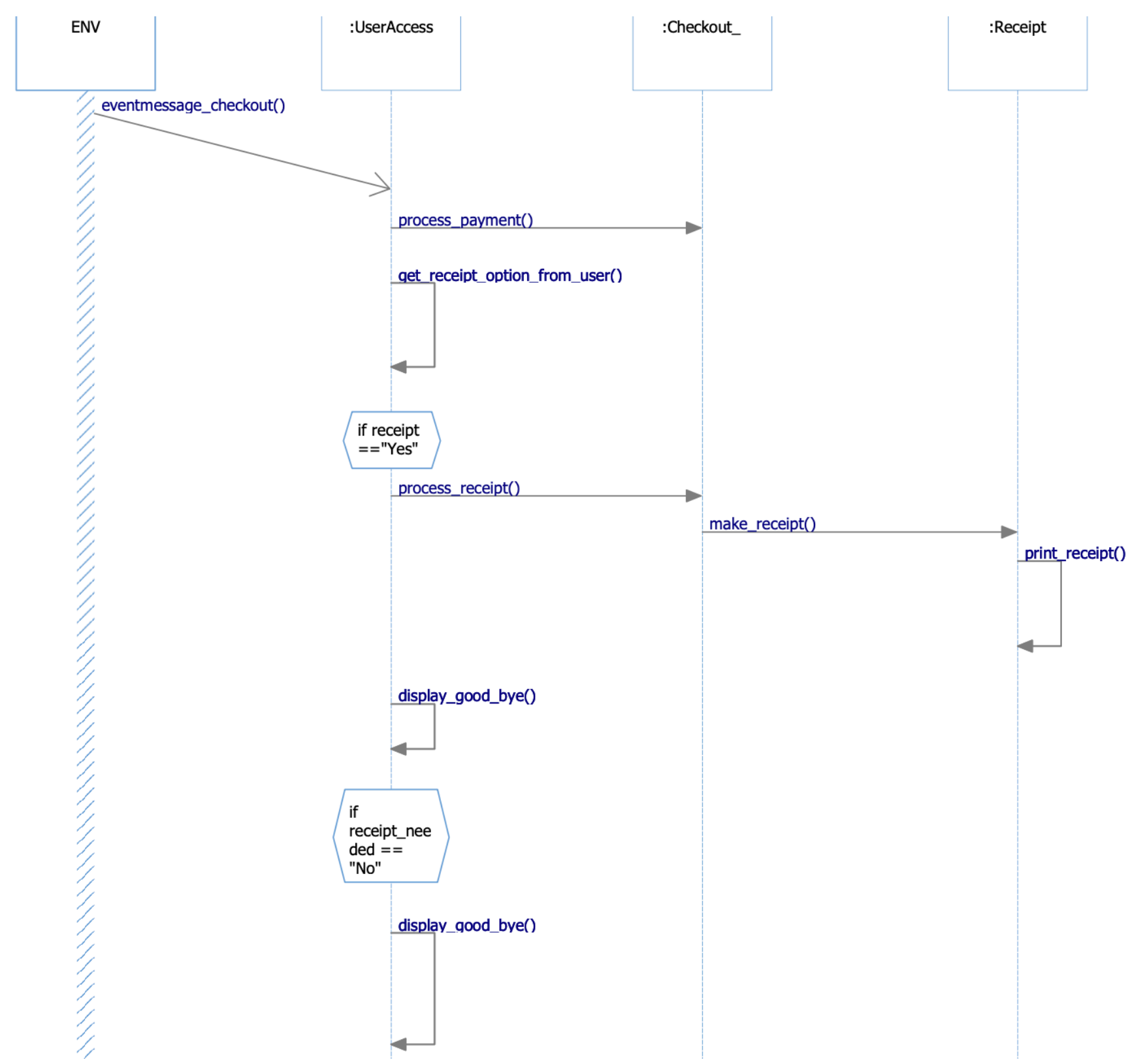
Delete item from cart



Shopping owner edit



Checkout



Logic -Pseudocode

```
Require gem colorise
Require gem tty-prompt
#Start of Class UserAccess
attr_reader is_customer
  #attributes
  public:
    @@is_customer = false

  #methods
  def checkUser
    #start of the program
    puts ("Are you a customer or Shop owner)
    If customer Then
      set customer is_customer = true
    elsif Shop_owner
      is_customer = false
    else
      throw error message to exception handler
      retry in exception handler
    End
  end
#End of Class UserAccess
```

Pseudo code - initial

```
#Main program  
start  
# Create new UserAccess object  
  create objUserAccess  
  
# create and initialize Product object  
  objProduct.initialize(objUserAccess.is_customer)  
end
```

Pseudo code - initial

```
#Start of Class Product
```

```
hashmapmain products[]  
hashmap products_selected[]  
  #method initialise()  
  def initialise(is_customer)  
    {  
      if (is_customer)  
      {  
        display menu for customer  
      }  
      else  
      {  
        display menu for shop owner  
      }  
      display_products()  
      display_product_selection_options()  
    }  
  # method initialize() end
```

Pseudocode

```
#method display_product_selection_options()
if (is_customer)
  1. "Select 1 for add a specific product to
cart"
  2. "Delete a product from the cart"
  if selected == 1
    puts ("Enter quatity")
    add_to_cart(product_id, quantity)
  elsif selected == 2
    puts ("Enter quatity to be deleted")
    delete_from_cart(product_id,
quantity)
  else
  end
end
end
```

Pseudocode

```
#if shop owner  
elsif (!is_customer)  
    1. "Add a new product"  
    2. "Delete a product"  
    3. "Change price of a product"  
    4. "Change stock quantity of a product"  
end
```

•

Pseudocode

```
#Start of class FileOperations  
#attributes  
its_JSON_file  
#methods  
open_JSON_file()  
read_and_map_JSON_to_hash()  
write_new_product_to_hash(product array)  
close_JSON_file()
```

```
#End of class FileOperations
```

Pseudocode

```
#Start of Class Cart  
#attributes  
itsCheckoutObj  
#methods  
add_to_cart()  
delete_from_cart()  
display_checkout_options()  
#End of Class Cart
```

Pseudocode

```
#Start of Class Checkout
```

```
#attributes
```

```
itsReceiptObj
```

```
#methods
```

```
process_payment()
```

```
ask_user_for_receipt()
```

```
def process_payment
```

```
    "Enter name"
```

```
    "Enter card number"
```

```
    check_card_type from_number()
```

```
    "Enter amount"
```

```
    "Payment succesful"
```

```
    ask_user_for_receipt()
```

```
end
```

```
def check_card_type_from_number
```

```
    if cardNUmber firt 4 numbers == 123
```

```
        card_type = "Mastercard"
```

```
    elsif cardNUmber == 456
```

```
        card_type == "Visa"
```

```
    else
```

```
        "Card not recognised"
```

```
    end
```

```
def ask_user_for_receipt
```

```
    "Do you want a receipt"
```

```
    if required
```

```
        itsReceiptObj.make_receipt
```

```
    else
```

```
        "Good bye"
```

```
    end
```

```
end
```

```
#End of Class Checkout
```

Challenges

- Did get trapped in 'planning' loop
- Did get trapped in 'Design fine tuning' loop



Improvements

- Need to focus on iterative development
- Test based development
- And to code without the fear of failure.

