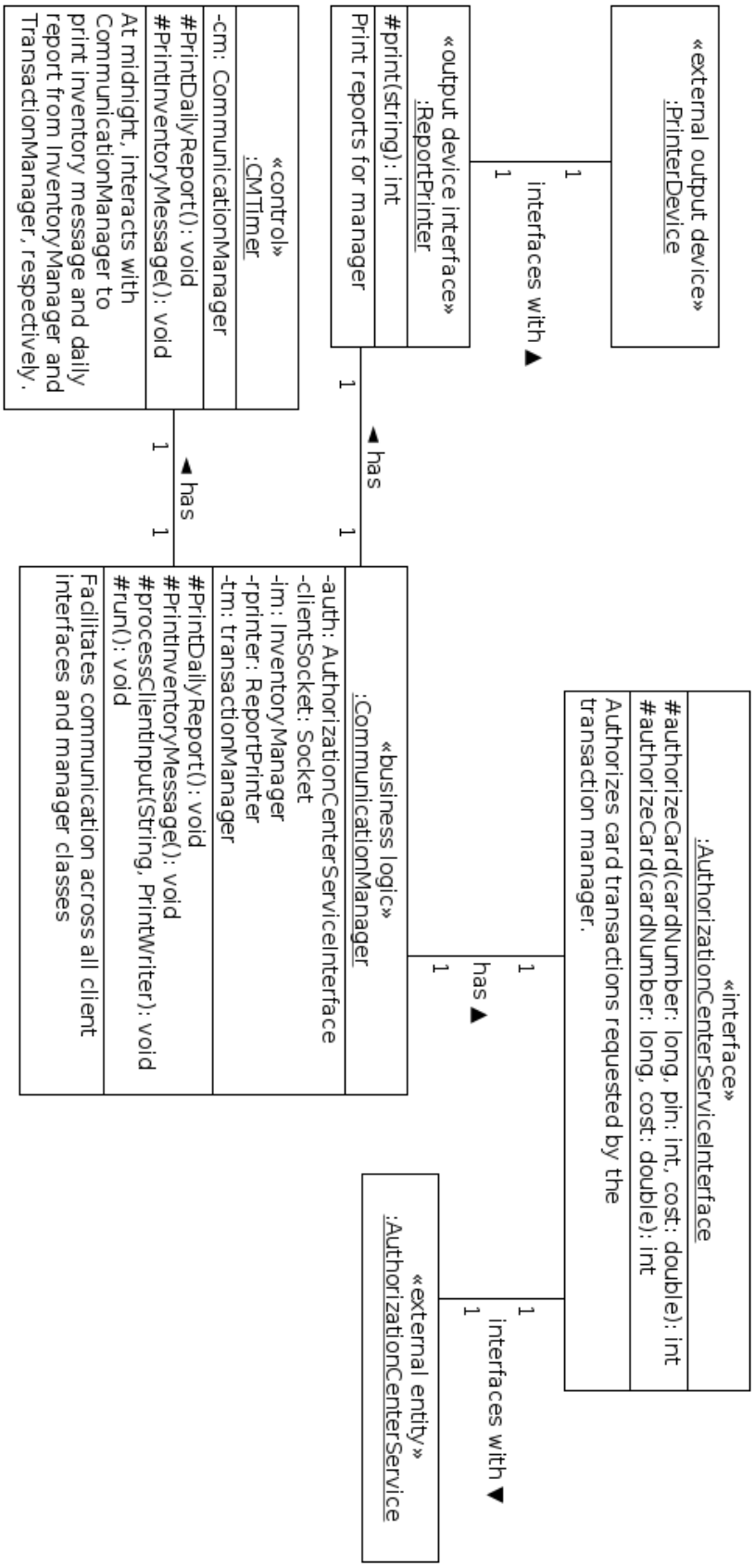
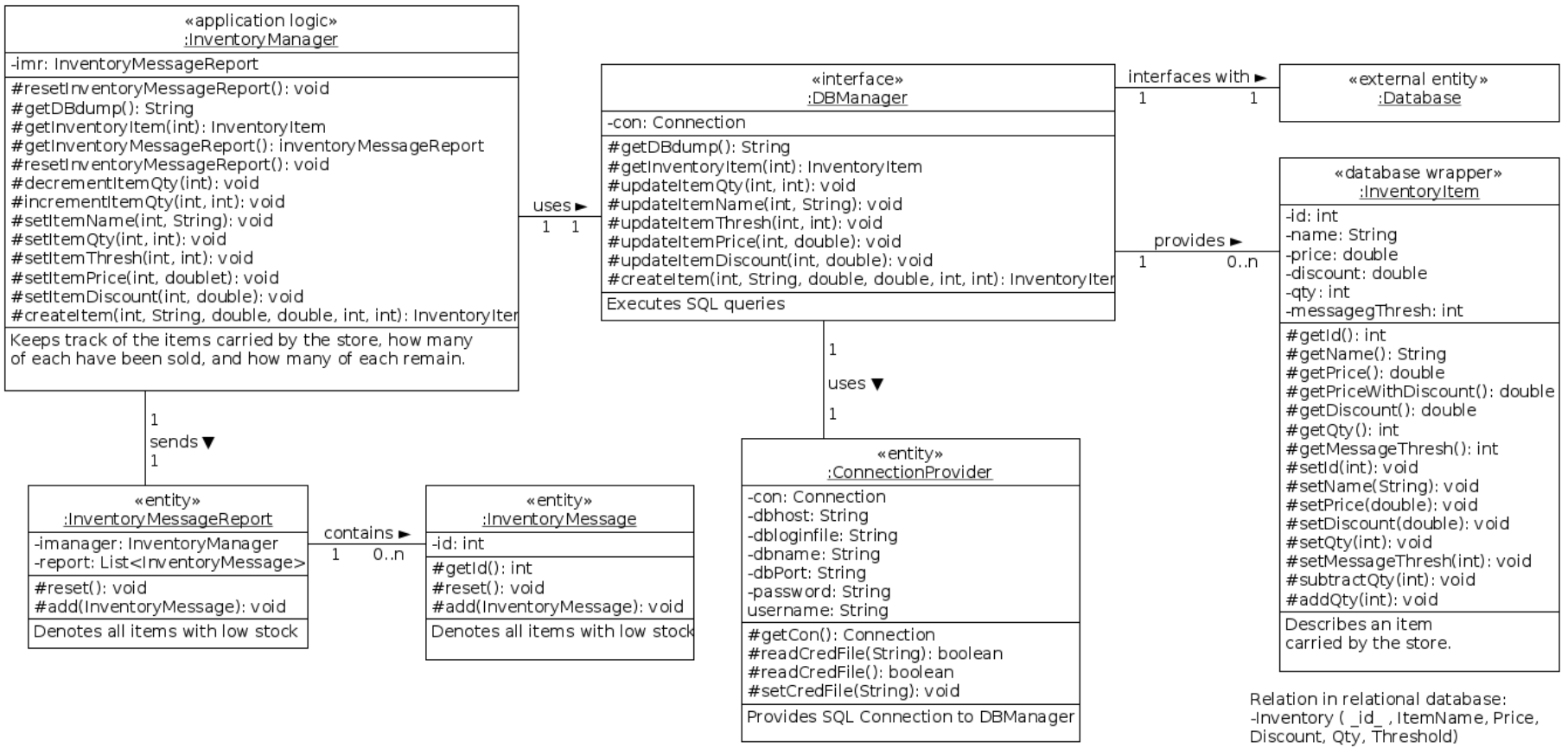
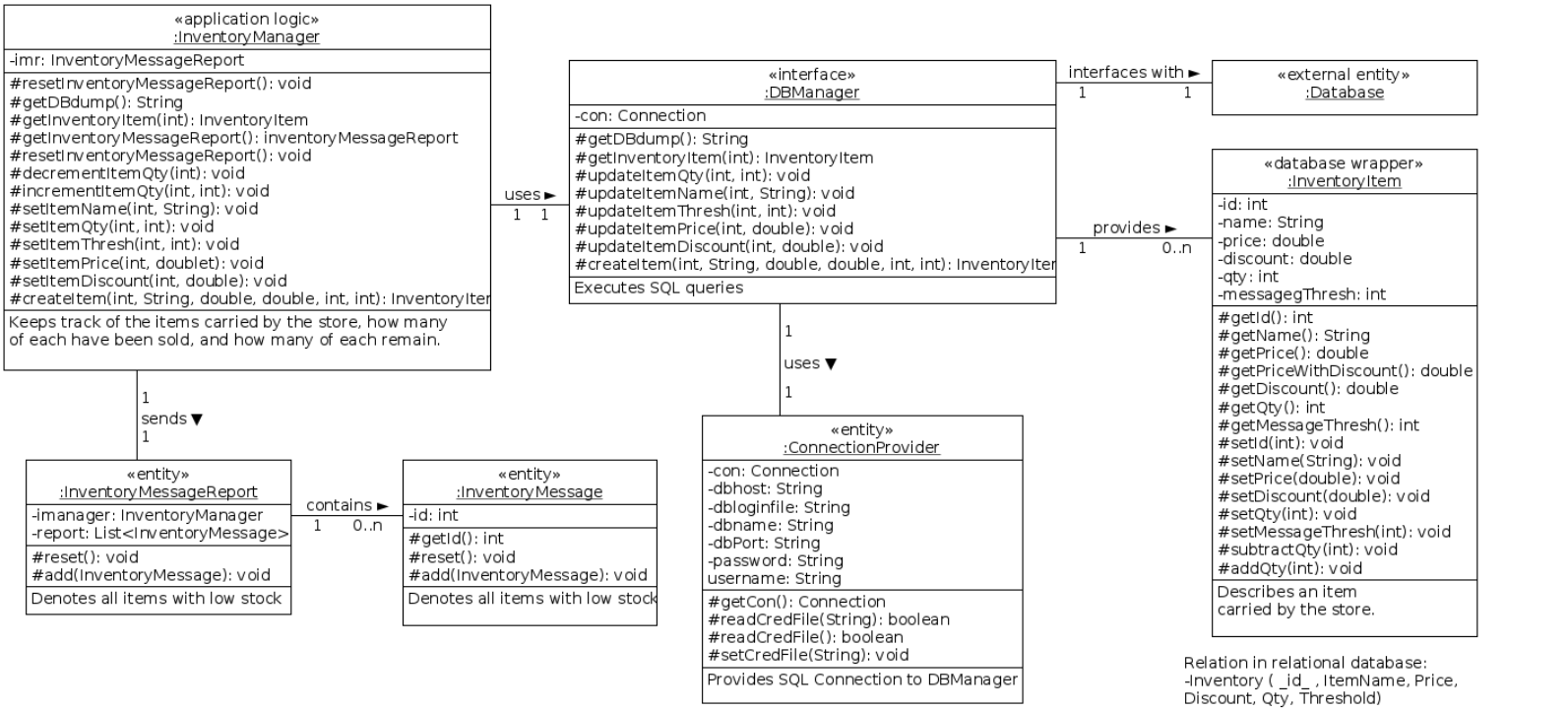
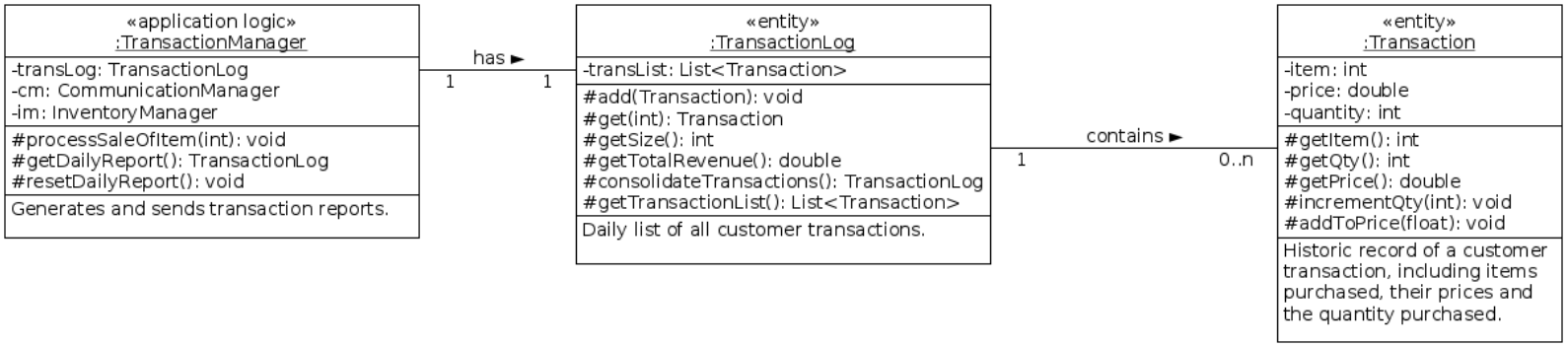
December Report

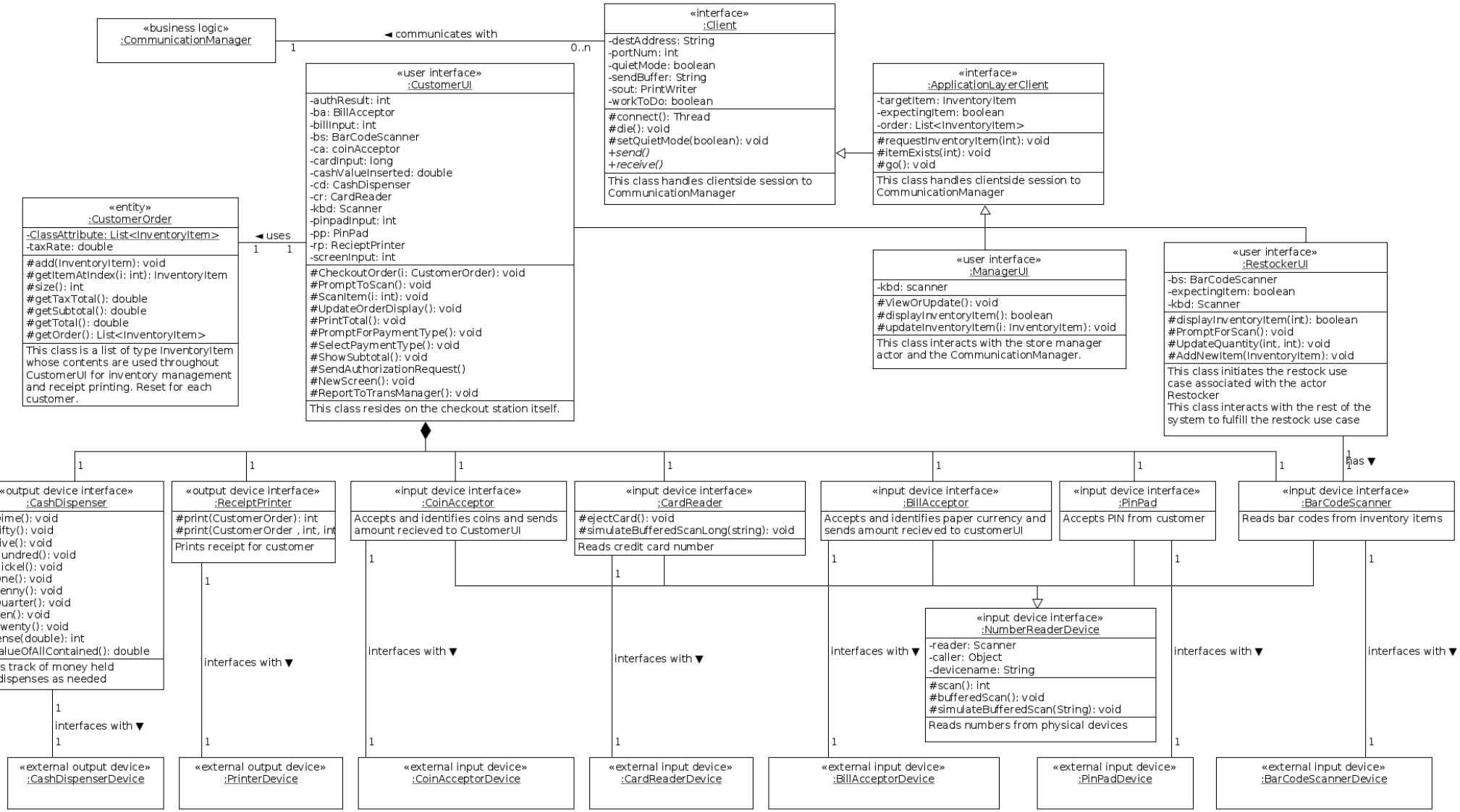
Static Models



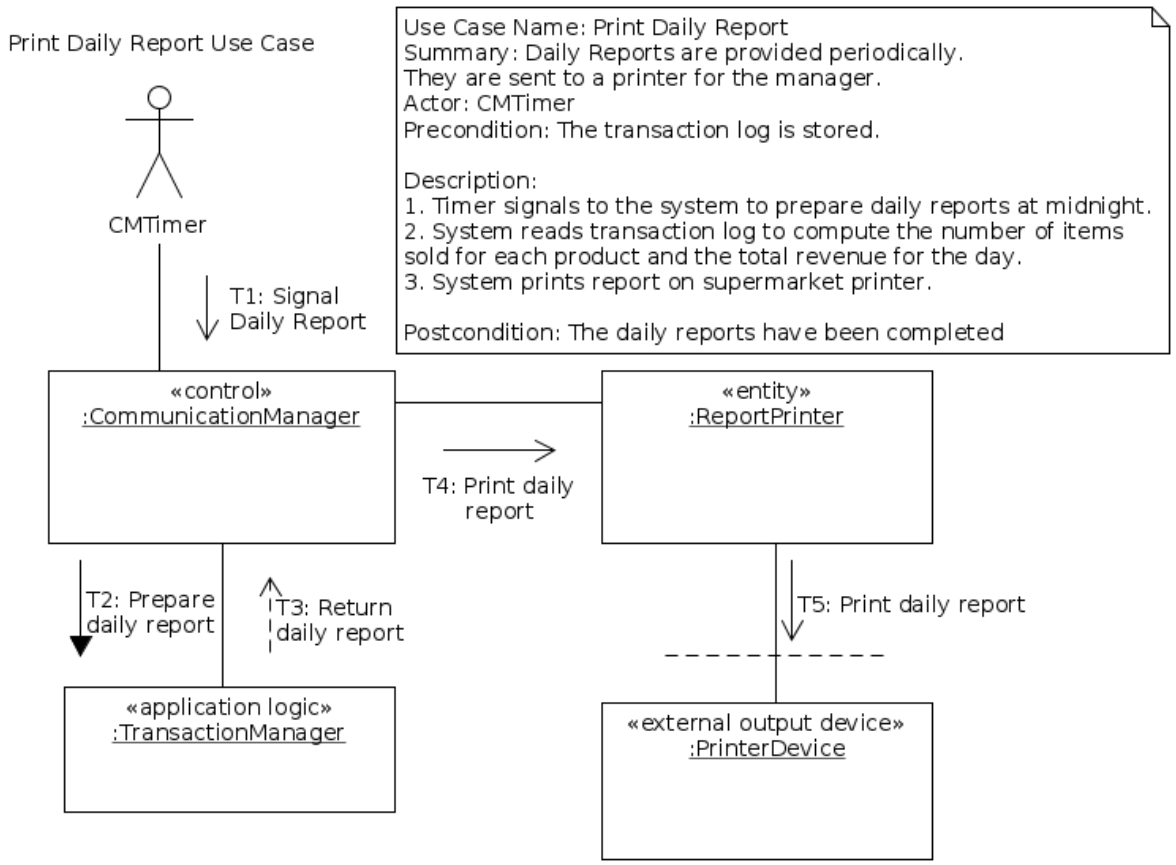

«application logic»
:InventoryManager
-imr: InventoryMessageReport
#resetInventoryMessageReport(): void
#getDBdump(): String
#getInventoryItem(int): InventoryItem
#getInventoryMessageReport(): inventoryMessageReport
#resetInventoryMessageReport(): void
#decrementItemQty(int): void
#incrementItemQty(int, int): void
#setItemName(int, String): void
#setItemQty(int, int): void
#setItemThresh(int, int): void
#setItemPrice(int, doublet): void
#setItemDiscount(int, double): void
#createItem(int, String, double, double, int, int): InventoryIter
Keeps track of the items carried by the store, how many of each have been sold, and how many of each remain.
uses ►
1 1
«interface»
:DBManager
-con: Connection
#getDBdump(): String
#getInventoryItem(int): InventoryItem
#updateItemQty(int, int): void
#updateItemName(int, String): void
#updateItemThresh(int, int): void
#updateItemPrice(int, double): void
#updateItemDiscount(int, double): void
#createItem(int, String, double, double, int, int): InventoryIter
Executes SQL queries
interfaces with ►
1 1
«external entity»
:Database
provides ►
1 0..n
«database wrapper»
:InventoryItem
-id: int
-name: String
-price: double
-discount: double
-qty: int
-messagegThresh: int
#getId(): int
#getName(): String
#getPrice(): double
#getPriceWithDiscount(): double
#getDiscount(): double
#getQty(): int
#getMessageThresh(): int
#setId(int): void
#setName(String): void
#setPrice(double): void
#setDiscount(double): void
#setQty(int): void
#setMessageThresh(int): void
#subtractQty(int): void
#addQty(int): void
Describes an item carried by the store.
1
sends ▼
1
«entity»
:InventoryMessageReport
-imanager: InventoryManager
-report: List<InventoryMessage>
#reset(): void
#add(InventoryMessage): void
Denotes all items with low stock
contains ►
1 0..n
«entity»
:InventoryMessage
-id: int
#getId(): int
#reset(): void
#add(InventoryMessage): void
Denotes all items with low stock
1
uses ▼
1
«entity»
:ConnectionProvider
-con: Connection
-dbhost: String
-dbloginfile: String
-dbname: String
-dbPort: String
-password: String
username: String
#getCon(): Connection
#readCredFile(String): boolean
#readCredFile(): boolean
#setCredFile(String): void
Provides SQL Connection to DBManager
Relation in relational database:
-Inventory (_id_ , ItemName, Price, Discount, Qty, Threshold)

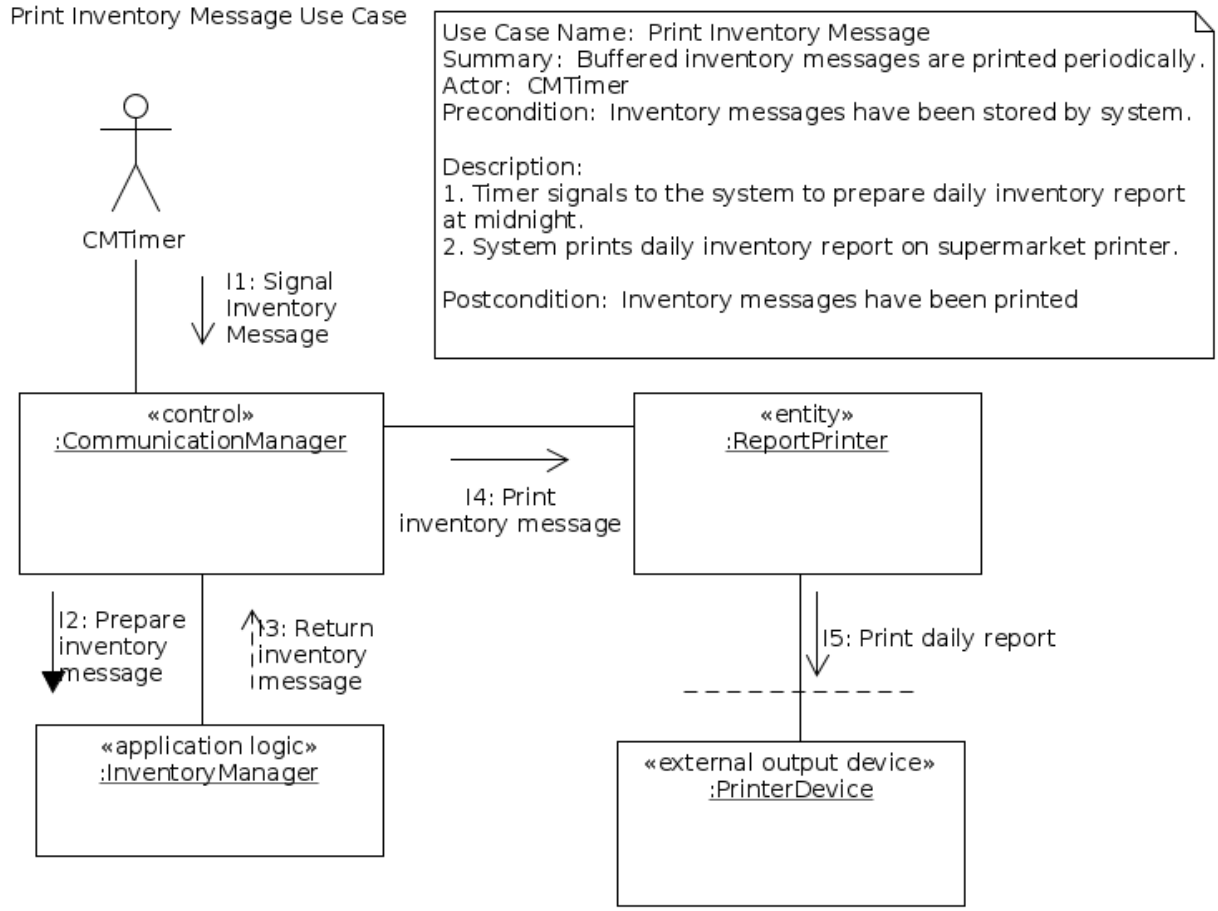

«application logic»
:InventoryManager
-imr: InventoryMessageReport
#resetInventoryMessageReport(): void
#getDBdump(): String
#getInventoryItem(int): InventoryItem
#getInventoryMessageReport(): inventoryMessageReport
#resetInventoryMessageReport(): void
#decrementItemQty(int): void
#incrementItemQty(int, int): void
#setItemName(int, String): void
#setItemQty(int, int): void
#setItemThresh(int, int): void
#setItemPrice(int, doublet): void
#setItemDiscount(int, double): void
#createItem(int, String, double, double, int, int): InventoryIter
Keeps track of the items carried by the store, how many of each have been sold, and how many of each remain.
uses ►
1 1
«interface»
:DBManager
-con: Connection
#getDBdump(): String
#getInventoryItem(int): InventoryItem
#updateItemQty(int, int): void
#updateItemName(int, String): void
#updateItemThresh(int, int): void
#updateItemPrice(int, double): void
#updateItemDiscount(int, double): void
#createItem(int, String, double, double, int, int): InventoryIter
Executes SQL queries
interfaces with ►
1 1
«external entity»
:Database
provides ►
1 0..n
«database wrapper»
:InventoryItem
-id: int
-name: String
-price: double
-discount: double
-qty: int
-messagegThresh: int
#getId(): int
#getName(): String
#getPrice(): double
#getPriceWithDiscount(): double
#getDiscount(): double
#getQty(): int
#getMessageThresh(): int
#setId(int): void
#setName(String): void
#setPrice(double): void
#setDiscount(double): void
#setQty(int): void
#setMessageThresh(int): void
#subtractQty(int): void
#addQty(int): void
Describes an item carried by the store.
1
sends ▼
1
1
uses ▼
1
«entity»
:InventoryMessageReport
-imanager: InventoryManager
-report: List<InventoryMessage>
#reset(): void
#add(InventoryMessage): void
Denotes all items with low stock
contains ►
1 0..n
«entity»
:InventoryMessage
-id: int
#getId(): int
#reset(): void
#add(InventoryMessage): void
Denotes all items with low stock
«entity»
:ConnectionProvider
-con: Connection
-dbhost: String
-dbloginfile: String
-dbname: String
-dbPort: String
-password: String
username: String
#getCon(): Connection
#readCredFile(String): boolean
#readCredFile(): boolean
#setCredFile(String): void
Provides SQL Connection to DBManager
Relation in relational database:
-Inventory (_id_ , ItemName, Price, Discount, Qty, Threshold)

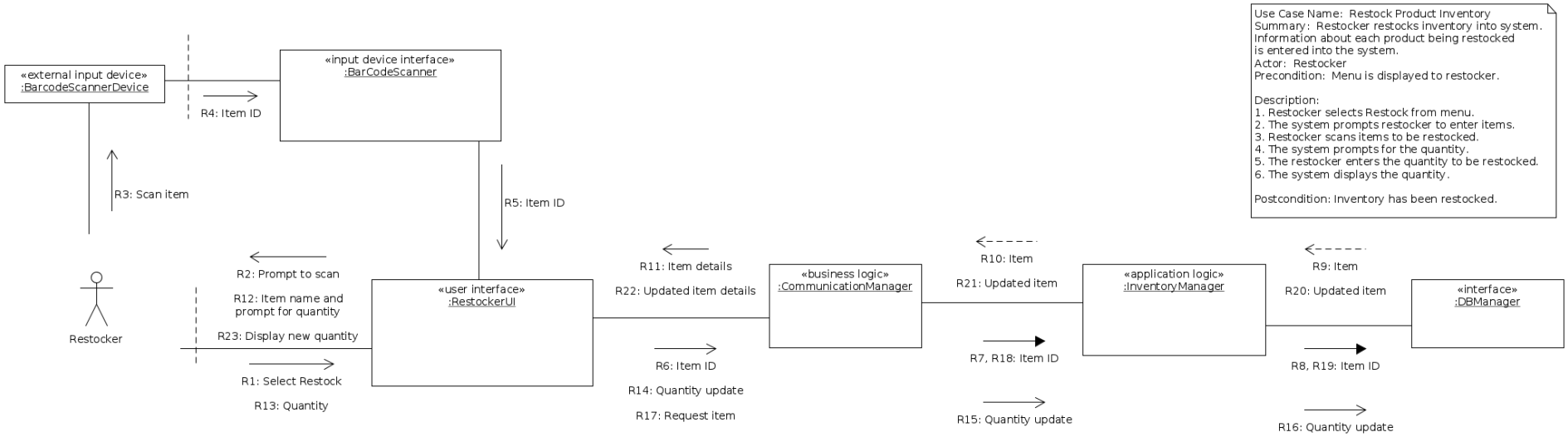

«application logic»
:TransactionManager
-transLog: TransactionLog
-cm: CommunicationManager
-im: InventoryManager
#processSaleOfItem(int): void
#getDailyReport(): TransactionLog
#resetDailyReport(): void
Generates and sends transaction reports.
has ►
1 1
«entity»
:TransactionLog
-transList: List<Transaction>
#add(Transaction): void
#get(int): Transaction
#getSize(): int
#getTotalRevenue(): double
#consolidateTransactions(): TransactionLog
#getTransactionList(): List<Transaction>
Daily list of all customer transactions.
contains ►
1 0..n
«entity»
:Transaction
-item: int
-price: double
-quantity: int
#getItem(): int
#getQty(): int
#getPrice(): double
#incrementQty(int): void
#addToPrice(float): void
Historic record of a customer transaction, including items purchased, their prices and the quantity purchased.

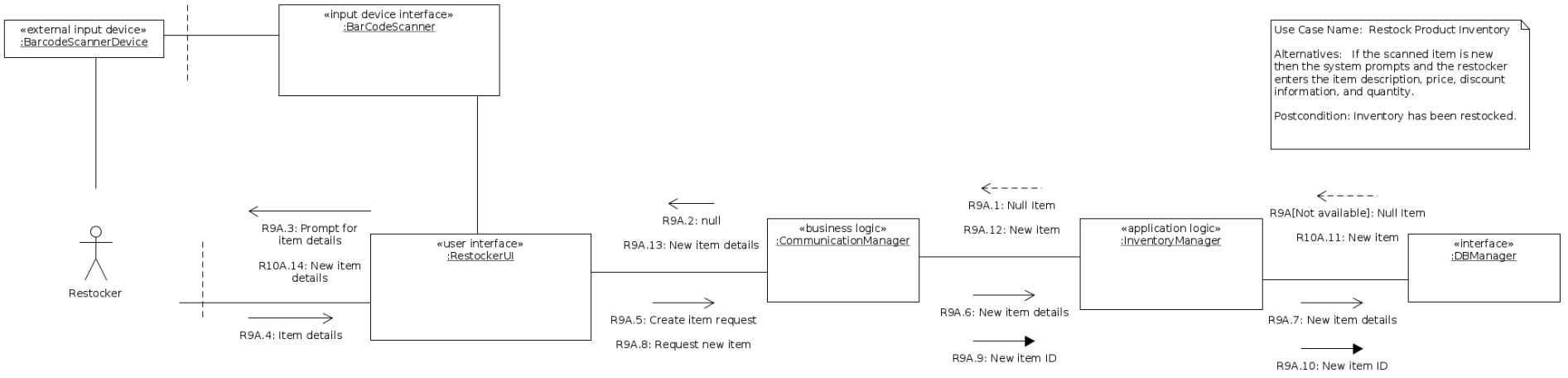

«business logic»
:CommunicationManager
◄ communicates with
1
0..n
«interface»
:Client
-destAddress: String
-portNum: int
-quietMode: boolean
-sendBuffer: String
-sout: PrintWriter
-workToDo: boolean
#connect(): Thread
#die(): void
#setQuietMode(boolean): void
+send()
+receive()
This class handles clientside session to CommunicationManager
«interface»
:ApplicationLayerClient
-targetItem: InventoryItem
-expectingItem: boolean
-order: List<InventoryItem>
#requestInventoryItem(int): void
#itemExists(int): void
#go(): void
This class handles clientside session to CommunicationManager
«user interface»
:CustomerUI
-authResult: int
-ba: BillAcceptor
-billInput: int
-bs: BarCodeScanner
-ca: coinAcceptor
-cardInput: long
-cashValueInserted: double
-cd: CashDispenser
-cr: CardReader
-kbd: Scanner
-pinpadInput: int
-pp: PinPad
-rp: RecieptPrinter
-screenInput: int
#CheckoutOrder(i: CustomerOrder): void
#PromptToScan(): void
#ScanItem(i: int): void
#UpdateOrderDisplay(): void
#PrintTotal(): void
#PromptForPaymentType(): void
#SelectPaymentType(): void
#ShowSubtotal(): void
#SendAuthorizationRequest()
#NewScreen(): void
#ReportToTransManager(): void
This class resides on the checkout station itself.
«entity»
:CustomerOrder
-ClassAttribute: List<InventoryItem>
-taxRate: double
#add(InventoryItem): void
#getItemAtIndex(i: int): InventoryItem
#size(): int
#getTaxTotal(): double
#getSubtotal(): double
#getTotal(): double
#getOrder(): List<InventoryItem>
This class is a list of type InventoryItem whose contents are used throughout CustomerUI for inventory management and receipt printing. Reset for each customer.
◄ uses
1
1
«user interface»
:ManagerUI
-kbd: scanner
#ViewOrUpdate(): void
#displayInventoryItem(): boolean
#updateInventoryItem(i: InventoryItem): void
This class interacts with the store manager actor and the CommunicationManager.
«user interface»
:RestockerUI
-bs: BarCodeScanner
-expectingItem: boolean
-kbd: Scanner
#displayInventoryItem(int): boolean
#PromptForScan(): void
#UpdateQuantity(int, int): void
#AddNewItem(InventoryItem): void
This class initiates the restock use case associated with the actor Restocker
This class interacts with the rest of the system to fulfill the restock use case
has ▼
«output device interface»
:CashDispenser
addDime(): void
addFifty(): void
addFive(): void
addHundred(): void
addNickel(): void
addOne(): void
addPenny(): void
addQuarter(): void
addTen(): void
addTwenty(): void
dispense(double): int
getValueOfAllContained(): double
Keeps track of money held and dispenses as needed
«output device interface»
:ReceiptPrinter
#print(CustomerOrder): int
#print(CustomerOrder , int, int
Prints receipt for customer
«input device interface»
:CoinAcceptor
Accepts and identifies coins and sends amount recieved to CustomerUI
«input device interface»
:CardReader
#ejectCard(): void
#simulateBufferedScanLong(string): void
Reads credit card number
«input device interface»
:BillAcceptor
Accepts and identifies paper currency and sends amount recieved to customerUI
«input device interface»
:PinPad
Accepts PIN from customer
«input device interface»
:BarCodeScanner
Reads bar codes from inventory items
«input device interface»
:NumberReaderDevice
-reader: Scanner
-caller: Object
-devicename: String
#scan(): int
#bufferedScan(): void
#simulateBufferedScan(String): void
Reads numbers from physical devices
interfaces with ▼
«external output device»
:CashDispenserDevice
«external output device»
:PrinterDevice
«external input device»
:CoinAcceptorDevice
«external input device»
:CardReaderDevice
«external input device»
:BillAcceptorDevice
«external input device»
:PinPadDevice
«external input device»
:BarCodeScannerDevice

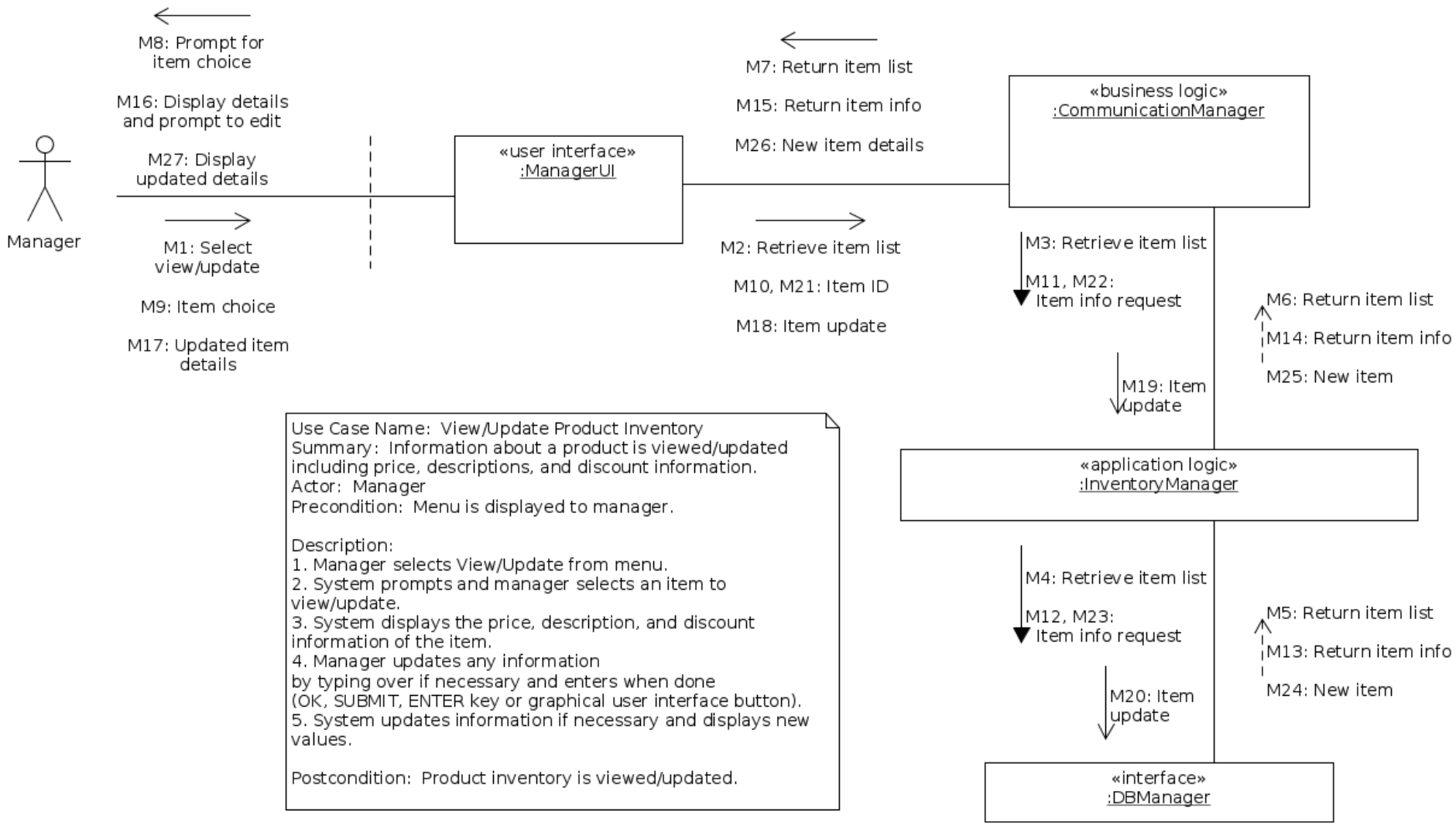
Dynamic Models






«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
R4: Item ID
R3: Scan item
R5: Item ID
Restocker
R2: Prompt to scan
R12: Item name and prompt for quantity
R23: Display new quantity
R1: Select Restock
R13: Quantity
«user interface»
:RestockerUI
R11: Item details
R22: Updated item details
R6: Item ID
R14: Quantity update
R17: Request item
«business logic»
:CommunicationManager
R10: Item
R21: Updated item
R7, R18: Item ID
R15: Quantity update
«application logic»
:InventoryManager
R9: Item
R20: Updated item
R8, R19: Item ID
R16: Quantity update
«interface»
:DBManager
Use Case Name: Restock Product Inventory
Summary: Restocker restocks inventory into system. Information about each product being restocked is entered into the system.
Actor: Restocker
Precondition: Menu is displayed to restocker.
Description:
1. Restocker selects Restock from menu.
2. The system prompts restocker to enter items.
3. Restocker scans items to be restocked.
4. The system prompts for the quantity.
5. The restocker enters the quantity to be restocked.
6. The system displays the quantity.
Postcondition: Inventory has been restocked.

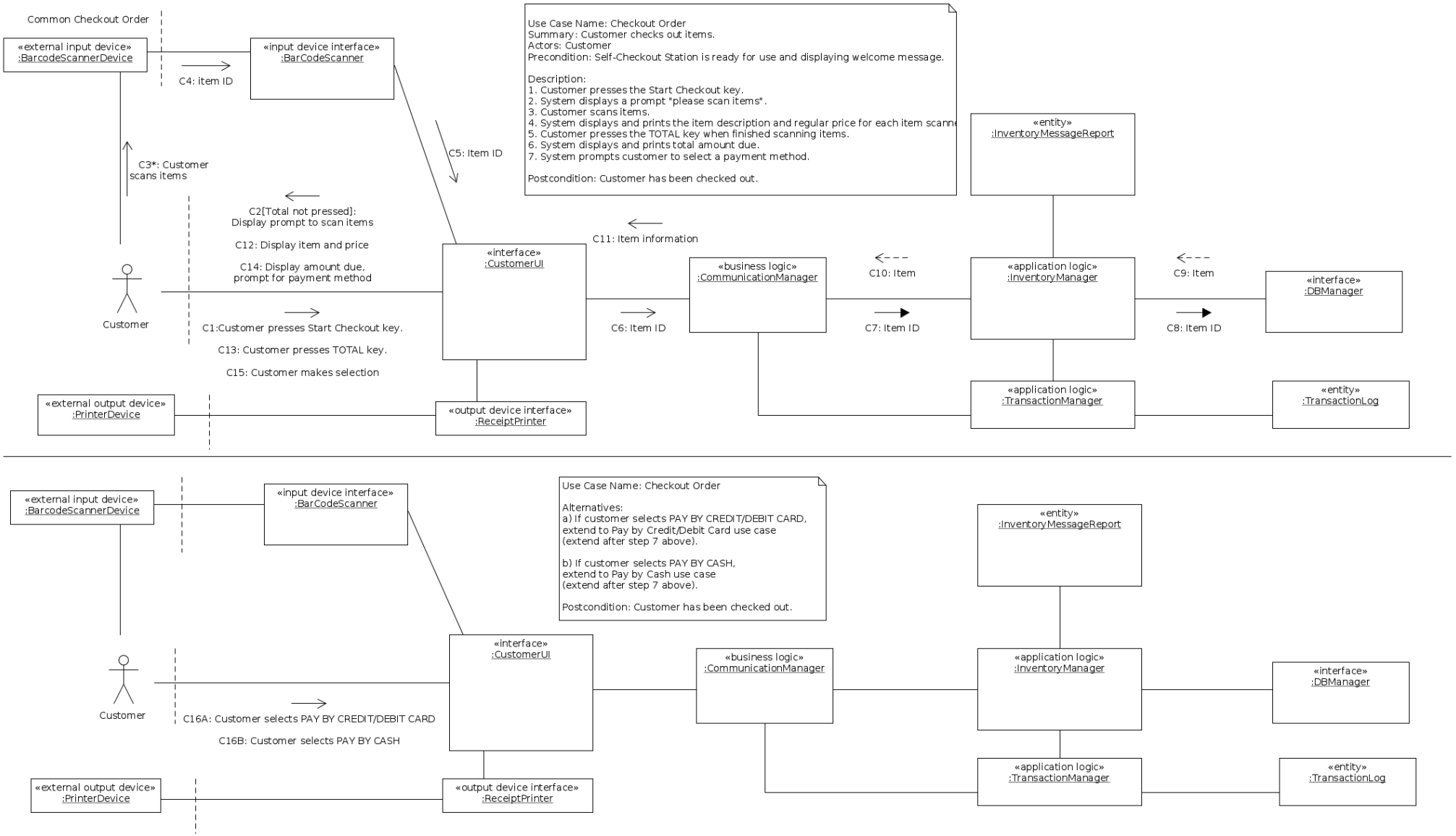

«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
Restocker
R9A.3: Prompt for item details
R10A.14: New item details
R9A.4: Item details
«user interface»
:RestockerUI
R9A.2: null
R9A.13: New item details
R9A.5: Create item request
R9A.8: Request new item
«business logic»
:CommunicationManager
R9A.1: Null Item
R9A.12: New item
R9A.6: New item details
R9A.9: New item ID
«application logic»
:InventoryManager
R9A[Not available]: Null Item
R10A.11: New item
R9A.7: New item details
R9A.10: New item ID
«interface»
:DBManager
Use Case Name: Restock Product Inventory
Alternatives: If the scanned item is new then the system prompts and the restocker enters the item description, price, discount information, and quantity.
Postcondition: Inventory has been restocked.

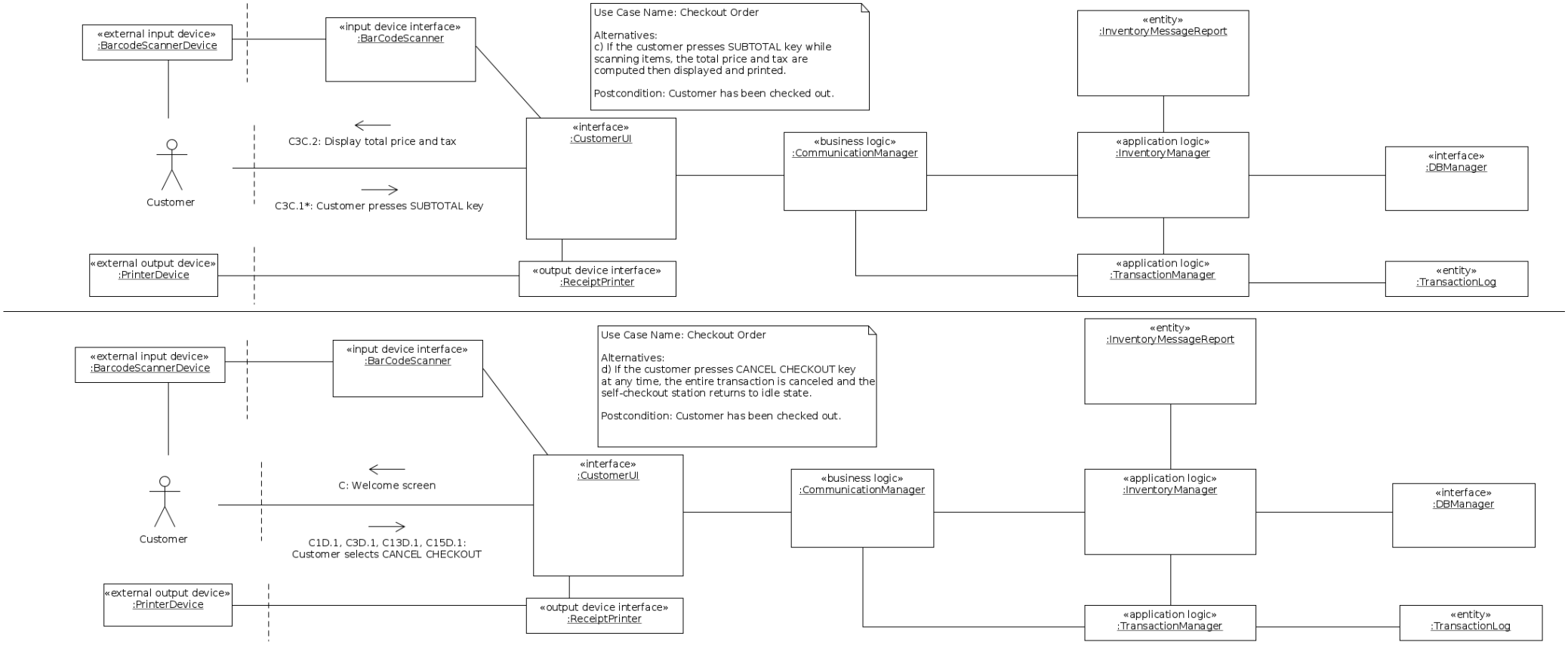


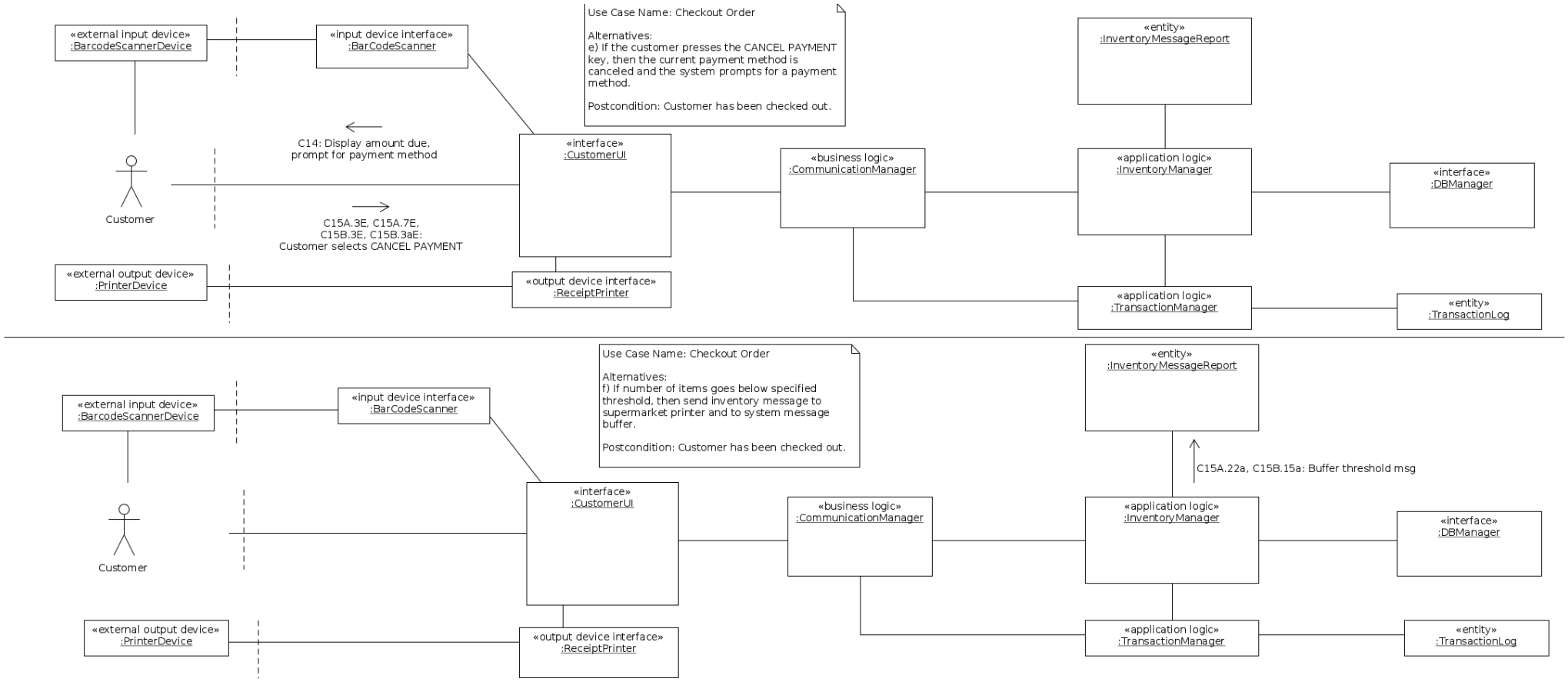
Use Case Name: View/Update Product Inventory
Summary: Information about a product is viewed/updated including price, descriptions, and discount information.
Actor: Manager
Precondition: Menu is displayed to manager.

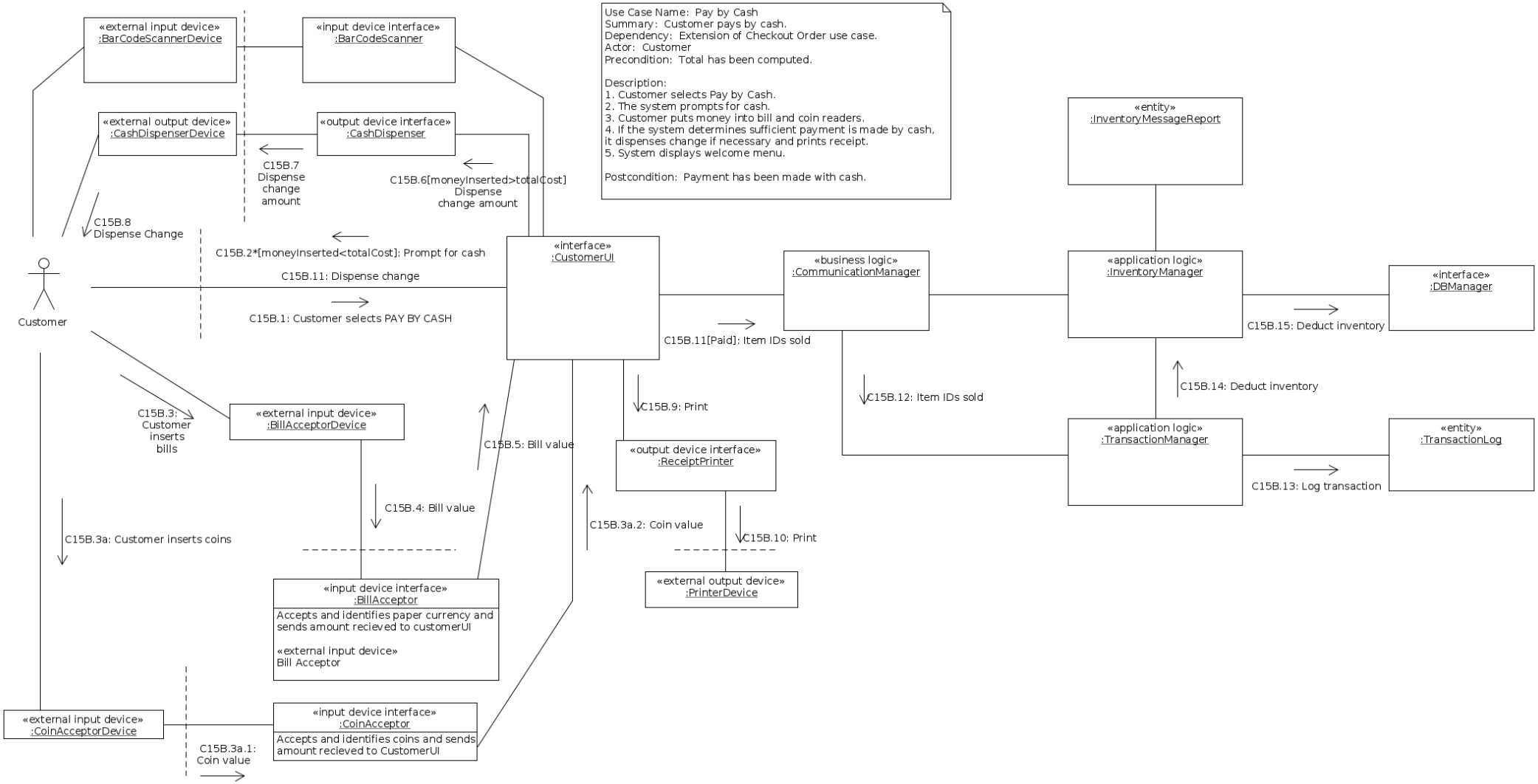
Description:
1. Manager selects View/Update from menu.
2. System prompts and manager selects an item to view/update.
3. System displays the price, description, and discount information of the item.
4. Manager updates any information
by typing over if necessary and enters when done
(OK, SUBMIT, ENTER key or graphical user interface button).
5. System updates information if necessary and displays new values.

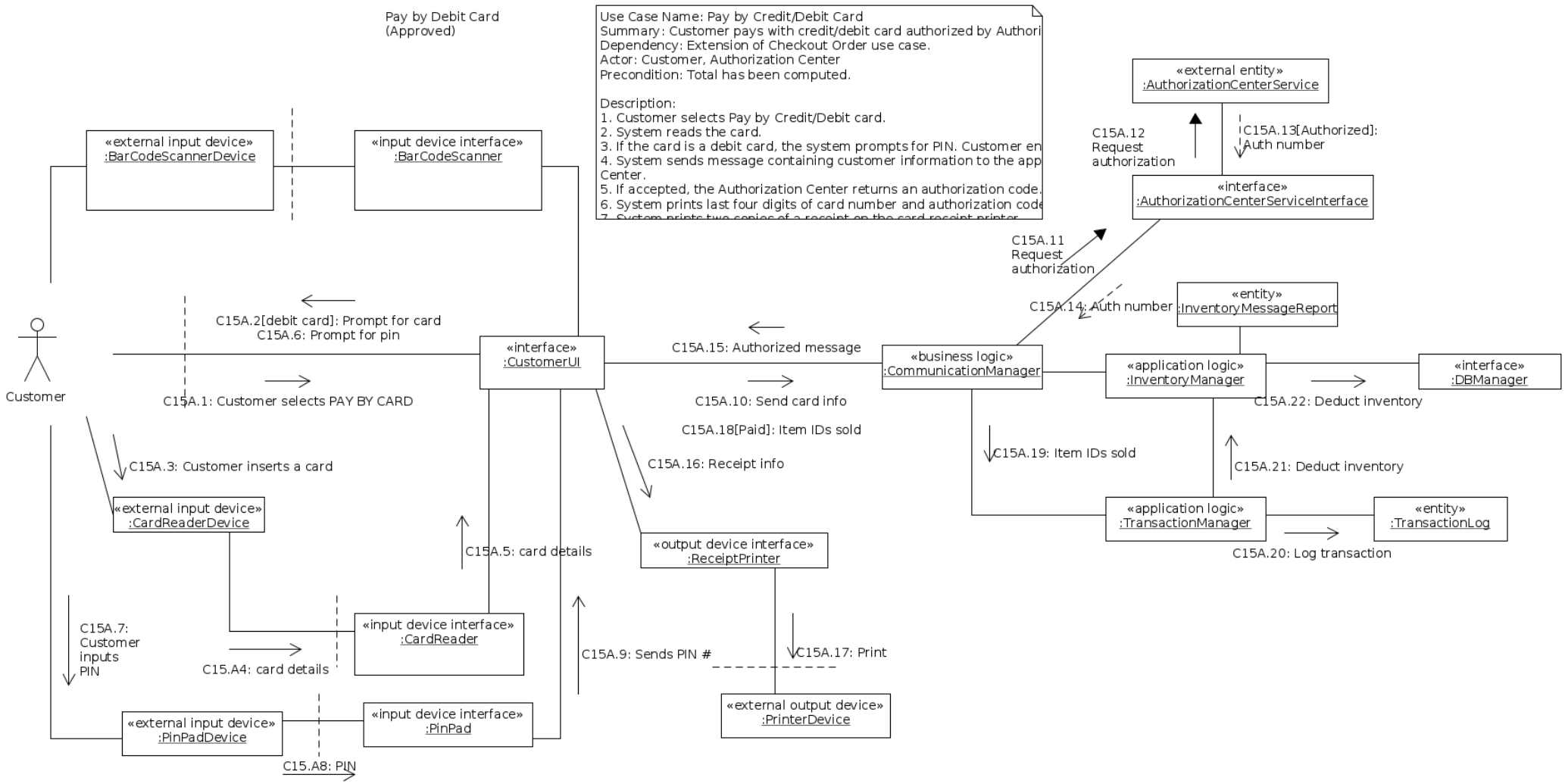
Postcondition: Product inventory is viewed/updated.

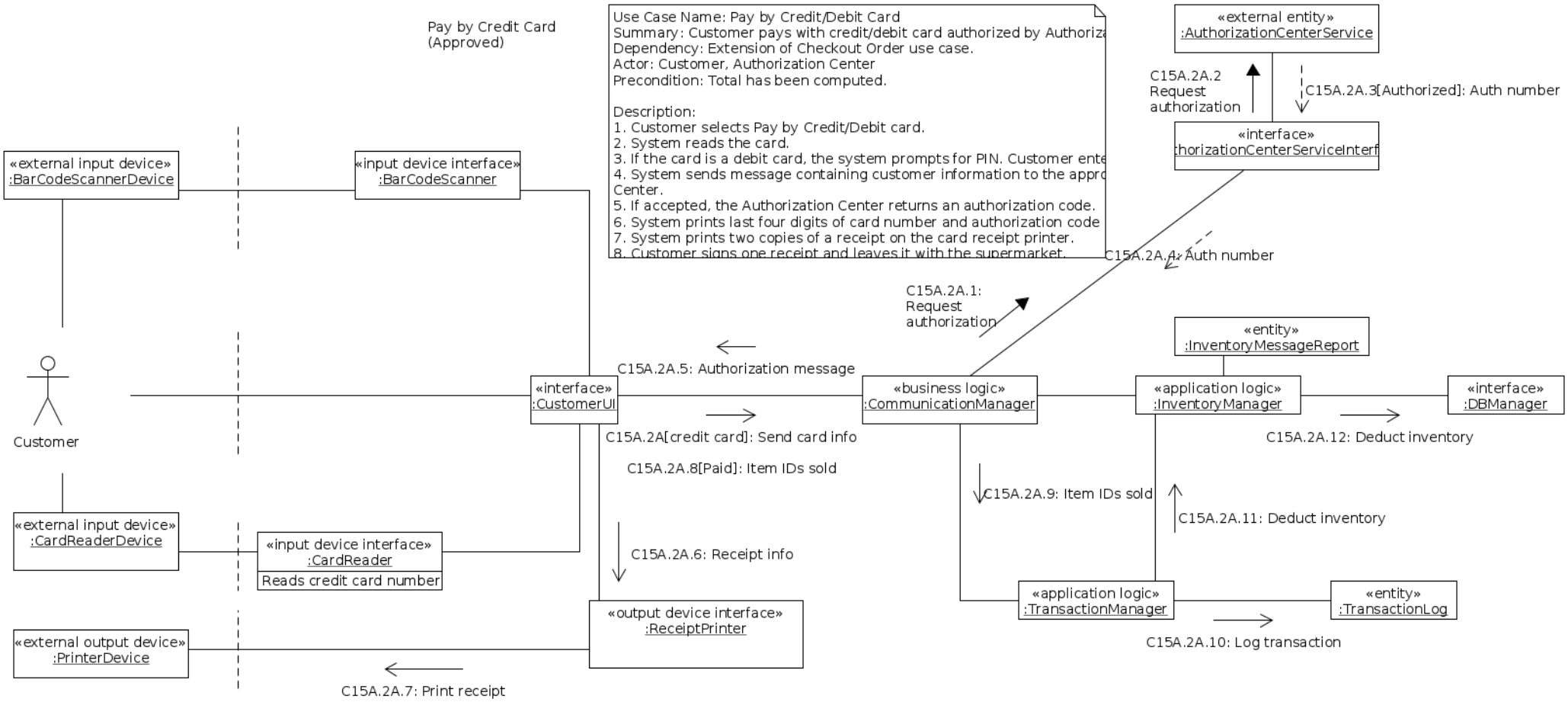

Common Checkout Order
«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
C4: item ID
C5: Item ID
C3*: Customer
scans items
C2[Total not pressed]:
Display prompt to scan items
C12: Display item and price
C14: Display amount due,
prompt for payment method
C1:Customer presses Start Checkout key.
C13: Customer presses TOTAL key.
C15: Customer makes selection
Customer
«interface»
:CustomerUI
C11: Item information
«business logic»
:CommunicationManager
C10: Item
C6: Item ID
C7: Item ID
«application logic»
:InventoryManager
C9: Item
C8: Item ID
«interface»
:DBManager
«entity»
:InventoryMessageReport
«application logic»
:TransactionManager
«entity»
:TransactionLog
«external output device»
:PrinterDevice
«output device interface»
:ReceiptPrinter
Use Case Name: Checkout Order
Summary: Customer checks out items.
Actors: Customer
Precondition: Self-Checkout Station is ready for use and displaying welcome message.
Description:
1. Customer presses the Start Checkout key.
2. System displays a prompt "please scan items".
3. Customer scans items.
4. System displays and prints the item description and regular price for each item scanne
5. Customer presses the TOTAL key when finished scanning items.
6. System displays and prints total amount due.
7. System prompts customer to select a payment method.
Postcondition: Customer has been checked out.
«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
«entity»
:InventoryMessageReport
«application logic»
:TransactionManager
«entity»
:TransactionLog
Customer
C16A: Customer selects PAY BY CREDIT/DEBIT CARD
C16B: Customer selects PAY BY CASH
«external output device»
:PrinterDevice
«output device interface»
:ReceiptPrinter
Use Case Name: Checkout Order
Alternatives:
a) If customer selects PAY BY CREDIT/DEBIT CARD,
extend to Pay by Credit/Debit Card use case
(extend after step 7 above).
b) If customer selects PAY BY CASH,
extend to Pay by Cash use case
(extend after step 7 above).
Postcondition: Customer has been checked out.

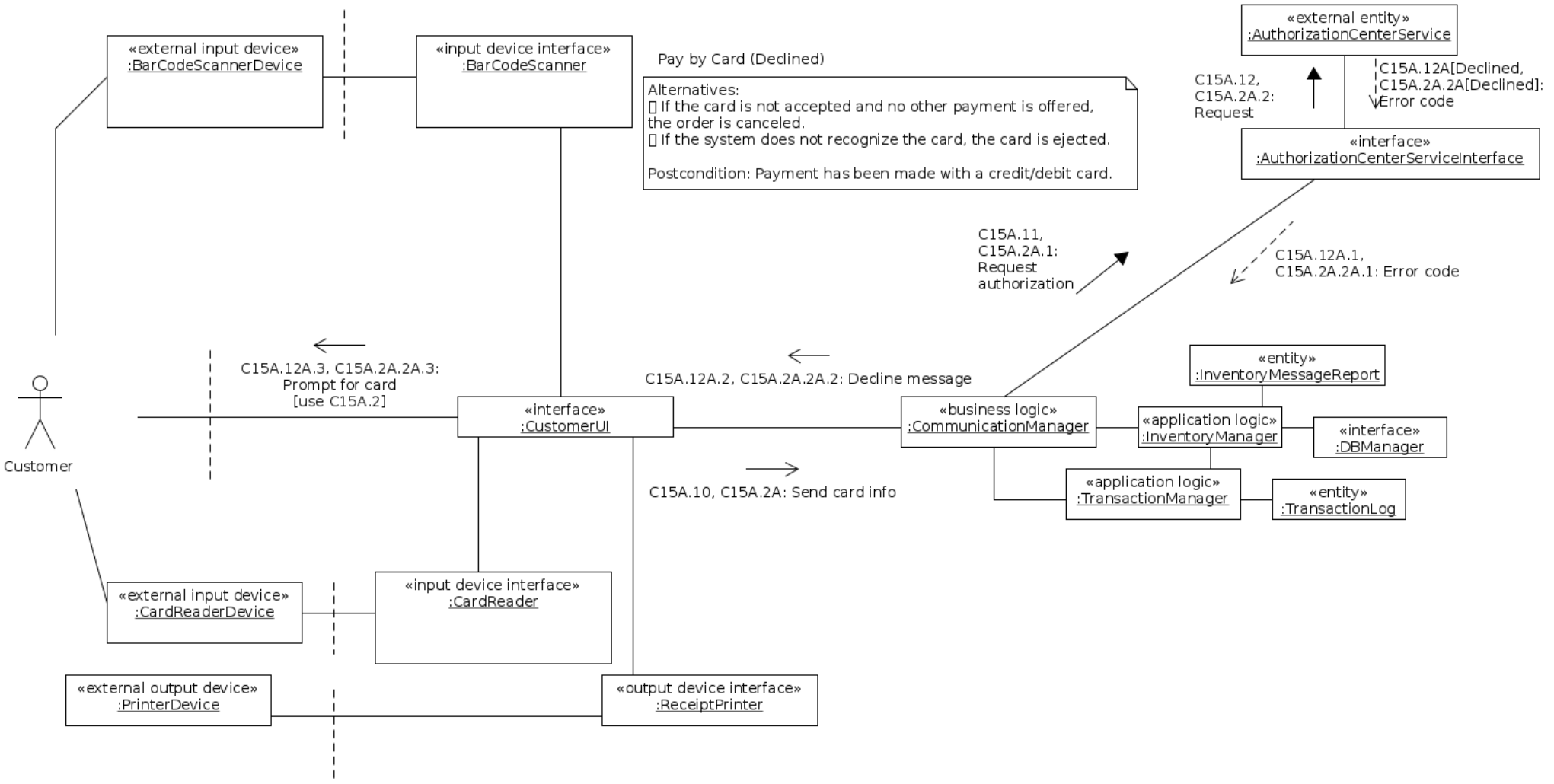

«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
Use Case Name: Checkout Order
Alternatives:
c) If the customer presses SUBTOTAL key while scanning items, the total price and tax are computed then displayed and printed.
Postcondition: Customer has been checked out.
«entity»
:InventoryMessageReport
C3C.2: Display total price and tax
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
Customer
C3C.1*: Customer presses SUBTOTAL key
«external output device»
:PrinterDevice
«output device interface»
:ReceiptPrinter
«application logic»
:TransactionManager
«entity»
:TransactionLog
«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
Use Case Name: Checkout Order
Alternatives:
d) If the customer presses CANCEL CHECKOUT key at any time, the entire transaction is canceled and the self-checkout station returns to idle state.
Postcondition: Customer has been checked out.
«entity»
:InventoryMessageReport
C: Welcome screen
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
Customer
C1D.1, C3D.1, C13D.1, C15D.1:
Customer selects CANCEL CHECKOUT
«external output device»
:PrinterDevice
«output device interface»
:ReceiptPrinter
«application logic»
:TransactionManager
«entity»
:TransactionLog


«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
Use Case Name: Checkout Order
Alternatives:
e) If the customer presses the CANCEL PAYMENT key, then the current payment method is canceled and the system prompts for a payment method.
Postcondition: Customer has been checked out.
«entity»
:InventoryMessageReport
C14: Display amount due, prompt for payment method
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
Customer
C15A.3E, C15A.7E,
C15B.3E, C15B.3aE:
Customer selects CANCEL PAYMENT
«external output device»
:PrinterDevice
«output device interface»
:ReceiptPrinter
«application logic»
:TransactionManager
«entity»
:TransactionLog
«external input device»
:BarcodeScannerDevice
«input device interface»
:BarCodeScanner
Use Case Name: Checkout Order
Alternatives:
f) If number of items goes below specified threshold, then send inventory message to supermarket printer and to system message buffer.
Postcondition: Customer has been checked out.
«entity»
:InventoryMessageReport
C15A.22a, C15B.15a: Buffer threshold msg
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
Customer
«external output device»
:PrinterDevice
«output device interface»
:ReceiptPrinter
«application logic»
:TransactionManager
«entity»
:TransactionLog

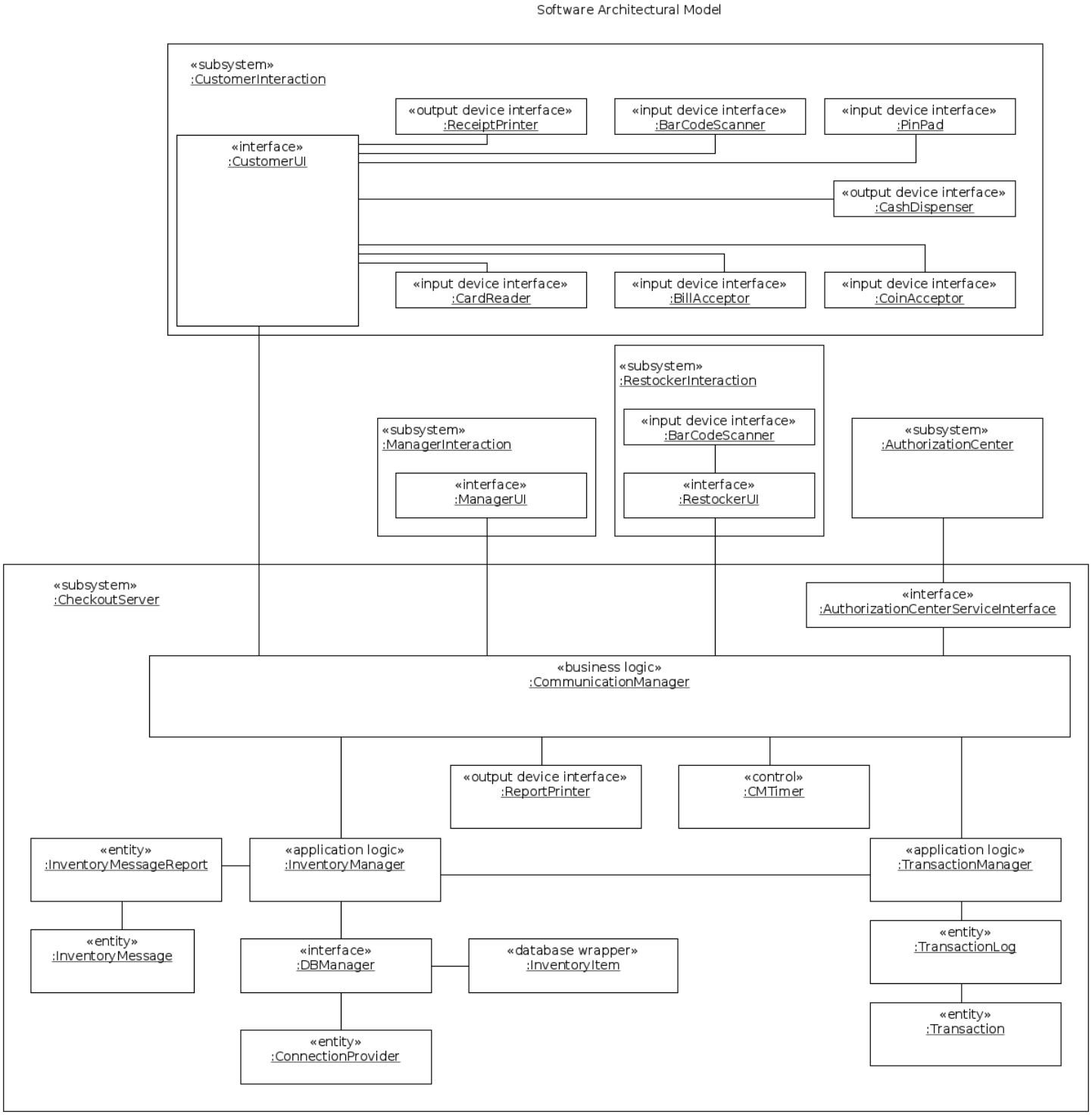

«external input device»
:BarCodeScannerDevice
«input device interface»
:BarCodeScanner
Use Case Name: Pay by Cash
Summary: Customer pays by cash.
Dependency: Extension of Checkout Order use case.
Actor: Customer
Precondition: Total has been computed.
Description:
1. Customer selects Pay by Cash.
2. The system prompts for cash.
3. Customer puts money into bill and coin readers.
4. If the system determines sufficient payment is made by cash, it dispenses change if necessary and prints receipt.
5. System displays welcome menu.
Postcondition: Payment has been made with cash.
«external output device»
:CashDispenserDevice
«output device interface»
:CashDispenser
«entity»
:InventoryMessageReport
C15B.7
Dispense
change
amount
C15B.6[moneyInserted>totalCost]
Dispense
change amount
C15B.8
Dispense Change
C15B.2*[moneyInserted<totalCost]: Prompt for cash
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
C15B.11: Dispense change
C15B.1: Customer selects PAY BY CASH
Customer
C15B.11[Paid]: Item IDs sold
C15B.15: Deduct inventory
C15B.12: Item IDs sold
C15B.14: Deduct inventory
C15B.3:
Customer
inserts
bills
«external input device»
:BillAcceptorDevice
C15B.9: Print
«application logic»
:TransactionManager
«entity»
:TransactionLog
C15B.5: Bill value
«output device interface»
:ReceiptPrinter
C15B.4: Bill value
C15B.3a: Customer inserts coins
C15B.3a.2: Coin value
C15B.10: Print
C15B.13: Log transaction
«input device interface»
:BillAcceptor
Accepts and identifies paper currency and sends amount recieved to customerUI
«external input device»
Bill Acceptor
«external output device»
:PrinterDevice
«external input device»
:CoinAcceptorDevice
«input device interface»
:CoinAcceptor
Accepts and identifies coins and sends amount recieved to CustomerUI
C15B.3a.1:
Coin value

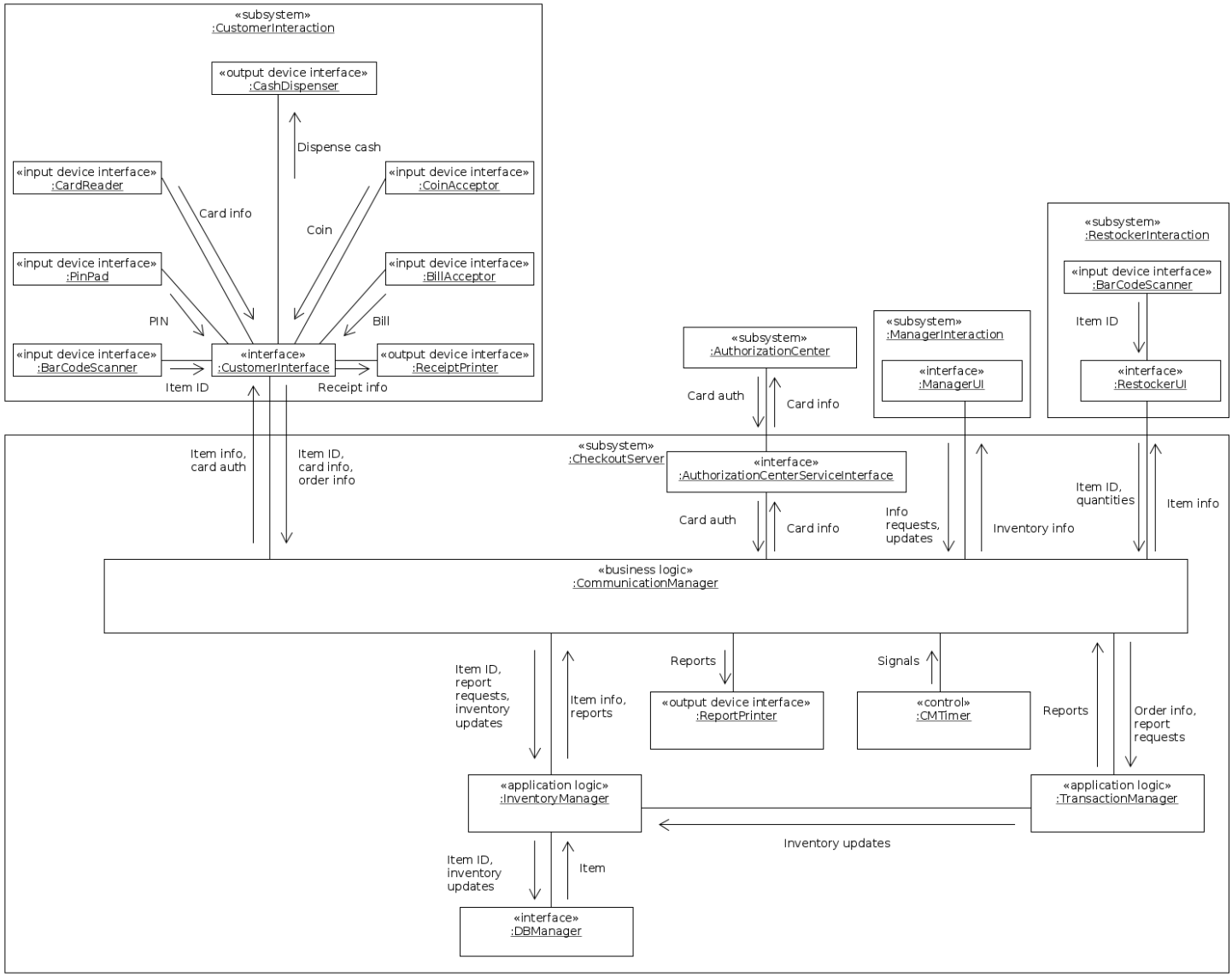

Pay by Debit Card
(Approved)
Use Case Name: Pay by Credit/Debit Card
Summary: Customer pays with credit/debit card authorized by Authori
Dependency: Extension of Checkout Order use case.
Actor: Customer, Authorization Center
Precondition: Total has been computed.
Description:
1. Customer selects Pay by Credit/Debit card.
2. System reads the card.
3. If the card is a debit card, the system prompts for PIN. Customer en
4. System sends message containing customer information to the app
Center.
5. If accepted, the Authorization Center returns an authorization code.
6. System prints last four digits of card number and authorization code
«external input device»
:BarCodeScannerDevice
«input device interface»
:BarCodeScanner
«external entity»
:AuthorizationCenterService
C15A.12
Request
authorization
C15A.13[Authorized]:
Auth number
«interface»
:AuthorizationCenterServiceInterface
C15A.11
Request
authorization
C15A.14: Auth number
«entity»
:InventoryMessageReport
C15A.2[debit card]: Prompt for card
C15A.6: Prompt for pin
Customer
C15A.1: Customer selects PAY BY CARD
«interface»
:CustomerUI
C15A.15: Authorized message
C15A.10: Send card info
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
C15A.22: Deduct inventory
C15A.18[Paid]: Item IDs sold
C15A.19: Item IDs sold
C15A.21: Deduct inventory
C15A.3: Customer inserts a card
C15A.16: Receipt info
«external input device»
:CardReaderDevice
«application logic»
:TransactionManager
«entity»
:TransactionLog
C15A.20: Log transaction
C15A.5: card details
«output device interface»
:ReceiptPrinter
C15A.7:
Customer
inputs
PIN
C15.A4: card details
«input device interface»
:CardReader
C15A.9: Sends PIN #
C15A.17: Print
«external output device»
:PrinterDevice
«external input device»
:PinPadDevice
«input device interface»
:PinPad
C15.A8: PIN


Pay by Credit Card
(Approved)
Use Case Name: Pay by Credit/Debit Card
Summary: Customer pays with credit/debit card authorized by Authoriz
Dependency: Extension of Checkout Order use case.
Actor: Customer, Authorization Center
Precondition: Total has been computed.
Description:
1. Customer selects Pay by Credit/Debit card.
2. System reads the card.
3. If the card is a debit card, the system prompts for PIN. Customer ente
4. System sends message containing customer information to the appro
Center.
5. If accepted, the Authorization Center returns an authorization code.
6. System prints last four digits of card number and authorization code
7. System prints two copies of a receipt on the card receipt printer.
8. Customer signs one receipt and leaves it with the supermarket.
«external entity»
:AuthorizationCenterService
C15A.2A.2
Request
authorization
C15A.2A.3[Authorized]: Auth number
«interface»
thorizationCenterServiceInterf
C15A.2A.4: Auth number
C15A.2A.1:
Request
authorization
«external input device»
:BarCodeScannerDevice
«input device interface»
:BarCodeScanner
C15A.2A.5: Authorization message
«entity»
:InventoryMessageReport
Customer
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
C15A.2A[credit card]: Send card info
C15A.2A.12: Deduct inventory
C15A.2A.8[Paid]: Item IDs sold
C15A.2A.9: Item IDs sold
C15A.2A.11: Deduct inventory
«external input device»
:CardReaderDevice
«input device interface»
:CardReader
Reads credit card number
C15A.2A.6: Receipt info
«output device interface»
:ReceiptPrinter
«application logic»
:TransactionManager
«entity»
:TransactionLog
«external output device»
:PrinterDevice
C15A.2A.10: Log transaction
C15A.2A.7: Print receipt


«external input device»
:BarCodeScannerDevice
«input device interface»
:BarCodeScanner
Pay by Card (Declined)
Alternatives:
If the card is not accepted and no other payment is offered, the order is canceled.
If the system does not recognize the card, the card is ejected.
Postcondition: Payment has been made with a credit/debit card.
«external entity»
:AuthorizationCenterService
C15A.12, C15A.2A.2: Request
C15A.12A[Declined], C15A.2A.2A[Declined]: Error code
«interface»
:AuthorizationCenterServiceInterface
C15A.11, C15A.2A.1: Request authorization
C15A.12A.1, C15A.2A.2A.1: Error code
C15A.12A.3, C15A.2A.2A.3: Prompt for card [use C15A.2]
C15A.12A.2, C15A.2A.2A.2: Decline message
«entity»
:InventoryMessageReport
Customer
«interface»
:CustomerUI
«business logic»
:CommunicationManager
«application logic»
:InventoryManager
«interface»
:DBManager
C15A.10, C15A.2A: Send card info
«application logic»
:TransactionManager
«entity»
:TransactionLog
«external input device»
:CardReaderDevice
«input device interface»
:CardReader
«external output device»
:PrinterDevice
«output device interface»
:ReceiptPrinter

Architectural Models




«subsystem»
:CustomerInteraction
«output device interface»
:CashDispenser
Dispense cash
«input device interface»
:CardReader
«input device interface»
:CoinAcceptor
Card info
Coin
«input device interface»
:PinPad
«input device interface»
:BillAcceptor
PIN
Bill
«input device interface»
:BarCodeScanner
«interface»
:CustomerInterface
«output device interface»
:ReceiptPrinter
Item ID
Receipt info
«subsystem»
:RestockerInteraction
«input device interface»
:BarCodeScanner
Item ID
«interface»
:RestockerUI
«subsystem»
:ManagerInteraction
«interface»
:ManagerUI
«subsystem»
:AuthorizationCenter
Card auth
Card info
Item info,
card auth
Item ID,
card info,
order info
«subsystem»
:CheckoutServer
«interface»
:AuthorizationCenterServiceInterface
Card auth
Card info
Info
requests,
updates
Inventory info
Item ID,
quantities
Item info
«business logic»
:CommunicationManager
Item ID,
report
requests,
inventory
updates
Item info,
reports
Reports
«output device interface»
:ReportPrinter
Signals
«control»
:CMTimer
Reports
Order info,
report
requests
«application logic»
:InventoryManager
«application logic»
:TransactionManager
Inventory updates
Item ID,
inventory
updates
Item
«interface»
:DBManager