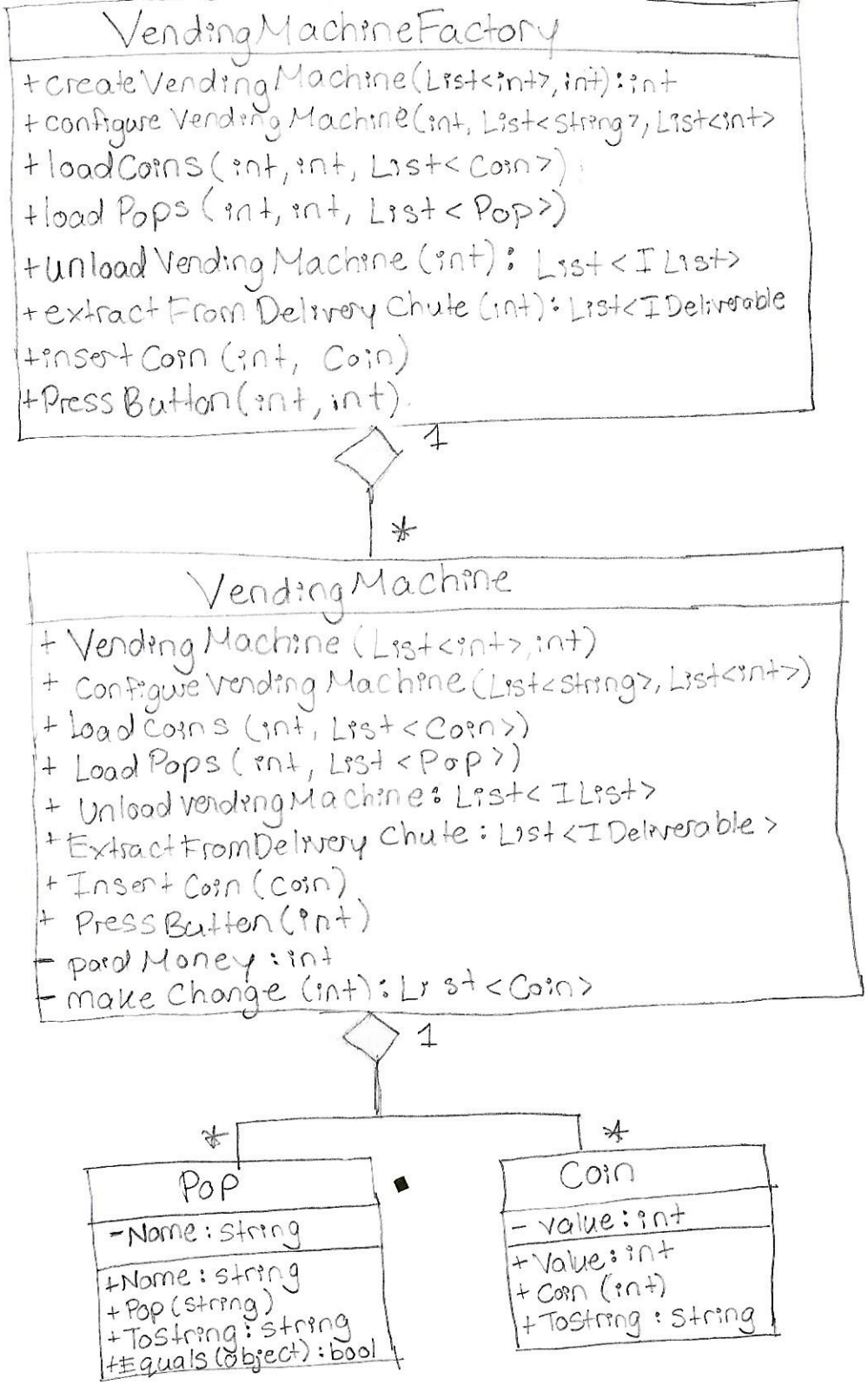
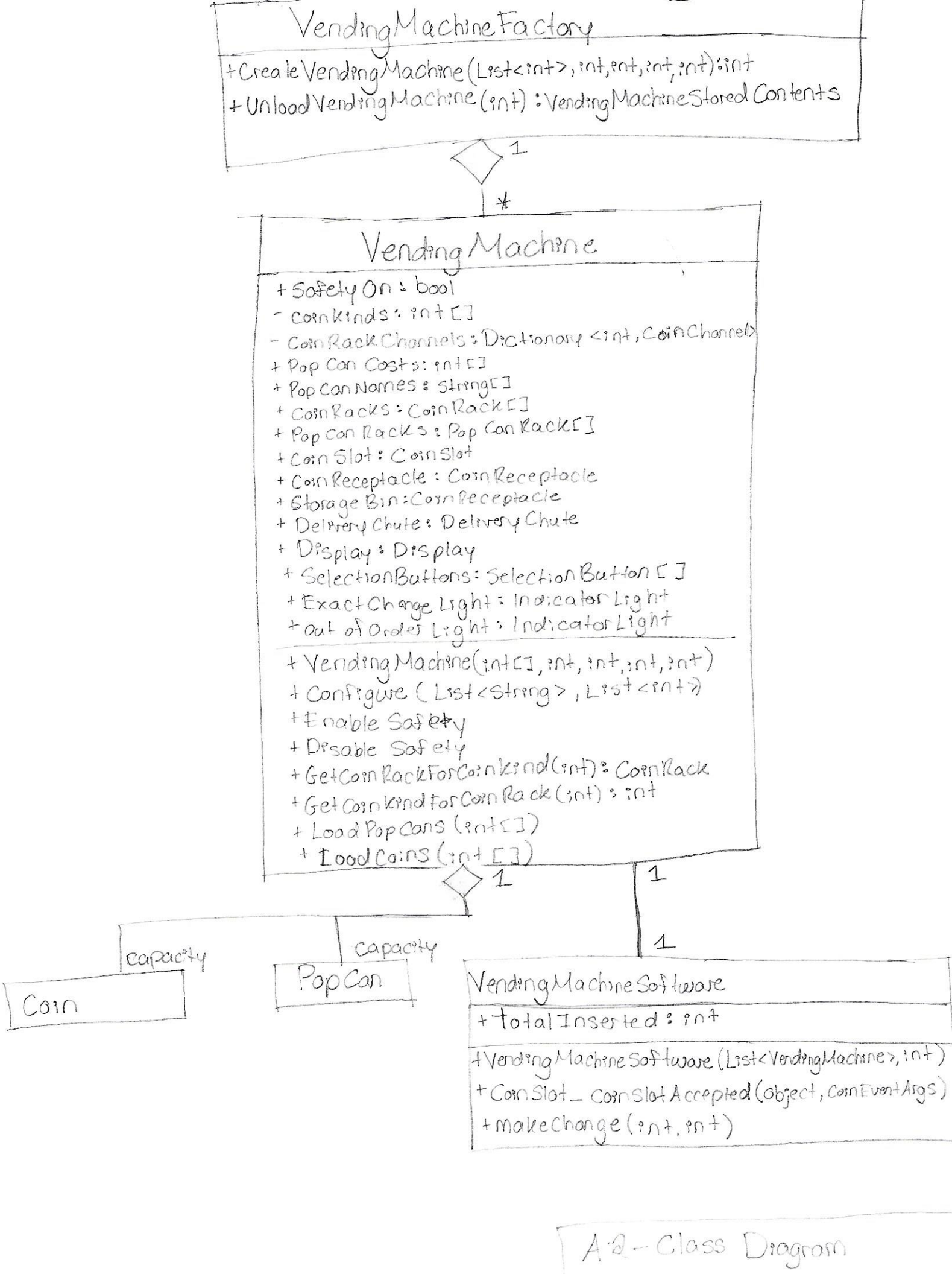
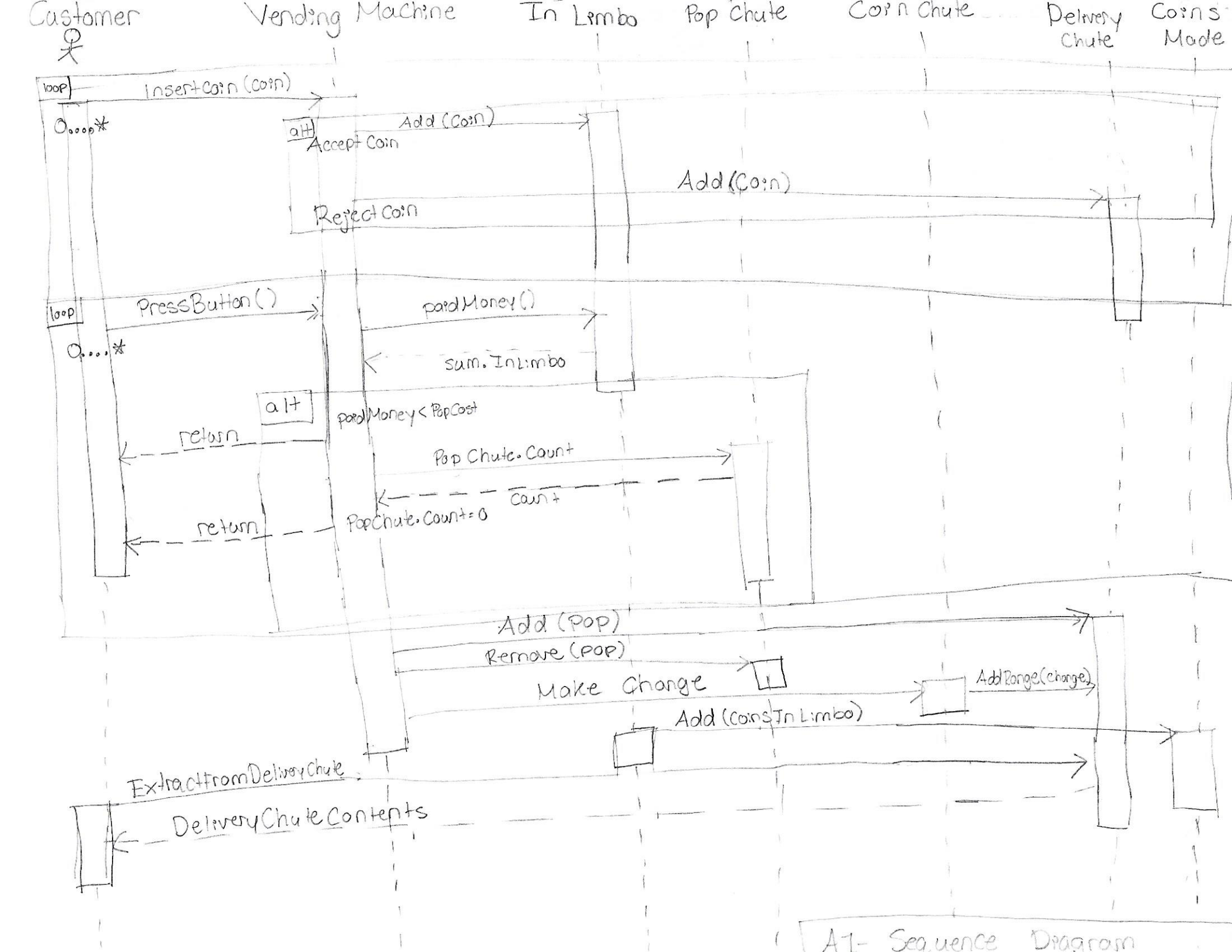
A1-Class Diagram

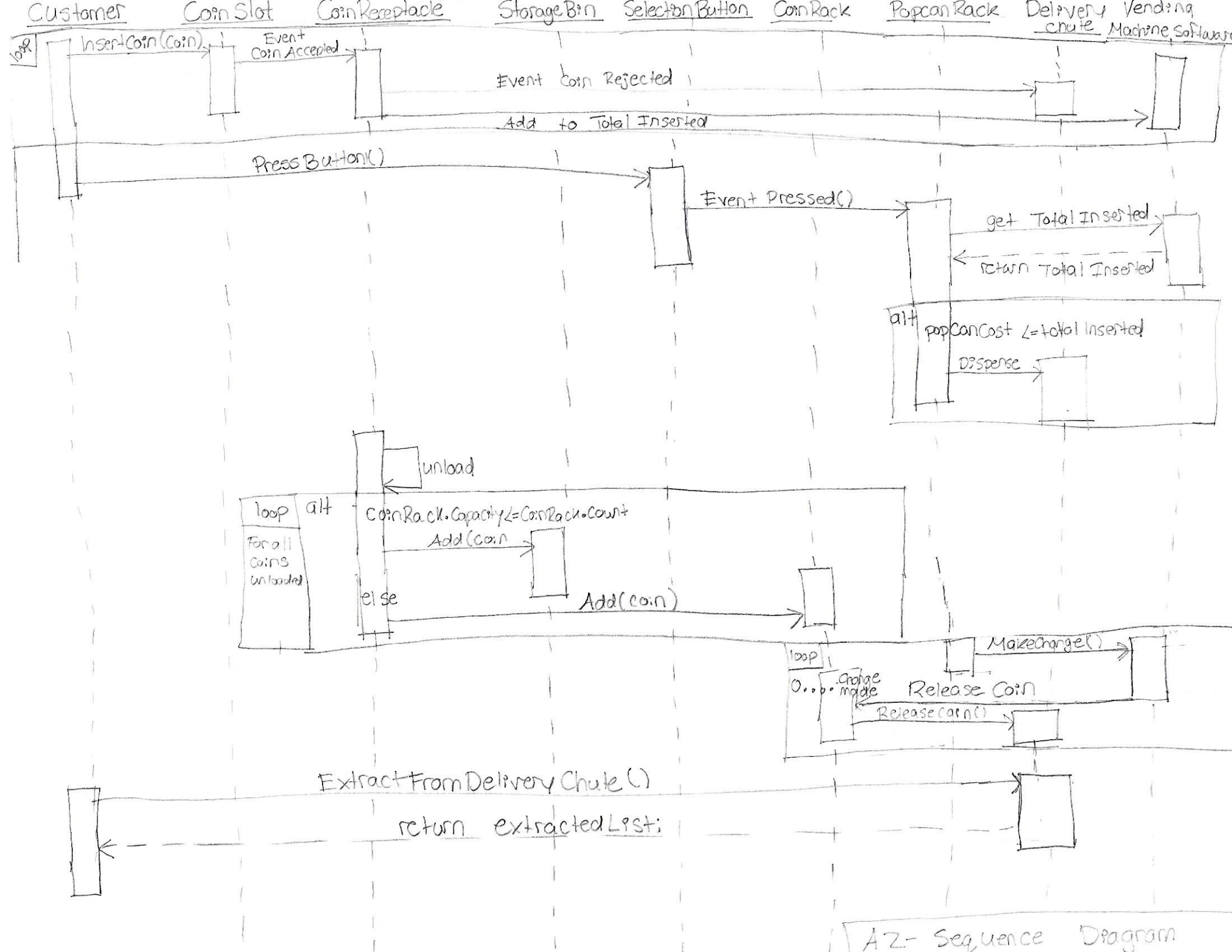


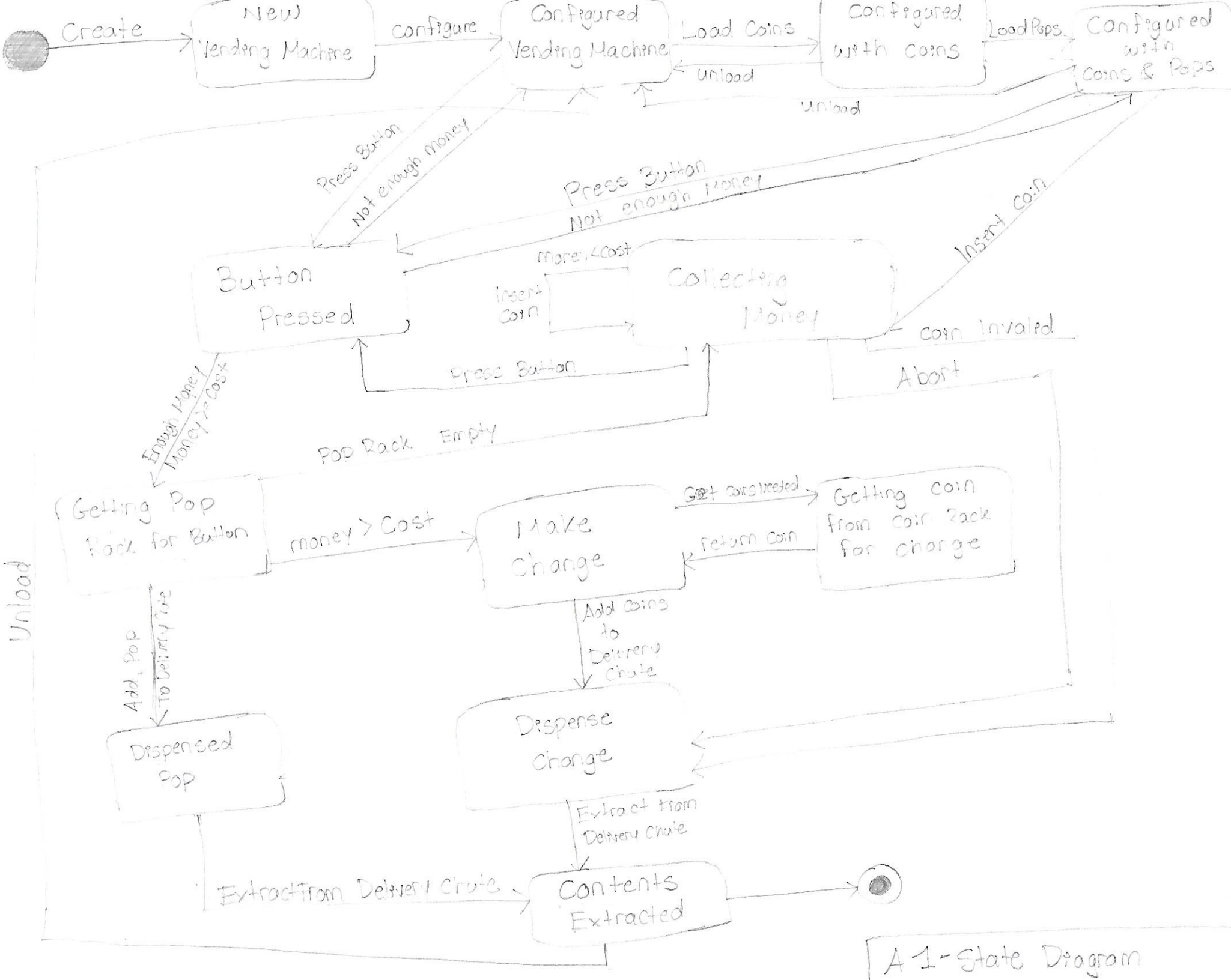


A2 - Class Diagram



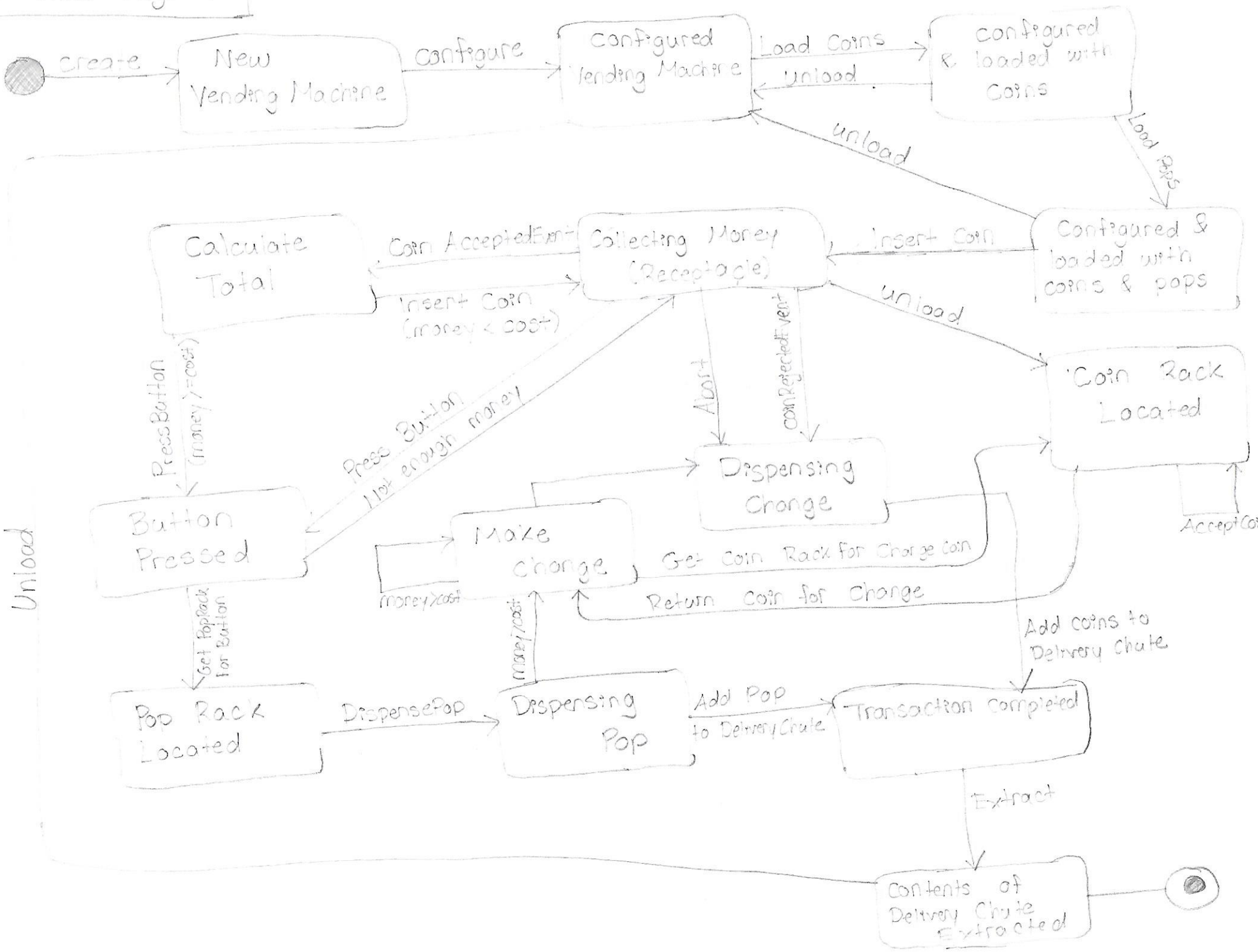
A1- Sequence Diagram


Customer
Coin Slot
Coin Receptacle
Storage Bin
Selection Button
Coin Rack
Popcan Rack
Delivery chute
Vending Machine Software
loop
InsertCoin(Coin)
Event Coin Accepted
Event Coin Rejected
Add to Total Inserted
PressButton()
Event Pressed()
get Total Inserted
return Total Inserted
alt
popCanCost <= total Inserted
Dispense
unload
loop
For all coins unloaded
alt
CoinRack.Capacity <= CoinRack.Count
Add(coin)
else
Add(coin)
loop
0...9 change made
MakeChange()
Release Coin
ReleaseCoin()
ExtractFromDeliveryChute()
return extractedList;
A2- Sequence Diagram



A 1-State Diagram

A2 - State Diagram



Lamess Kharfan
SENG 301- A2

Diagrams Explanations

Class Diagrams:

A1 Class Diagram: The A1 class diagram shows the hierarchy of how a vending machine factory has 0 to infinitely many vending machines, and how each vending machine has from 0 to infinitely many coins and pops, since there are no capacity parameters on the pop and coin chutes in A1.

A2 Class Diagram: Only the differences between A1 and A2 are shown in the UML Classes in the diagrams. Vending Machine Factory on the A2 diagram only shows 2 methods, create and unload, because they are the only ones that differ from the Vending Machine Factory in A1, therefore it can be assumed that any of the remaining methods in Vending Machine Factory on the A1 diagram that are not shown can be assumed to be present with the exact same signature in A2.

Parser, Script Processor and Analyzer are not shown in the class diagrams because they differ very little between the two assignments, and so the need to include them on the diagrams felt to be wasteful of space. The only true differences would be in how the unload method of A2 returned contents and how the create method is used, with an extra 3 parameters for capacities. For the purposes of abstraction, they have been left out. If there were on the diagram though, Script processor would be on the top, and would be associated with analyzer and parser and vending machine factory using just a line, and the script processor only has one of each of these.

Sequence Diagrams:

The differences between the two would be the that in A1, our “customer”, which is our script, interacts with the vending machine object mainly, and the vending machine object takes care of how each interaction is handled using its software and hardware. In A2, the customer can directly interact with the vending machines hardware components, so instead of inserting a coin into the vending machine object, the customer inserts the coin into the coin slot of the vending machine, and the only thing done by the vending machine itself is drive the software components based on what a customer does with the hardware components.

The sequences diagrams model a customer interaction with the vending machine. It is possible to also model the Vending Machine Factory being created by a vending machine factory owner for example, as well as a technician loading pops and coins, but instead I chose to only model an interaction with the main system, the vending machine, and the customer. Although many methods are not using in this interaction, such as configure, create, load pops, load coins, and unload, it does still show the main interaction and function of a vending machine.

State diagrams:

The state diagrams mainly model a customer interaction, but also do go back as far as creation and configuration of the vending machine. This does not show the creation of the vending machine factory. Instead it focused on the states of only one system, the vending machine. It is possible to show states of coins and pops also, such as when coins are “in limbo” in A1, however this defers from focusing on the states of the vending machine itself, and so they have been left out. A1 and A2 are very similar in the states they can be in, the main difference being that transitions are sometimes event based in A2, and in A1 they are always based on the calling of methods, and the results of what happens in the methods.