Vending Machine Design Document

Date: 03/08/2024

Author: Garrett Engelder

Abstract

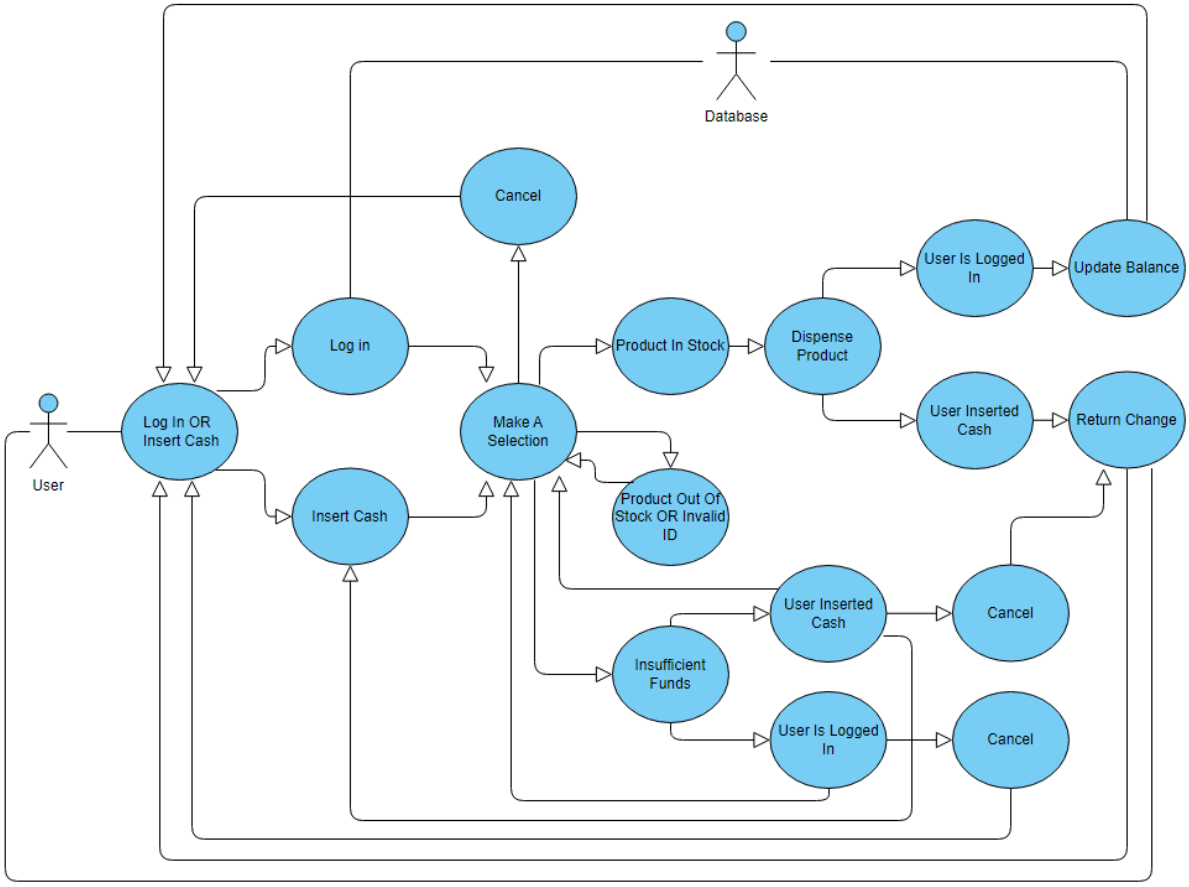
The Vending Machine program in Java is designed to simulate the functionality of a vending machine, allowing users to login to an account previously established in the vending machine's database or input cash and make purchases by selecting products through a keypad.

Table of Contents

Requirements

- mysql-connector-j-8.3.0.jar
- Oracle Open JDK 21
- MYSQL database with proper table names, variables, and sign in information. See P## E.R.D. for table names and variables. See readme for database sign in information

Use Case Diagram



Program Flow

1.0 Initialization:

- Initialize the program.
- Display to the user “Log in OR Insert Funds”.
- Prompt the user with two JOptionPanes
    - The first JOptionPane asks the user to “Enter Face Hash ID: ”
    - The second JOptionPane asks the user to “Insert Cash: ”

2.0 Flow - Option Cash:

- User inputs cash.
- Display “Balance: (Inserted Cash Balance)”

2.0 Flow - Option Log In

- User entered valid face hash ID
- Display “Welcome (User’s name)”
- Display “Balance: (Database Cash Balance)”

2.1 Flow Cont.

- Display “Make a selection”

2.2 Flow - User Inputs “cancel”

- Return to 1.0 Initialization - Display to the user “Log in OR Insert Funds”

2.3 Flow Cont.

- User inputs slot ID selection
- Run validation checks on slot ID
- Run check on price of slot versus balance
- Dispense product OR display validation check flag

Program Flow Cont.

2.4 Flow - Option Cash

- Return total balance to user

2.4 Flow - Option Log In

- Update user balance in database

2.5 Flow Cont.

- Return to 1.0 Initialization - Display to the user “Log in OR Insert Funds”

3.0 Program Validation Flag Checks:

- **If the user selects a slotID that is of greater value than total balance in machine & They are not logged in:**
    - Display “Insufficient Funds”
    - Display “Balance: (Inserted Cash Balance)”
    - Display “Price: (Price of slot)”
    - Prompt user with menu asking “Do you want to insert more funds?”
        - **If the user selects “Yes”**
            - Return to 1.0 Initialization - Display to the user “Log in OR Insert Funds”
                - **If user Logs in at this point, refund previously entered cash balance to user.**
        - **If the user selects “No”**
            - Return to 2.1 Flow Cont. - Display “Make a selection”
        - **If the user selects “Cancel”**
            - Display “Returning Balance: (Inserted Cash Balance)”

Program Flow Cont.

- Return to 1.0 Initialization - Display to the user “Log in OR Insert Funds”

- **If the user selects a slotID that is of greater value than total balance in machine & They are logged in:**
    - Display “Insufficient Funds”
    - Display “Balance: (Inserted Cash Balance)”
    - Display “Price: (Price of slot)”
    - Display “Make a selection”
        - **If the user enters “cancel”**
            - Return to 1.0 Initialization - Display to the user “Log in OR Insert Funds”
- **If the user selects an invalid slotID:**
    - Display “Invalid ID”
    - Allow for user to select a different slot
- **If the user selects a product out of stock:**
    - Display “Out of Stock”
    - Allow for user to make a different selection

Program Flow Cont.

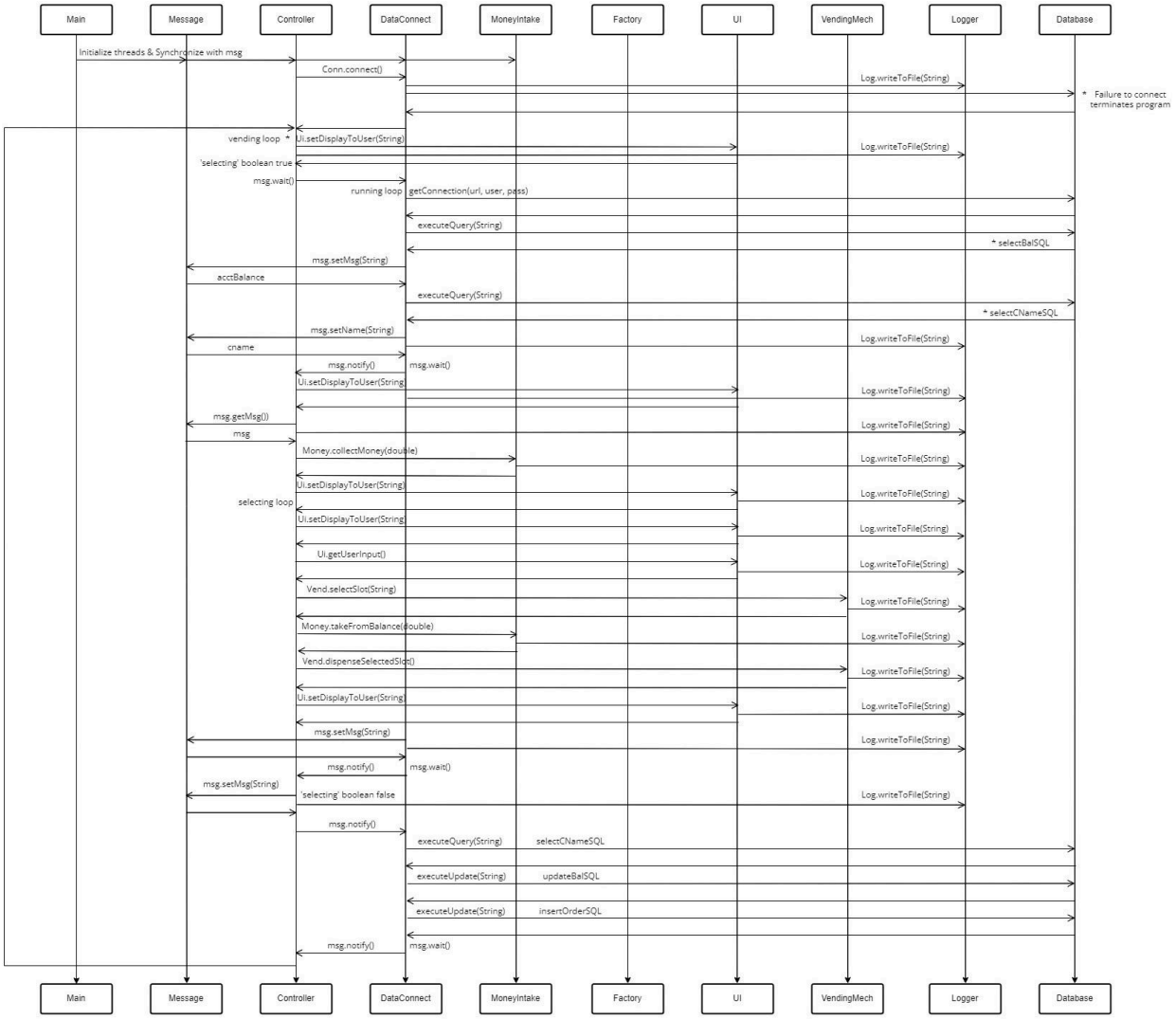
4.0 Program Completion:

- **If the user selects a valid slotID, the product is in stock, and the total balance is greater than price of product:**
    - Take price of product from user balance in vending machine
        - **If the user is logged in:**
            - Update user balance in database to new balance
        - **If the user is not logged in:**
            - Return remaining balance to user
    - Dispense product to user
    - Return to 1.0 Initialization - Display to the user “Log in OR Insert Funds”

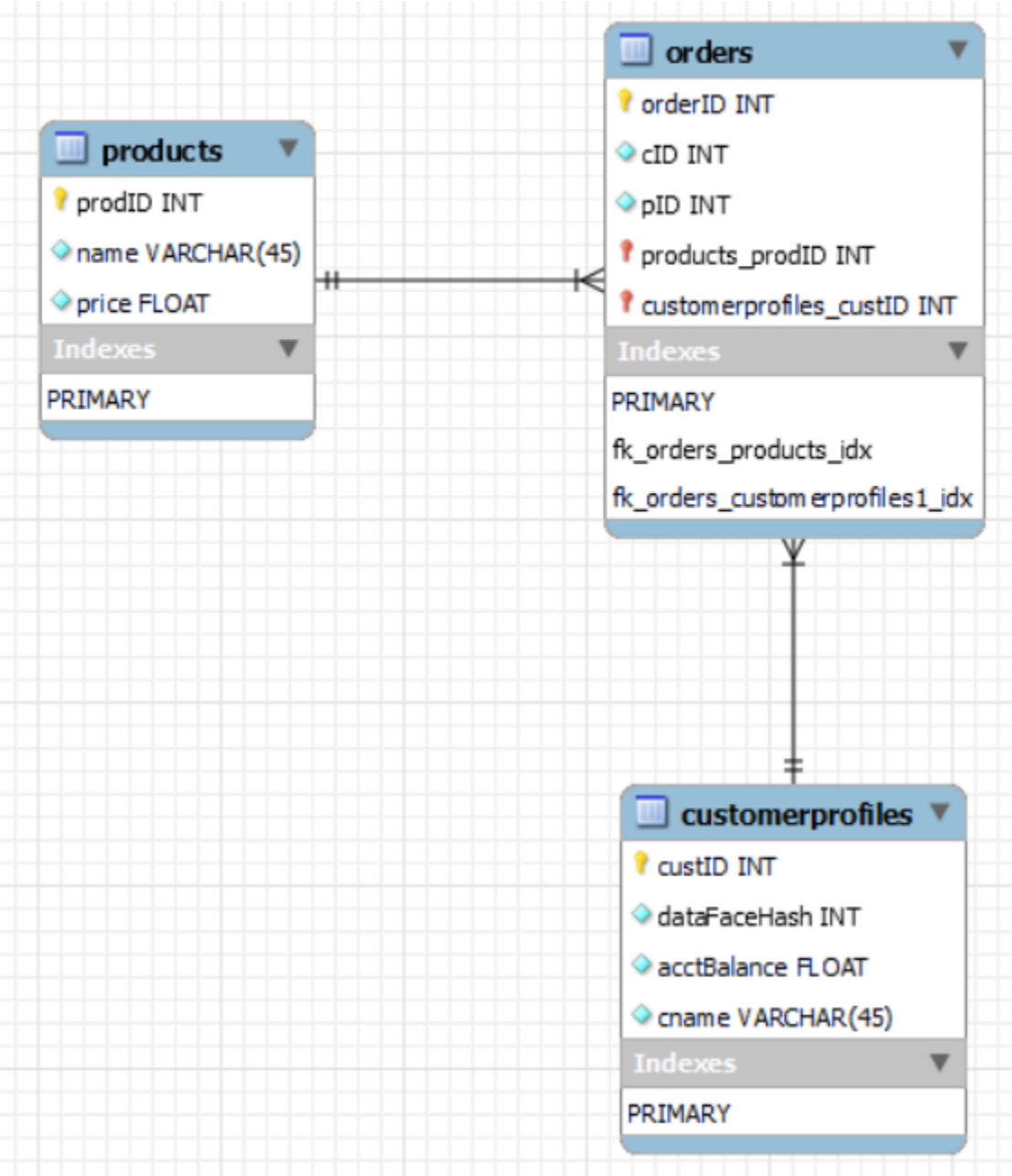
5.0 Restart Option:

- The user can input “cancel” at any time to retrieve any money they had inserted into the machine if they inserted cash
- Return to 1.0 Initialization - Display to the user “Log in OR Insert Funds”

Sequence Diagram



E.R.D.


products
prodID INT
name VARCHAR(45)
price FLOAT
Indexes
PRIMARY
orders
orderID INT
cID INT
pID INT
products_prodID INT
customerprofiles_custID INT
Indexes
PRIMARY
fk_orders_products_idx
fk_orders_customerprofiles1_idx
customerprofiles
custID INT
dataFaceHash INT
acctBalance FLOAT
cname VARCHAR(45)
Indexes
PRIMARY

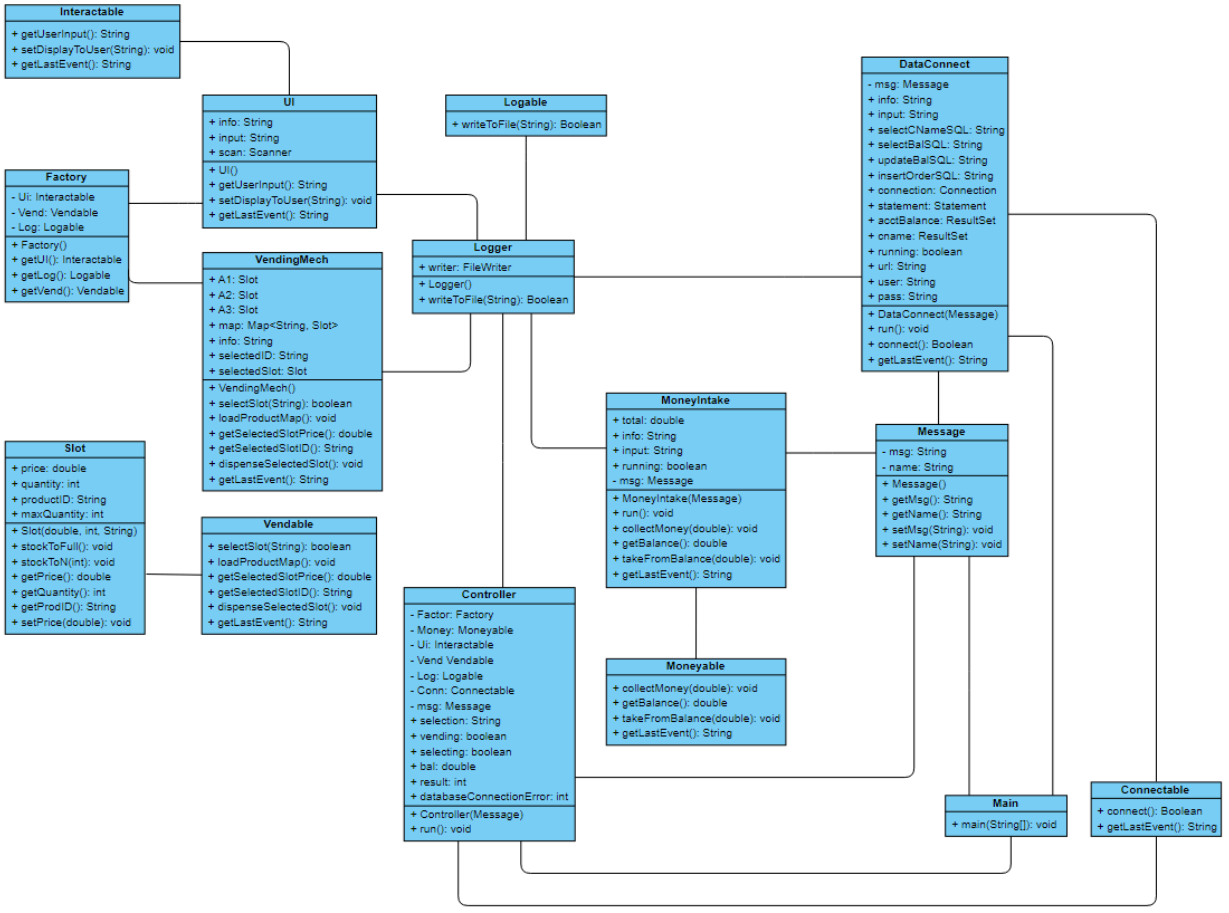
Populated Tables

prodID	name	price
11111111	Coke	2.99
22222222	Pepsi	4.99
33333333	Sprite	2.79
NULL	NULL	NULL

orderID	cID	pID
1	2	11111111
2	2	22222222
3	2	22222222
4	2	22222222
5	1	11111111
6	2	11111111
7	2	11111111
8	1	22222222
9	2	22222222
NULL	NULL	NULL

custID	dataFaceHash	acctBalance	cname
1	111	5.01	Jerry
2	222	4.02	Gary
NULL	NULL	NULL	NULL

Class Diagram



Program Structure

Main Program:

- Initializes main
- Synchronizes “Controller”, “MoneyIntake”, and “DataConnect” threads with shared message object
- Starts “Controller”, “MoneyIntake”, and “DataConnect” threads

Program Functions:

- getLastEvent() - return info set by other methods
- dispenseSelectedSlot(String slotID) - dispense product selected by user inputted slotID
- getSelectedSlotQuantity(String slotID) - return selected slot quantity
- getSelectedSlotPrice(String slotID) - return selected slot price
- getSelectedSlotID() - return selected slot ID
- selectSlot(String slotID) - validate user input is a valid slot ID
- VendingMech() - calls loadProductMap method upon creation of class
- setDisplayToUser(String display) - displays local variable ‘display’ information to user on vending machine display
- getUserInput() - initializes and calls a scanner to retrieve user input
- Slot(double price, int quantity, String productID) - receive created slot information to allow for reading or writing of custom Slot
- stockToFull() - change quantity of slot to full (Not yet implemented)
- stockToN(int n) - change quantity of slot to a specific number (Not yet implemented)
- getPrice() - return price of selected Slot
- getQuantity() - return quantity of selected Slot

Program Structure Cont.

- getProdID() - return ID of selected Slot
- setPrice(double amount) - change the price of a slot to a specific number (Not yet implemented)
- collectMoney(double money) - add user inputted cash value to total balance in vending machine
- getBalance() - return total balance in vending machine
- takeFromBalance(double money) - subtract an amount from total balance in vending machine
- writeToFile(String info) - write info to local file log.txt
- Factory() - allow the sub-classes to choose the type of objects to create
- getUI() - return class Ui
- getMoney() - return class Money
- getLog() - return class Log
- getVend() - return class Vend
- getConn() - return class Conn
- connect() - connect to MYSQL database

UI Summary

- Display to user "Insufficient Funds"
- Display to user "Balance: $#.##"
- Display to user "Price: $#.##"
- Display to user "Returning Balance: $#.##"
- Display to user "Invalid ID OR Product Out of Stock"
- Display to user "Change: $#.##"
- Display to user "Dispensing"
- Display to user "Returning Balance: $#.##"
- Display to user "Make a selection"
- Display to user "Balance: $#.##"
- Display to user "Welcome (Name of user)"
- Display to user "Log in OR Insert Funds"

Error Handling

- msg.wait() can cause an Interrupted Exception and will be caught if presented
- Double.parseDouble(msg.getMsg()) can cause a Number Format Exception and will be caught if presented
- Thread.sleep() can cause an Interrupted Exception and will be caught if presented
- FileWriter() can cause an IO Exception and will be caught if presented
- writer.write() can cause an IO Exception and will be caught if presented
- t3.join(), t1.join(), and t2.join() can cause an Interrupted Exception and will be caught if presented
- statement.executeUpdate() can cause an SQL Exception and will be caught if presented
- cname.next() can cause an SQL Exception and will be caught if presented
- statement.executeQuery() can cause an SQL Exception and will be caught if presented
- cname.getString() can cause an SQL Exception and will be caught if presented
- acctBalance.getString() can cause an SQL Exception and will be caught if presented
- acctBalance.next() can cause an SQL Exception and will be caught if presented
- connection.createStatement() can cause an SQL Exception and will be caught if presented
- DriverManager.getConnection() can cause an SQL Exception and will be caught if presented

Development Challenges and Current Issues

- Developing the sequence diagram for this program has been the most difficult trial throughout this entire process. It is far from perfect as it currently is and I do not wish to revisit it anytime soon.
- The JOptionPanes do not currently properly handle empty string inputs.
- The JOptionPanes have racing and priority issues.
- Developing the program with the use of threading seems to have caused more issues than it has resolved throughout the development process.
- The program may contain too many comments and many of the redundant ones should most likely be removed.

Future Enhancements

- Handle empty string inputs in JOptionPanes
- Implement facial recognition to properly execute face hash ID system
- Either remove threading or figure out a method to allow input on both JOptionPanes without having to cancel the first one
- Implement a visual representation of the vending machine to show the user the items and slots
- Add debit/credit as a payment type
- Either remove or implement methods: stockToN(int), stockToFull(), setPrice(double)
- Implement getLastEvent(String) into its own class with its own interface that all classes who might need access can access.
- The passing of messages between threads using the Message class could be better executed with an additional variable attached.

Conclusion

The Vending Machine program provides a foundational framework for creating a simulated vending machine experience. With the inclusion of a database the information that has been and will be stored can be used in a manner that benefits the owner of the vending machine and the product owners/distributors to further enhance business and provide new opportunities.