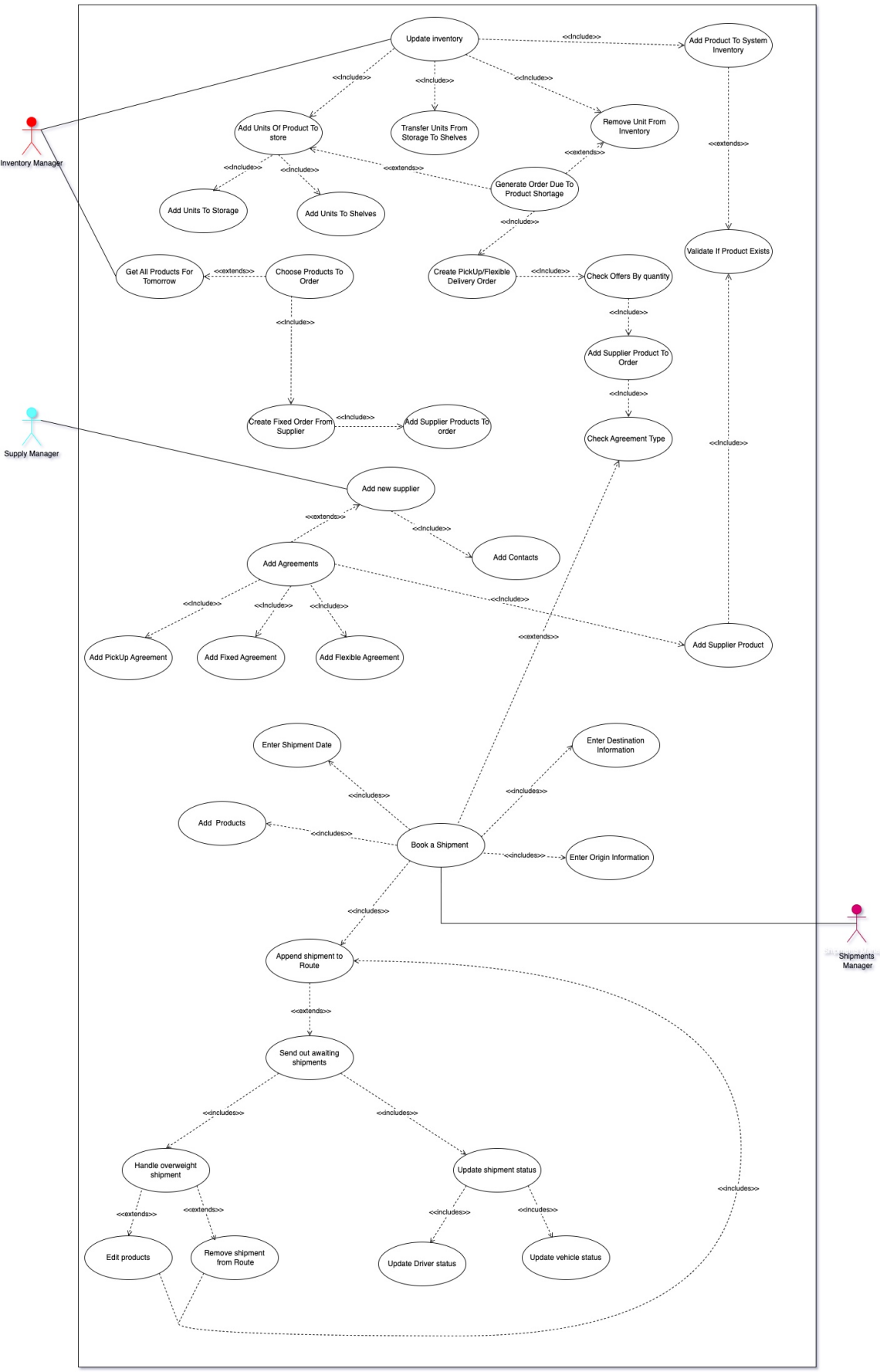
UseCase Diagram



Use Case E

1. Detailed format:

Use Case Name: Create Fixed Order From Supplier

Textual Description: This use case starts when the daily fixed-order process is triggered for next-day supply. The system retrieves all active fixed supplier agreements scheduled for tomorrow, displays their product lists to the Inventory Manager for selection, then automatically generates a fixed order containing the chosen items, selects the cheapest supplier offer for each product, invokes the shipment service if the agreement specifies pickup, and finally saves the completed order to history and returns the new order ID.

List of Actors:

- **Inventory Manager -** Initiates the fixed-order process by selecting which products to include.

Pre-conditions

- The Inventory Manager is authenticated and has access to the fixed-order interface
- One or more fixed supplier agreements for next-day delivery exist in the system.
- Each agreement includes at least one product that is available for ordering.

Post-conditions

- A new fixed order is recorded and linked to the chosen supplier agreement.
- The order contains exactly the products the Inventory Manager selected.
- The order is scheduled with a delivery date of tomorrow.
- The order appears in the system's order history for tracking and reporting.
- The Inventory Manager receives confirmation that the fixed order was successfully created.

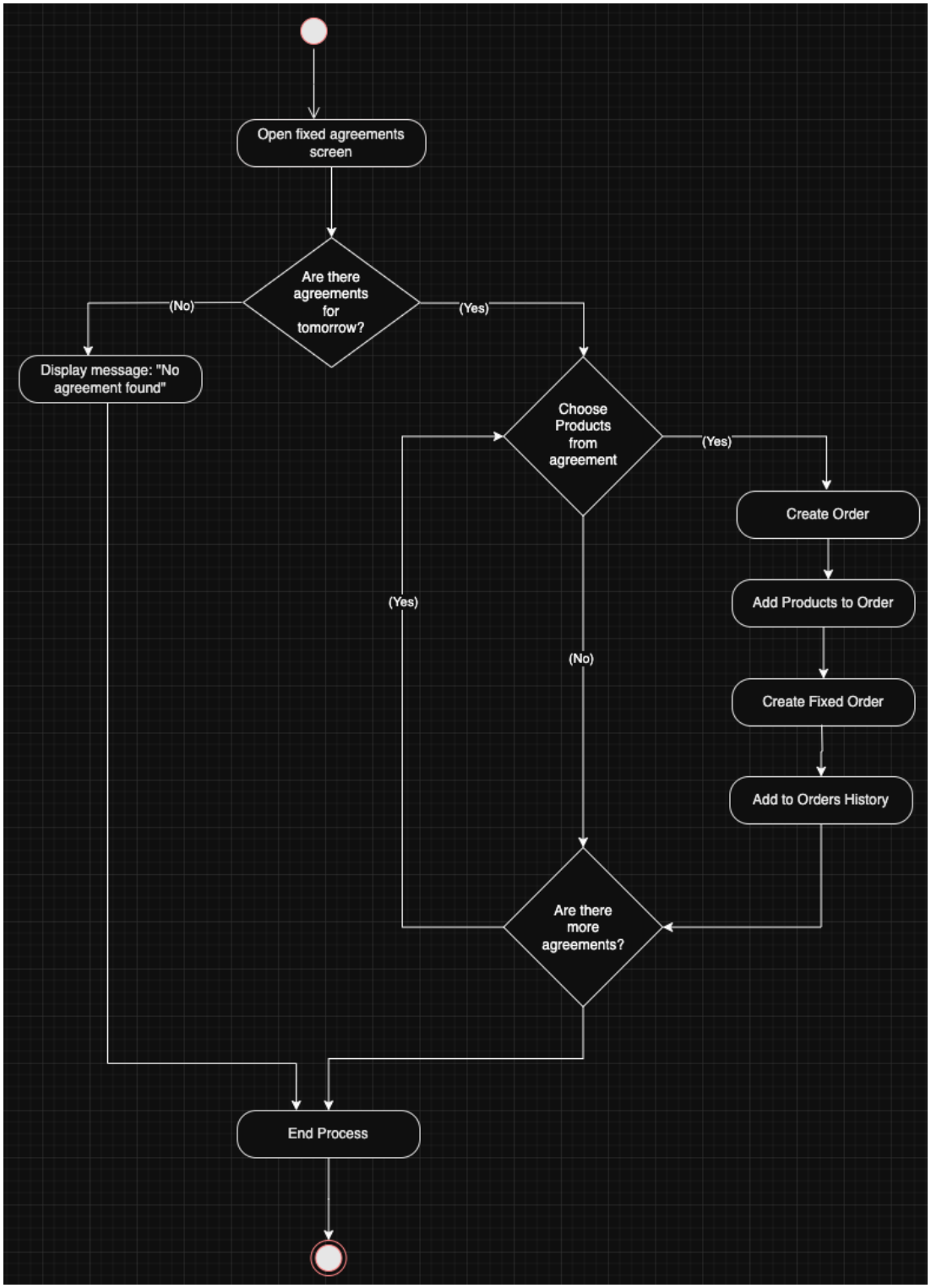
Main Success Scenario

- The Inventory Manager opens the fixed-order creation screen.
- The system retrieves and displays all active fixed supplier agreements scheduled for delivery tomorrow, along with their available products.
- The Inventory Manager selects the required products and quantities from one or more agreements.
- The system generates a new fixed order record based on those selections.
- For each selected product, the system identifies and applies the most cost-effective supplier offer.
- If the agreement specifies a pickup contract, the system initiates the shipment service to arrange collection.
- The system stamps the order with tomorrow's delivery date and saves it to the order history.
- The system confirms successful creation and returns the new order's reference number to the Inventory Manager.

Alternatives/Extensions

- **No agreement found:** the system finds no fixed agreements for tomorrow – the system displayes " No agreements found for tomorrow" and the use case ends with no order created.
- **Agreement has no available products:** the system detects thar a particular agreement's product list is empty or all products are unavailable – the system skips that agreement and proceeds with any remaining agreements.

2. Activity Diagram:


Open fixed agreements screen
Are there agreements for tomorrow?
(No)
(Yes)
Display message: "No agreement found"
Choose Products from agreement
(Yes)
Create Order
Add Products to Order
Create Fixed Order
Add to Orders History
(Yes)
(No)
Are there more agreements?
End Process

3. Contract:

Operation: createFixedOrder(agreementID : AgreementID, items : List<ProductID, quantity>)
Cross References: Create Fixed Order From Supplier

Pre Conditions:

- Inventory manager is authenticated and authorized to place orders.
- A fixed agreement with ID = agreementID exists and is active for delivery “tomorrow”.
- For every (ProductID, quantity) in items:
  - OrderService.isSupplierProductExistsByPID(ProductID) return True.
  - Quantity > 0.

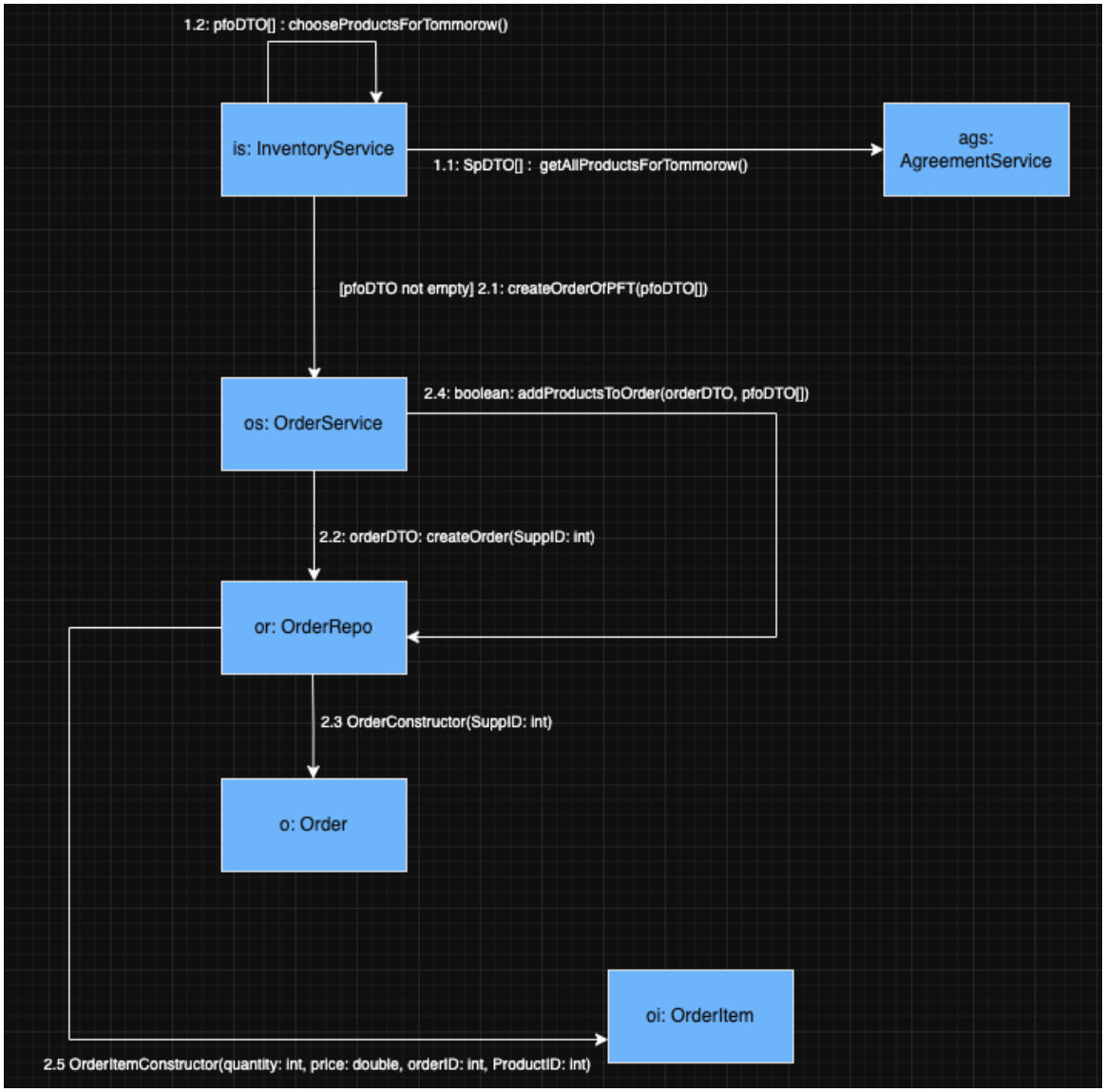
Post Conditions:

- Order creation:
  - OrderService.createOrder(agreementID) has been called, producing a new OrderDTO (with unique orderID) which is included in OrderRepo.
- Product selection (for each ProductID, quantity in items):
  - SupplierProduct,chooseSPProductByCheapestOffer(ProductID, quantity) was called, returning an spDTO with price and supplier info.
  - OrderService.addProductsToOrder(orderID, spDTOList) was called, resulting in one OrderItem per spDTO, each included with : orderID, ProductID, quantity and spDTO.price.

  A new fixed order instance is created and linked to the specified supplier agreement.

- Optional shipment:
  - If Order.DTO.isPickupOrder() == True, then ShipmentService.createShipment(orderDTO) was created.
- orderService.addOrderToOrderHistory(orderDTO) was called, so the complete order has been inserted to the orders history.
- The operation returned the new unique orderID to the caller.

4. Interaction Diagram:



Use Case F

1. Detailed format:

Use Case Name: Generate Order Due to Product Shortage

Textual Description: This use case starts when the stock of a product goes below its minimum level. The system creates a new order in the delivery system module for the desired product. Then, the system checks the available price offers for the product and selects the cheapest one. According to the contract type of the selected offer (pickup or delivery), the system handles the order accordingly. If it is a pickup contract, the system contacts the shipment service.

List of Actors:

- **Inventory Manager** – initiates the process indirectly by removing items from inventory.

Pre-conditions:

- The product exists in the system.
- The system has inventory records for the product (i.e., at least one unit is registered).
- One of the following conditions occurs:
    - A unit is removed from the inventory, causing the stock to fall below the product's defined minimum level.
    - A first-time addition of the product to the inventory occurs, and the added quantity is still below the minimum level.

Post-conditions:

- A new order has been created in the delivery system.
- The product has been added to the order.
- The shipping type (Pickup or Delivery) has been determined based on the selected offer's contract.
- If the selected contract type is Pickup, a shipment request has been created and sent to the shipment service.
-

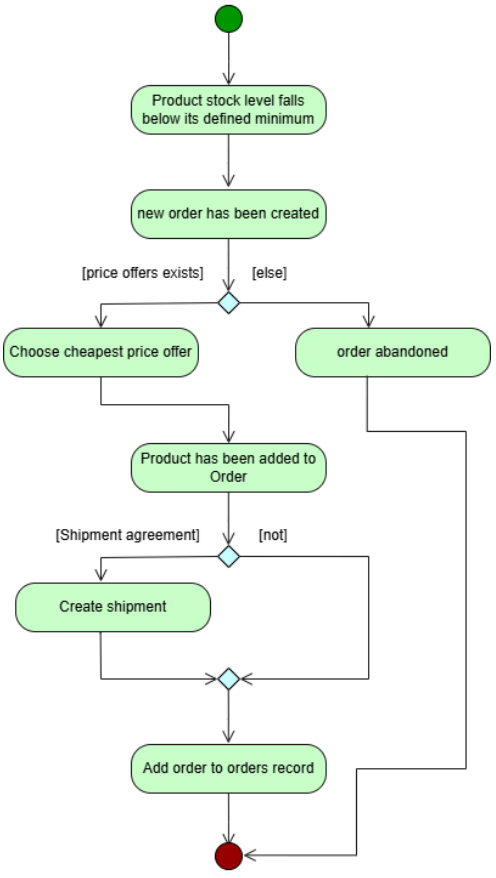
Main Success Scenario:

1. The stock level of a product falls below its defined minimum.
2. The system creates a new order in the delivery module.
3. The system checks all available price offers for the product.
4. The system selects the cheapest offer.
5. The product is added to the order.
6. The system determines the shipping type (Pickup or Delivery) based on the selected offer's contract.
7. If the shipping type is Pickup, the system sends a shipment request to the shipment service.

Alternatives / Extensions:

- **No price offers found:** If no valid price offers exist for the product (i.e., no supplier contracts are available), the order process is aborted and the product is not added to the order.

2. Activity Diagram:



3. Contract:

Operation: generateOrderDueToShortage(productID: int, amount: int)

Cross Reference: Generate Order Due to Product Shortage

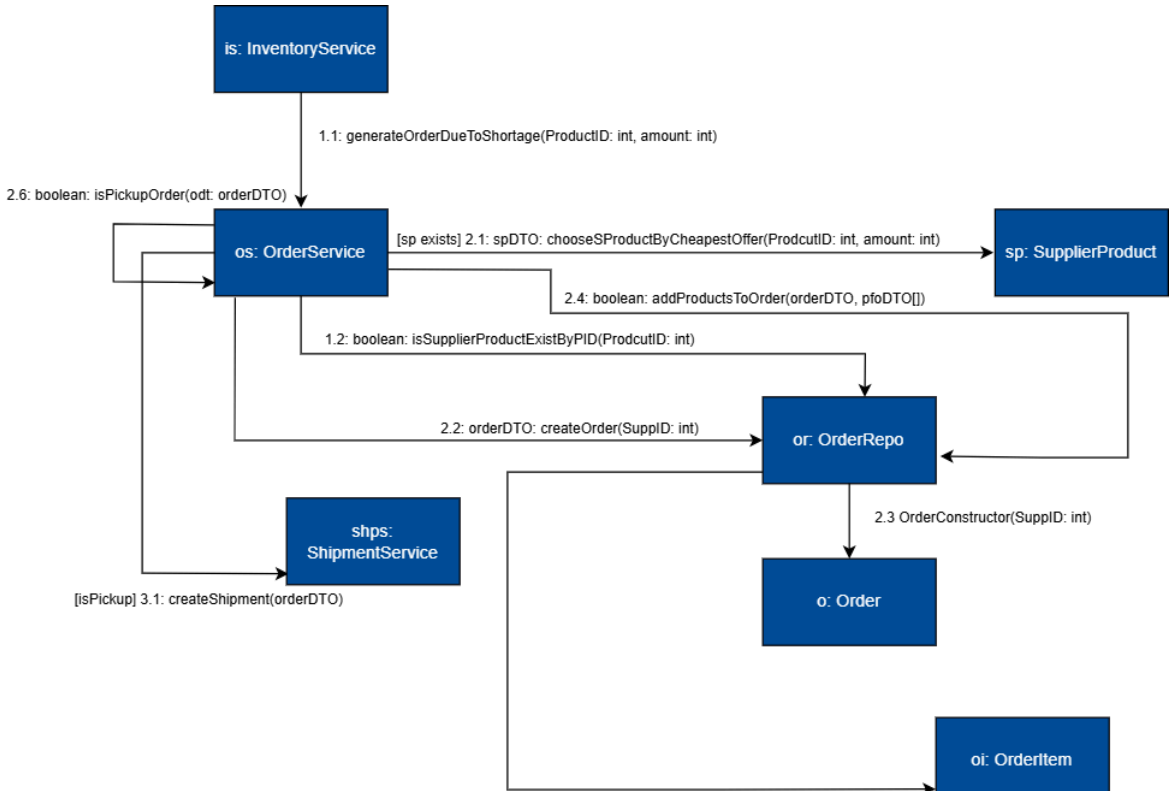
Preconditions:

- A Product instance with the given productID exists in the system.
- The inventory amount of the product is defined and less than its minimum amount.
- A SupplierProduct instance may exist that can supply the product with valid price offers.
- Each price offer is linked to an Agreement that defines the shipping type (Pickup or Delivery).

Postconditions:

- If no valid SupplierProduct or offer exists, the operation is aborted and no Order is created.
- Otherwise:
    - A new Order instance o was created through the OrderRepo.
    - The cheapest valid SupplierProduct offer was selected.
    - An OrderItem was created and added to the Order with the selected product, amount, and price.
    - The shipping type of the order was determined based on the Agreement linked to the selected SupplierProduct.
    - If the shipping type is Pickup, a ShipmentRequest was created and associated with the Order.

4. Interaction Diagram:



Use Case H

1) Detailed Format

Use Case Name: Book a Shipment

Textual Description: A Shipment Manager creates a new shipment request so that products can be collected from a supplier and dispatched to a store. A shipment is done by a driver and a vehicle of the company.
The booking is stored in the **Booking System** with status **PENDING** and will later be send by the Route-Planning logic.
* No need for the Shipment Manager to identify.

List of Actors:

- **Shipments Manager** – Initiates the process and responsible for shipments in the system (can also register as Admin with password).

Pre-conditions:

- A vehicle weighing appropriately for the weight of the products exist.
- A driver whose license fits for the type of the vehicle exist.
- Proper system upload.
- A Scanner instance exist (to get input from user

Post-conditions:

- A New Shipment instance has been created with:
  * Unique ID
  * Origin and Destination locations
  * Valid Date and Time
  * List of the shipment's products
- A driver whose license fits to the vehicle's type has been assigned.
- The shipment has been sent out to road as part of a Route.
- Shipment status has been updated to **Shipped.** Also the statuses of the driver and vehicle.

Main success scenario:

1. User uploads the system by empty/preset Initialization.
2. User selects option 1: 'Book a new shipment' in Main Menu.
3. User enters shipment information:
    3.1 User enters information for Origin and Destination as follow:
        3.1.1 Contact name and phone number
        3.1.2 Street and House number
        3.1.3 District (South, North, Central)
    3.2 User enters a valid future date for the shipment to be executed.
4. User handle products.
    4.1 User adds product information as follow:
        4.1.1 product Name
        4.1.2 Weight Per Unit (kg)
        4.1.3 Amount of units
    4.2 User receive a message: " Product '_name_' added successfully!".
User repeats step 4 until indicates done.

5. User is asking if he want the shipment to be delivered (y) or
    self pick-up (n)? User chooses 'y'.
7. User receive a message that his shipment will be handled shortly.
8. System handles shipment.
    8.1 Execution process:
        8.1.1 Shipment status updates to 'Pending'.
        8.1.2 Push into pending shipments queue.
        8.1.3 System executes shipment alone or with another shipments in the
            same route, according to location and time automations.
    8.2 Shipment status updates to 'Shipped'.
.9. The system presents the Main Menu again for another operation

Alternatives/Extension:

Invalid ORIGIN / DESTINATION field entered .1

Issue: User types empty street, phone/house number that are not digits, or district not in {South, North, Central}.

1.1 System displays error message: “Invalid input was given!"

1.2 System asks: "Would you like to try again? (y/n)"

1.2.1 User answers 'y'

System re-displays Step 1 (asks for Origin information)

1.2.2 User answers 'n'

System prints "Thank you for using our service! Returning to … ."main menu

Use-case terminates and system display Main Menu.

2. Invalid Shipment DATE

Issue: day ∉ [1...31] OR month ∉ [1...12] OR year has been passed.

2.1 System displays error message (e,g.: "Invalid value for MonthOfYear (valid values 1 - 12). Invalid input was given!")

2.2 System asks: "Would you like to try again? (y/n)"

System returns step 1.2

3. Invalid product's WEIGHT-PER-UNIT or AMOUNT

Issue: The values entered are not numeric.

3.1 System catches NumberFormatException

3.2 System displays error message: (e.g: "Weight must be an integer (kg).").

3.3 System asks: "Would you like to try again? (y/n)"

System returns step 1.2

4. User cancels product loop

Alternative: When prompted "Add another product? (y/n)" - User answers 'n'

4.1 System skips remaining product loop and proceeds to confirmation - step 5.

5. User decides to choose Self-Delivered option

Alternative: System asks “Do you want shipment number <ID> delivered? (y/n) - user enters 'n'.

5.1 System updates shipment status "Self Delivered".

5.2 Shipment is added to bookingHistory.

5.3 System displays: " Your self-booking has been confirmed"
"Thank you for booking a shipment!".
Use-case terminates and system displays Main Menu.

6. System falls according to Insert DB fail
Issue: When trying to add a new shipment to DB, the system falls.
It can happen according to one of the following scenarios:
- Network problem mid-transaction.
- Unique-key violation.
- DB is full (very rare)

6.1 DB behavior: aborts the statement and marks it as Marked ROLLBACK ONLY which means that the transaction failed - needs to rollback().
6.2 Java behavior: throws SQLException with vendor code.
6.3 DB is not updated with new shipment.
6.4 Use-case terminates and user pops out of the system.

7. User closes the console window during runtime
7.1 OS terminates the process.
7.2 None of the data committed.
7.3 When the program restarts, the user is presented with the Main Menu.

2) Contract: BookingMenu.run

Operation: BookingMenu.run(Scanner scanner) – main function of the process

References: Use Case Book a Shipment

Preconditions:

- A vehicle weighing appropriately for the weight of the products exists.
- A driver whose license fits for the type of the vehicle exists.
- Proper system upload.

Postconditions:

- A Shipment instance newShipment was created (instance creation).
- originLocation, destinationLocation, shipmentDate, departureTime has been updated (attribute modification).
- productsInShipment List has been updated with user's choices (attribute modification).
- newShipment has been sent to bookMadeShipment() for the user to choose shipment type (Self/Delivery).
- if Delivery chosen – newShipment was associated with DrivingRoute (association formed).
- Shipment status updated to (Shipped/Self-Delivered).
- newShipment was associated with bookingController (association formed); (to add it to DB – historical log of completed shipments).

3) Interaction Diagram