Cody Luhmann

2:

Project Description:

My project is basically an Online Skateshop, one where people can select certain types of decks, boards, wheels, and trucks. It implements the memento design pattern at the end of the selection asking if the user wants to go back and select an entirely new board and parts, or if they just want to select new parts.

3:

User Features:

ID:	Requirements:	Responsibilities:
UC-01	User is able to select board and parts	System keeps track of customers board and parts, and only allows one deck, truck, and wheel part to be selected for one board
UC-02	User is able to browse inventory	System displays all inventory for decks, trucks, and wheels
UC-03	User can go back to a previous state in order to re choose his board or parts	System uses memento to keep track of customers certain states, so that they can re choose their board or their parts

System Features:

ID:	Requirements:	Responsibilities:
SF-01	System displays customers order at certain points	Keeps track of the customers order throughout the program and displays each option that they chose right after they pick it.
SF-02	System uses mementos	Creates mementos of certain states so that the user can go back and select a new board or new parts
SF-03	System checks for a valid credit card	Checks if a credit card is valid or not, this code was used from geeksforgeeks.com all credit is given toward them.

Cody Luhmann

4:

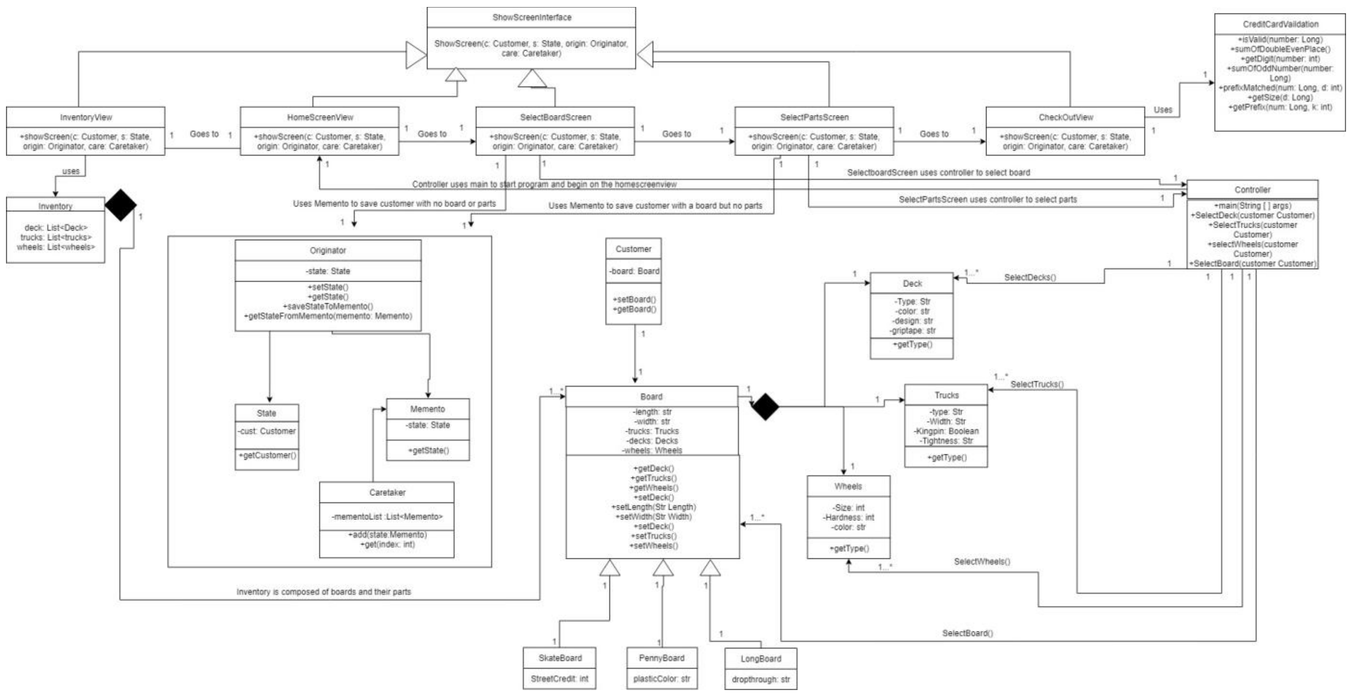
Features not implemented:

ID:	Requirements:	Responsibilities:
UC-04	User can browse a database (Database was not implemented)	System will use a database to keep track of the inventory
SF-04	Different boardtypes have different parts.(Not Implemented)	Based on the board selected, would limit the options you have for each type of board, so a skateboard would have different parts to select than a longboard.

5:

Many things changed in my class diagram, for starters I know included a controller as well as all my views for certain types of screens. All these screens implements the interface for showscreen in order to show what they need to. There is also added visibility modifiers as well since I forgot those in my previous class diagram. Another thing added was my memento design pattern, which is used in certain views in order to capture the state of the customer. I did not have all these things in my previous class diagram simply because I did not really know what I would end up using as a design pattern and my in experience with creating a class diagram before this semester.

This is a much more zoomed out version of my Class Diagram, but don't worry I will be zooming in to show you where my design pattern was implemented.

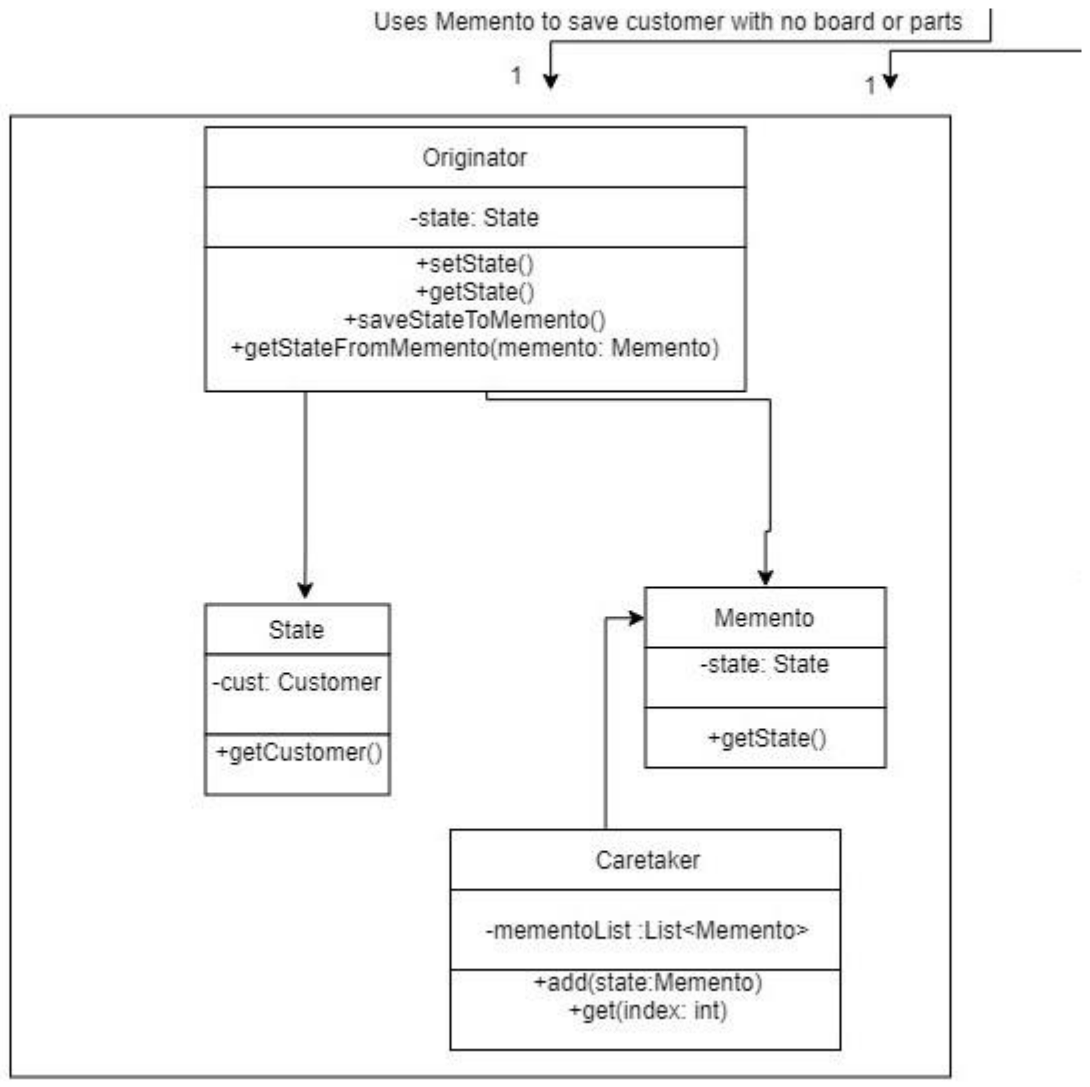


Cody Luhmann

Continue down to see zoomed in photos.

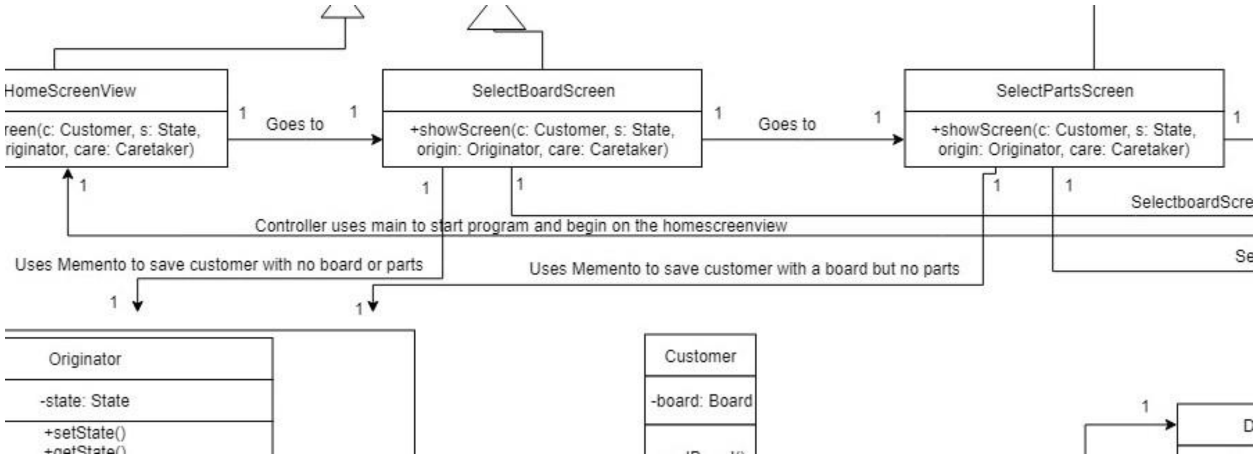
6:

Here is the memento design pattern which uses the current state of my program which contains the customer and his board. I used this so that the customer could rethink his order if he decides to do so when he is at checkout. That way if he does not like some aspect of his order he can change it! The moments where memento is being used is inside the select board screen in order to keep track of the customer but with no board or parts selected, and its being used inside the SelectPartsView in order to save the state of a customer that has a board selected but not parts selected. Then, in checkoutview is where the memento design pattern recalls the past states depending on if the user wants to go back and select everything or just go back and select his parts.



Cody Luhmann

You can see below that SelectboardScreen and SelectPartsScreen both use memento in order to create saved states of the customer. The Checkoutview then calls the caretaker in order to find the states and pass the customer through the showScreen method so that the customers order reverts back to nothing or reverts to just having the board.


HomeScreenView
reen(c: Customer, s: State, riginator, care: Caretaker)
Goes to
SelectBoardScreen
+showScreen(c: Customer, s: State, origin: Originator, care: Caretaker)
Goes to
SelectPartsScreen
+showScreen(c: Customer, s: State, origin: Originator, care: Caretaker)
SelectboardScre
Controller uses main to start program and begin on the homescreenview
Uses Memento to save customer with no board or parts
Uses Memento to save customer with a board but no parts
Originator
-state: State
+setState()
Customer
-board: Board
1

7:

I have learned a lot through this course project, as I had no prior experience with Java before this class. I now know about class diagrams as well as sequence, activity, and user case diagrams. I know that in the future I should start with making these diagrams before I try anything too greedy in my code. That is partly why my first class diagram was so bad, I did not really plan ahead for what was to come, now I know better. I also learned about the mvc framework and how useful it is to separate your program into these three components. In fact the biggest thing I have learned from this process is how much time is needed to truly complete a project. I thought I was going to be able to implement everything I set out to do, but once I started to get stuck on “simple” things it became apparent that I would need a lot more time to implement everything I wanted. Next time I will be much more prepared now knowing what it takes to design a system like this!