3.46 Planning

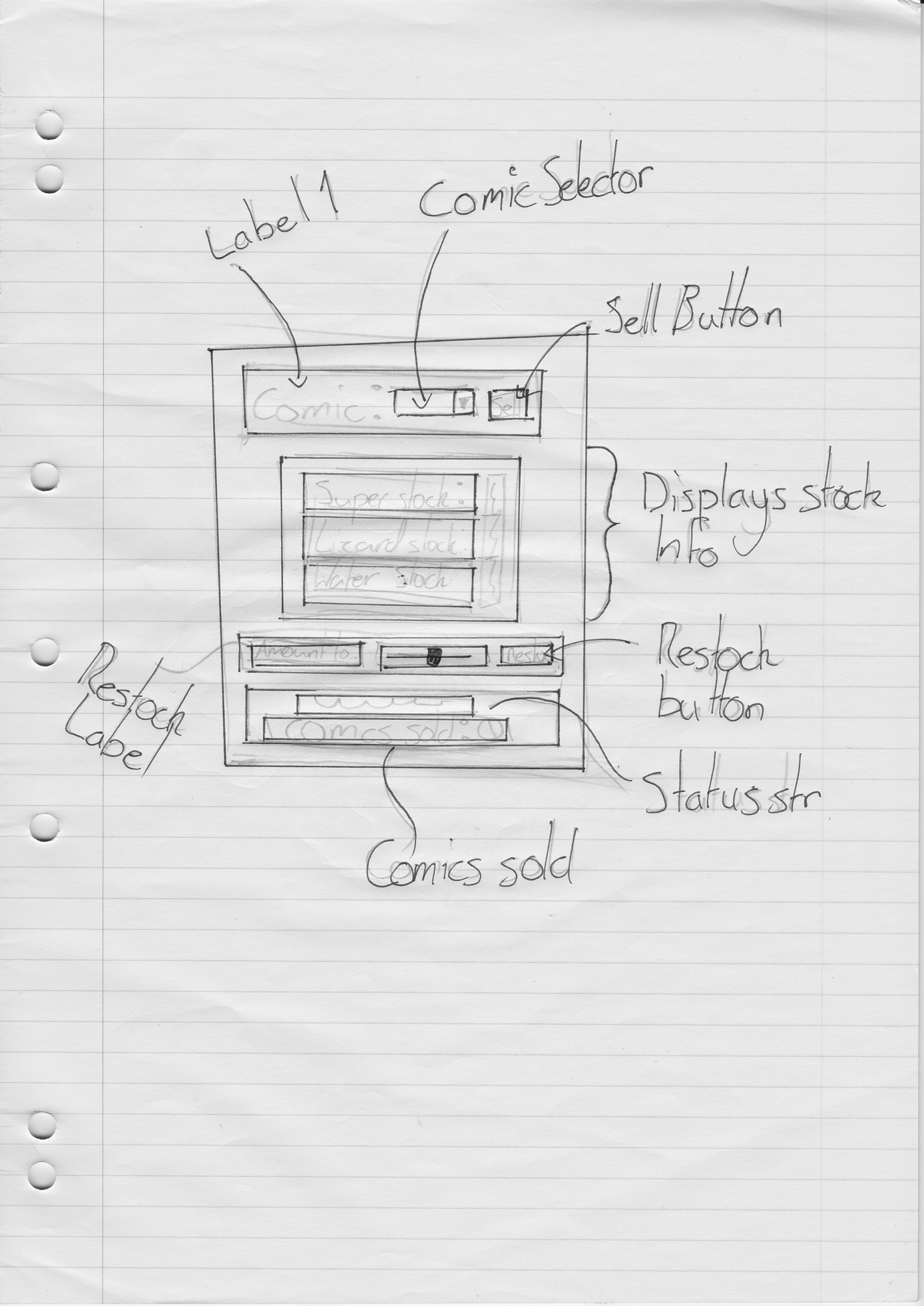
**User inputs:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name:** | **Input Type:** | **Input Method (widget)** | **Reason For Using the chosen input method** |
| Sell Button | Click (run function) | Button | Because a button is the easiest way to trigger a command |
| Comic Selector | Option select (string) | Option box | This way the user only has a preset number of choices, they cannot choose comics they do not have. It removes human errors like spelling mistakes and it’s easy to use/understand |
| Restock Button | Click (run function) | Button | Because a button is the easiest way to trigger a command |
| Restock input | Integer | Input | This allows the user to input whatever number they want. It also has boundary values so the user cannot enter negative numbers or decimals or strings. |

Information Displays Needed:

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Information Type | What will it display? | Widget Used? |
| Self.label | String | Each comic’s name and stock. It’s stored in the comic class | Label |
| Status\_str | String | Wether or not the last action the user tried to do was successful. If not it will display an appropriate error message | Label |
| Restock\_label | String | “Amount to restock: “ | Label |
| Amount\_sold | Integer | The amount of comics sold | Label |
| Amount\_sold\_text | String | “amount of comics sold: | Label |
| Label1 | String | “comic: “ | Label |

**Design interface sketch:**



**Classes Required:**

Class: Comic

This class is used to make each of the comics, it stores things like the amount of stock and runs the sell and restock functions.

**Constraints or existing Data:**

The user will have to enter values into the data.txt file when they first run the program so it display’s their desired comics/stocks. After that the autosave features will allow the program what the user’s previous values were.

**Indexed Data Structures**

I have a dictionary (comics\_dict) that stores the name of the comic and its’ instance of the comic class.

**What calculations are nessescary?**

* How many comics have been sold: comics sold += 1 whenever sell button is clicked
* Can the user sell this comic?: check that the chosen comics stock is > 0
* Restock amount: chosen comic’s amount + restock slider value
* Sell a comic: chosen comic’s stock – 1

**Functions Used**:

Sell(): (note this is contained within the comic class)

Check that’s its stock > 0, if it is:

Minus 1 from its stock and update any labels

Set the status string to reflect the successful transaction

Update the comics sold value

Else:

Set the status string to notify the user of the failed attempt and the reason for the failure

Restock(): (note this is contained within the comic class)

Get the value from the restock slider

Add the restock slider value to the chosen comic’s stock

Update the chosen comics stock label

Update the status string to reflect the successful transactions

**Test cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable being tested** | **Input** | **Expected Result** | **Result** | **Reason for test** | **Pass/Fail** |
| Stocks | Sell button when stock is at 0 | Error shown in status string | Error shown in status string | Too check that thee user cannot sell stock the do not have | Pass |
| Sell button when stock is not at 0 | Stock goes down by one for chosen comic, comics sold label is updated to reflect the new sale and the status string reflects the successful attempt | Stock goes down by one for chosen comic, comics sold label is updated to reflect the new sale and the status string reflects the successful attempt | To check the sell function works correctly | Pass |
| Comic Option box | Change the selected comic and try selling/ restocking | Updates the correct values for the correct comic | Updates the correct values for the correct comic and staus string reflects the changes | To check the option selector box return the correct values for the restock and sell functions. And to. Check that the status string updates on restock attempts | Pass |

**Images of testing:**

|  |  |
| --- | --- |
|  | Checking that the user cannot sell stock they do not have.  The stock did not change and the status string showed an easy to understand error message. In this example no matter how many times the sell button was clicked the super dude stock did not decrease because it was at 0. |
|  | Checking that everything updates when when a comic is sold:  The correct stock (water woman) was sold, it went from 12 (in prev screen shot) to 1. Also the status\_str successfully updated to show it was a successful sell. And the number of comics sold also updated correctly. |
|  | Checking that when the selected comic changes, the correct values are updated.  The comic was changed to super dude and 124 stock was added (it was at 0.)  The stock was updated, the status\_str was updated as well. Nothing else updated when it was not supposed to |
|  | Check that the restock input/button work correctly:  12 stock was added to the lizard man comic. It’s stock label went from 9 to 21, the math was correct. Also the status\_str updated successfully. |
|  | Check that the comic’s stock is saved in-between closes.  As you can see from the terminal window the code was closed and re-opened. Also the. Status string says “start-up successful!” so you can tell the code has just been opened. All the stocks match what they were when the code was closed (check screen shot) so therefore it all works successfully. |