# GTU Department of Computer Engineering CSE 222/505 - Spring 2020
# Homework 07 Report

**ESRA ERYILMAZ**
**171044046**

## - Q1 -

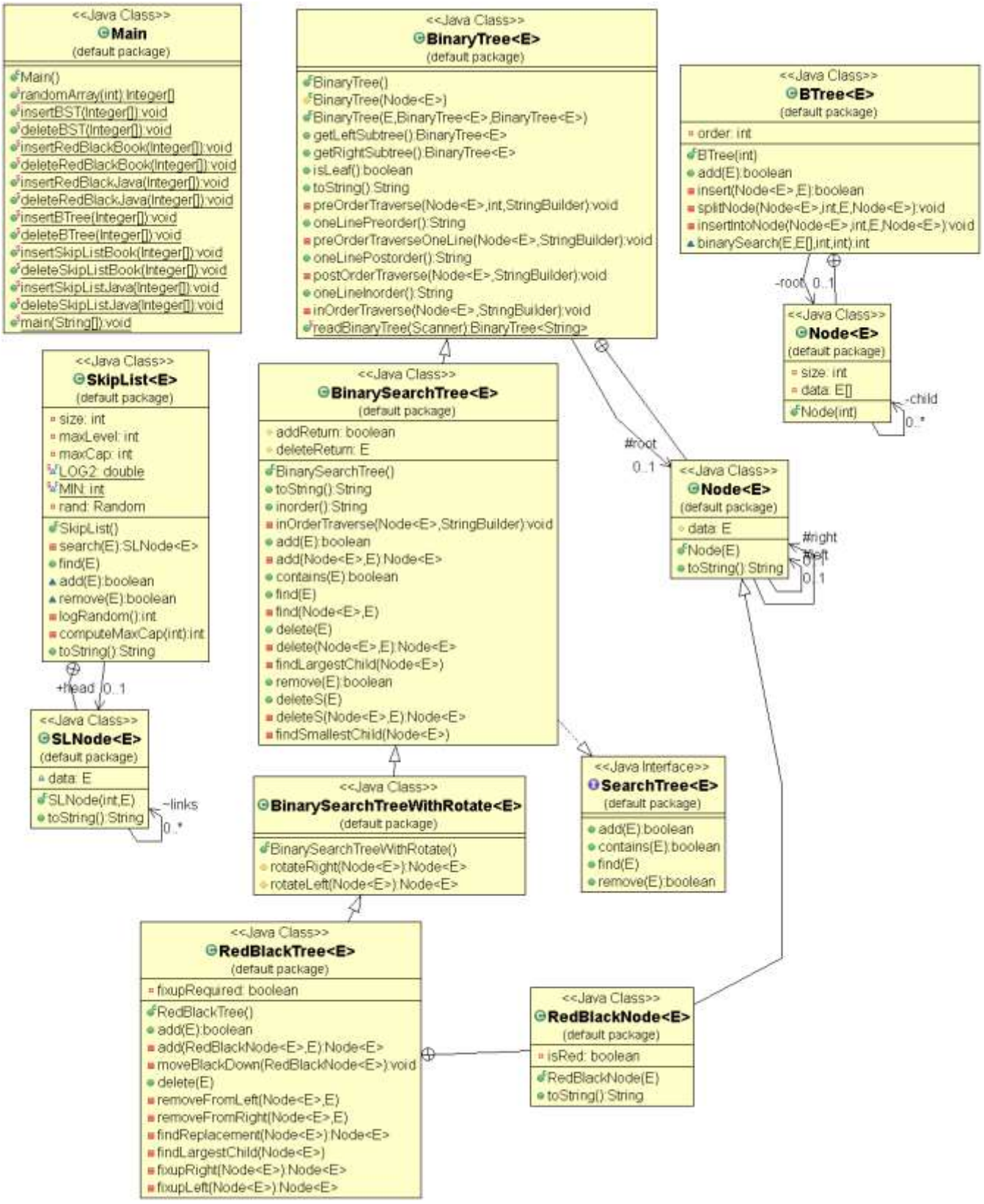
*Q1 is in the 171044046.pdf file.*

## - Q2 -

*I could not implement it.*

# - Q3 -

## 1. CLASS DIAGRAM

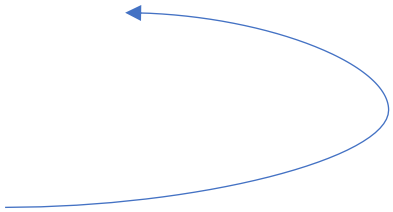


## 2. PROBLEM SOLUTION APPROACH

I used the add and remove methods in the book and java.

## 3. TEST CASES

- Regular binary search tree
- Red-Black tree implementation in the book
- Red Black tree implementation in java
- B-tree implementation in the book
- Skip list implementation in the book
- Skip list implementation in java



I calculated the run times of these classes using their add and remove methods.
P.S. But for BTree there is no remove method.

P.S. I saved the results in the excel file. (q3)

## 4. RUNNING AND RESULTS

# INSERTION (IN MILLISECONDS)

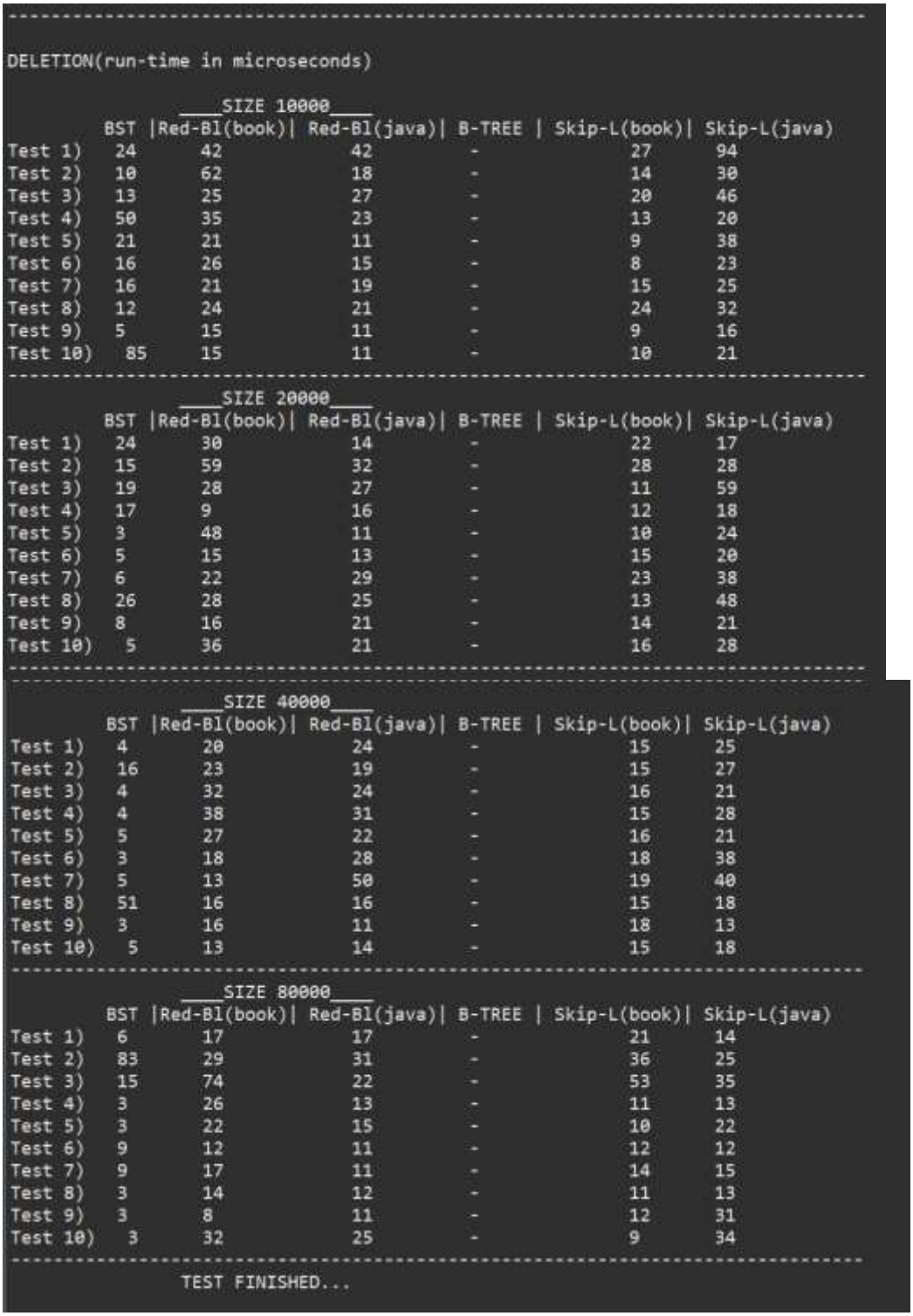
```
              TEST STARTING...

INSERTION (run-time in milliseconds)

-------------------------------------------------------------------------------
                ____SIZE 10000____
         BST |Red-Bl(book)| Red-Bl(java)| B-TREE | Skip-L(book)| Skip-L(java)
Test 1)   3       5             4          6           9              11
Test 2)   1       2             1          1           10             3
Test 3)   1       2             1          3           5              3
Test 4)   1       2             1          1           4              2
Test 5)   1       2             1          1           4              2
Test 6)   1       1             1          1           4              2
Test 7)   1       2             1          1           4              2
Test 8)   1       1             3          0           5              2
Test 9)   1       1             1          0           3              2
Test 10)   1      1             1          0           3              2
-------------------------------------------------------------------------------
                ____SIZE 20000____
         BST |Red-Bl(book)| Red-Bl(java)| B-TREE | Skip-L(book)| Skip-L(java)
Test 1)   2       3             2          1           8              5
Test 2)   2       3             2          1           8              4
Test 3)   2       3             2          1           8              4
Test 4)   2       3             2          1           8              5
Test 5)   2       3             2          1           8              4
Test 6)   2       3             2          1           8              4
Test 7)   2       3             2          1           8              4
Test 8)   2       3             2          1           9              4
Test 9)   2       3             2          1           8              4
Test 10)   2      3             2          1           8              4
-------------------------------------------------------------------------------
                ____SIZE 40000____
         BST |Red-Bl(book)| Red-Bl(java)| B-TREE | Skip-L(book)| Skip-L(java)
Test 1)   5       7             5          2           20             10
Test 2)   5       8             5          2           20             10
Test 3)   5       7             5          2           20             10
Test 4)   5       8             5          2           22             11
Test 5)   6       7             6          3           23             10
Test 6)   6       7             6          3           23             10
Test 7)   5       8             5          2           21             10
Test 8)   5       8             6          2           26             13
Test 9)   9       8             6          2           21             11
Test 10)   5      8             5          2           23             11
-------------------------------------------------------------------------------
                ____SIZE 80000____
         BST |Red-Bl(book)| Red-Bl(java)| B-TREE | Skip-L(book)| Skip-L(java)
Test 1)   12      16            12         4           56             24
Test 2)   12      16            12         4           56             23
Test 3)   12      16            12         4           60             24
Test 4)   12      16            12         7           68             24
Test 5)   12      16            12         7           61             25
Test 6)   13      16            11         4           55             25
Test 7)   12      16            11         4           72             24
Test 8)   12      16            11         4           56             24
Test 9)   12      16            11         4           55             24
Test 10)   12     15            12         4           48             22
-------------------------------------------------------------------------------
              TEST FINISHED...
```

# DELETION(IN MICROSECONDS)

*It seems like deletion takes longer time then insertion ,*
*but time units are different. If I did millisecond for deletion it would always return 0. So I did microseconds for deletion.*



DELETION(run-time in microseconds)

____SIZE 10000____

| | BST | Red-Bl(book) | Red-Bl(java) | B-TREE | Skip-L(book) | Skip-L(java) |
|---|---|---|---|---|---|---|
| Test 1) | 24 | 42 | 42 | - | 27 | 94 |
| Test 2) | 10 | 62 | 18 | - | 14 | 30 |
| Test 3) | 13 | 25 | 27 | - | 20 | 46 |
| Test 4) | 50 | 35 | 23 | - | 13 | 20 |
| Test 5) | 21 | 21 | 11 | - | 9 | 38 |
| Test 6) | 16 | 26 | 15 | - | 8 | 23 |
| Test 7) | 16 | 21 | 19 | - | 15 | 25 |
| Test 8) | 12 | 24 | 21 | - | 24 | 32 |
| Test 9) | 5 | 15 | 11 | - | 9 | 16 |
| Test 10) | 85 | 15 | 11 | - | 10 | 21 |

____SIZE 20000____

| | BST | Red-Bl(book) | Red-Bl(java) | B-TREE | Skip-L(book) | Skip-L(java) |
|---|---|---|---|---|---|---|
| Test 1) | 24 | 30 | 14 | - | 22 | 17 |
| Test 2) | 15 | 59 | 32 | - | 28 | 28 |
| Test 3) | 19 | 28 | 27 | - | 11 | 59 |
| Test 4) | 17 | 9 | 16 | - | 12 | 18 |
| Test 5) | 3 | 48 | 11 | - | 10 | 24 |
| Test 6) | 5 | 15 | 13 | - | 15 | 20 |
| Test 7) | 6 | 22 | 29 | - | 23 | 38 |
| Test 8) | 26 | 28 | 25 | - | 13 | 48 |
| Test 9) | 8 | 16 | 21 | - | 14 | 21 |
| Test 10) | 5 | 36 | 21 | - | 16 | 28 |

____SIZE 40000____

| | BST | Red-Bl(book) | Red-Bl(java) | B-TREE | Skip-L(book) | Skip-L(java) |
|---|---|---|---|---|---|---|
| Test 1) | 4 | 20 | 24 | - | 15 | 25 |
| Test 2) | 16 | 23 | 19 | - | 15 | 27 |
| Test 3) | 4 | 32 | 24 | - | 16 | 21 |
| Test 4) | 4 | 38 | 31 | - | 15 | 28 |
| Test 5) | 5 | 27 | 22 | - | 16 | 21 |
| Test 6) | 3 | 18 | 28 | - | 18 | 38 |
| Test 7) | 5 | 13 | 50 | - | 19 | 40 |
| Test 8) | 51 | 16 | 16 | - | 15 | 18 |
| Test 9) | 3 | 16 | 11 | - | 18 | 13 |
| Test 10) | 5 | 13 | 14 | - | 15 | 18 |

____SIZE 80000____

| | BST | Red-Bl(book) | Red-Bl(java) | B-TREE | Skip-L(book) | Skip-L(java) |
|---|---|---|---|---|---|---|
| Test 1) | 6 | 17 | 17 | - | 21 | 14 |
| Test 2) | 83 | 29 | 31 | - | 36 | 25 |
| Test 3) | 15 | 74 | 22 | - | 53 | 35 |
| Test 4) | 3 | 26 | 13 | - | 11 | 13 |
| Test 5) | 3 | 22 | 15 | - | 10 | 22 |
| Test 6) | 9 | 12 | 11 | - | 12 | 12 |
| Test 7) | 9 | 17 | 11 | - | 14 | 15 |
| Test 8) | 3 | 14 | 12 | - | 11 | 13 |
| Test 9) | 3 | 8 | 11 | - | 12 | 31 |
| Test 10) | 3 | 32 | 25 | - | 9 | 34 |

TEST FINISHED...

## Time Complexities

Run time increases as array size increases.

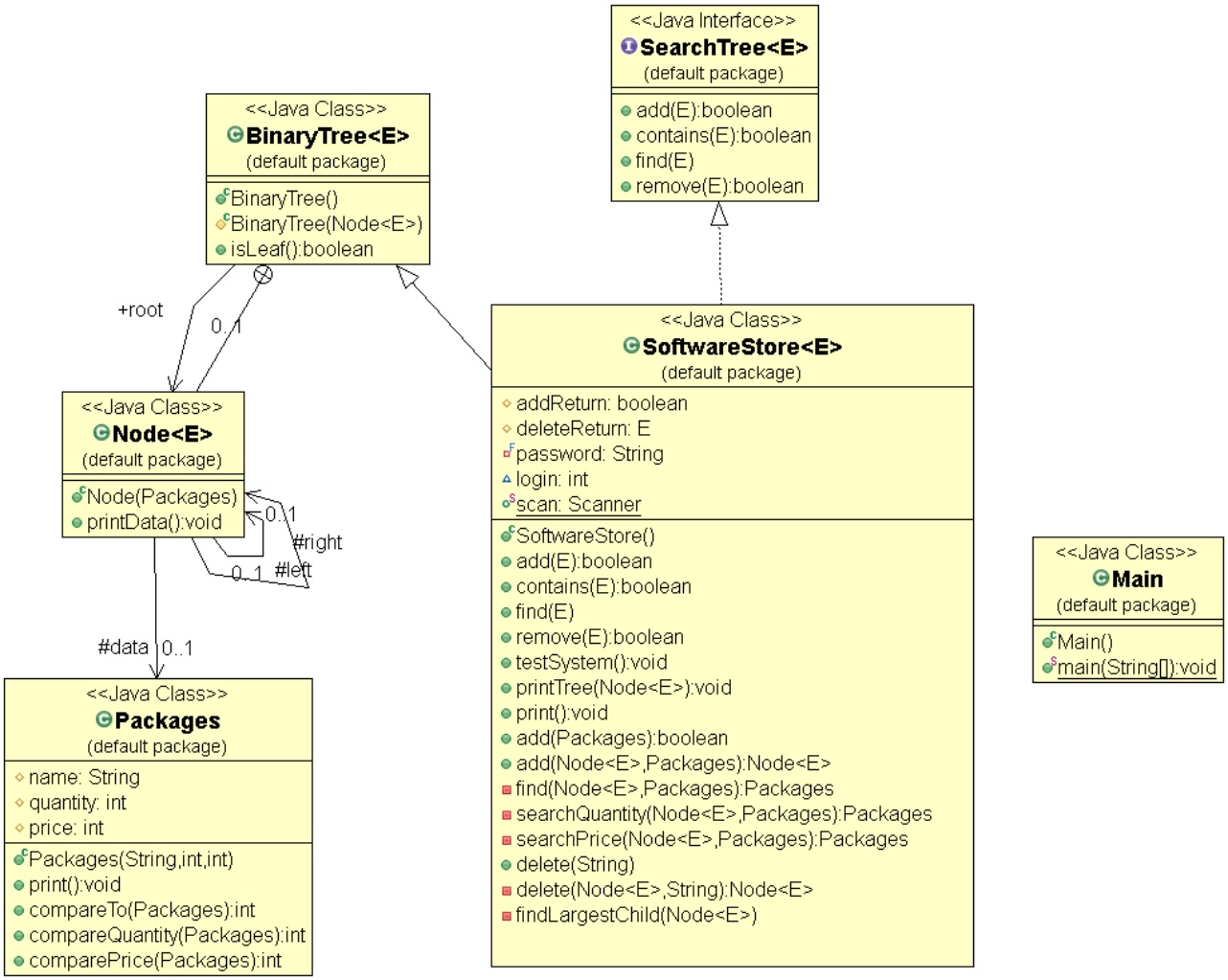
For all classes;

Average case :
insert and deletion operations takes O(logn)

Worst case :
Binary search tree and skip list implementations,they take O(n)
The rest are the same as average case.

# - Q4 -

## 1. CLASS DIAGRAM



## 2. PROBLEM SOLUTION APPROACH

I implement search tree interface. I used Binary Search Tree to implement methods. I used Binary Tree class to keep Node. Inside Node class I keep data in Packages type. Packages class includes :

- String name
- int quantity
- int price

Packages class is comparable and inside Packages class there are 3 important methods.

- compareTo() -> to compare software names
- compareQuantity() -> to compare software quantity
- comparePrice() -> to compare software price

(I used that compare methods inside search methods.)

I have SoftwareStore class. Inside that class , I do all operations.
The most important method which is testSystem(), that method takes inputs from console and do the operations.

When I create the SoftwareStore object in Main, it automatically adds some packages to the system.

I printed the tree in preorder after many operations
Admin must enter the "1234" password to enter the system.

## 3. TEST CASES

**ADMIN TESTS**

| Test Case ID | Test Method | Test Input | Test Output | Pass/Fail |
|---|---|---|---|---|
| **T1** | add() method | Console input | - | Pass |
| **T2** | remove() method | Console input | - | Pass |
| **T3** | update() method | Console input | - | Pass |

**USER TESTS**

| Test Case ID | Test Method | Test Input | Test Output | Pass/Fail |
|---|---|---|---|---|
| **T1** | search() method<br>-by name<br>-by quantity<br>-by price | Console input | - | Pass |

## 4. RUNNING AND RESULTS

- *I write menu-driven program, so I tested all methods with console input.*

packages when the program first opened :

```
Name                Quantity   Price
Adobe Photoshop 6.0  ,  10  ,  600
Adobe Flash 3.3       ,  40  ,  200
Adobe Flash 4.0       ,  35  ,  400
Adobe Photoshop 6.2  ,  20  ,  700
Norton 4.5           ,  30  ,  200
Norton 5.5           ,  30  ,  300
```

# ADMIN TESTS

Firstly you should enter user type.

'1' for admin

'2' for user

If you enter 1 , you should enter admin password to be able to do operations.

```
                TEST STARTING...
------------------------------------------------
Are you admin or user?
'1' for admin
'2' for user
INPUT : 1
Enter password : 1234

Admin login successful !

[Software packages in the store]
[PRINT] :
Name              Quantity      Price
Adobe Photoshop 6.0  ,  10  ,  600
Adobe Flash 3.3  ,  40  ,  200
Adobe Flash 4.0  ,  35  ,  400
Adobe Photoshop 6.2  ,  20  ,  700
Norton 4.5  ,  30  ,  200
Norton 5.5  ,  30  ,  300

What do you want to do?
1) Add package.
2) Remove package.
3) Update package.
4) Exit.
```

LOGIN

```
INPUT : 1
Package name (String) : Avast 2.0
Package quantity (int) : 50
Package price (int) : 350
[PRINT] :
Name              Quantity      Price
Adobe Photoshop 6.0  ,  10  ,  600
Adobe Flash 3.3  ,  40  ,  200
Adobe Flash 4.0  ,  35  ,  400
Adobe Photoshop 6.2  ,  20  ,  700
Norton 4.5  ,  30  ,  200
Avast 2.0  ,  50  ,  350
Norton 5.5  ,  30  ,  300
```

ADD PACKAGE

```
What do you want to do?
1) Add package.
2) Remove package.
3) Update package.
4) Exit.
INPUT : 2
Package name (String) : Adobe Flash 3.3
[PRINT] :
Name              Quantity      Price
Adobe Photoshop 6.0  ,  10  ,  600
Adobe Flash 4.0  ,  35  ,  400
Adobe Photoshop 6.2  ,  20  ,  700
Norton 4.5  ,  30  ,  200
Avast 2.0  ,  50  ,  350
Norton 5.5  ,  30  ,  300
```

REMOVE PACKAGE

```
What do you want to do?
1) Add package.
2) Remove package.
3) Update package.
4) Exit.
INPUT : 3
        1) Update sold out packages.
        2) Update new software packages.
        INPUT : 1
Package name (String) : Norton 5.5
[PRINT] :
Name              Quantity      Price
Adobe Photoshop 6.0  ,  10  ,  600
Adobe Flash 4.0  ,  35  ,  400
Adobe Photoshop 6.2  ,  20  ,  700
Norton 4.5  ,  30  ,  200
Avast 2.0  ,  50  ,  350
```

UPDATE PACKAGE

```
What do you want to do?
1) Add package.
2) Remove package.
3) Update package.
4) Exit.
INPUT : 3
        1) Update sold out packages.
        2) Update new software packages.
        INPUT : 2
Package name (String) : Avast 3.0
Package quantity (int) : 20
Package price (int) : 450
[PRINT] :
Name              Quantity      Price
Adobe Photoshop 6.0  ,  10  ,  600
Adobe Flash 4.0  ,  35  ,  400
Adobe Photoshop 6.2  ,  20  ,  700
Norton 4.5  ,  30  ,  200
Avast 2.0  ,  50  ,  350
Avast 3.0  ,  20  ,  450
```

UPDATE PACKAGE

# USER TESTS

If you enter 2 :

```
               TEST STARTING...
-------------------------------------------------
Are you admin or user?
'1' for admin
'2' for user
INPUT : 2

User login successful !

[Software packages in the store]
[PRINT] :
Name              Quantity      Price
Adobe Photoshop 6.0  ,  10  ,  600
Adobe Flash 3.3  ,  40  ,  200
Adobe Flash 4.0  ,  35  ,  400
Adobe Photoshop 6.2  ,  20  ,  700
Norton 4.5  ,  30  ,  200
Norton 5.5  ,  30  ,  300
```

LOGIN

```
What do you want to do?
1) Search software by name.
2) Search software by quantity.
3) Search software by price.
4) Exit.
INPUT : 1
Package name (String) : Norton 4.5
Search Results : Quantity : 30 ,  Price : 200
```

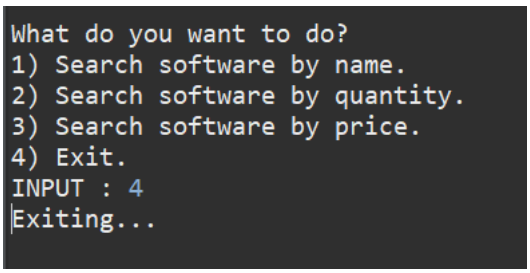
SEARCH by NAME

```
What do you want to do?
1) Search software by name.
2) Search software by quantity.
3) Search software by price.
4) Exit.
INPUT : 2
Package quantity (int) : 10
Search Results -> Name : Adobe Photoshop 6.0 ,  Price : 600
```

SEARCH by QUANTITY

```
What do you want to do?
1) Search software by name.
2) Search software by quantity.
3) Search software by price.
4) Exit.
INPUT : 3
Package price (int) : 300
Search Results -> Name : Norton 5.5 ,  Quantity : 30
```

SEARCH by PRICE


What do you want to do?
1) Search software by name.
2) Search software by quantity.
3) Search software by price.
4) Exit.
INPUT : 4
Exiting...

EXIT...