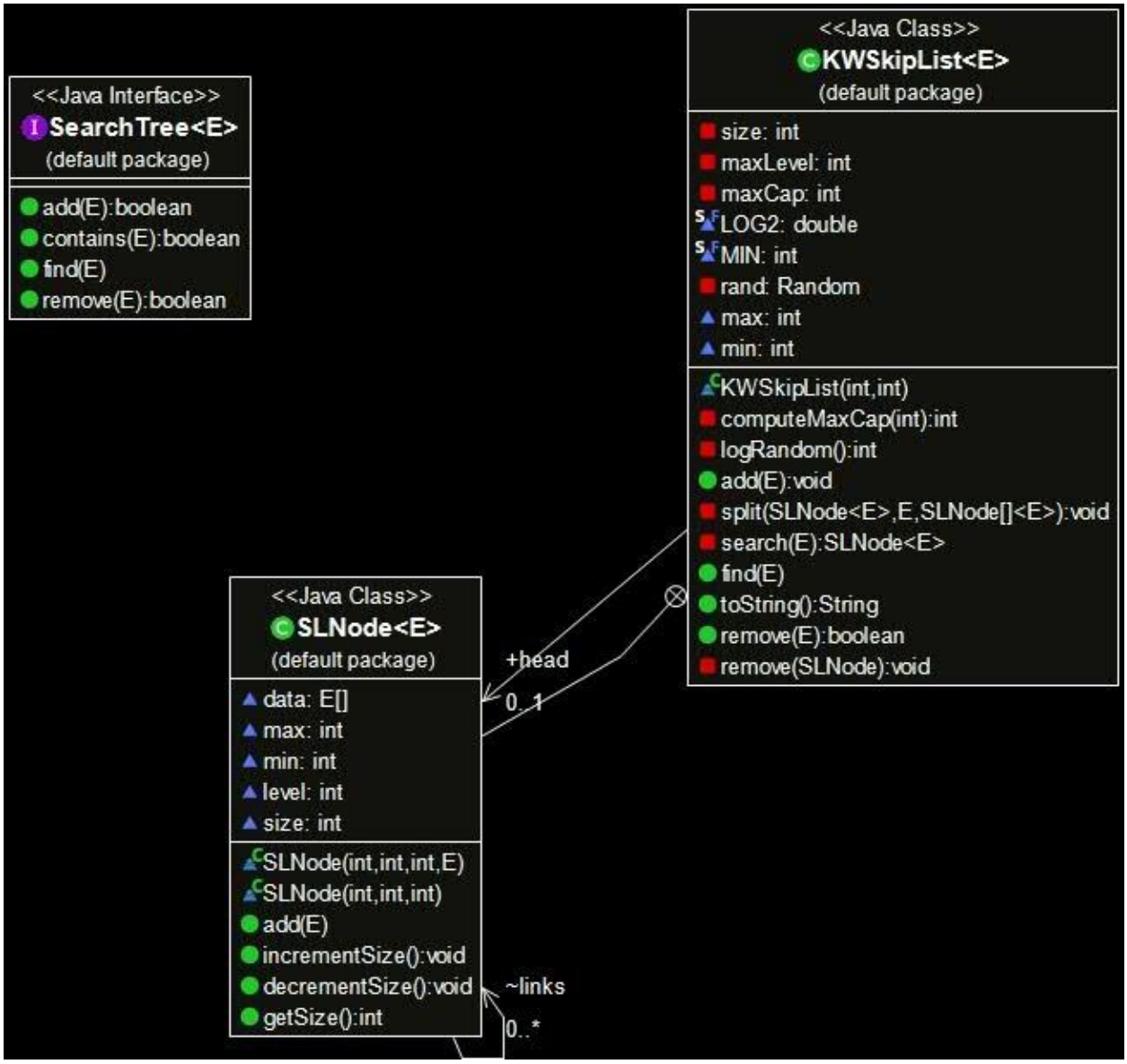
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 7 Report

161044063

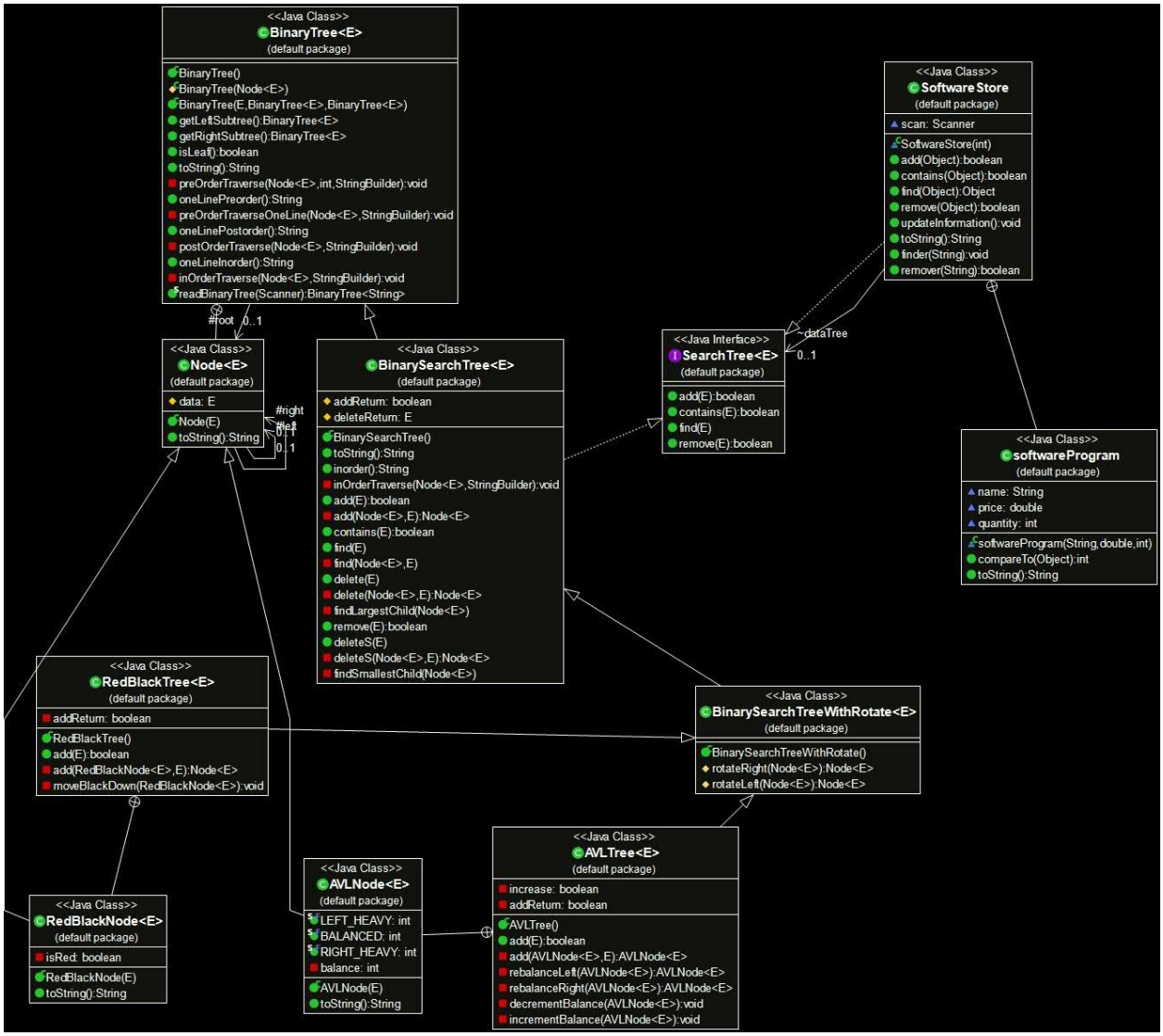
ZAFER ALTAY

1-)CLASS DIAGRAM

FOR PART2:



FOR PART4:



2-)TEST CASES

****Comparisons of Part3 will be discarded as separate files.**

For SkipList:

TEST ID	T.SCENERIO	TEST DATA	EXPECTED R.	PASS/FAIL
T01	**create list, add a few elements and print them**	**add 40,30,20,50,90, 60 sequently**	**20 in1th Node and the node has3 elements 30 in1th Node and the node has3 elements 40 in1th Node and the node has3 elements 50 in2th Node and the node has3 elements 60 in2th Node and the node has3 elements 90 in2th Node and the node has3 elements**	**PASS**
T02	**add a few more elements for controlling split**	**add (1,11,18,16,111, 123) sequently**	**1 in1th Node and the node has3 elements 11 in1th Node and the node has3 elements 16 in1th Node and the node has3 elements 18 in2th Node and the node has2 elements 20 in2th Node and the node has2 elements 30 in3th Node and the node has2 elements 40 in3th Node and the node has2 elements 50 in4th Node and the node has3 elements 60 in4th Node and the node has3 elements 90 in4th Node and the node has3 elements 111 in5th Node and the node has2 elements 123 in5th Node and the node has2 elements**	**PASS**

T03	**add same elemnt for duplicate test**	**add 40**	**1 in1th Node and the node has3 elements 11 in1th Node and the node has3 elements 16 in1th Node and the node has3 elements 18 in2th Node and the node has2 elements 20 in2th Node and the node has2 elements 30 in3th Node and the node has2 elements 40 in3th Node and the node has2 elements 50 in4th Node and the node has3 elements 60 in4th Node and the node has3 elements 90 in4th Node and the node has3 elements 111 in5th Node and the node has2 elements 123 in5th Node and the node has2 elements**	**PASS**
T04	**delete a few elements and control changing node num**	**delete 123,16,50,31,11**	**1 in1th Node and the node has1 elements 18 in2th Node and the node has2 elements 20 in2th Node and the node has2 elements 40 in3th Node and the node has1 elements 60 in4th Node and the node has2 elements 90 in4th Node and the node has2 elements**	**PASS**

			111 in5th Node and the node has1 elements	

For SoftwareStore:

TEST ID	T.SCENERIO	TEST DATA	EXPECTED R.	PASS/FAIL
T0	Automatic item creation and screen printing	Adobe Photoshop 6.0 Adobe Photoshop 6.2 Norton 4.5 Norton 5.5 Adobe Flash3.3 Adobe Flash 4.0	Adobe Photoshop 6.0 Adobe Photoshop 6.2 Norton 4.5 Norton 5.5 Adobe Flash3.3 Adobe Flash 4.0	PASS
T1	Delete one of the items and show the remaining	Adobe Photoshop 6.2	Adobe Photoshop 6.0 Norton 4.5 Norton 5.5 Adobe Flash3.3 Adobe Flash 4.0	PASS
T2	adding a new item	Name: My Program Price:45,99 Quantity: 2	Adobe Photoshop 6.0 Norton 4.5 Norton 5.5 Adobe Flash3.3 Adobe Flash 4.0 My Program	PASS
T3	Search by name	Norton 4.5	Norton 4.5	PASS
T4	change price	Norton 4.5 price: 99,99	Norton 4.5 price: 99,99	PASS
T5	increased quantity	Adobe Flash 4.0 quantity:1 add +10	Adobe Flash 4.0 quantity:11	PASS

T6	**reduced quantity**	**Adobe Flash 4.0 quantity:11 delete :15**	**The app is not in stock**	**PASS**

3-) Running command and results

For KWSkipList:

```
First I set a skipList with a maximum of 3 and a minimum of 1
Now, we add elements in an unordered manner, these will be in the middle of the list, between and to the beginning.(T01)
Now we print them on the screen

20  in1th Node and the node has3 elements
30  in1th Node and the node has3 elements
40  in1th Node and the node has3 elements
50  in2th Node and the node has3 elements
60  in2th Node and the node has3 elements
90  in2th Node and the node has3 elements
```

```
We add a few more elements to see the change of node numbers(T02)
1  in1th Node and the node has3 elements
11  in1th Node and the node has3 elements
16  in1th Node and the node has3 elements
18  in2th Node and the node has2 elements
20  in2th Node and the node has2 elements
30  in3th Node and the node has2 elements
40  in3th Node and the node has2 elements
50  in4th Node and the node has3 elements
60  in4th Node and the node has3 elements
90  in4th Node and the node has3 elements
111  in5th Node and the node has2 elements
123  in5th Node and the node has2 elements
```

```
Now I test whether the same element is added or not(T03)
1  in1th Node and the node has3 elements
11  in1th Node and the node has3 elements
16  in1th Node and the node has3 elements
18  in2th Node and the node has2 elements
20  in2th Node and the node has2 elements
30  in3th Node and the node has2 elements
40  in3th Node and the node has2 elements
50  in4th Node and the node has3 elements
60  in4th Node and the node has3 elements
90  in4th Node and the node has3 elements
111  in5th Node and the node has2 elements
123  in5th Node and the node has2 elements
```

```
Now we will delete a few elements,so the node numbers will also change(T04)

1  in1th Node and the node has1 elements
18  in2th Node and the node has2 elements
20  in2th Node and the node has2 elements
40  in3th Node and the node has1 elements
60  in4th Node and the node has2 elements
90  in4th Node and the node has2 elements
111  in5th Node and the node has1 elements
```

For Software Store:

```
First we created a new store
I write the whole tree on the screen to check the packages we created automatically(T0)
App name: Adobe Photoshop 6.0
Price: 25.99 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 3.3
Price: 20.0 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 4.0
Price: 27.5 Dollars
Piece:1 pieces in stock

App name: Adobe Photoshop 6.2
Price: 30.0 Dollars
Piece:1 pieces in stock

App name: Norton 4.5
Price: 15.99 Dollars
Piece:1 pieces in stock

App name: Norton 5.5
Price: 19.99 Dollars
Piece:1 pieces in stock
```

```
Now, by testing the deletion method, I remove photoshop6.2 from the programs.(T1)
I test again to see the results.
App name: Adobe Photoshop 6.0
Price: 25.99 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 3.3
Price: 20.0 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 4.0
Price: 27.5 Dollars
Piece:1 pieces in stock

App name: Norton 4.5
Price: 15.99 Dollars
Piece:1 pieces in stock

App name: Norton 5.5
Price: 19.99 Dollars
Piece:1 pieces in stock
```

```
Now I'm testing the adding method(T2)
Enter price
45,99
How many do you want to add
2
I'm testing to see it added
App name: Adobe Photoshop 6.0
Price: 25.99 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 3.3
Price: 20.0 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 4.0
Price: 27.5 Dollars
Piece:1 pieces in stock

App name: Norton 4.5
Price: 15.99 Dollars
Piece:1 pieces in stock

App name: My Program
Price: 45.99 Dollars
Piece:2 pieces in stock

App name: Norton 5.5
Price: 19.99 Dollars
Piece:1 pieces in stock
```

```
Now I'm testing the search method(T3)
App name: Norton 4.5
Price: 15.99 Dollars
Piece:1 pieces in stock
```

```
First, we change the price of one of the programs and test its new version.(T4)
App name: Adobe Photoshop 6.0
Price: 25.99 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 3.3
Price: 20.0 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 4.0
Price: 27.5 Dollars
Piece:1 pieces in stock

App name: Norton 4.5
Price: 15.99 Dollars
Piece:1 pieces in stock

App name: My Program
Price: 45.99 Dollars
Piece:2 pieces in stock

App name: Norton 5.5
Price: 19.99 Dollars
Piece:1 pieces in stock


Please enter a name
Norton 4.5
What do you want
1-)Add in stock
2-)Remove from stock
3-)Sell
4-)Update price
4
```

```
4
How much money new price of program
99,99
Last status of program
App name: Norton 4.5
Price: 99.99 Dollars
Piece:1 pieces in stock
Now we are looking at the latest version of a program that we have increased quantity(T5)
App name: Adobe Photoshop 6.0
Price: 25.99 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 3.3
```

```
App name: Adobe Flash 3.3
Price: 20.0 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 4.0
Price: 27.5 Dollars
Piece:1 pieces in stock

App name: Norton 4.5
Price: 99.99 Dollars
Piece:1 pieces in stock

App name: My Program
Price: 45.99 Dollars
Piece:2 pieces in stock

App name: Norton 5.5
Price: 19.99 Dollars
Piece:1 pieces in stock


Please enter a name
Adobe Flash 4.0
What do you want
1-)Add in stock
2-)Remove from stock
3-)Sell
4-)Update price
1
How many do you want to add
10
Last status of program
App name: Adobe Flash 4.0
Price: 27.5 Dollars
Piece:11 pieces in stock
```

```
Now we are drastically reducing the number of programs(T6)
App name: Adobe Photoshop 6.0
Price: 25.99 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 3.3
Price: 20.0 Dollars
Piece:1 pieces in stock

App name: Adobe Flash 4.0
Price: 27.5 Dollars
Piece:11 pieces in stock

App name: Norton 4.5
Price: 99.99 Dollars
Piece:1 pieces in stock

App name: My Program
Price: 45.99 Dollars
Piece:2 pieces in stock

App name: Norton 5.5
Price: 19.99 Dollars
Piece:1 pieces in stock


Please enter a name
Adobe Flash 4.0
What do you want
1-)Add in stock
2-)Remove from stock
3-)Sell
4-)Update price
2
How many do you want to remove
15
The program removed from stock completely
```

4-)Problem Solution Approach

For SoftwareStore at q4:

First of all,If the user chooses which tree, I cast dataTree to that tree in the constructor method.Second, I have set a general password for this system. If the user chose admin login when he\she ran the application, I asked for a password and compared it to the general password. I listed the tasks he\she can do if the

passwords match. if the user chose guest login, I just listed the search functions, the user can select one of them and run it.

If we talk about methods;

add: I cast the object reference I received into the string. Then I created this string with the constructor of my own node and sent it to the tree's find method.If it already exists, I increased the quantity by 1. If not, I took the price and the quantity from user and created an object.

remove:In this method, I cast the reference I received into the string. And I created an object with this name. I sent the object to the find method of the tree.If the tree has more than one of the same element, I reduced the quantity by 1. If there is one, I removed it from the tree.

updateInformation: In this method, I listed all the elements first. I asked which application to choose. Then I presented a menu for the things you wanted to do.If the user wanted to increase the quantity of applications that it chose, I bought and increased the quantity of that node that how many to increase.

If the user chose to uninstall the application, I asked how many they wanted to uninstall.If the quantity he chose is already larger than the number of apps, I completely uninstalled the app. If not, I reduced the quantity.

If the sell was selected, I reduced the quantity of applications by 1.

If she\he chose to update the price, I bought and updated the new price.

And I printed the final version of the application on the screen.

find: I converted my reference to string and created an object with that string.And I sent that object to the find method of the tree. If the element returned by the tree is null, it means that it is not in the tree, if it is not null, it is in the tree.I was able to do this search only for the application name because the compareTo method of the node I created compares only the name.

toString: I used toString method of tree.It uses the toString method of the node I created.My node's toString method shows the application name, quantity and price.

For KWSkipList:

First of all, I created an array with a maximum size array and level that can hold my own elements, whatever links I have in node constructor.My Node contains information such as array elements, links, and the number of elements it holds.

In the constructor method, we got the maximum and minimum element numbers of the knot we received from the user.

add : In the add method, I sent my reference item to the search method.Then I checked the node that the first element of pred showed.If the displayed element is

head, I added at link[0] to the element that pred showed.If the displayed node already has the maximum number of elements, I divided the node into 2 parts.I set the links of node that have smaller values to show the new node.If head links[0] is null, I created a new node because there are no elements added before.

If pred[0] does not show the head, I checked the element of the node to be added, and if it does not exceed the maximum value, I sent the item to the method of add of the node.If it already holds the maximum number of elements, I sent it to the auxiliary split method.

split:I compared item to be added with the last item of node. If item is bigger, I added it to the new node.If the item is small, I sent the last item of the node to the new node, then reduced the size of the node by 1 and i send the item to the add method of node.I reduced the size because I did not want to exceed the size of the array in the add method.I added the items from the added and sorted node to the new node.I sent it to the new one, starting from the middle member until the last element.And I adjusted the links again so that a new node was added.

Remove : Finds the node to be deleted with search.If the number of elements of the node is not minimum, it scrolls to the item to be deleted.If it is the minimum, it takes the smallest element of the next node and adds the node that falls below the minimum. Private remove methods run recursively for get elements from the next node. If the number of members of the node it receives is minimum, it works again.

Size: I returned node num.