DATA STRUCTURE HOMEWORK 07

MOHAMMAD ASHRAF YAWAR

161044123

PART02:

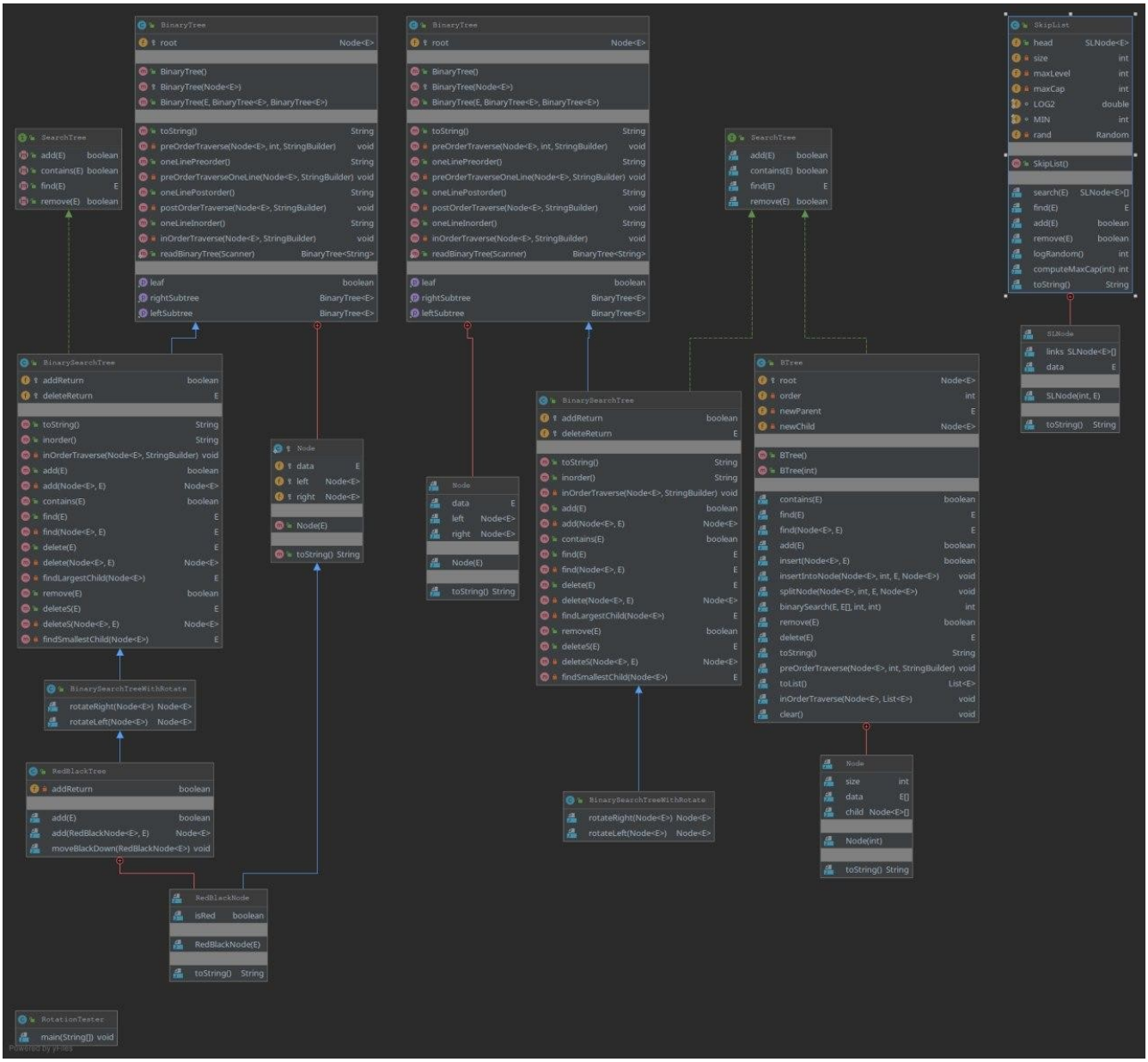
UML:

SCREEN SHOTS:

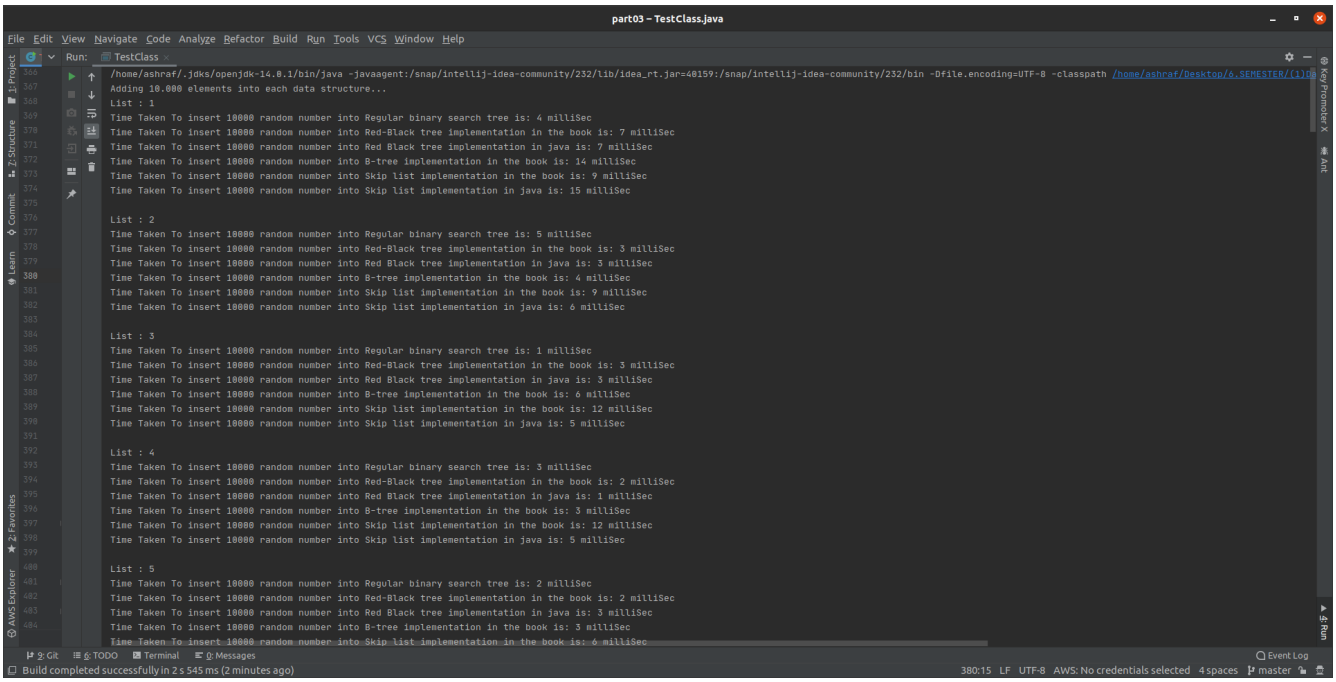
TEST CASES:

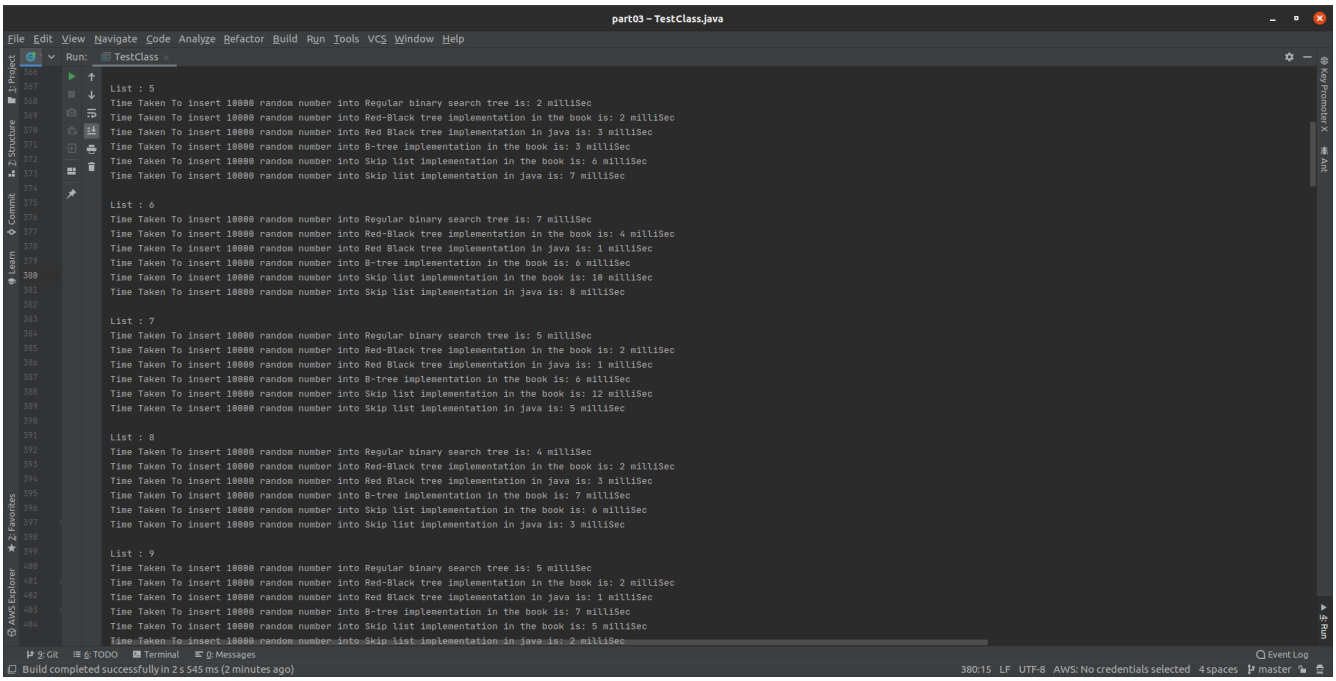
PROBLEM SOL APPROACH:

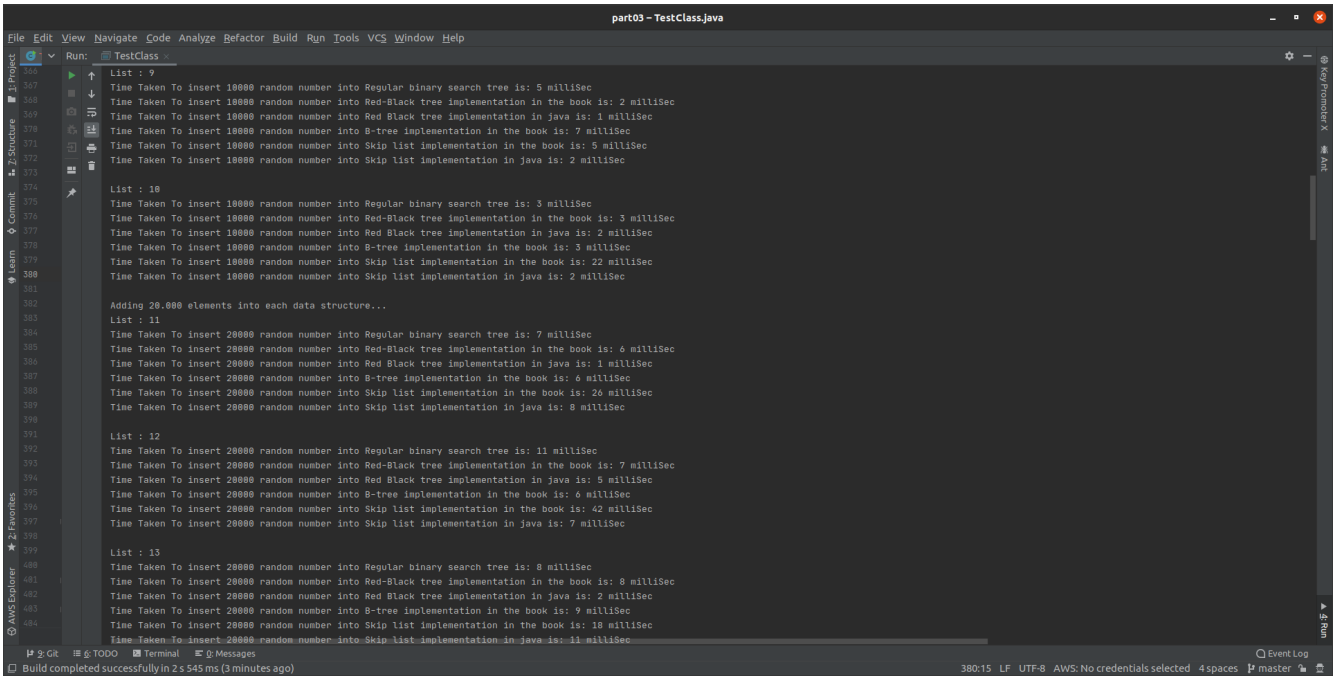
PART03:
UML:

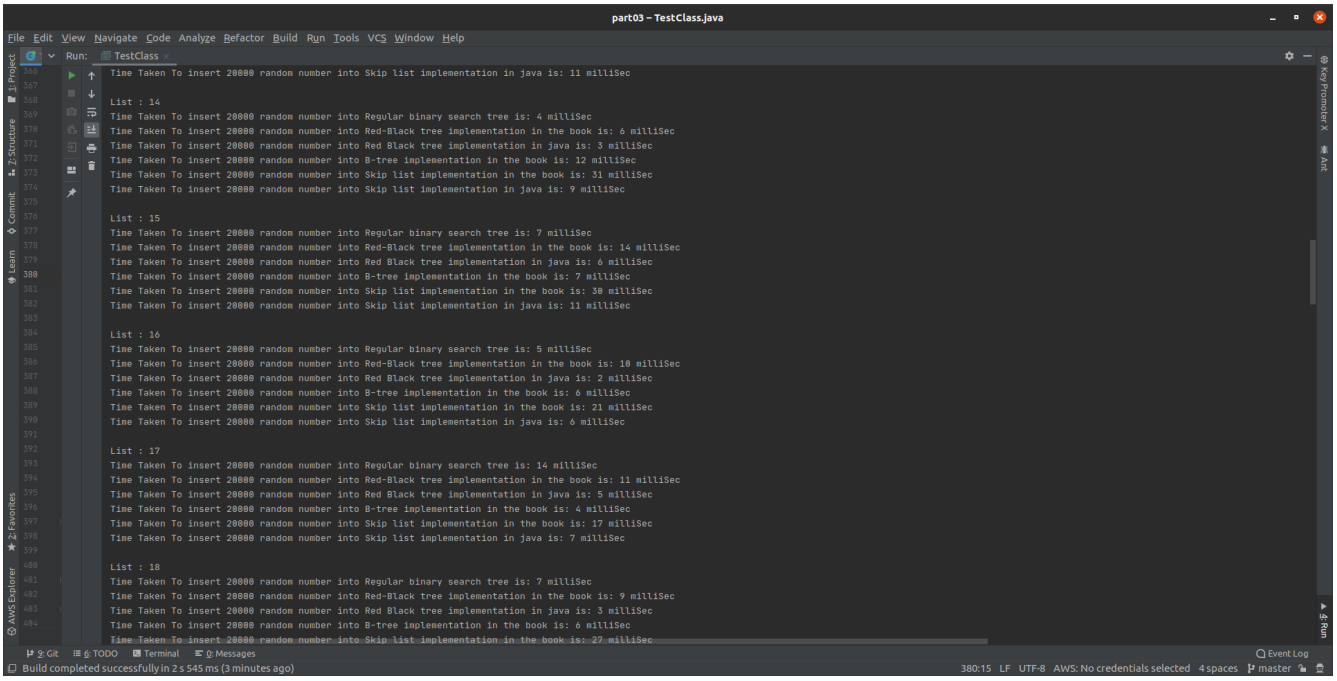

BinaryTree
root Node<E>
BinaryTree()
BinaryTree(Node<E>)
BinaryTree(E, BinaryTree<E>, BinaryTree<E>)
toString() String
preOrderTraverse(Node<E>, int, StringBuilder) void
oneLinePreorder() String
preOrderTraverseOneLine(Node<E>, StringBuilder) void
oneLinePostorder() String
postOrderTraverse(Node<E>, StringBuilder) void
oneLineInorder() String
inOrderTraverse(Node<E>, StringBuilder) void
readBinaryTree(Scanner) BinaryTree<String>
leaf boolean
rightSubtree BinaryTree<E>
leftSubtree BinaryTree<E>
SkipList
head SLNode<E>
size int
maxLevel int
maxCap int
LOG2 double
MIN int
rand Random
SkipList()
search(E) SLNode<E>[]
find(E) E
add(E) boolean
remove(E) boolean
logRandom() int
computeMaxCap(int) int
toString() String
SLNode
links SLNode<E>[]
data E
SLNode(int, E)
toString() String
SearchTree
add(E) boolean
contains(E) boolean
find(E) E
remove(E) boolean
BinarySearchTree
addReturn boolean
deleteReturn E
inorder() String
inOrderTraverse(Node<E>, StringBuilder) void
add(Node<E>, E) Node<E>
contains(E) boolean
delete(E) E
delete(Node<E>, E) Node<E>
findLargestChild(Node<E>) E
deleteS(E) E
deleteS(Node<E>, E) Node<E>
findSmallestChild(Node<E>) E
Node
data E
left Node<E>
right Node<E>
Node(E)
BTree
root Node<E>
order int
newParent E
newChild Node<E>
BTree()
BTree(int)
insert(Node<E>, E) boolean
insertIntoNode(Node<E>, int, E, Node<E>) void
splitNode(Node<E>, int, E, Node<E>) void
binarySearch(E, E[], int, int) int
toList() List<E>
inOrderTraverse(Node<E>, List<E>) void
clear() void
size int
data E[]
child Node<E>[]
Node(int)
BinarySearchTreeWithRotate
rotateRight(Node<E>) Node<E>
rotateLeft(Node<E>) Node<E>
RedBlackTree
addReturn boolean
add(E) boolean
add(RedBlackNode<E>, E) Node<E>
moveBlackDown(RedBlackNode<E>) void
RedBlackNode
isRed boolean
RedBlackNode(E)
RotationTester
main(String[]) void

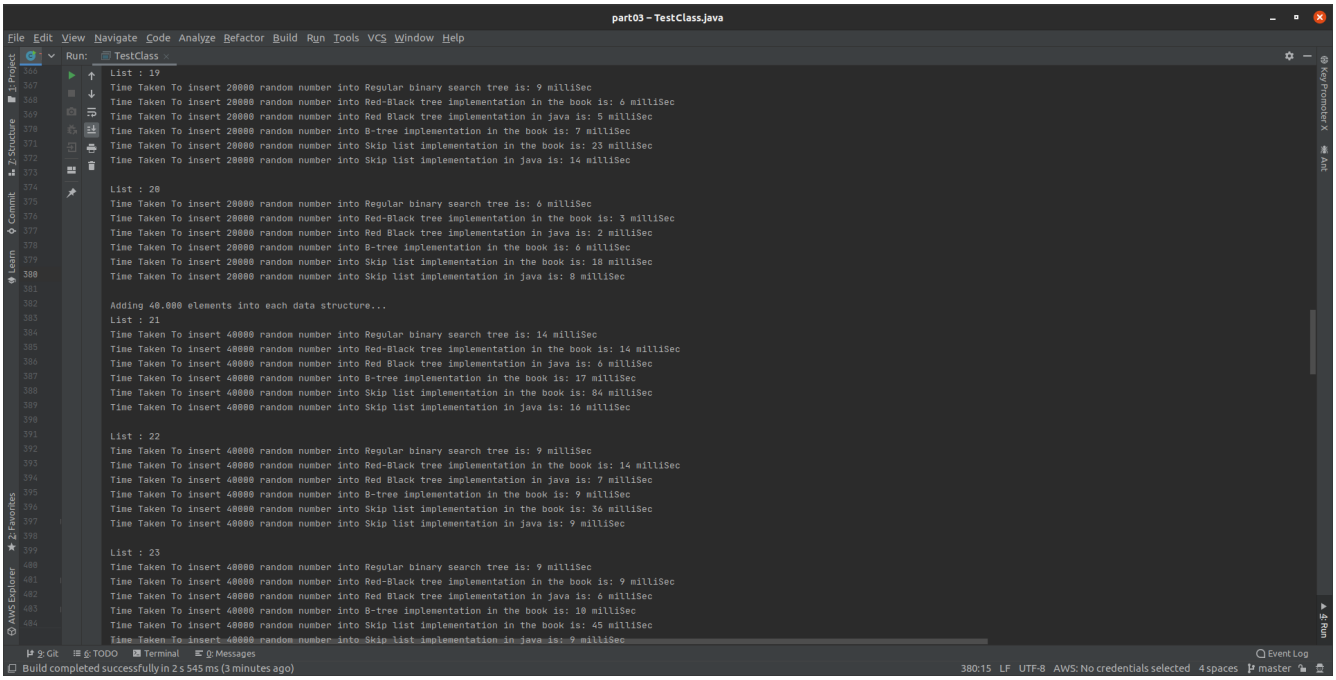
SCREEN SHOTS:

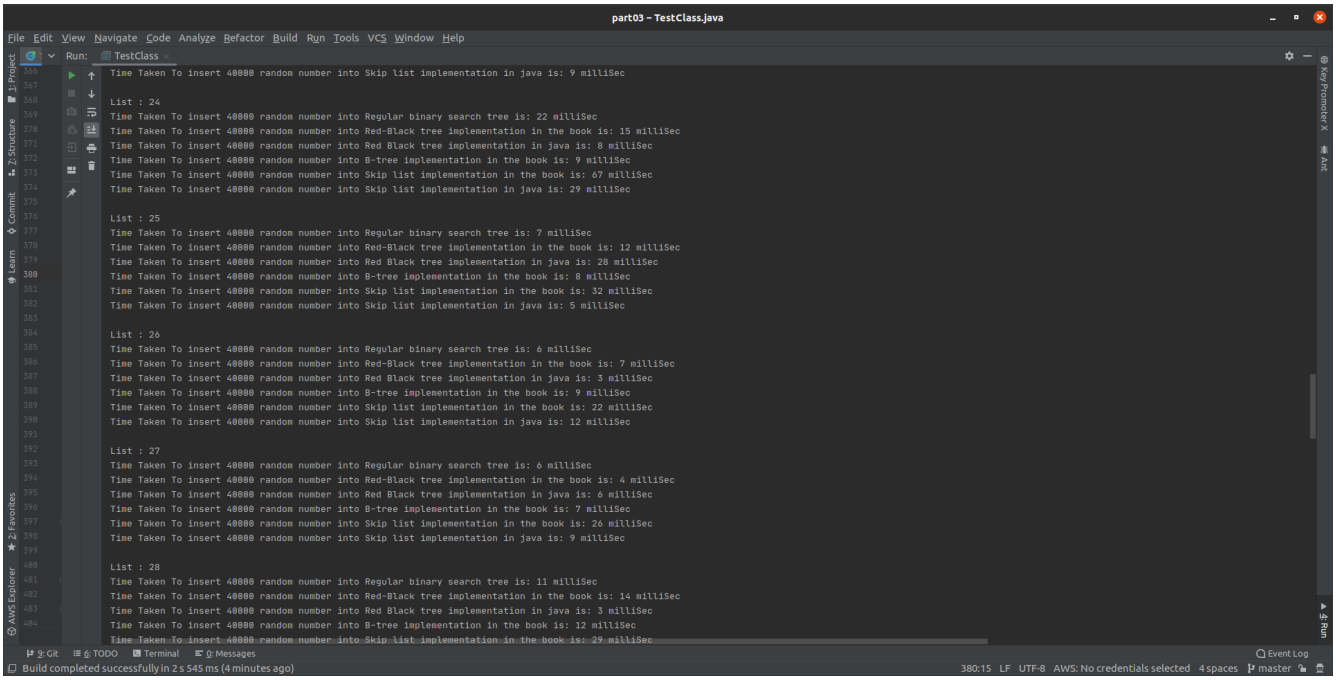


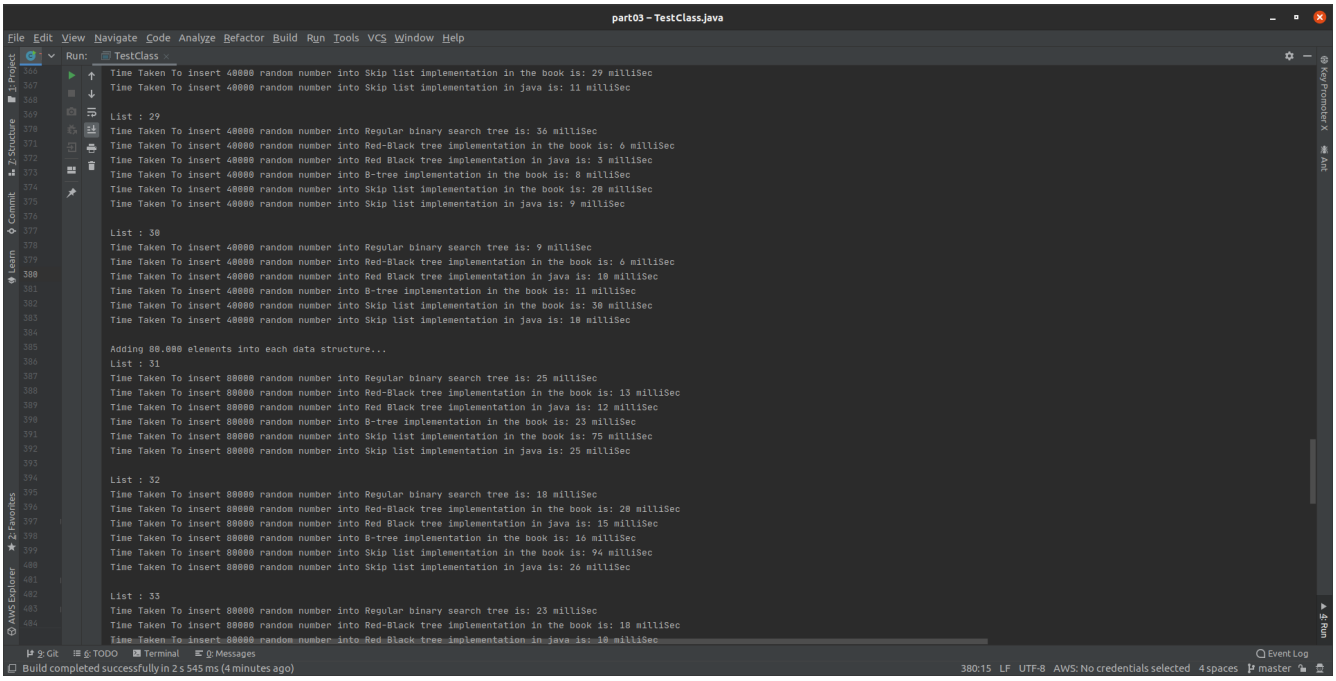


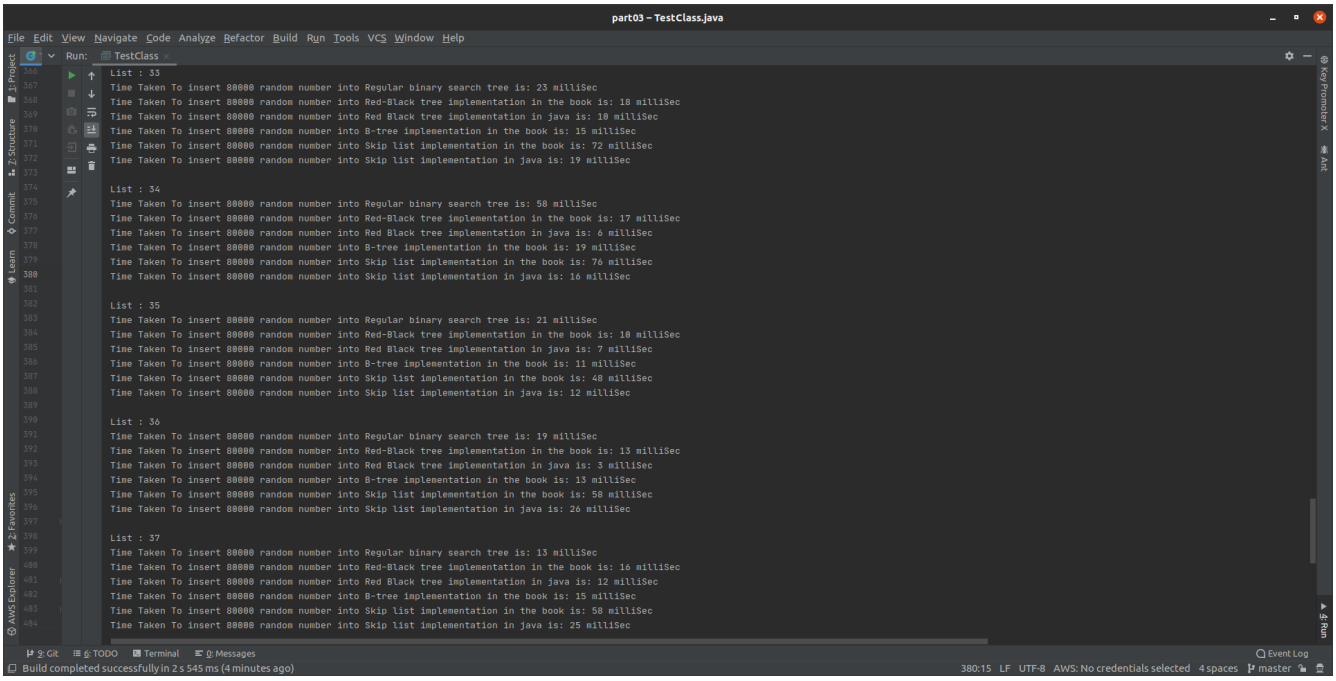

part03 – TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
List : 9
Time Taken To insert 10000 random number into Regular binary search tree is: 5 milliSec
Time Taken To insert 10000 random number into Red-Black tree implementation in the book is: 2 milliSec
Time Taken To insert 10000 random number into Red Black tree implementation in java is: 1 milliSec
Time Taken To insert 10000 random number into B-tree implementation in the book is: 7 milliSec
Time Taken To insert 10000 random number into Skip list implementation in the book is: 5 milliSec
Time Taken To insert 10000 random number into Skip list implementation in java is: 2 milliSec
List : 10
Time Taken To insert 10000 random number into Regular binary search tree is: 3 milliSec
Time Taken To insert 10000 random number into Red-Black tree implementation in the book is: 3 milliSec
Time Taken To insert 10000 random number into Red Black tree implementation in java is: 2 milliSec
Time Taken To insert 10000 random number into B-tree implementation in the book is: 3 milliSec
Time Taken To insert 10000 random number into Skip list implementation in the book is: 22 milliSec
Time Taken To insert 10000 random number into Skip list implementation in java is: 2 milliSec
Adding 20.000 elements into each data structure...
List : 11
Time Taken To insert 20000 random number into Regular binary search tree is: 7 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 1 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 26 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 8 milliSec
List : 12
Time Taken To insert 20000 random number into Regular binary search tree is: 11 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 7 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 5 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 42 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 7 milliSec
List : 13
Time Taken To insert 20000 random number into Regular binary search tree is: 8 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 8 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 2 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 9 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 18 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 11 milliSec
9: Git 6: TODO Terminal 0: Messages Event Log
Build completed successfully in 2 s 545 ms (3 minutes ago)
380:15 LF UTF-8 AWS: No credentials selected 4 spaces master

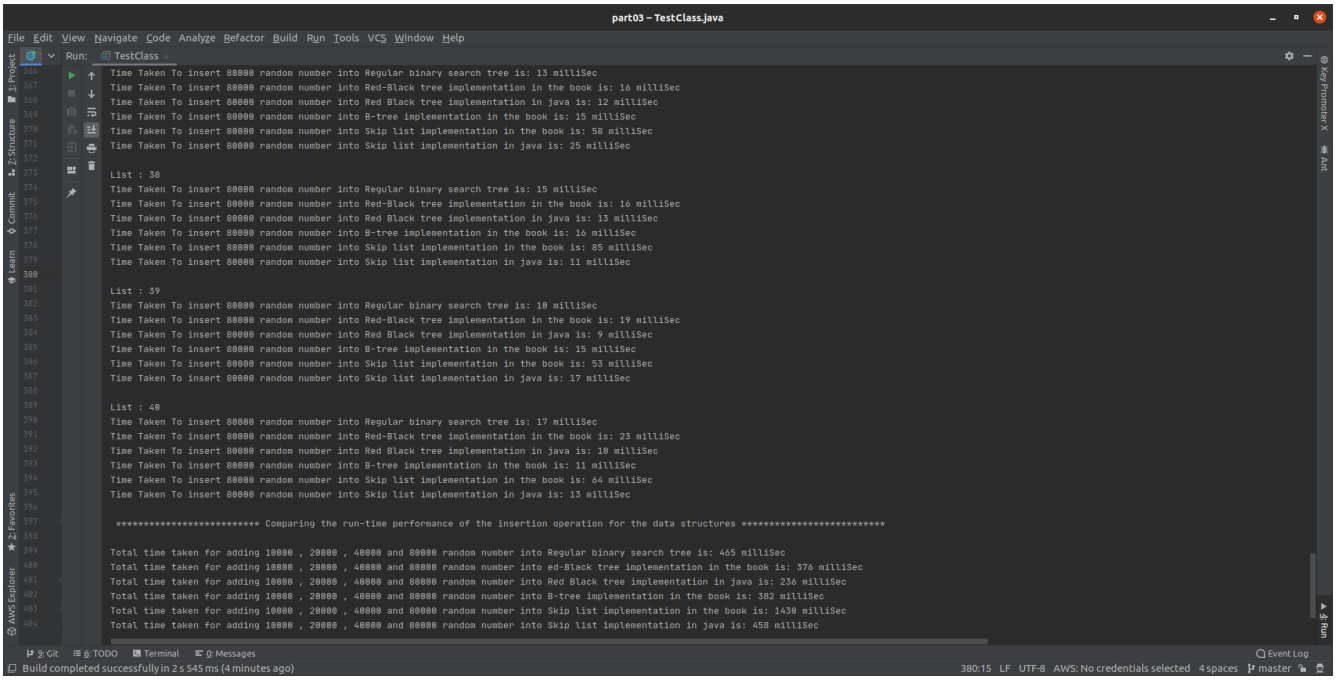

part03 – TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
Time Taken To insert 20000 random number into Skip list implementation in java is: 11 milliSec
List : 14
Time Taken To insert 20000 random number into Regular binary search tree is: 4 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 3 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 12 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 31 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 9 milliSec
List : 15
Time Taken To insert 20000 random number into Regular binary search tree is: 7 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 14 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 6 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 7 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 30 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 11 milliSec
List : 16
Time Taken To insert 20000 random number into Regular binary search tree is: 5 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 10 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 2 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 21 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 6 milliSec
List : 17
Time Taken To insert 20000 random number into Regular binary search tree is: 14 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 11 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 5 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 4 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 17 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 7 milliSec
List : 18
Time Taken To insert 20000 random number into Regular binary search tree is: 7 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 9 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 3 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 27 milliSec
9: Git 6: TODO Terminal 0: Messages Event Log
Build completed successfully in 2 s 545 ms (3 minutes ago)
380:15 LF UTF-8 AWS: No credentials selected 4 spaces master

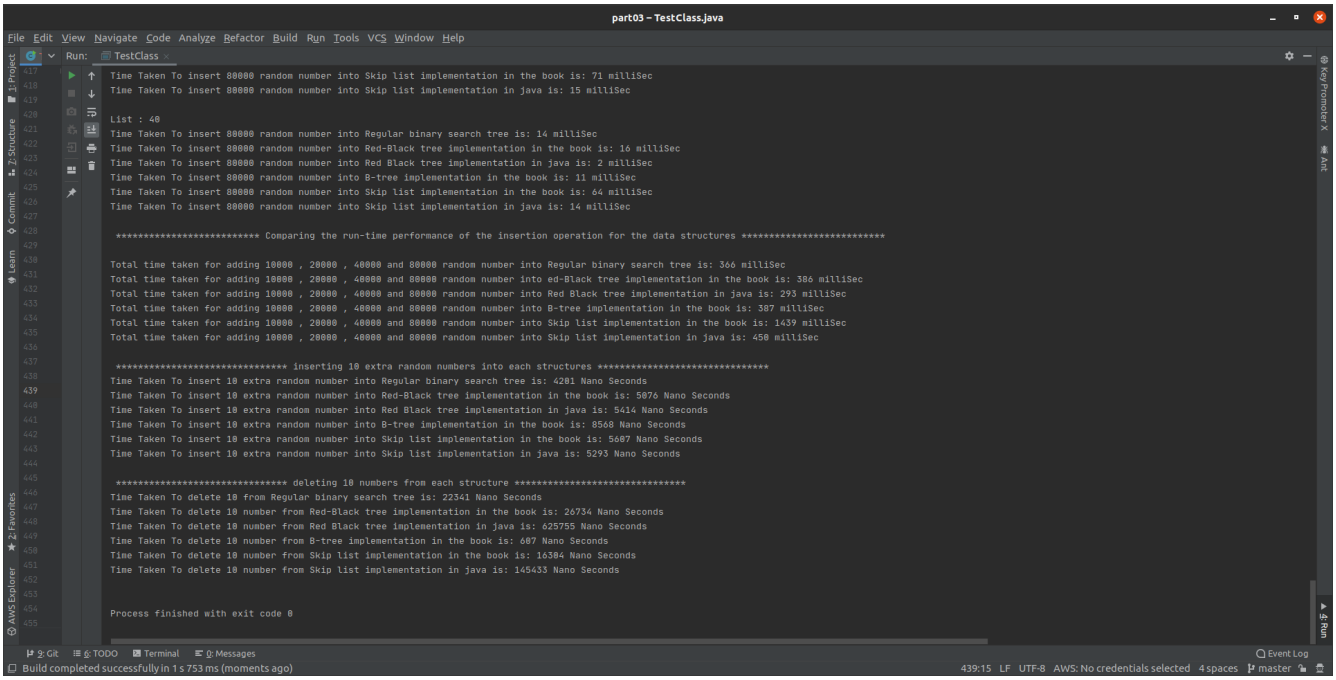

part03 – TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
List : 19
Time Taken To insert 20000 random number into Regular binary search tree is: 9 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 5 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 7 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 23 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 14 milliSec
List : 20
Time Taken To insert 20000 random number into Regular binary search tree is: 6 milliSec
Time Taken To insert 20000 random number into Red-Black tree implementation in the book is: 3 milliSec
Time Taken To insert 20000 random number into Red Black tree implementation in java is: 2 milliSec
Time Taken To insert 20000 random number into B-tree implementation in the book is: 6 milliSec
Time Taken To insert 20000 random number into Skip list implementation in the book is: 18 milliSec
Time Taken To insert 20000 random number into Skip list implementation in java is: 8 milliSec
Adding 40.000 elements into each data structure...
List : 21
Time Taken To insert 40000 random number into Regular binary search tree is: 14 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 14 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 6 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 17 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 84 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 16 milliSec
List : 22
Time Taken To insert 40000 random number into Regular binary search tree is: 9 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 14 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 7 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 9 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 36 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 9 milliSec
List : 23
Time Taken To insert 40000 random number into Regular binary search tree is: 9 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 9 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 6 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 10 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 45 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 9 milliSec
9: Git 6: TODO Terminal 0: Messages
Event Log
Build completed successfully in 2 s 545 ms (3 minutes ago)
380:15 LF UTF-8 AWS: No credentials selected 4 spaces master


part03 – TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
Time Taken To insert 40000 random number into Skip list implementation in java is: 9 milliSec
List : 24
Time Taken To insert 40000 random number into Regular binary search tree is: 22 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 15 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 8 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 9 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 67 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 29 milliSec
List : 25
Time Taken To insert 40000 random number into Regular binary search tree is: 7 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 12 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 28 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 8 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 32 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 5 milliSec
List : 26
Time Taken To insert 40000 random number into Regular binary search tree is: 6 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 7 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 3 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 9 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 22 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 12 milliSec
List : 27
Time Taken To insert 40000 random number into Regular binary search tree is: 6 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 4 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 6 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 7 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 26 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 9 milliSec
List : 28
Time Taken To insert 40000 random number into Regular binary search tree is: 11 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 14 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 3 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 12 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 29 milliSec
9: Git 6: TODO Terminal 0: Messages
Event Log
Build completed successfully in 2 s 545 ms (4 minutes ago)
380:15 LF UTF-8 AWS: No credentials selected 4 spaces master


part03 – TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
Time Taken To insert 40000 random number into Skip list implementation in the book is: 29 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 11 milliSec
List : 29
Time Taken To insert 40000 random number into Regular binary search tree is: 36 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 6 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 3 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 8 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 28 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 9 milliSec
List : 30
Time Taken To insert 40000 random number into Regular binary search tree is: 9 milliSec
Time Taken To insert 40000 random number into Red-Black tree implementation in the book is: 6 milliSec
Time Taken To insert 40000 random number into Red Black tree implementation in java is: 10 milliSec
Time Taken To insert 40000 random number into B-tree implementation in the book is: 11 milliSec
Time Taken To insert 40000 random number into Skip list implementation in the book is: 30 milliSec
Time Taken To insert 40000 random number into Skip list implementation in java is: 10 milliSec
Adding 80.000 elements into each data structure...
List : 31
Time Taken To insert 80000 random number into Regular binary search tree is: 25 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 13 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 12 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 23 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 75 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 25 milliSec
List : 32
Time Taken To insert 80000 random number into Regular binary search tree is: 18 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 20 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 15 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 16 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 94 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 26 milliSec
List : 33
Time Taken To insert 80000 random number into Regular binary search tree is: 23 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 18 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 10 milliSec
9: Git 6: TODO Terminal 0: Messages
Event Log
Build completed successfully in 2 s 545 ms (4 minutes ago)
380:15 LF UTF-8 AWS: No credentials selected 4 spaces master


part03 – TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
List : 33
Time Taken To insert 80000 random number into Regular binary search tree is: 23 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 18 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 10 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 15 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 72 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 19 milliSec
List : 34
Time Taken To insert 80000 random number into Regular binary search tree is: 58 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 17 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 6 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 19 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 76 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 16 milliSec
List : 35
Time Taken To insert 80000 random number into Regular binary search tree is: 21 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 10 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 7 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 11 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 48 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 12 milliSec
List : 36
Time Taken To insert 80000 random number into Regular binary search tree is: 19 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 13 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 3 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 13 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 58 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 26 milliSec
List : 37
Time Taken To insert 80000 random number into Regular binary search tree is: 13 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 16 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 12 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 15 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 58 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 25 milliSec
9: Git 6: TODO Terminal 0: Messages
Event Log
Build completed successfully in 2 s 545 ms (4 minutes ago)
380:15 LF UTF-8 AWS: No credentials selected 4 spaces master

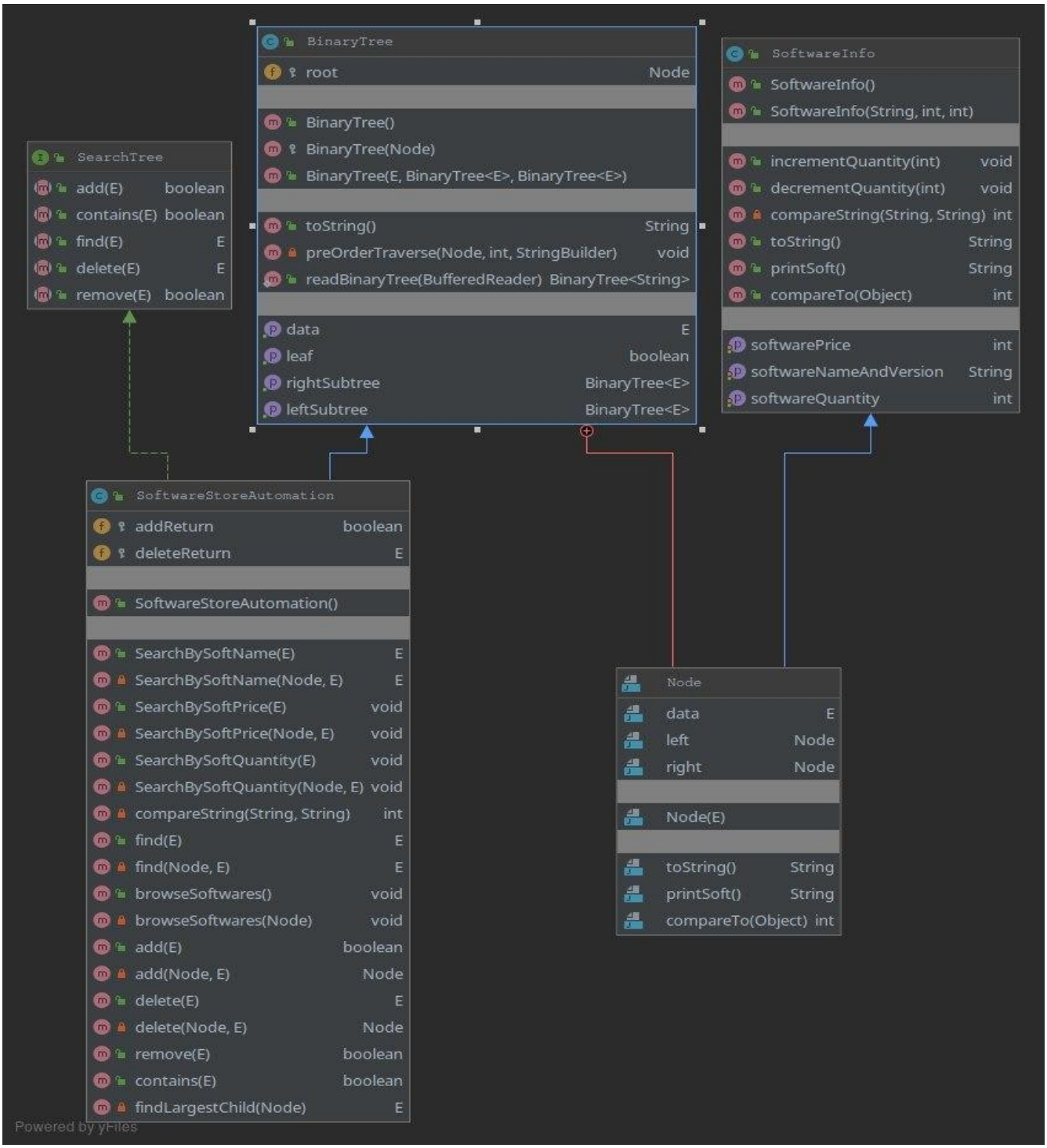

part03 – TestClass.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run: TestClass
Time Taken To insert 80000 random number into Regular binary search tree is: 13 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 16 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 12 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 15 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 58 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 25 milliSec
List : 38
Time Taken To insert 80000 random number into Regular binary search tree is: 15 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 16 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 13 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 16 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 85 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 11 milliSec
List : 39
Time Taken To insert 80000 random number into Regular binary search tree is: 10 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 19 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 9 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 15 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 53 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 17 milliSec
List : 40
Time Taken To insert 80000 random number into Regular binary search tree is: 17 milliSec
Time Taken To insert 80000 random number into Red-Black tree implementation in the book is: 23 milliSec
Time Taken To insert 80000 random number into Red Black tree implementation in java is: 10 milliSec
Time Taken To insert 80000 random number into B-tree implementation in the book is: 11 milliSec
Time Taken To insert 80000 random number into Skip list implementation in the book is: 64 milliSec
Time Taken To insert 80000 random number into Skip list implementation in java is: 13 milliSec
********** Comparing the run-time performance of the insertion operation for the data structures **********
Total time taken for adding 10000 , 20000 , 40000 and 80000 random number into Regular binary search tree is: 465 milliSec
Total time taken for adding 10000 , 20000 , 40000 and 80000 random number into ed-Black tree implementation in the book is: 376 milliSec
Total time taken for adding 10000 , 20000 , 40000 and 80000 random number into Red Black tree implementation in java is: 236 milliSec
Total time taken for adding 10000 , 20000 , 40000 and 80000 random number into B-tree implementation in the book is: 382 milliSec
Total time taken for adding 10000 , 20000 , 40000 and 80000 random number into Skip list implementation in the book is: 1438 milliSec
Total time taken for adding 10000 , 20000 , 40000 and 80000 random number into Skip list implementation in java is: 458 milliSec
9: Git 6: TODO Terminal 0: Messages Event Log
Build completed successfully in 2 s 545 ms (4 minutes ago)
380:15 LF UTF-8 AWS: No credentials selected 4 spaces master



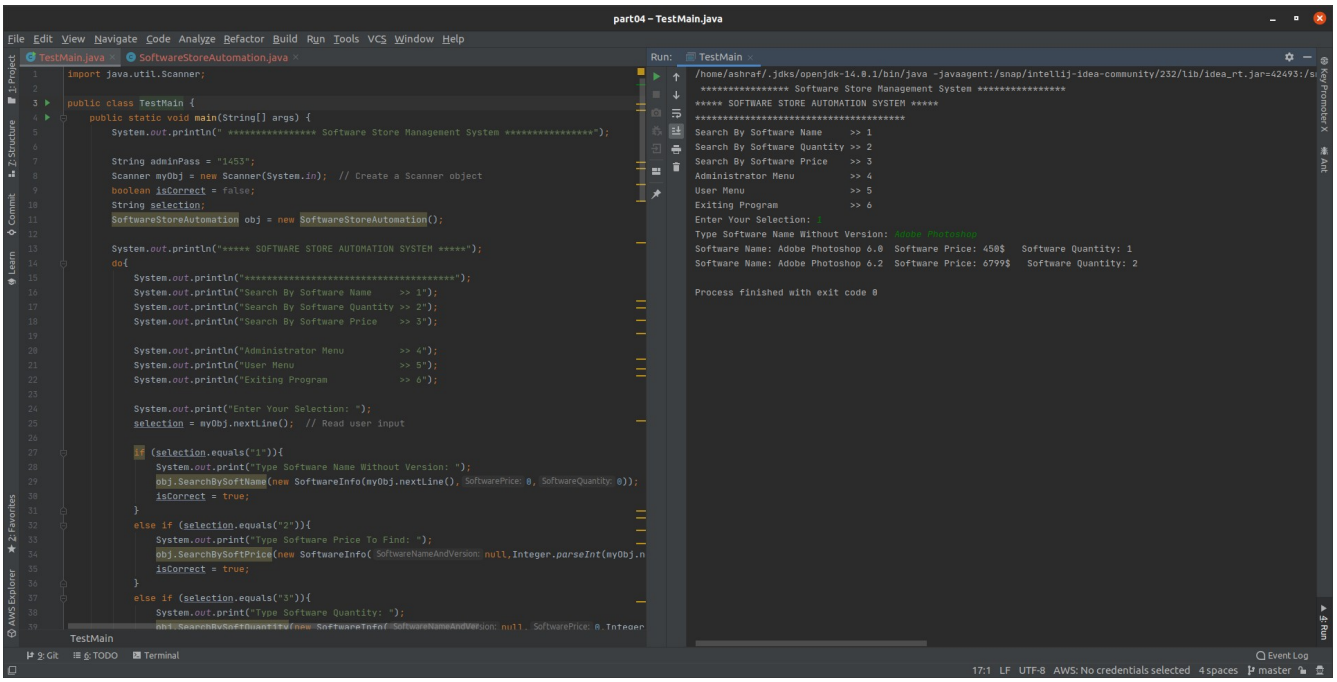
TEST CASES:
PROBLEM SOL APPROACH:

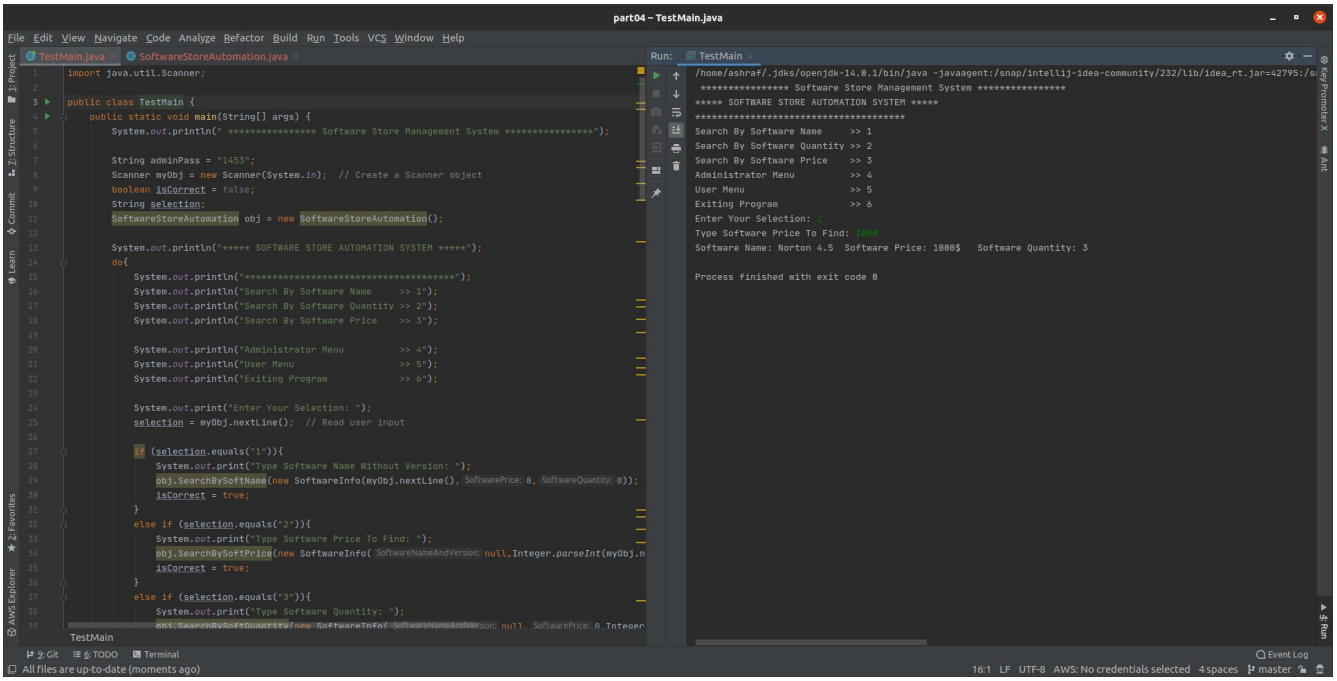
→ test is done for each data structure.
→ I have perform desired operations on each of data structure and calculated running time for each data structure.

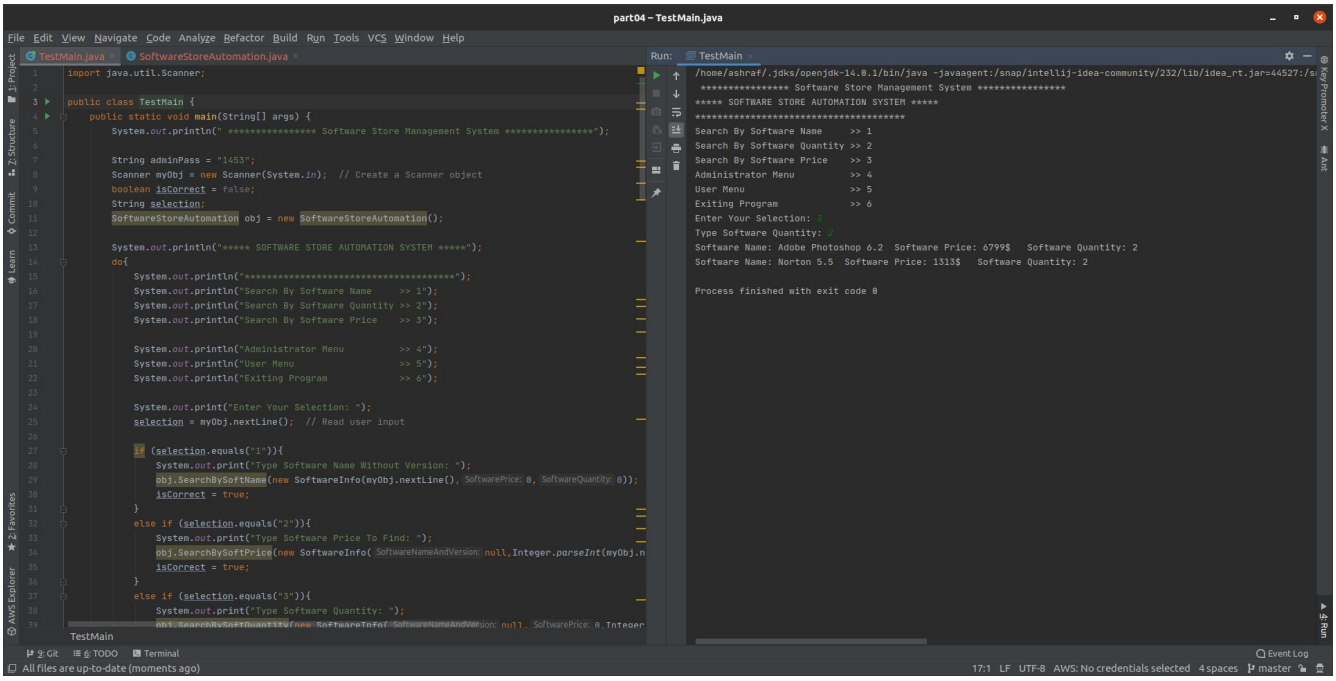
PART04:
UML:

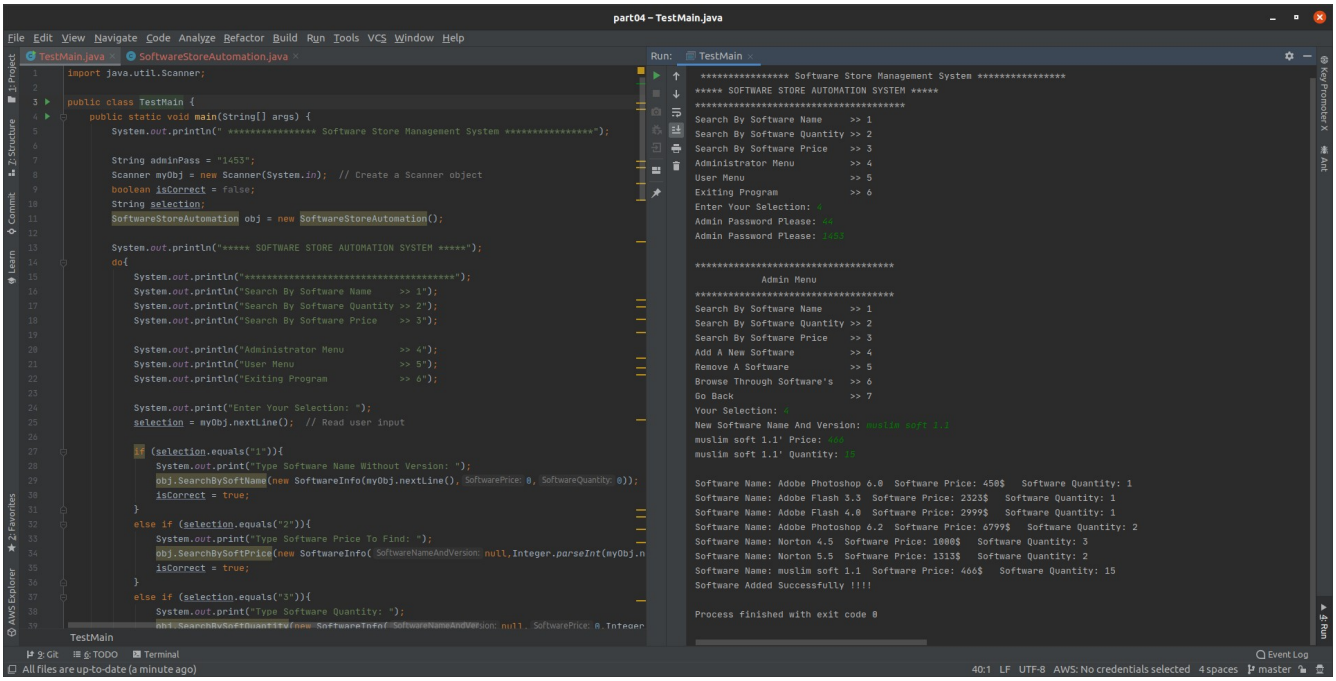

BinaryTree
root Node
BinaryTree()
BinaryTree(Node)
BinaryTree(E, BinaryTree<E>, BinaryTree<E>)
toString() String
preOrderTraverse(Node, int, StringBuilder) void
readBinaryTree(BufferedReader) BinaryTree<String>
data E
leaf boolean
rightSubtree BinaryTree<E>
leftSubtree BinaryTree<E>
SearchTree
add(E) boolean
contains(E) boolean
find(E) E
delete(E) E
remove(E) boolean
SoftwareInfo
SoftwareInfo()
SoftwareInfo(String, int, int)
incrementQuantity(int) void
decrementQuantity(int) void
compareString(String, String) int
toString() String
printSoft() String
compareTo(Object) int
softwarePrice int
softwareNameAndVersion String
softwareQuantity int
SoftwareStoreAutomation
addReturn boolean
deleteReturn E
SoftwareStoreAutomation()
SearchBySoftName(E) E
SearchBySoftName(Node, E) E
SearchBySoftPrice(E) void
SearchBySoftPrice(Node, E) void
SearchBySoftQuantity(E) void
SearchBySoftQuantity(Node, E) void
compareString(String, String) int
find(E) E
find(Node, E) E
browseSoftwares() void
browseSoftwares(Node) void
add(E) boolean
add(Node, E) Node
delete(E) E
delete(Node, E) Node
remove(E) boolean
contains(E) boolean
findLargestChild(Node) E
Node
data E
left Node
right Node
Node(E)
toString() String
printSoft() String
compareTo(Object) int
Powered by yFiles

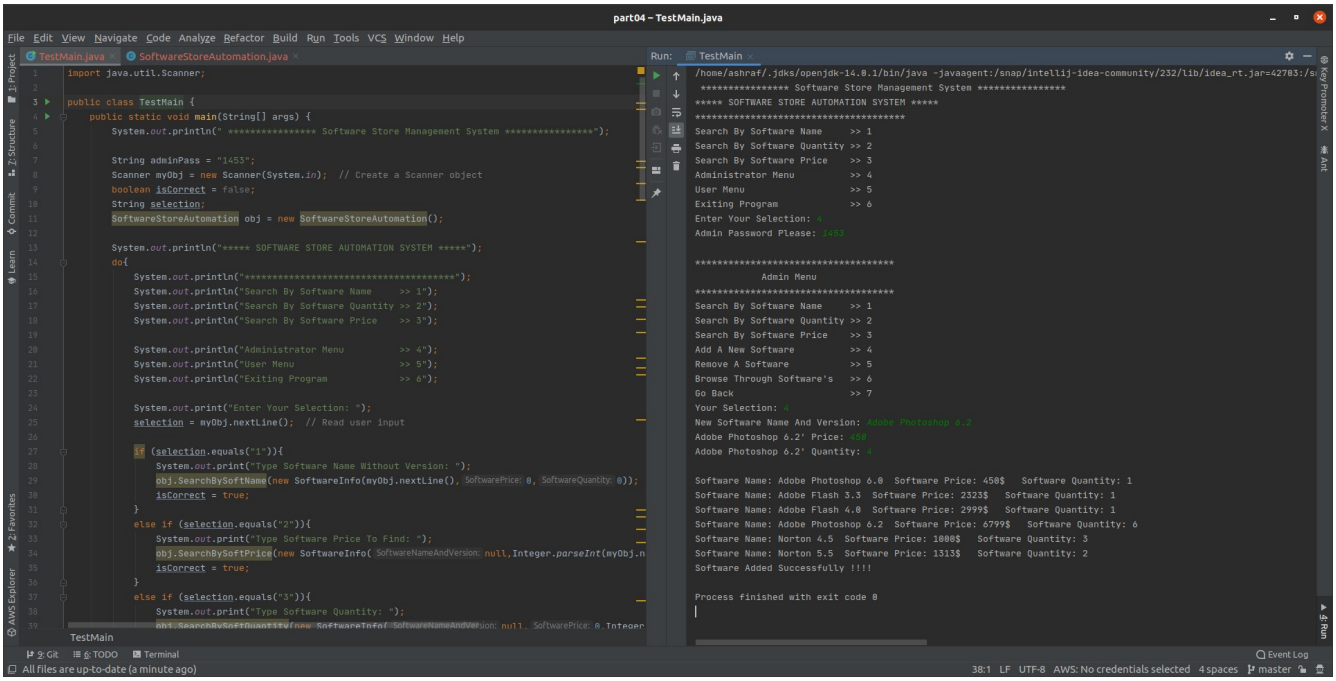
SCREEN SHOTS:

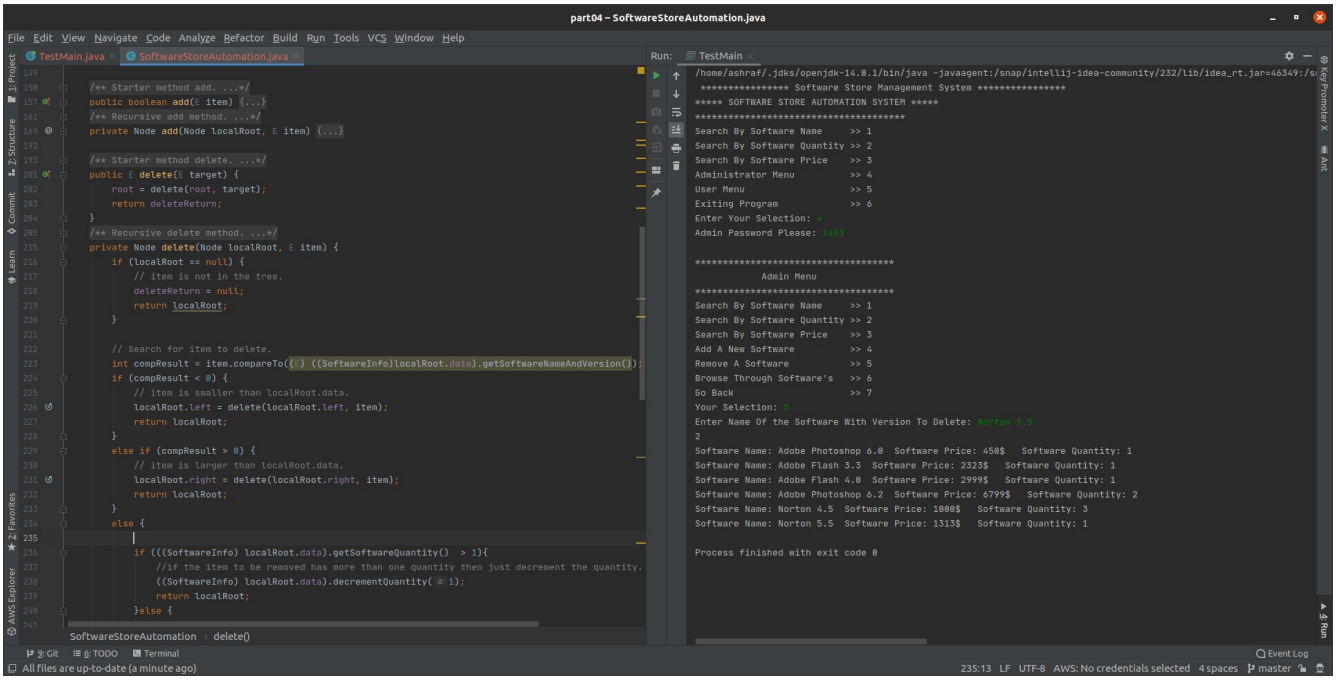


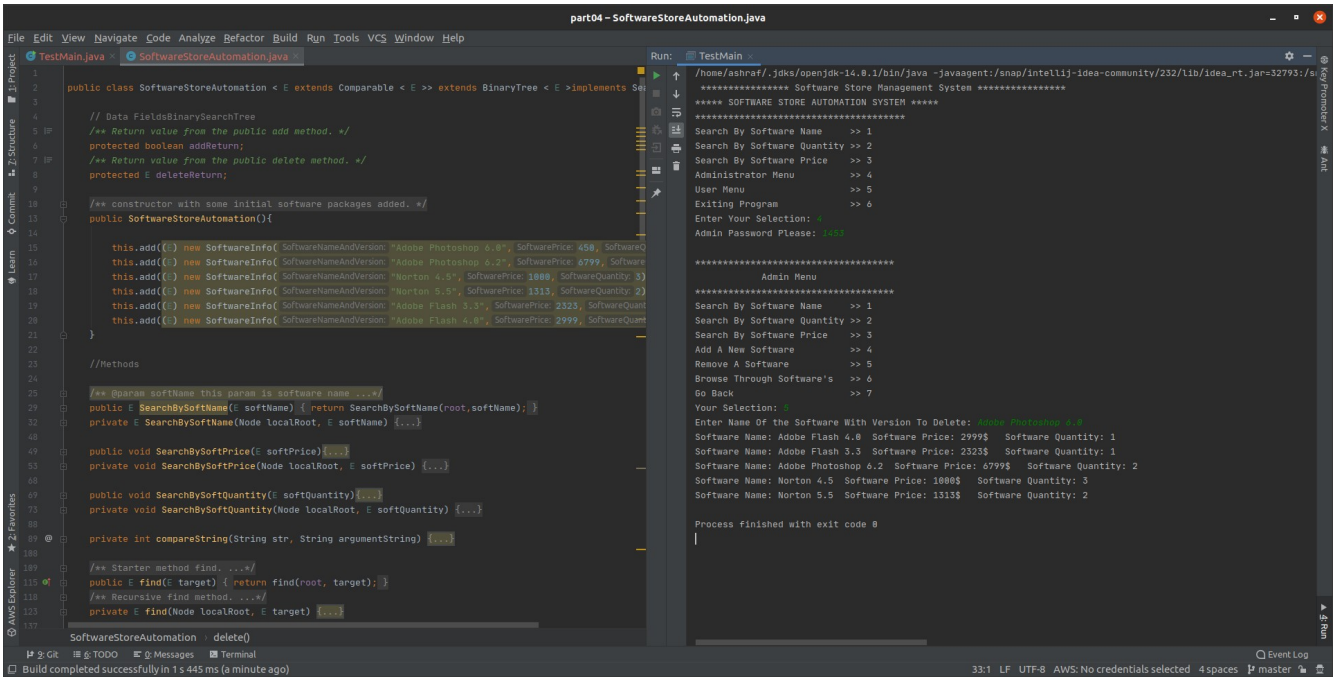


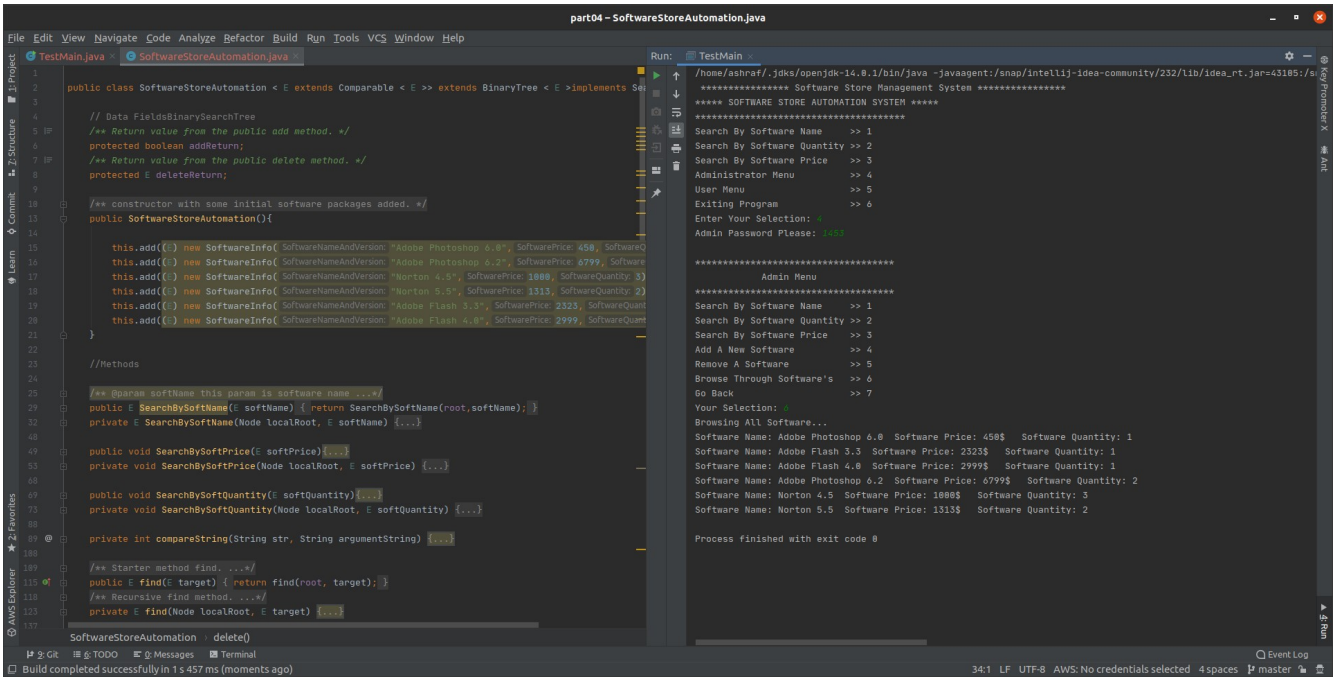

part04 – TestMain.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
import java.util.Scanner;
public class TestMain {
public static void main(String[] args) {
System.out.println(" ********** Software Store Management System **********");
String adminPass = "1453";
Scanner myObj = new Scanner(System.in); // Create a Scanner object
boolean isCorrect = false;
String selection;
SoftwareStoreAutomation obj = new SoftwareStoreAutomation();
System.out.println("***** SOFTWARE STORE AUTOMATION SYSTEM *****");
do{
System.out.println("**************************************");
System.out.println("Search By Software Name >> 1");
System.out.println("Search By Software Quantity >> 2");
System.out.println("Search By Software Price >> 3");
System.out.println("Administrator Menu >> 4");
System.out.println("User Menu >> 5");
System.out.println("Exiting Program >> 6");
System.out.print("Enter Your Selection: ");
selection = myObj.nextLine(); // Read user input
if (selection.equals("1")){
System.out.print("Type Software Name Without Version: ");
obj.SearchBySoftName(new SoftwareInfo(myObj.nextLine(), SoftwarePrice: 0, SoftwareQuantity: 0));
isCorrect = true;
}
else if (selection.equals("2")){
System.out.print("Type Software Price To Find: ");
obj.SearchBySoftPrice(new SoftwareInfo(SoftwareNameAndVersion: null,Integer.parseInt(myObj.n
isCorrect = true;
}
else if (selection.equals("3")){
System.out.print("Type Software Quantity: ");
TestMain
Run: TestMain
/home/ashraf/.jdks/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=44527:/s
********** Software Store Management System **********
***** SOFTWARE STORE AUTOMATION SYSTEM *****
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 3
Type Software Quantity: 2
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 2
Software Name: Norton 5.5 Software Price: 1313$ Software Quantity: 2
Process finished with exit code 0
9: Git
6: TODO
Terminal
Event Log
All files are up-to-date (moments ago)
17:1 LF UTF-8 AWS: No credentials selected 4 spaces master

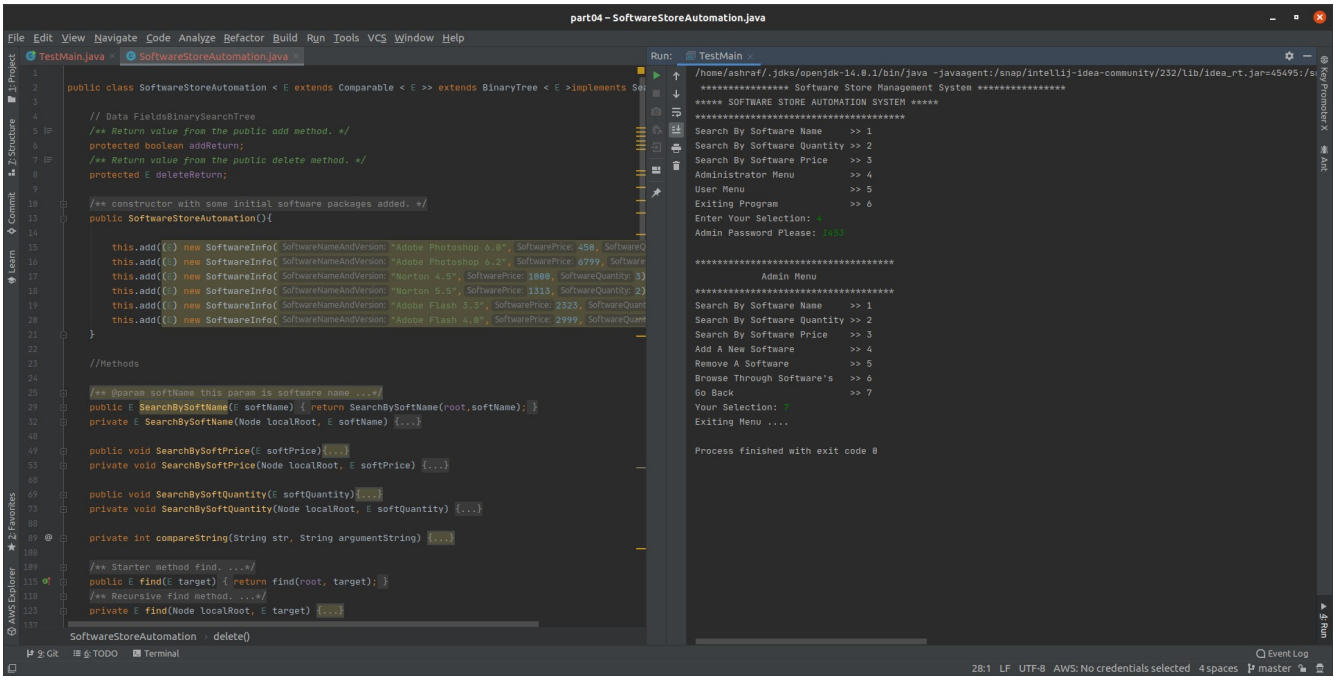

part04 – TestMain.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
import java.util.Scanner;
public class TestMain {
public static void main(String[] args) {
System.out.println(" ********** Software Store Management System **********");
String adminPass = "1453";
Scanner myObj = new Scanner(System.in); // Create a Scanner object
boolean isCorrect = false;
String selection;
SoftwareStoreAutomation obj = new SoftwareStoreAutomation();
System.out.println("***** SOFTWARE STORE AUTOMATION SYSTEM *****");
do{
System.out.println("**************************************");
System.out.println("Search By Software Name >> 1");
System.out.println("Search By Software Quantity >> 2");
System.out.println("Search By Software Price >> 3");
System.out.println("Administrator Menu >> 4");
System.out.println("User Menu >> 5");
System.out.println("Exiting Program >> 6");
System.out.print("Enter Your Selection: ");
selection = myObj.nextLine(); // Read user input
if (selection.equals("1")){
System.out.print("Type Software Name Without Version: ");
obj.SearchBySoftName(new SoftwareInfo(myObj.nextLine(), SoftwarePrice: 0, SoftwareQuantity: 0));
isCorrect = true;
}
else if (selection.equals("2")){
System.out.print("Type Software Price To Find: ");
obj.SearchBySoftPrice(new SoftwareInfo(SoftwareNameAndVersion: null,Integer.parseInt(myObj.n
isCorrect = true;
}
else if (selection.equals("3")){
System.out.print("Type Software Quantity: ");
TestMain
Run: TestMain
********** Software Store Management System **********
***** SOFTWARE STORE AUTOMATION SYSTEM *****
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 4
Admin Password Please: 44
Admin Password Please: 1453
****
Admin Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Add A New Software >> 4
Remove A Software >> 5
Browse Through Software's >> 6
Go Back >> 7
Your Selection: 4
New Software Name And Version: muslim soft 1.1
muslim soft 1.1' Price: 466
muslim soft 1.1' Quantity: 15
Software Name: Adobe Photoshop 6.0 Software Price: 450$ Software Quantity: 1
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 2
Software Name: Norton 4.5 Software Price: 1000$ Software Quantity: 3
Software Name: Norton 5.5 Software Price: 1313$ Software Quantity: 2
Software Name: muslim soft 1.1 Software Price: 466$ Software Quantity: 15
Software Added Successfully !!!!
Process finished with exit code 0
9: Git
6: TODO
Terminal
Event Log
All files are up-to-date (a minute ago)
40:1 LF UTF-8 AWS: No credentials selected 4 spaces master

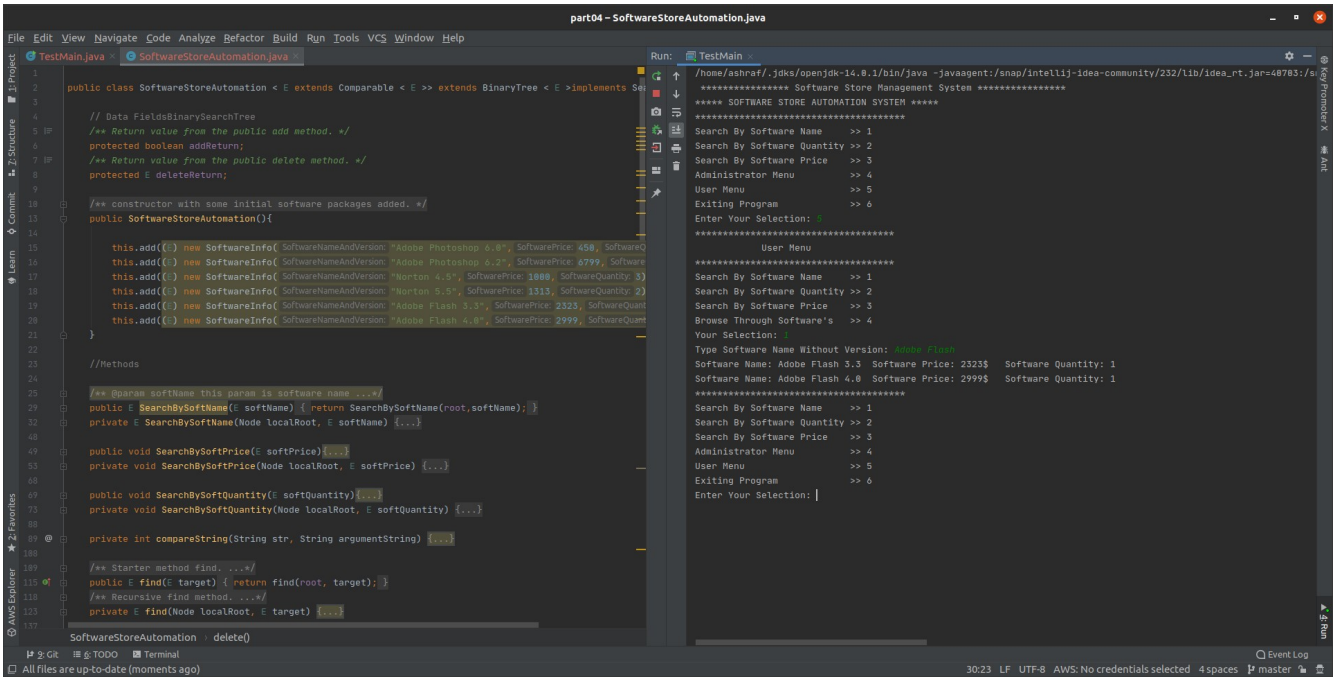

part04 – TestMain.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
import java.util.Scanner;
public class TestMain {
public static void main(String[] args) {
System.out.println(" ********** Software Store Management System **********");
String adminPass = "1453";
Scanner myObj = new Scanner(System.in); // Create a Scanner object
boolean isCorrect = false;
String selection;
SoftwareStoreAutomation obj = new SoftwareStoreAutomation();
System.out.println("***** SOFTWARE STORE AUTOMATION SYSTEM *****");
do{
System.out.println("**************************************");
System.out.println("Search By Software Name >> 1");
System.out.println("Search By Software Quantity >> 2");
System.out.println("Search By Software Price >> 3");
System.out.println("Administrator Menu >> 4");
System.out.println("User Menu >> 5");
System.out.println("Exiting Program >> 6");
System.out.print("Enter Your Selection: ");
selection = myObj.nextLine(); // Read user input
if (selection.equals("1")){
System.out.print("Type Software Name Without Version: ");
obj.SearchBySoftName(new SoftwareInfo(myObj.nextLine(), SoftwarePrice: 0, SoftwareQuantity: 0));
isCorrect = true;
}
else if (selection.equals("2")){
System.out.print("Type Software Price To Find: ");
obj.SearchBySoftPrice(new SoftwareInfo(SoftwareNameAndVersion: null,Integer.parseInt(myObj.n
isCorrect = true;
}
else if (selection.equals("3")){
System.out.print("Type Software Quantity: ");
TestMain
Run: TestMain
/home/ashraf/.jdks/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=42703:/s
********** Software Store Management System **********
***** SOFTWARE STORE AUTOMATION SYSTEM *****
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 4
Admin Password Please: 1453
****
Admin Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Add A New Software >> 4
Remove A Software >> 5
Browse Through Software's >> 6
Go Back >> 7
Your Selection: 4
New Software Name And Version: Adobe Photoshop 6.2
Adobe Photoshop 6.2' Price: 450
Adobe Photoshop 6.2' Quantity: 4
Software Name: Adobe Photoshop 6.0 Software Price: 450$ Software Quantity: 1
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 6
Software Name: Norton 4.5 Software Price: 1080$ Software Quantity: 3
Software Name: Norton 5.5 Software Price: 1313$ Software Quantity: 2
Software Added Successfully !!!!
Process finished with exit code 0
9: Git
6: TODO
Terminal
Event Log
All files are up-to-date (a minute ago)
38:1 LF UTF-8 AWS: No credentials selected 4 spaces master

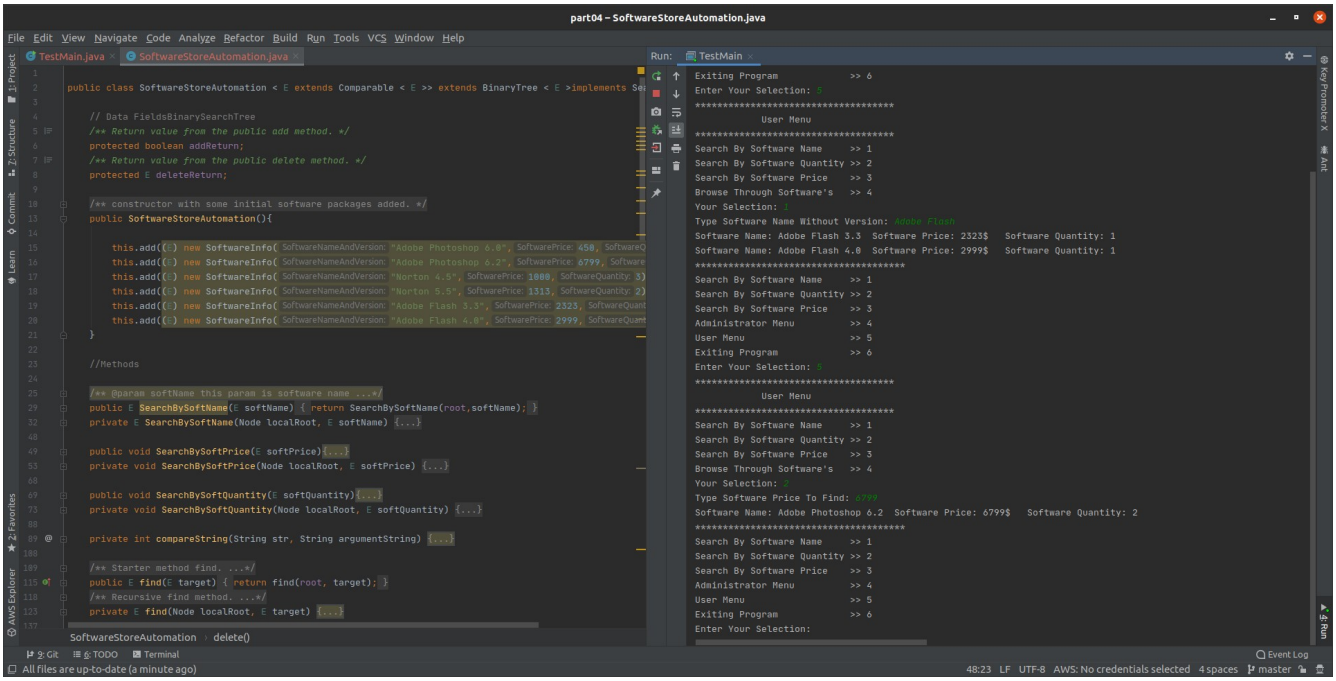

part04 – SoftwareStoreAutomation.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
/** Starter method add. ...*/
public boolean add(E item) {...}
/** Recursive add method. ...*/
private Node add(Node localRoot, E item) {...}
/** Starter method delete. ...*/
public E delete(E target) {
root = delete(root, target);
return deleteReturn;
}
/** Recursive delete method. ...*/
private Node delete(Node localRoot, E item) {
if (localRoot == null) {
// item is not in the tree.
deleteReturn = null;
return localRoot;
}
// Search for item to delete.
int compResult = item.compareTo((E) ((SoftwareInfo)localRoot.data).getSoftwareNameAndVersion());
if (compResult < 0) {
// item is smaller than localRoot.data.
localRoot.left = delete(localRoot.left, item);
return localRoot;
}
else if (compResult > 0) {
// item is larger than localRoot.data.
localRoot.right = delete(localRoot.right, item);
return localRoot;
}
else {
if (((SoftwareInfo) localRoot.data).getSoftwareQuantity() > 1){
//if the item to be removed has more than one quantity then just decrement the quantity.
((SoftwareInfo) localRoot.data).decrementQuantity(d: 1);
return localRoot;
}else {
SoftwareStoreAutomation › delete()
Run: TestMain
/home/ashraf/.jdks/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=46349:/s
********** Software Store Management System **********
***** SOFTWARE STORE AUTOMATION SYSTEM *****
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 4
Admin Password Please: 1453
****
Admin Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Add A New Software >> 4
Remove A Software >> 5
Browse Through Software's >> 6
Go Back >> 7
Your Selection: 5
Enter Name Of the Software With Version To Delete: Norton 5.5
2
Software Name: Adobe Photoshop 6.0 Software Price: 450$ Software Quantity: 1
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 2
Software Name: Norton 4.5 Software Price: 1080$ Software Quantity: 3
Software Name: Norton 5.5 Software Price: 1313$ Software Quantity: 1
Process finished with exit code 0
9: Git
6: TODO
Terminal
Event Log
All files are up-to-date (a minute ago)
235:13 LF UTF-8 AWS: No credentials selected 4 spaces master

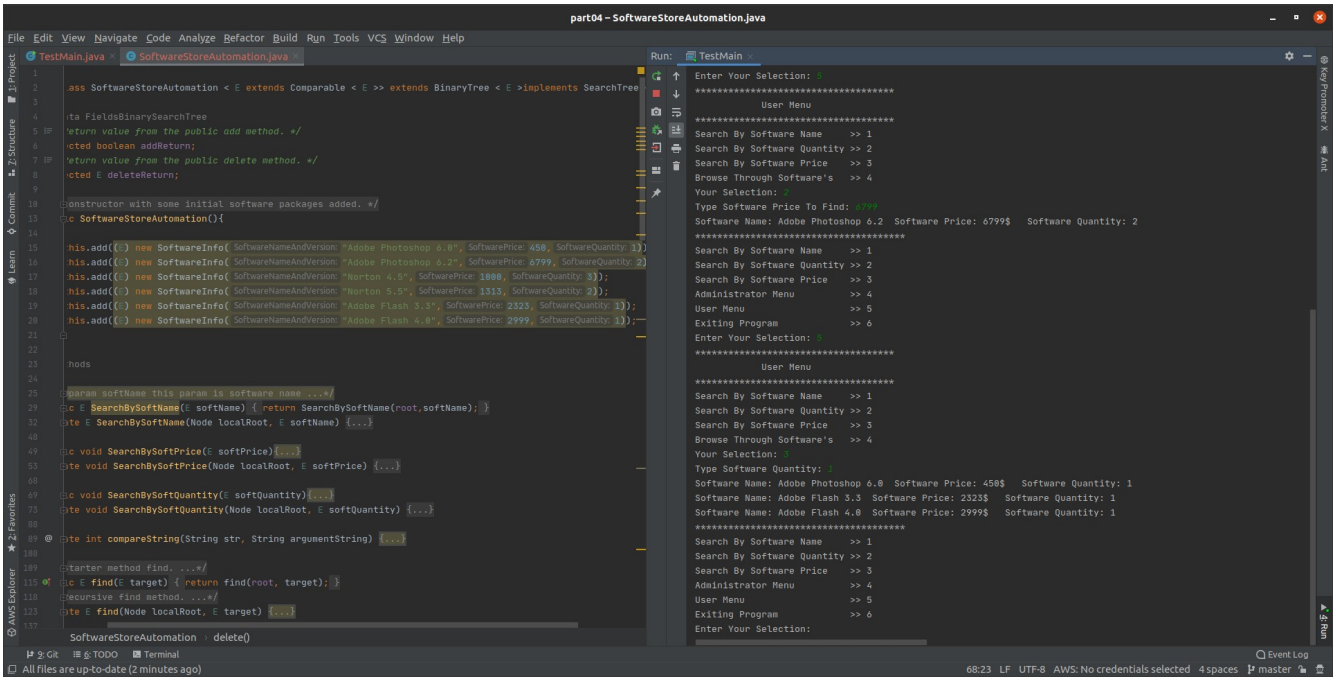

part04 – SoftwareStoreAutomation.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
public class SoftwareStoreAutomation < E extends Comparable < E >> extends BinaryTree < E >implements Se
// Data FieldsBinarySearchTree
/** Return value from the public add method. */
protected boolean addReturn;
/** Return value from the public delete method. */
protected E deleteReturn;
/** constructor with some initial software packages added. */
public SoftwareStoreAutomation(){
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.0", SoftwarePrice: 450, SoftwareQ
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.2", SoftwarePrice: 6799, Software
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 4.5", SoftwarePrice: 1000, SoftwareQuantity: 3)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 5.5", SoftwarePrice: 1313, SoftwareQuantity: 2)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 3.3", SoftwarePrice: 2323, SoftwareQuant
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 4.0", SoftwarePrice: 2999, SoftwareQuant
}
//Methods
/** @param softName this param is software name ...*/
public E SearchBySoftName(E softName) { return SearchBySoftName(root,softName); }
private E SearchBySoftName(Node localRoot, E softName) {...}
public void SearchBySoftPrice(E softPrice){...}
private void SearchBySoftPrice(Node localRoot, E softPrice) {...}
public void SearchBySoftQuantity(E softQuantity){...}
private void SearchBySoftQuantity(Node localRoot, E softQuantity) {...}
private int compareString(String str, String argumentString) {...}
/** Starter method find. ...*/
public E find(E target) { return find(root, target); }
/** Recursive find method. ...*/
private E find(Node localRoot, E target) {...}
SoftwareStoreAutomation › delete()
Run: TestMain
/home/ashraf/.jdks/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=52793:/s
**************** Software Store Management System ****************
***** SOFTWARE STORE AUTOMATION SYSTEM *****
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 4
Admin Password Please: 1453
****
Admin Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Add A New Software >> 4
Remove A Software >> 5
Browse Through Software's >> 6
Go Back >> 7
Your Selection: 5
Enter Name Of the Software With Version To Delete: Adobe Photoshop 6.0
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 2
Software Name: Norton 4.5 Software Price: 1000$ Software Quantity: 3
Software Name: Norton 5.5 Software Price: 1313$ Software Quantity: 2
Process finished with exit code 0
9: Git 6: TODO 0: Messages Terminal
Event Log
Build completed successfully in 1 s 445 ms (a minute ago)
33:1 LF UTF-8 AWS: No credentials selected 4 spaces master

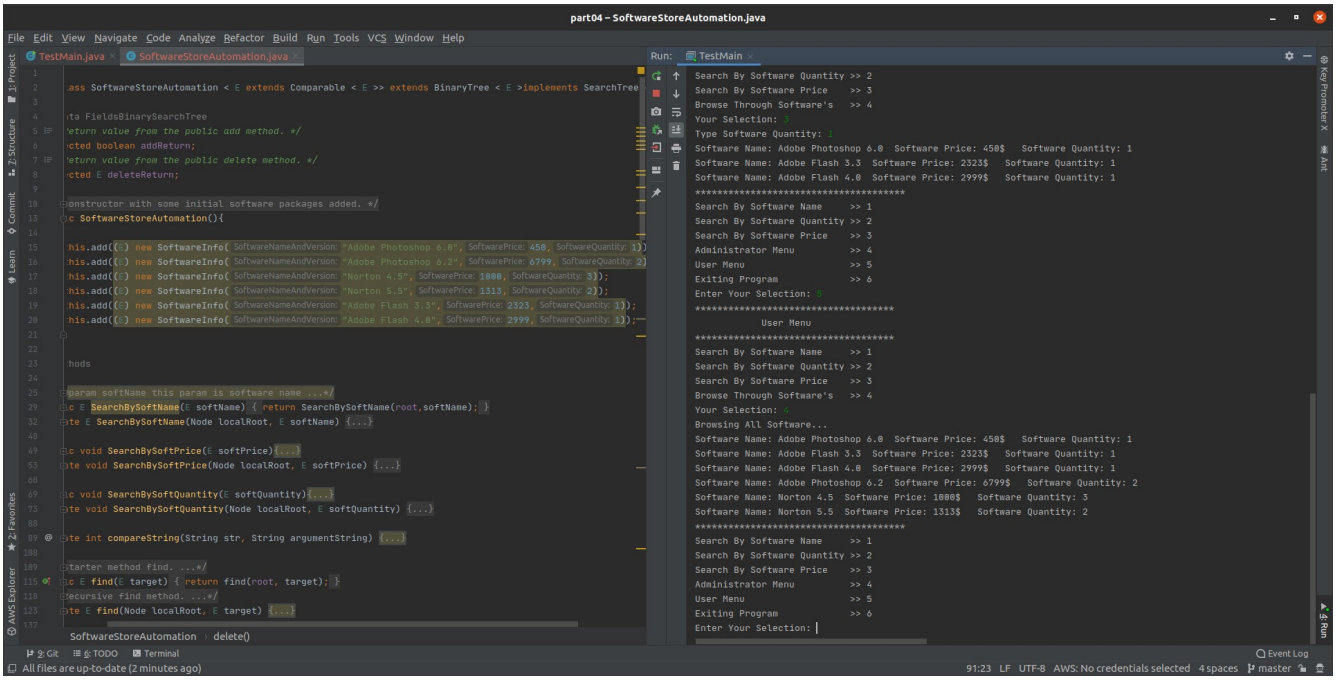

part04 – SoftwareStoreAutomation.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
public class SoftwareStoreAutomation < E extends Comparable < E >> extends BinaryTree < E >implements Se
// Data FieldsBinarySearchTree
/** Return value from the public add method. */
protected boolean addReturn;
/** Return value from the public delete method. */
protected E deleteReturn;
/** constructor with some initial software packages added. */
public SoftwareStoreAutomation(){
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.0", SoftwarePrice: 450, SoftwareQ
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.2", SoftwarePrice: 6799, Software
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 4.5", SoftwarePrice: 1000, SoftwareQuantity: 3)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 5.5", SoftwarePrice: 1313, SoftwareQuantity: 2)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 3.3", SoftwarePrice: 2323, SoftwareQuant
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 4.0", SoftwarePrice: 2999, SoftwareQuant
}
//Methods
/** @param softName this param is software name ...*/
public E SearchBySoftName(E softName) { return SearchBySoftName(root,softName); }
private E SearchBySoftName(Node localRoot, E softName) {...}
public void SearchBySoftPrice(E softPrice){...}
private void SearchBySoftPrice(Node localRoot, E softPrice) {...}
public void SearchBySoftQuantity(E softQuantity){...}
private void SearchBySoftQuantity(Node localRoot, E softQuantity) {...}
private int compareString(String str, String argumentString) {...}
/** Starter method find. ...*/
public E find(E target) { return find(root, target); }
/** Recursive find method. ...*/
private E find(Node localRoot, E target) {...}
SoftwareStoreAutomation › delete()
Run: TestMain
/home/ashraf/.jdks/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=43185:/s
**************** Software Store Management System ****************
***** SOFTWARE STORE AUTOMATION SYSTEM *****
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 4
Admin Password Please: 1453
****
Admin Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Add A New Software >> 4
Remove A Software >> 5
Browse Through Software's >> 6
Go Back >> 7
Your Selection: 6
Browsing All Software...
Software Name: Adobe Photoshop 6.0 Software Price: 450$ Software Quantity: 1
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 2
Software Name: Norton 4.5 Software Price: 1000$ Software Quantity: 3
Software Name: Norton 5.5 Software Price: 1313$ Software Quantity: 2
Process finished with exit code 0
9: Git 6: TODO 0: Messages Terminal
Event Log
Build completed successfully in 1 s 457 ms (moments ago)
34:1 LF UTF-8 AWS: No credentials selected 4 spaces master


part04 – SoftwareStoreAutomation.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
public class SoftwareStoreAutomation < E extends Comparable < E >> extends BinaryTree < E >implements Se
// Data FieldsBinarySearchTree
/** Return value from the public add method. */
protected boolean addReturn;
/** Return value from the public delete method. */
protected E deleteReturn;
/** constructor with some initial software packages added. */
public SoftwareStoreAutomation(){
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.0", SoftwarePrice: 450, SoftwareQ
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.2", SoftwarePrice: 6799, Software
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 4.5", SoftwarePrice: 1000, SoftwareQuantity: 3)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 5.5", SoftwarePrice: 1313, SoftwareQuantity: 2)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 3.3", SoftwarePrice: 2323, SoftwareQuant
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 4.0", SoftwarePrice: 2999, SoftwareQuant
}
//Methods
/** @param softName this param is software name ...*/
public E SearchBySoftName(E softName) { return SearchBySoftName(root,softName); }
private E SearchBySoftName(Node localRoot, E softName) {...}
public void SearchBySoftPrice(E softPrice){...}
private void SearchBySoftPrice(Node localRoot, E softPrice) {...}
public void SearchBySoftQuantity(E softQuantity){...}
private void SearchBySoftQuantity(Node localRoot, E softQuantity) {...}
private int compareString(String str, String argumentString) {...}
/** Starter method find. ...*/
public E find(E target) { return find(root, target); }
/** Recursive find method. ...*/
private E find(Node localRoot, E target) {...}
SoftwareStoreAutomation › delete()
Run: TestMain
/home/ashraf/.jdks/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=45495:/s
Software Store Management System
SOFTWARE STORE AUTOMATION SYSTEM
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 4
Admin Password Please: 1453
Admin Menu
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Add A New Software >> 4
Remove A Software >> 5
Browse Through Software's >> 6
Go Back >> 7
Your Selection: 7
Exiting Menu ....
Process finished with exit code 0
9: Git 6: TODO Terminal
Event Log
28:1 LF UTF-8 AWS: No credentials selected 4 spaces master


part04 – SoftwareStoreAutomation.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
public class SoftwareStoreAutomation < E extends Comparable < E >> extends BinaryTree < E >implements Se
// Data FieldsBinarySearchTree
/** Return value from the public add method. */
protected boolean addReturn;
/** Return value from the public delete method. */
protected E deleteReturn;
/** constructor with some initial software packages added. */
public SoftwareStoreAutomation(){
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.0", SoftwarePrice: 450, SoftwareQ
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.2", SoftwarePrice: 6799, Software
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 4.5", SoftwarePrice: 1000, SoftwareQuantity: 3)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 5.5", SoftwarePrice: 1313, SoftwareQuantity: 2)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 3.3", SoftwarePrice: 2323, SoftwareQuant
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 4.0", SoftwarePrice: 2999, SoftwareQuant
}
//Methods
/** @param softName this param is software name ...*/
public E SearchBySoftName(E softName) { return SearchBySoftName(root,softName); }
private E SearchBySoftName(Node localRoot, E softName) {...}
public void SearchBySoftPrice(E softPrice){...}
private void SearchBySoftPrice(Node localRoot, E softPrice) {...}
public void SearchBySoftQuantity(E softQuantity){...}
private void SearchBySoftQuantity(Node localRoot, E softQuantity) {...}
private int compareString(String str, String argumentString) {...}
/** Starter method find. ...*/
public E find(E target) { return find(root, target); }
/** Recursive find method. ...*/
private E find(Node localRoot, E target) {...}
SoftwareStoreAutomation › delete()
Run: TestMain
/home/ashraf/.jdks/openjdk-14.0.1/bin/java -javaagent:/snap/intellij-idea-community/232/lib/idea_rt.jar=40703:/s
Software Store Management System
SOFTWARE STORE AUTOMATION SYSTEM
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 5
User Menu
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Browse Through Software's >> 4
Your Selection: 1
Type Software Name Without Version: Adobe Flash
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection:
9: Git 6: TODO Terminal
Event Log
All files are up-to-date (moments ago)
30:23 LF UTF-8 AWS: No credentials selected 4 spaces master


part04 – SoftwareStoreAutomation.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
public class SoftwareStoreAutomation < E extends Comparable < E >> extends BinaryTree < E >implements Sea
// Data FieldsBinarySearchTree
/** Return value from the public add method. */
protected boolean addReturn;
/** Return value from the public delete method. */
protected E deleteReturn;
/** constructor with some initial software packages added. */
public SoftwareStoreAutomation(){
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.0", SoftwarePrice: 450, SoftwareQ
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Photoshop 6.2", SoftwarePrice: 6799, Software
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 4.5", SoftwarePrice: 1800, SoftwareQuantity: 3)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Norton 5.5", SoftwarePrice: 1313, SoftwareQuantity: 2)
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 3.3", SoftwarePrice: 2323, SoftwareQuant
this.add((E) new SoftwareInfo(SoftwareNameAndVersion: "Adobe Flash 4.0", SoftwarePrice: 2999, SoftwareQuant
}
//Methods
/** @param softName this param is software name ...*/
public E SearchBySoftName(E softName) { return SearchBySoftName(root,softName); }
private E SearchBySoftName(Node localRoot, E softName) {...}
public void SearchBySoftPrice(E softPrice){...}
private void SearchBySoftPrice(Node localRoot, E softPrice) {...}
public void SearchBySoftQuantity(E softQuantity){...}
private void SearchBySoftQuantity(Node localRoot, E softQuantity) {...}
private int compareString(String str, String argumentString) {...}
/** Starter method find. ...*/
public E find(E target) { return find(root, target); }
/** Recursive find method. ...*/
private E find(Node localRoot, E target) {...}
SoftwareStoreAutomation › delete()
Run: TestMain
Exiting Program >> 6
Enter Your Selection: 6
****
User Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Browse Through Software's >> 4
Your Selection: 1
Type Software Name Without Version: Adobe Flash
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 6
****
User Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Browse Through Software's >> 4
Your Selection: 2
Type Software Price To Find: 6799
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 2
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection:
9: Git 6: TODO Terminal
Event Log
All files are up-to-date (a minute ago)
48:23 LF UTF-8 AWS: No credentials selected 4 spaces master


part04 – SoftwareStoreAutomation.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TestMain.java
SoftwareStoreAutomation.java
Run: TestMain
Enter Your Selection: 6
****
User Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Browse Through Software's >> 4
Your Selection: 2
Type Software Price To Find: 6799
Software Name: Adobe Photoshop 6.2 Software Price: 6799$ Software Quantity: 2
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection: 6
****
User Menu
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Browse Through Software's >> 4
Your Selection: 3
Type Software Quantity: 2
Software Name: Adobe Photoshop 6.0 Software Price: 450$ Software Quantity: 1
Software Name: Adobe Flash 3.3 Software Price: 2323$ Software Quantity: 1
Software Name: Adobe Flash 4.0 Software Price: 2999$ Software Quantity: 1
****
Search By Software Name >> 1
Search By Software Quantity >> 2
Search By Software Price >> 3
Administrator Menu >> 4
User Menu >> 5
Exiting Program >> 6
Enter Your Selection:
SoftwareStoreAutomation › delete()
9: Git 6: TODO Terminal
Event Log
All files are up-to-date (2 minutes ago)
68:23 LF UTF-8 AWS: No credentials selected 4 spaces master



TEST CASES:

Test Case ID	Test Scenarios	Test Data	Expected Results	Actual Result	Pass/Fail
t0	Confirm admin password	input	1453	1453	Pass
t1	Confirm admin password	input	1453	Any number other than 1453	Fail
t2	Add new software	add(softwareinfo)	software name version price,Quantity	If all correct add the software to store	Pass
t3	Add new software	add(softwareinfo)	software name version price,Quantity	If any one not correct do not add to store	fail
t4	Remove software	remove(software name)	Software name n	If all correct then remove the software from store	Pass

t5	Remove software	remove(software name)	software name	If any not correct	Fail
t6	browse softwares	browse()		Print all the softwares	Pass
t7	Search for a software by name	searachBysoftName(String)	software name	If all correct/if any one not correct	Pass/fail
t8	Search for a software by price	searchByPrice(integer)	input software price	Print all softs with same price	Pass
t9	Search for a software by quantity	searchByQuantity(integer)	input software Quntity	Print all soft with same Quantity	Pass

PROBLEM SOL APPROACH:

→ system start with menu
→ by selecting different number you can navigate through different menus
→ three main menu
    1:main manu
    2:admin menu
    3:user menu
→ search is public and can be found in any menus
→ add method:
    - while adding if we had the same software name and version regardless of price and quantity then we simply increase the quantity of that software in the store system and update the quantity(while addin a new software we can give as much software quantity as we want for examples if we had a software X with quantity 5 in our store and we want to add the same software X to the system with given quantity of 3 then the total quantity of the software X will be 8) otherwise we add new software with new information taken from use.

→ remove
- the reverse is applicable to remove for examples we have a software named X in our system with quantity more than 1 then if we want to remove the software X form system it will just decrease the quantity of the software instead of removing is from the system and otherwise if we had software X with quantity of 1 then it will remove the software with all information related to that software from the system.

→ browse method:
    – browse method will print all software’s available in the store.

→ as a data structure I made use of binary search tree by implementing Comparable interface.