Software Engineering I – Final Project

The A Team

Member:

Zachary Simpson

Isaac Simpson

Huyen Vu

Wenhao Ge

Class/Attribute Descriptions:

<<Entity>>Item:

The *item* class creates an item object that can be scanned into the checkout system so a customer can purchase the item. Item objects contain the following attributes:

ID: An int variable containing the objects id number.

inventoryQuantity: An int value containing the quantity of an item in the inventory list.

discountPercent: An int value giving the percentage of discount on an item.

itemName: A string value containing the items name.

description: A string value giving the description of the item.

isAlcohol: A boolean value that states whether or not an item is classes as alcohol.

Price: A double value that gives the price of an item.

barCode: A string value that gives the barcode information.

inventoryLevel: A double value that gives the lowest inventory quantity allowed before having to reorder additional stock.

<<Entity>> TransactionEntity:

A *transaction entity* is a class object that contain all the information created from an individual transaction. It will contain information such as the date of transaction, total cost that the customer pays, the items purchased, and the prices of each item. A TransactionEntity will contain the following attributes:

ID: A string value that gives the transaction id.

Total: A double value that gives the total cost of a transaction.

createdDate: A string value that gives the date of the transaction.

items: A list object that contains *items* objects.

<<Interface>>BankInterface:

The *BankInterface* class provides a means for the checkout system to contact a bank and exchange customer information to verify a credit/debit cards authenticity as well as accept payments. It contains the following functions:

authorizeCardInfo() is a function that gets a confirmation number from a bank to verify and accept payment for a transaction.

<<Interface>> CheckoutInterface:

The *CheckoutInterface* class prompts a customer for input into a transaction and provides a way for the customer to interact with the checkout system to make a purchase. It contains the follwing function:

checkoutOrder() is a function that prompts the customer for items to be purchased and allows the customer to scan items into a transaction.

<<Interface>>PrinterInterface:

This *PrinterInterface* class allows the checkout system to interface with the printer to print the daily report and inventory report. It contains the following functions:

printDailyReport(): This function prints the daily inventory report.

printInvetoryReport(): This function prints the inventory report.

<<Interface>>ViewUpdateInterface:

The *ViewUpdateInterface* class provides an interface to view item information and update the information. It contains the following functions:

viewItem(): This function displays item information and asks for user input to call updateItem().

updateItem(): This function gets user input and changes item information for any of an items attributes.

viewUpdateProduct(): This function calls a list of items and gets user input to select an item to call *viewItem*() using an item id.

<<Interface>>RestockInterface:

The *RestockInterface* class provides an interface for the checkout system to access the inventory. It contains the following function:

RestockInventory(): This function calls *restockNewItem*() with a barCode when the inventory level reaches a null stock count.

<<Interface>> ReceiptPrinterInterface

The *ReceiptPrinterInterface* class is an interface class between the checkout system and the receipt printer at checkout to allow the system to print a customer receipt. It contains the following functions:

printReceipt():This function prints a receipt for a customer after a completed transaction.

printSubTotal(): This function prints the subtotal of a transaction on the receipt.

<<Business Logic>>ViewUpdateManager:

The *ViewUpdateManager* class allows the store manager to be able to view item information in the inventory and update the information. It contains the following functions:

viewItem(): This function returns item information for an item called with its' id.

updateItem(): This function updates the item information.

<<Business logic>>CashierManager:

The *CashierManager* class is a logic class that gets user input from the customer to determine the method that a customer will use to pay for a transaction. It contains the following functions:

checkDebitCard(): This function checks if the payment is with a debit card.

payByCash(): This function proceed payment by cash process.

payByCard(): This function checks if the payment is a debit or credit card. If a debit card is used then the function will prompt for user pin before calling the authorizeCardInfo function using cardNumber(). If a credit card is used, there is no need for pin input.

<<Business logic>> RestockManager:

The *RestockManager* class allows the user to input new information for input information for new or existing items. It contains the following functions:

restock(): When called this function allows the user to update information for an existing item in the inventory.

restockNewItem(): This function allows a user to input item information for a new item into the inventory.

<<Business logic>>CheckoutManager:

The *CheckoutManager* class is the main logic class that contains the main functions used to make a purchase with the checkout system. It contains the following functions:

addItemToCart(): Checks the barcode of a scanned item and checks to see if the item is alcohol. The function will then add the item to the card if validated.

getTotal(): This function returns the total price of the cart in the current transaction.

printSubtotal(): This function calculate total of current cart and returns for the customer to see.

checkout(): This function displays the current cart information such as the total price for the user and it prompts the user for a payment method. It will then complete the transaction if the payment is accepted, call the print receipt function and complete the purchase.

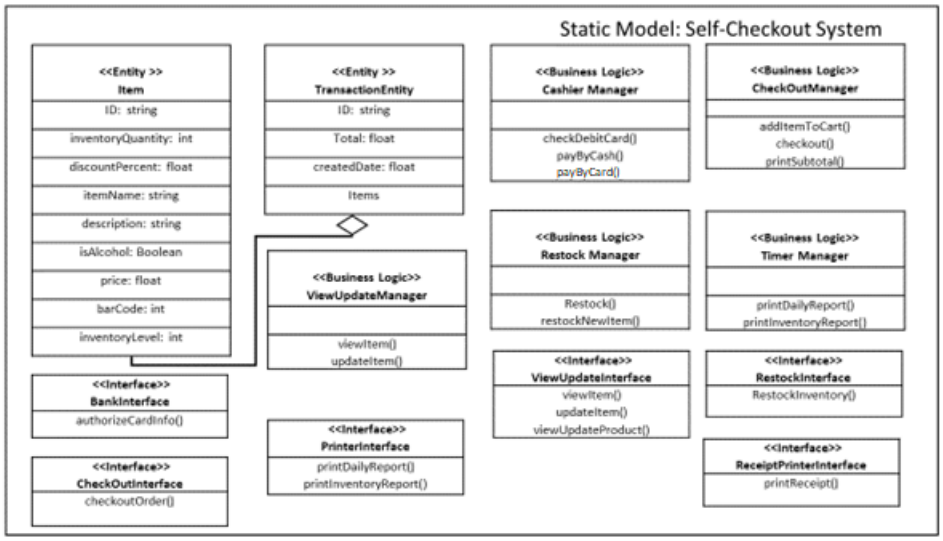
<<Business logic>> TimerManager:

The *TimerManager* class uses the current date and time to call specific functions. It is primarily used to print the daily report and inventory report. It contains the following functions:

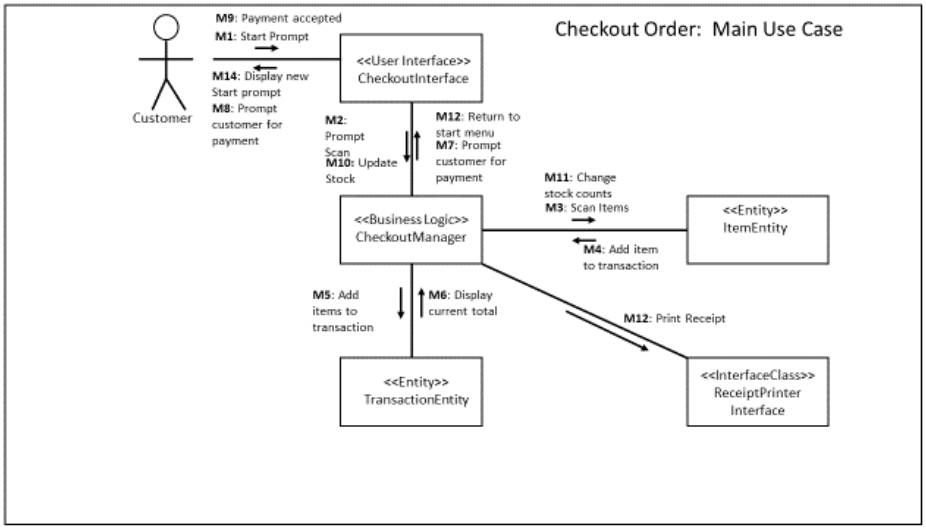
printDailyReport(): This function is activated by the system timer and prints a daily report of all transactions made in the checkout system.

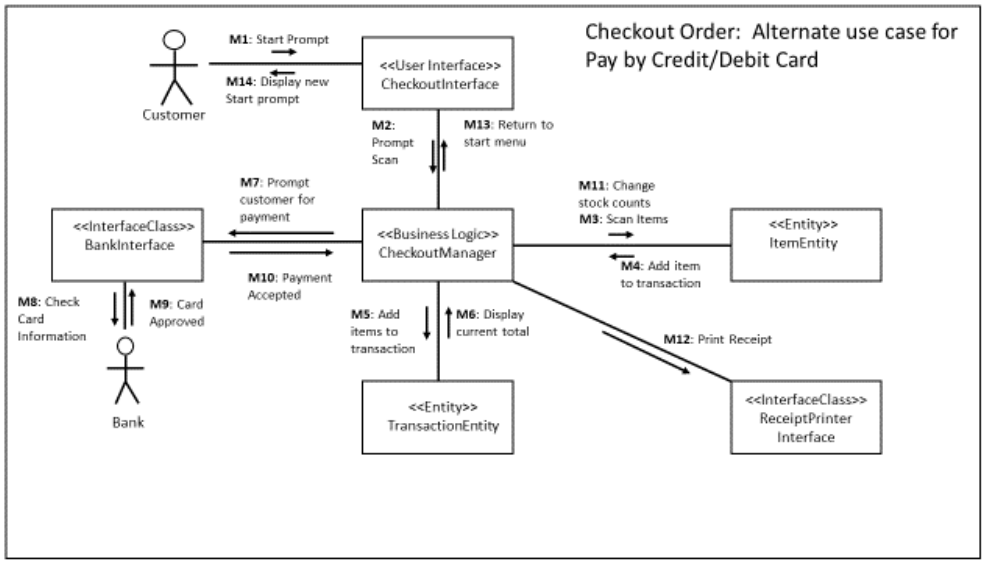
printInventoryReport(): When called, this function prints a report of all items whose inventory level is below its stated amount in the variable inventoryLevel. The report is then sent to the printer through the PrinterInterface class to print the report.

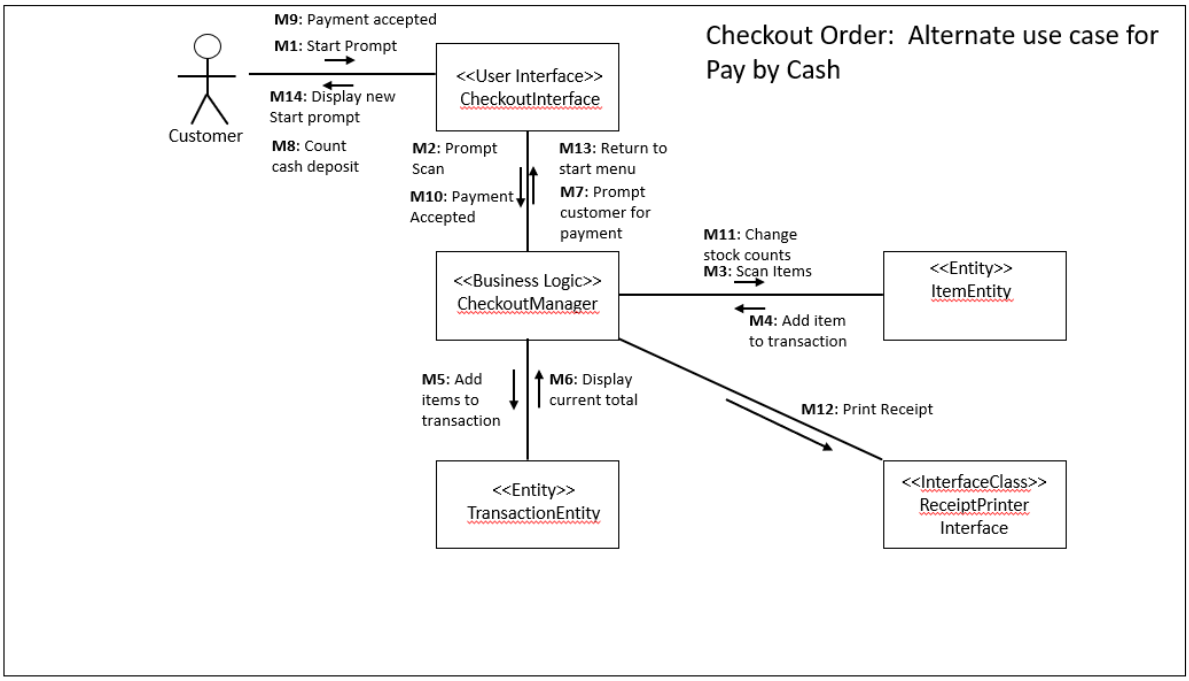
Static Model With operations:

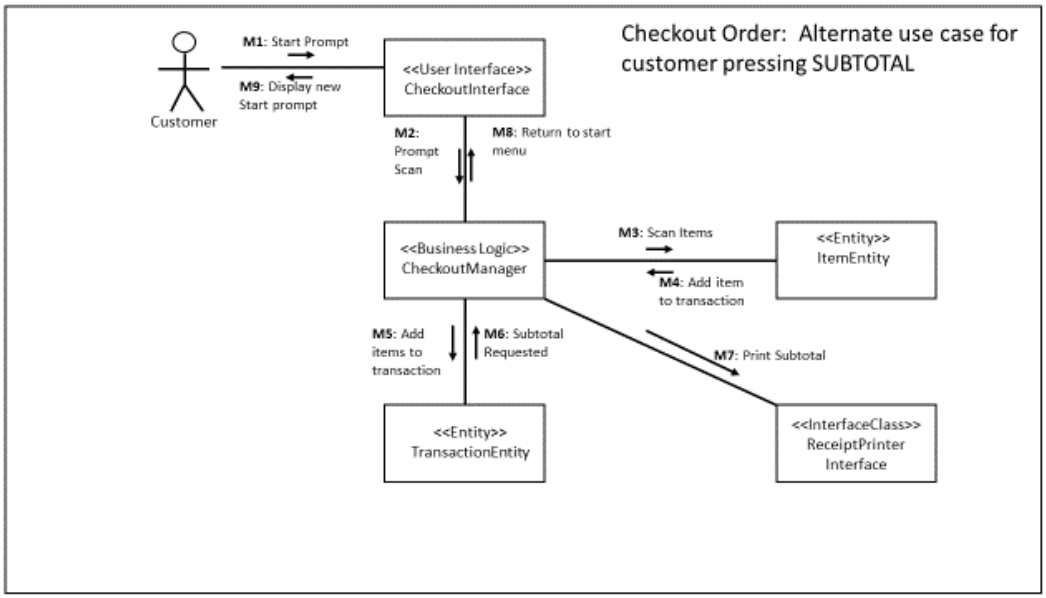


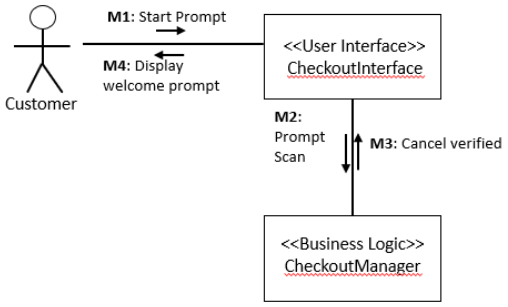
Interaction Models:



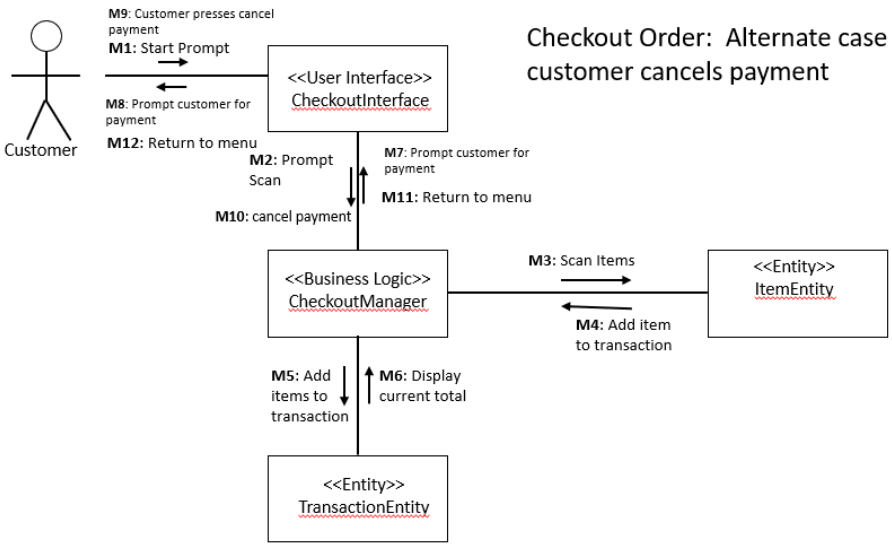



Checkout Order: Alternate use case for Pay by Cash
Customer
M9: Payment accepted
M1: Start Prompt
M14: Display new Start prompt
M8: Count cash deposit
<<User Interface>> CheckoutInterface
M2: Prompt Scan
M10: Payment Accepted
M13: Return to start menu
M7: Prompt customer for payment
<<Business Logic>> CheckoutManager
M11: Change stock counts
M3: Scan Items
M4: Add item to transaction
<<Entity>> ItemEntity
M5: Add items to transaction
M6: Display current total
M12: Print Receipt
<<Entity>> TransactionEntity
<<InterfaceClass>> ReceiptPrinter Interface

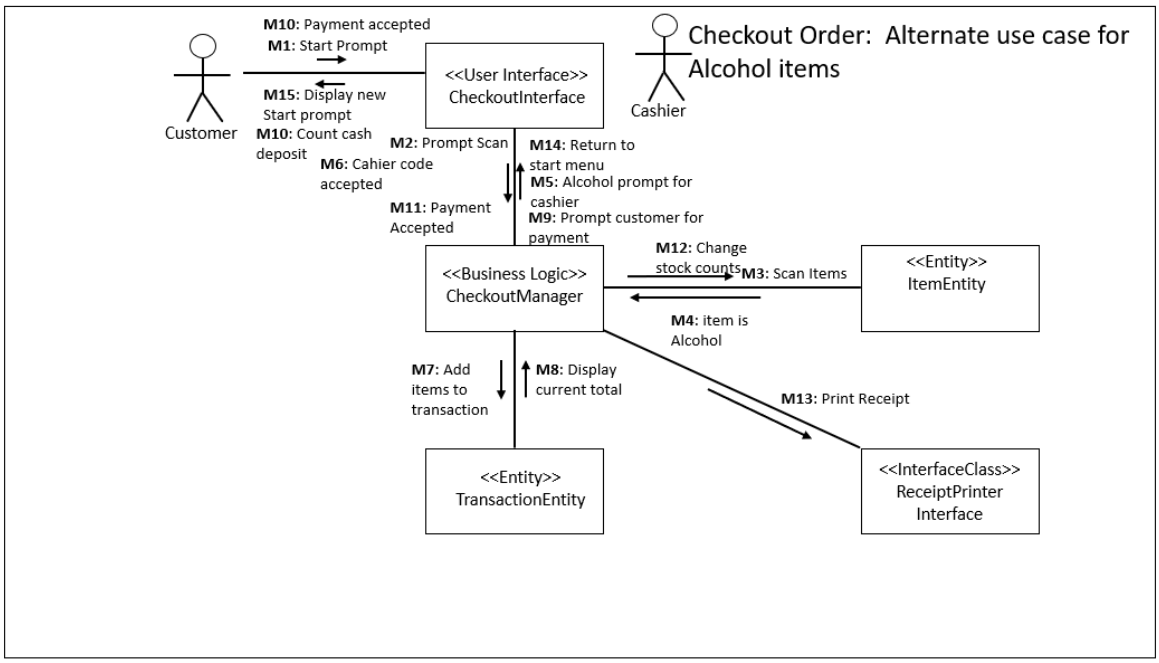

Checkout Order: Alternate use case for customer pressing SUBTOTAL
Customer
M1: Start Prompt
M9: Display new Start prompt
<<User Interface>> CheckoutInterface
M2: Prompt Scan
M8: Return to start menu
<<Business Logic>> CheckoutManager
M3: Scan Items
M4: Add item to transaction
<<Entity>> ItemEntity
M5: Add items to transaction
M6: Subtotal Requested
M7: Print Subtotal
<<Entity>> TransactionEntity
<<InterfaceClass>> ReceiptPrinter Interface

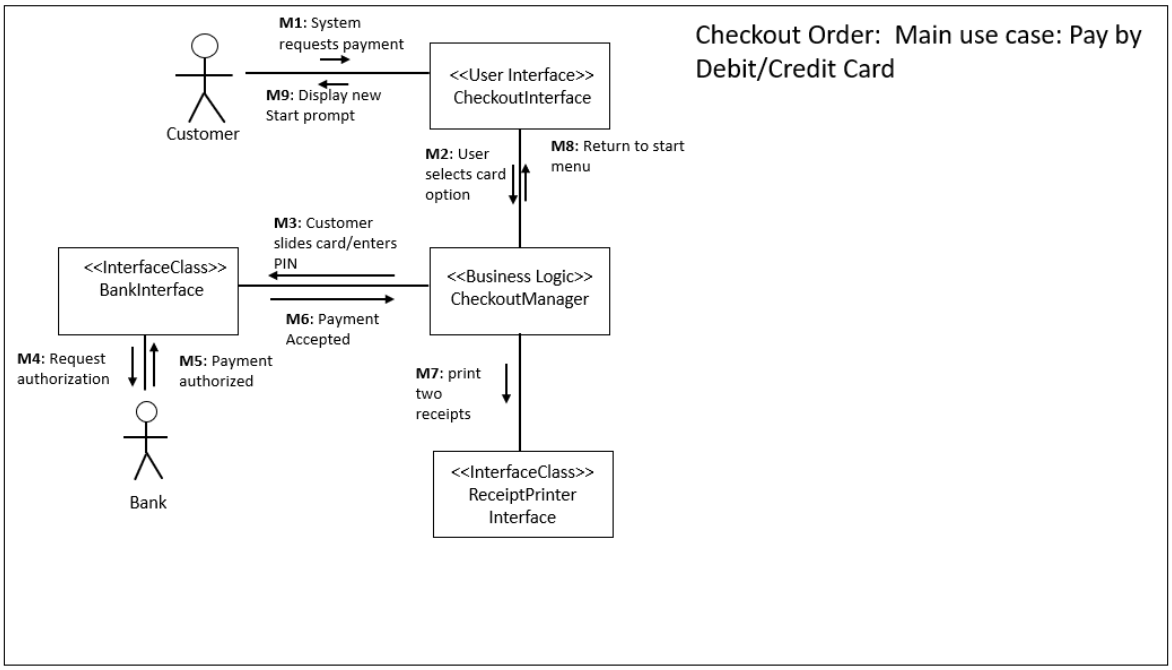


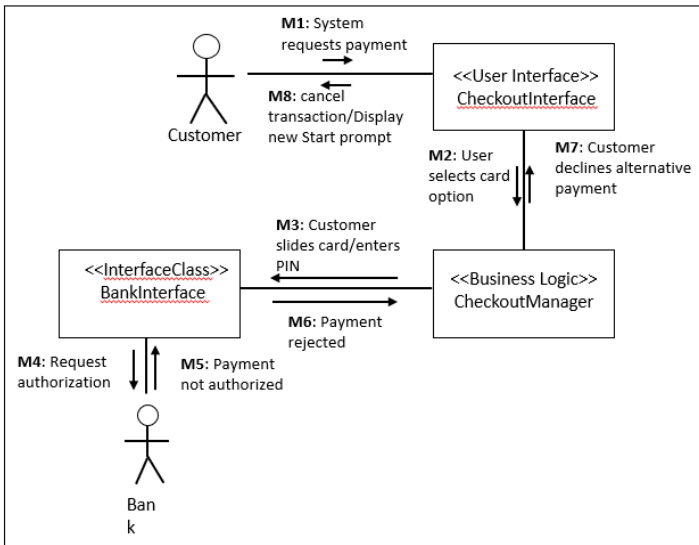
Checkout Order: Alternate use case for customer Cancels Checkout



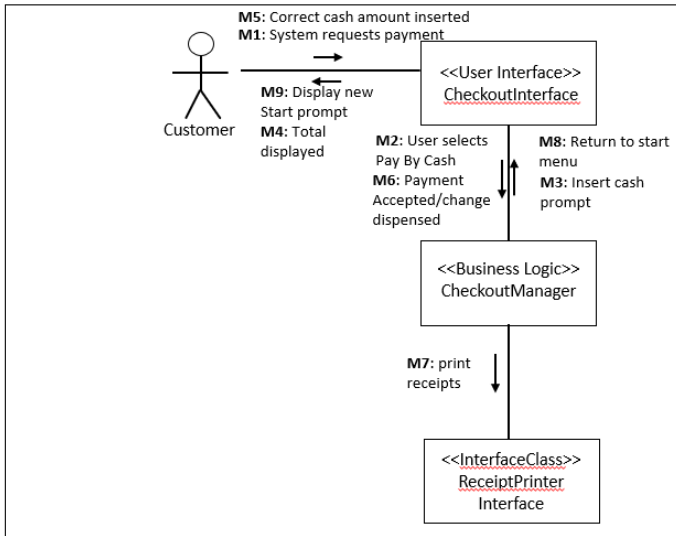
Checkout Order: Alternate case for customer cancels payment


Checkout Order: Alternate use case for Alcohol items
Customer
Cashier
<<User Interface>>
CheckoutInterface
<<Business Logic>>
CheckoutManager
<<Entity>>
ItemEntity
<<Entity>>
TransactionEntity
<<InterfaceClass>>
ReceiptPrinter
Interface
M10: Payment accepted
M1: Start Prompt
M15: Display new Start prompt
M10: Count cash deposit
M2: Prompt Scan
M6: Cahier code accepted
M11: Payment Accepted
M14: Return to start menu
M5: Alcohol prompt for cashier
M9: Prompt customer for payment
M12: Change stock counts
M3: Scan Items
M4: item is Alcohol
M7: Add items to transaction
M8: Display current total
M13: Print Receipt

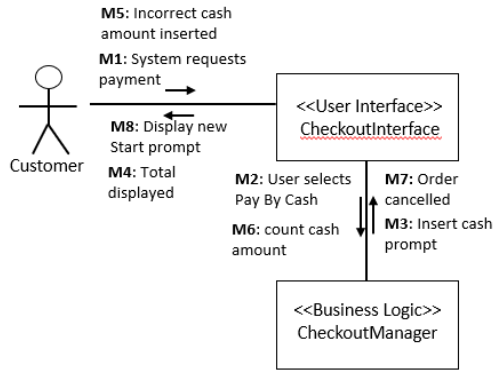

Checkout Order: Main use case: Pay by Debit/Credit Card
Customer
Bank
<<User Interface>>
CheckoutInterface
<<Business Logic>>
CheckoutManager
<<InterfaceClass>>
BankInterface
<<InterfaceClass>>
ReceiptPrinter
Interface
M1: System requests payment
M9: Display new Start prompt
M2: User selects card option
M8: Return to start menu
M3: Customer slides card/enters PIN
M6: Payment Accepted
M4: Request authorization
M5: Payment authorized
M7: print two receipts



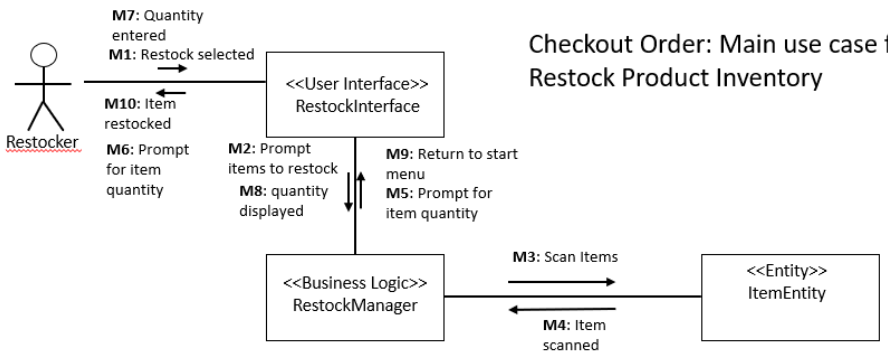
Checkout Order: Alternate use case for rejected card



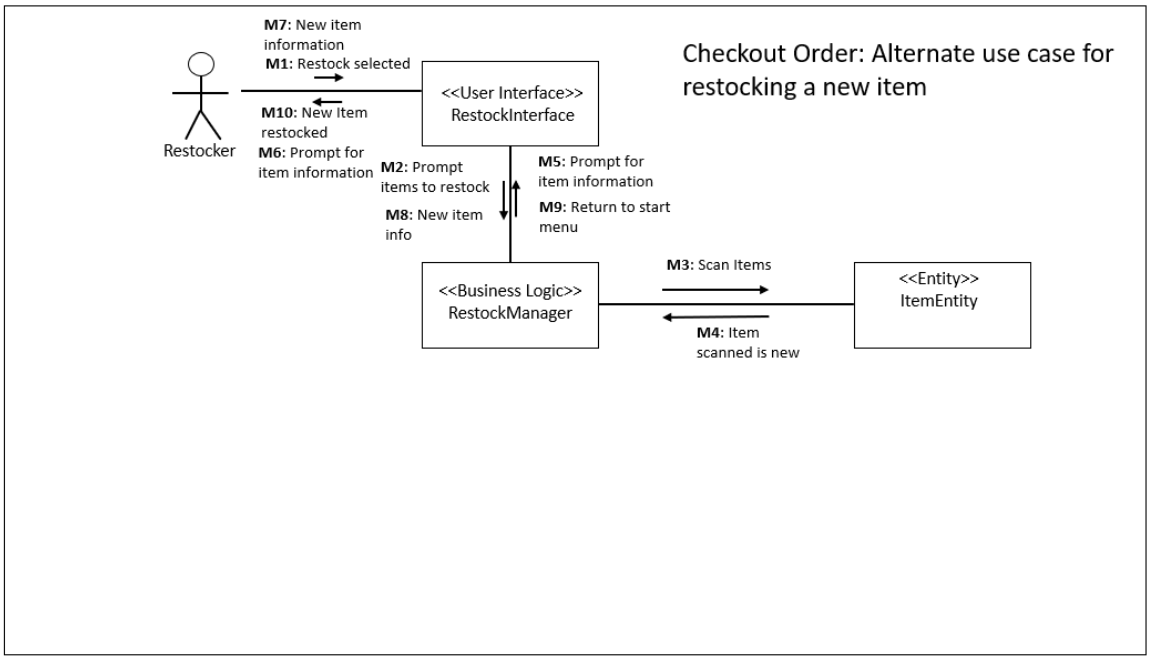
Checkout Order: Main use case: Pay by Cash

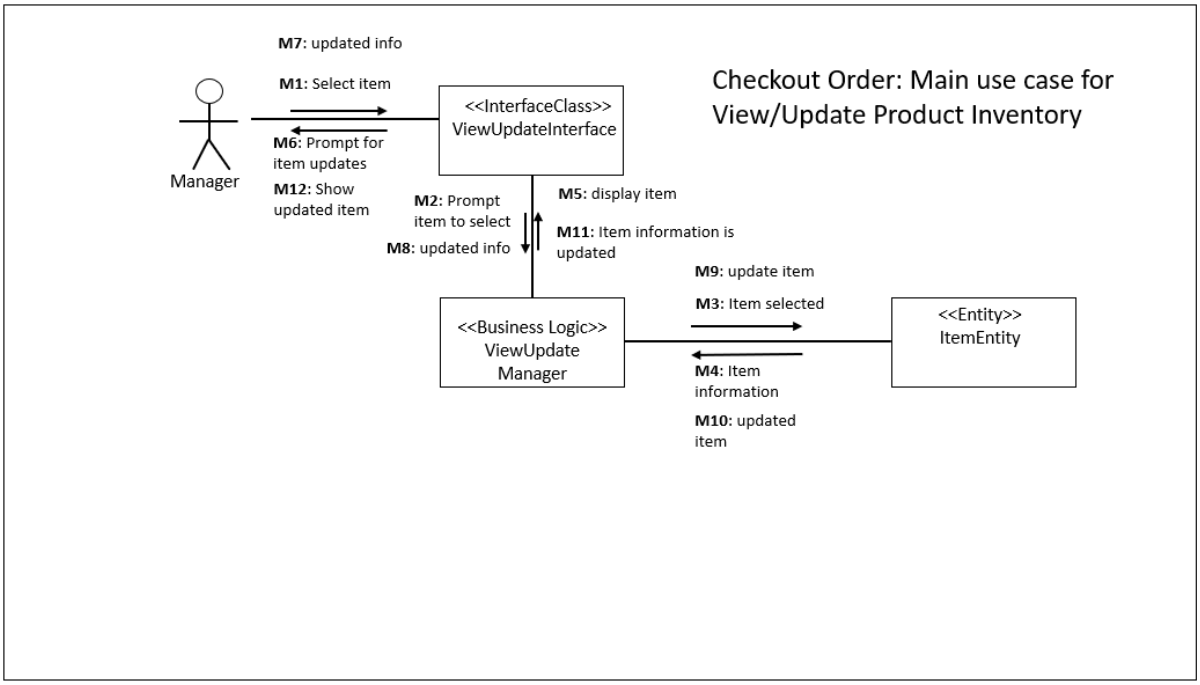


Checkout Order: Alternate use case: insufficient cash amount

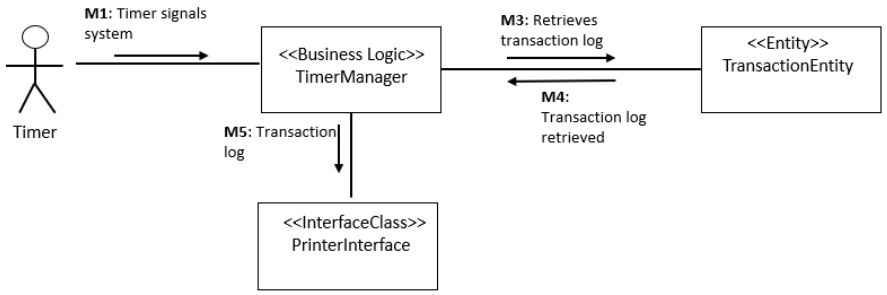


Checkout Order: Main use case for Restock Product Inventory

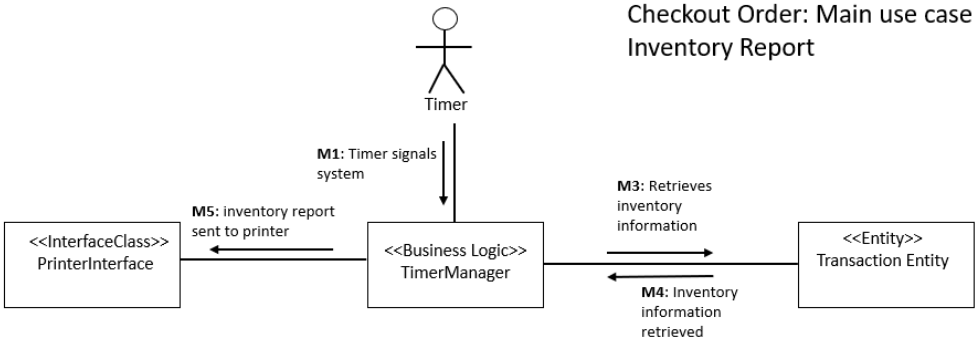

Checkout Order: Alternate use case for restocking a new item
Restocker
M7: New item information
M1: Restock selected
M10: New Item restocked
M6: Prompt for item information
<<User Interface>>
RestockInterface
M2: Prompt items to restock
M8: New item info
M5: Prompt for item information
M9: Return to start menu
<<Business Logic>>
RestockManager
M3: Scan Items
M4: Item scanned is new
<<Entity>>
ItemEntity


Checkout Order: Main use case for View/Update Product Inventory
Manager
M7: updated info
M1: Select item
M6: Prompt for item updates
M12: Show updated item
<<InterfaceClass>>
ViewUpdateInterface
M2: Prompt item to select
M8: updated info
M5: display item
M11: Item information is updated
M9: update item
M3: Item selected
M4: Item information
M10: updated item
<<Business Logic>>
ViewUpdate
Manager
<<Entity>>
ItemEntity

Checkout Order: Main use case for Print Daily Report



Checkout Order: Main use case for Print Inventory Report



Software Architecture Model: