Steven Valero

Joel Diaz

Java Report

Some of the patterns that I used in this project were composite, iterator, observer, and decorator. The composite is based on the fact that I was using Jframes. The iterator came into play when I created a Jlist and had to iterate through the memory of data to grab all the price values. The observer exists when the code waits for a user to click on a certain Jbutton and then an action happens. Finally, the decorator comes in from the fact that I used borders to separate my Jframes.

Next, i have to talk about my code. Well I figure that is what the presentation is for but I can attempt to talk a little about my code on paper. Essentially I used the NetBeans swing UI which lets me build a very nice user interface without too much hassle. I started with just setting up the UI and then figuring out the code later. Once I had the UI how I wanted it I started thinking about what each button should do. First I made certain text field uneditable, meaning I do not want the user to be able to change them since it would not make sense. Next I thought about the increment or decrement buttons. If a user hits one of these buttons the quantity counter should change going in either direction but if the user attempts to go below zero then it would just stay at zero. You can’t have a negative amount of an item. At the same time if the increment or decrement buttons got pressed the price of that item would have to adjust and so I added that to code. Once I had the increment and decrement buttons working I had to think about how to add an item to some kind of list, like a cart. Well I came up with the idea of a jList which lets me added or remove items in a list. When the button is pressed to add an item to the cart it adds a String to the jList which is implemented using a DefaultListModel which is similar to a Vector or ArrayList. I can literaly just add the String to the Model. As soon as I add the String to the model it appears on the list. Okay so now I have my list populated but what if a user wants to remove an item from the list that they accidently put on their. I then added a Remove item button which allows you to select an item from the list and remove it. All the button does is grab the index of the selected item of the list, which also corresponds to the location of where this item is stored in the Model and removes it. Now I am down to the final stretch which is just a checkout button. Now I could not think of a unique way of coming up with a check out button so I just made a small text field that had the price of what you would have to pay at the end. The checkout button goes through all the data that the model has and parses the String, since the model only contains full String, for only the price and adds them up together at the end resulting in the final price.

Finally, the experience with this final project was interesting. I attempted to go about this project so many different ways but at the end I had to just go with what was easier. I did not have enough time to attempt anything to fancy I’m still learning java. A lot of what I learned was through using Google. I had fun working on this final project , if I had more time and less classes to deal with I would of definitely went for a better look but hey this is what I got for now.