Vivek Modi(A00268807)

Mobile management system.

# Abstract

Mobile Management system is a management system which will search for a model of mobile if the customer wishes to buy it. The objective of the system is to add the Details of a Particular brand mobile i.e. Price and Model once the mobile phone model is added the customer section you will get all the details of the mobile in the drop down list fetched and once clicking on particular model adding other details of customer you can print the bill from the customer billing section which is in customer search module which makes this system as an perfect billing system with respect to mobile management as product gets managed and used in other part’s Mobile Management System aims to create a platform where the user can access the system add mobiles and customers and generate the reports.

# Introduction

Mobile management system tends to deliver user friendly access to the end user as it is very simple to add the model and price after adding the model it gets updated to the customer module from which you can add the customer bill and the you can search and print the same. It is a combination of various features of java programming who’s goal is to make user friendly system.

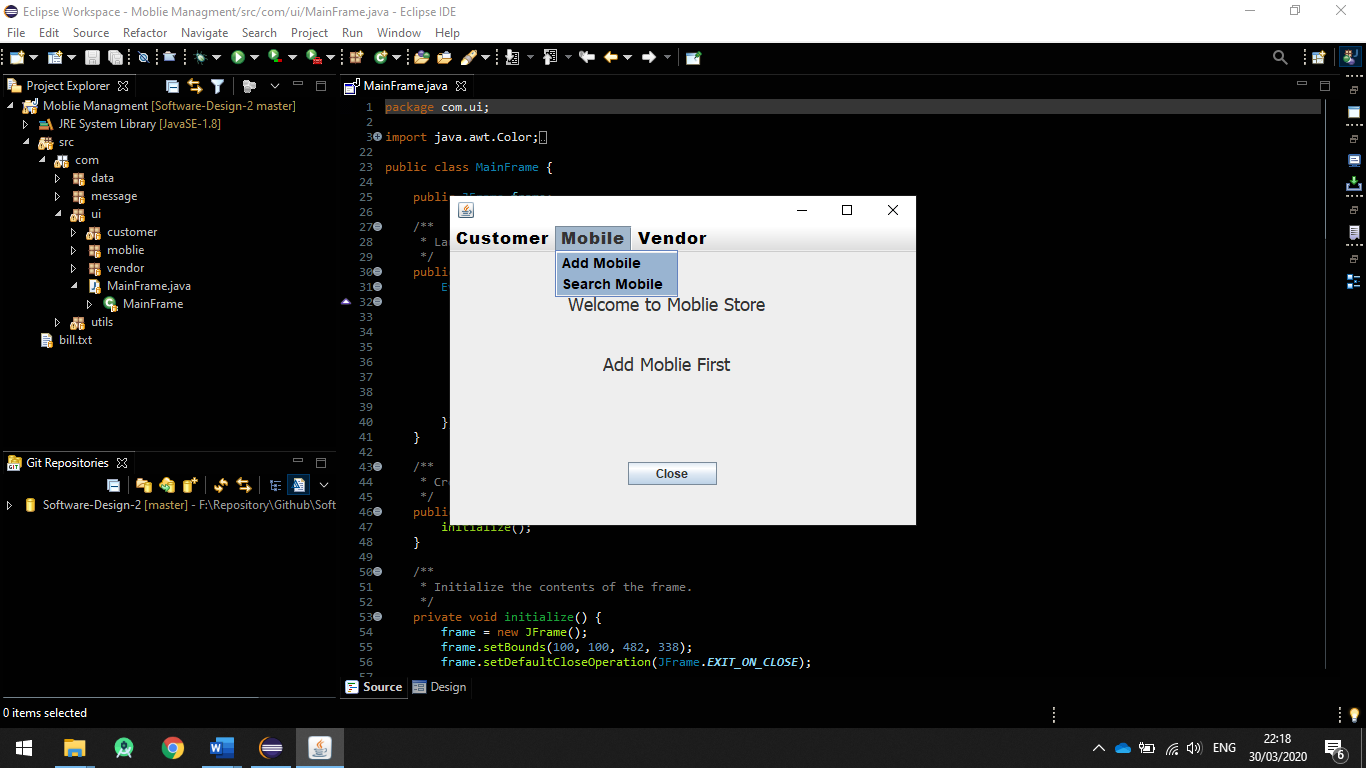
# Features

Various combination of java features and collection frameworks are used in this system which are as follows:-

1. Use of Binary search tree and hash table at different places for multiple time to search and insert the data in modules.
2. Priority Queue: -
   1. After inserting the data into the hash table used priority queue to count the number of times the product i.e. Mobile phone is searched so that we can get the number of times a mobile is bought by the customer.
3. Singleton: -
   1. Singleton is used so that it can avoid creating many instances instead just creating one instance which helped to hold object for **e.g**. used to hold the object of priority queue in the program so instead of creating new instance every time just using one instance all the time.
4. Visitor: -
   1. Visitor pattern used to add new model without making any changes to other features and its hierarchy.
5. Generic: -
   1. Use of generic enables to specify with a single declaration or multiple declaration which saves compile time and also helps to identify invalid types.
6. Comparator: -
   1. Comparator is used in class mobile comparator to compare the number of counts of search between two model, count is used in priority queue before using this count comparing to get the comparison value.
7. Tree Set: -
   1. This Data structure is used to store the data, which is sorted, on a red/black tree algorithm, this provides O(log(N)) complexity for operations. This is used in add Customer Module.

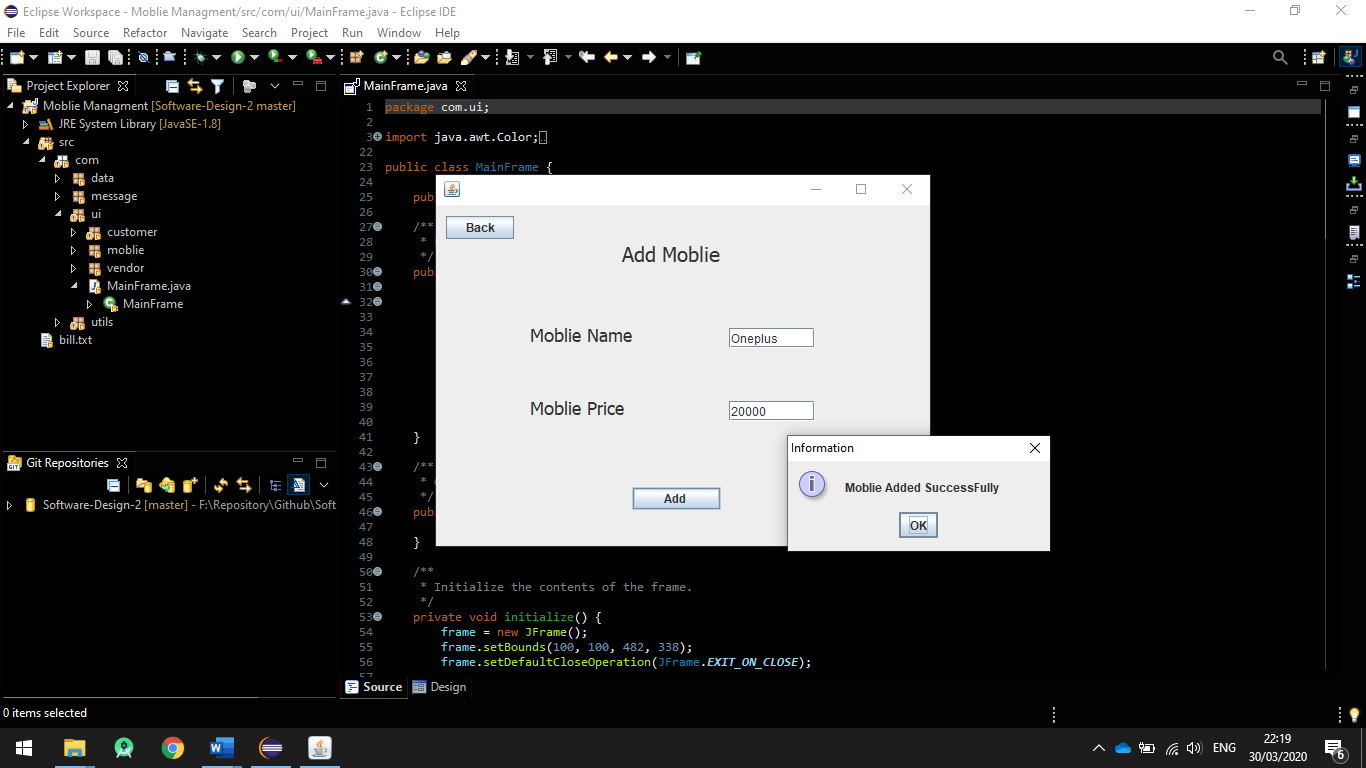
# SnapShots: -

# Mainframe

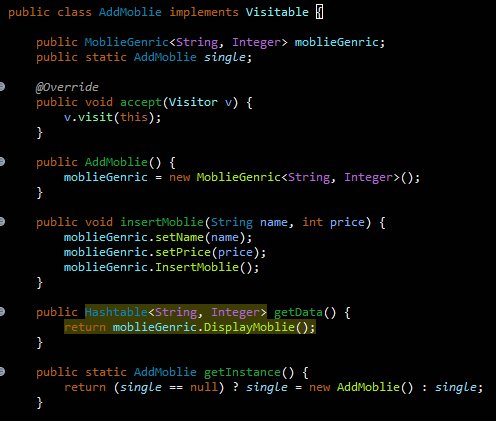


This is the home screen of the application where you can find the Customer, Mobile and vendor as drop-down list.

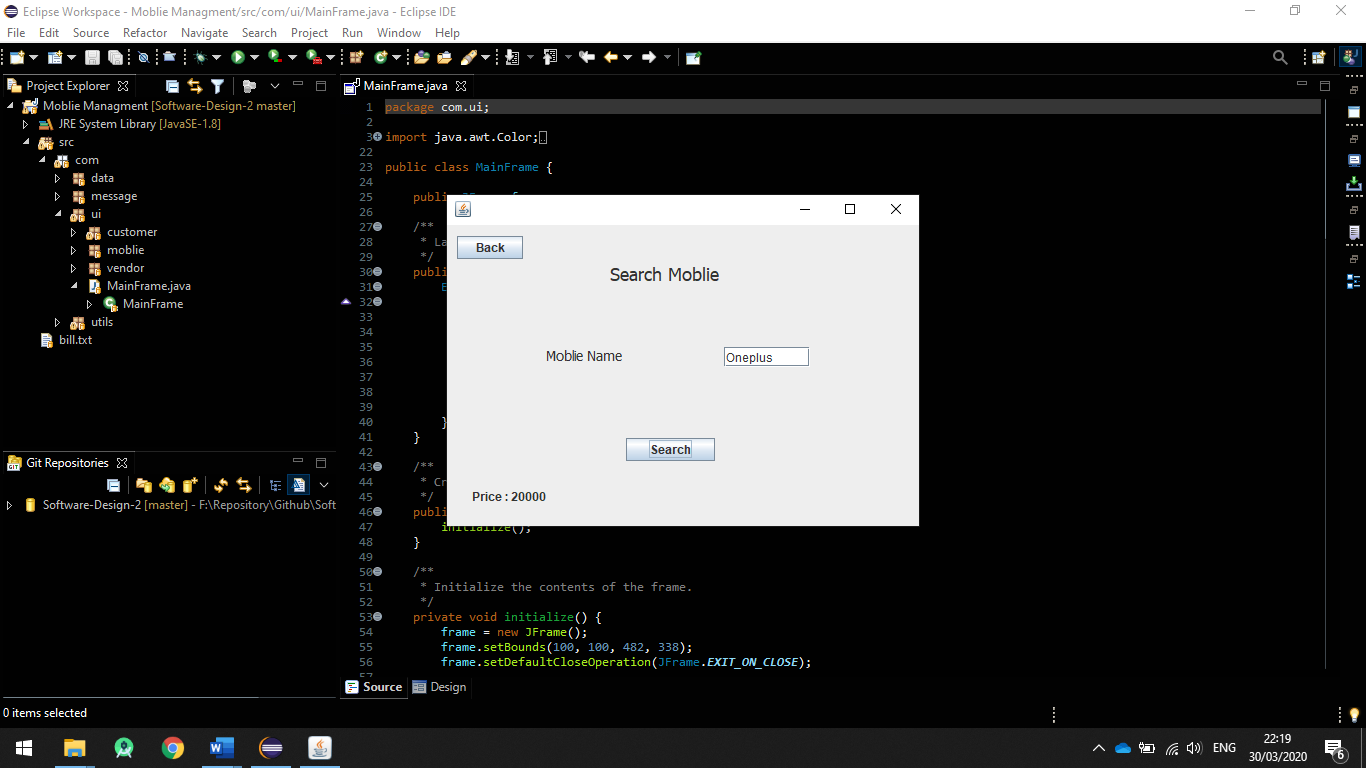
# Add Mobile



This is the Add mobile menu from drop down in which you need to add the mobile name and price in order to get it in the customer billing module.

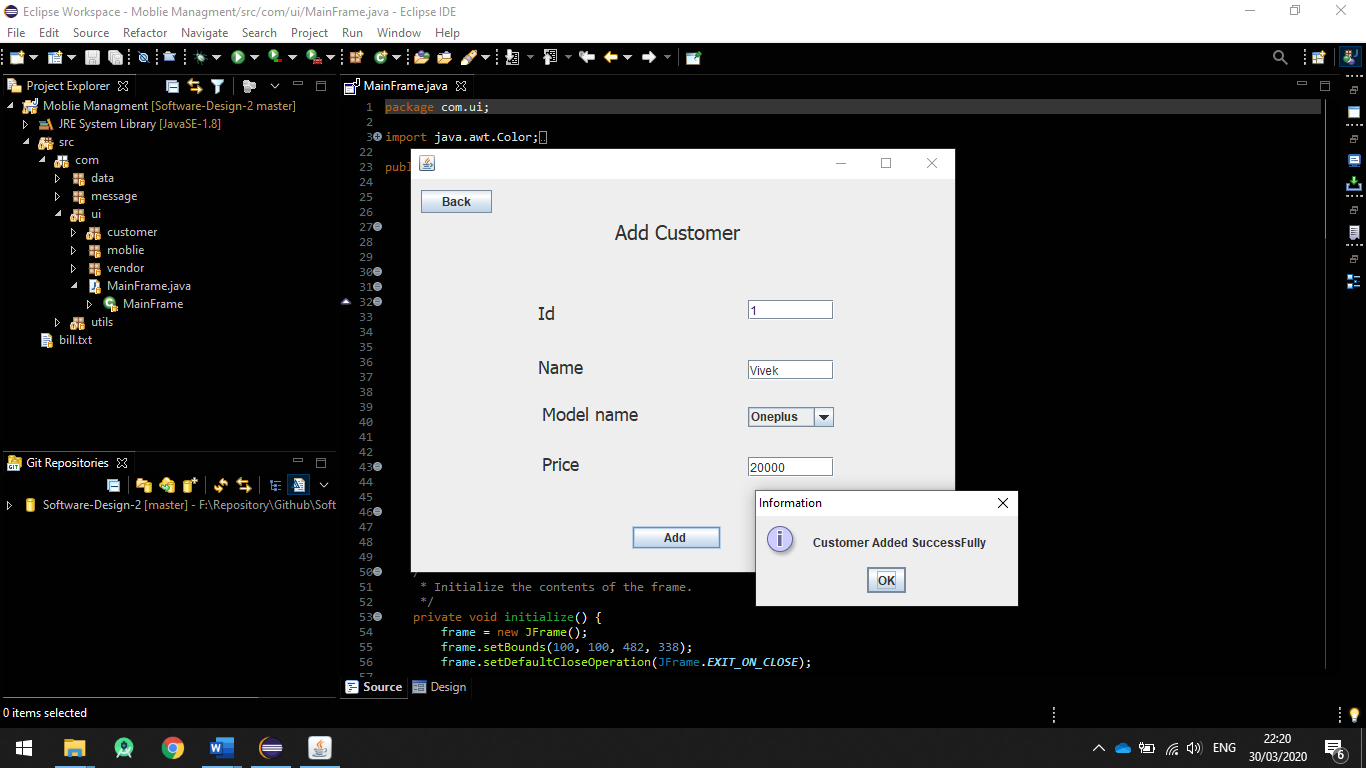


# 3.search



This is search mobile module in which you cant put the mobile phone name and it will show the price if present or else it will show an error.

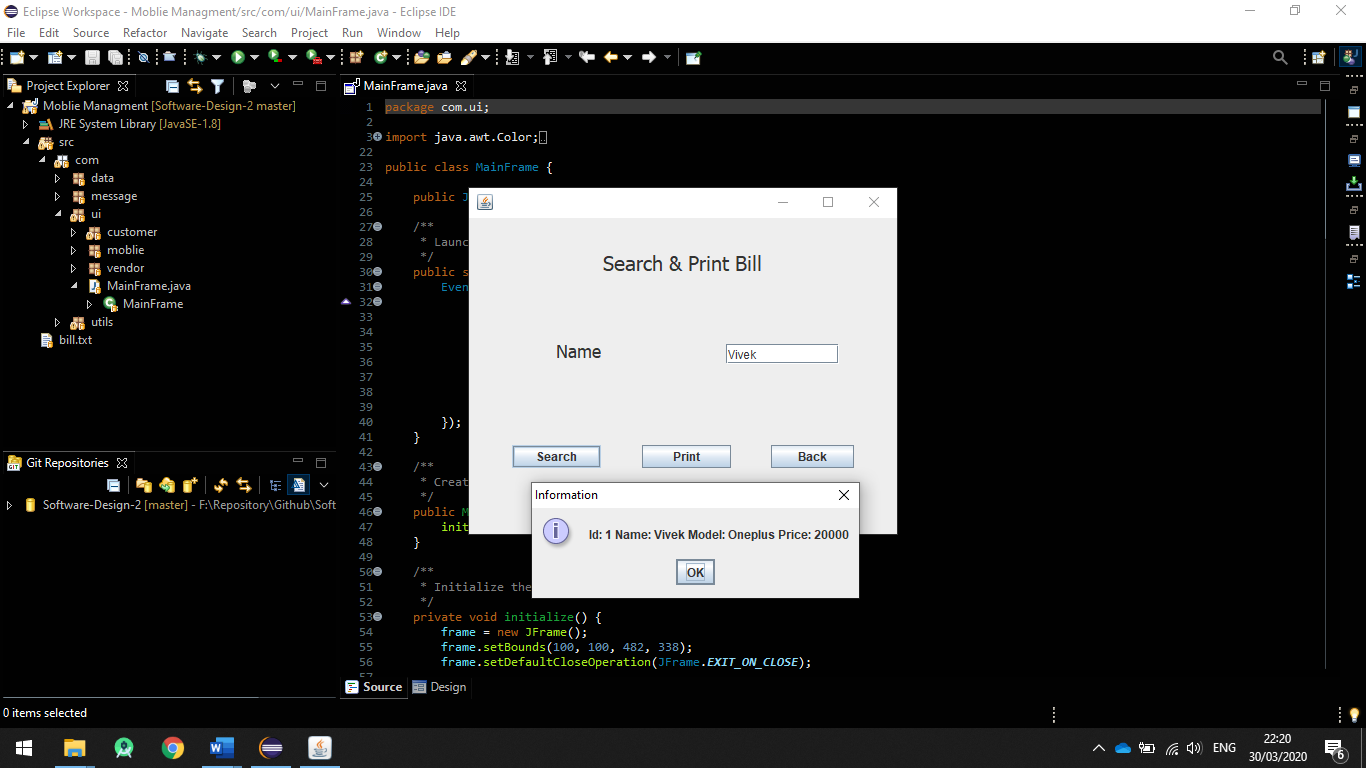


4. Add customer with Sortedset

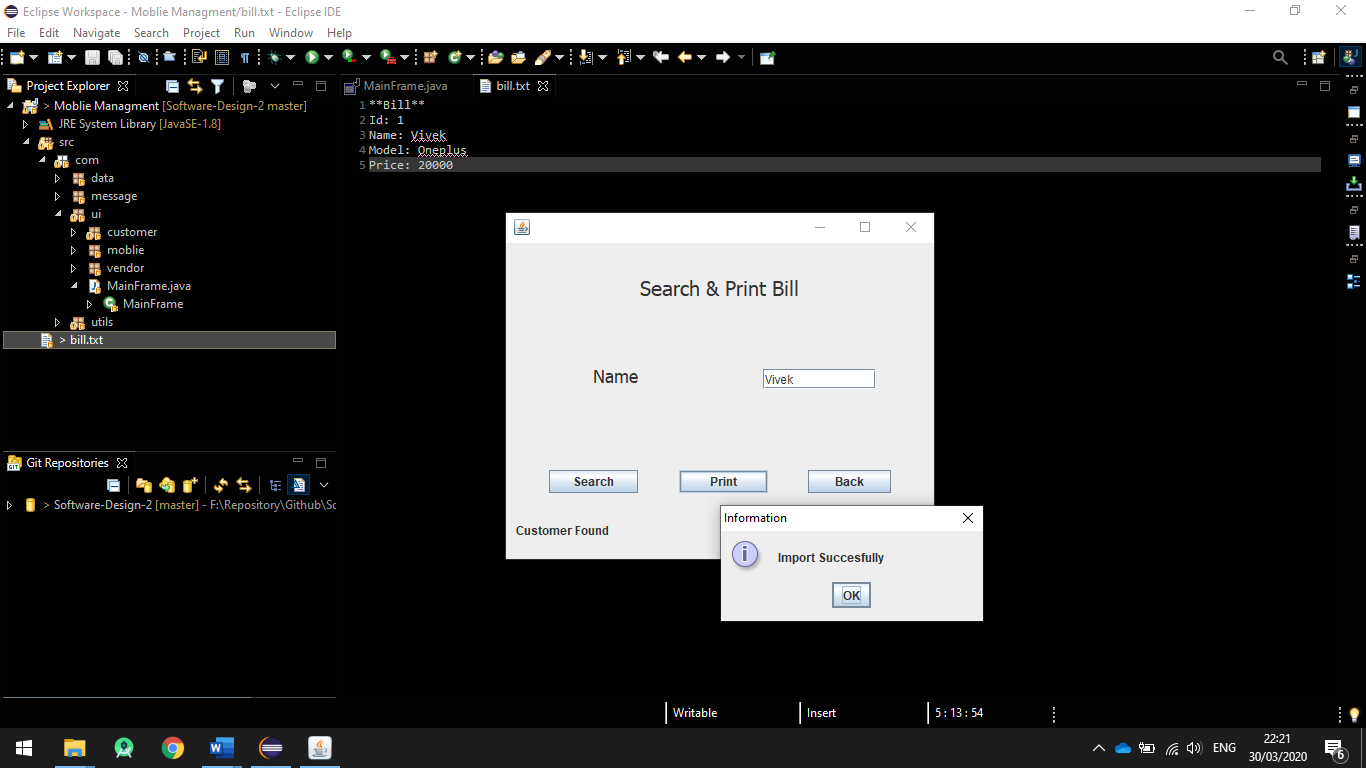
Adding the customer with mobile name already present in the drop down with price and after clicking on button add show customer added successfully.

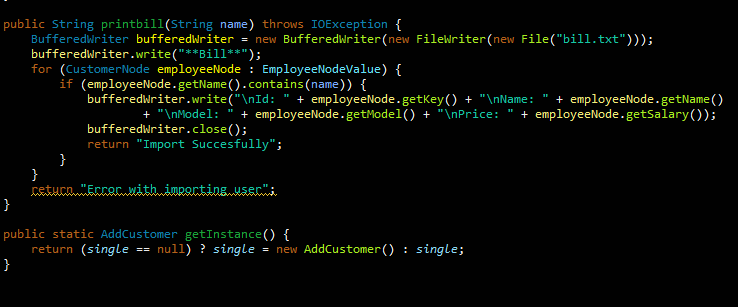


**5.Search and Print Bill: -**



Searching the Customer with customer name it shows details and the mobile phone which customer bought with price, this also can be printed which will make it as an bill that will be imported in form of text file.





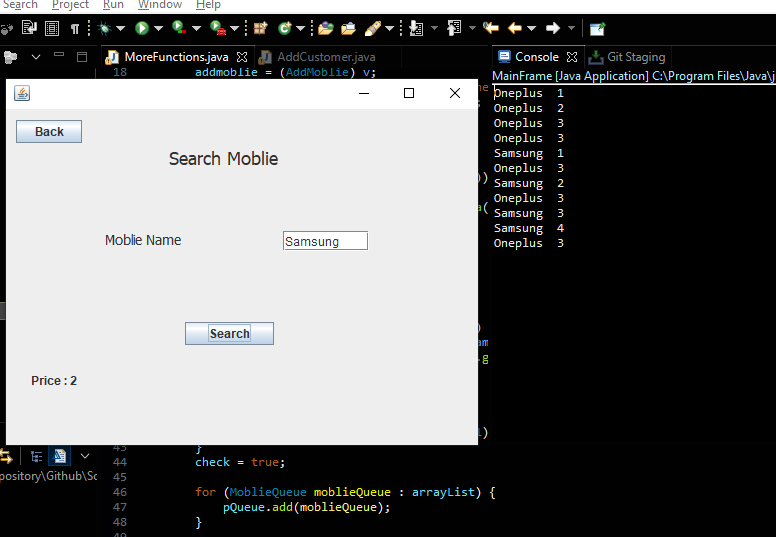
# 6.binary search tree implementation

# 

# 7.hash table implemetation with genric

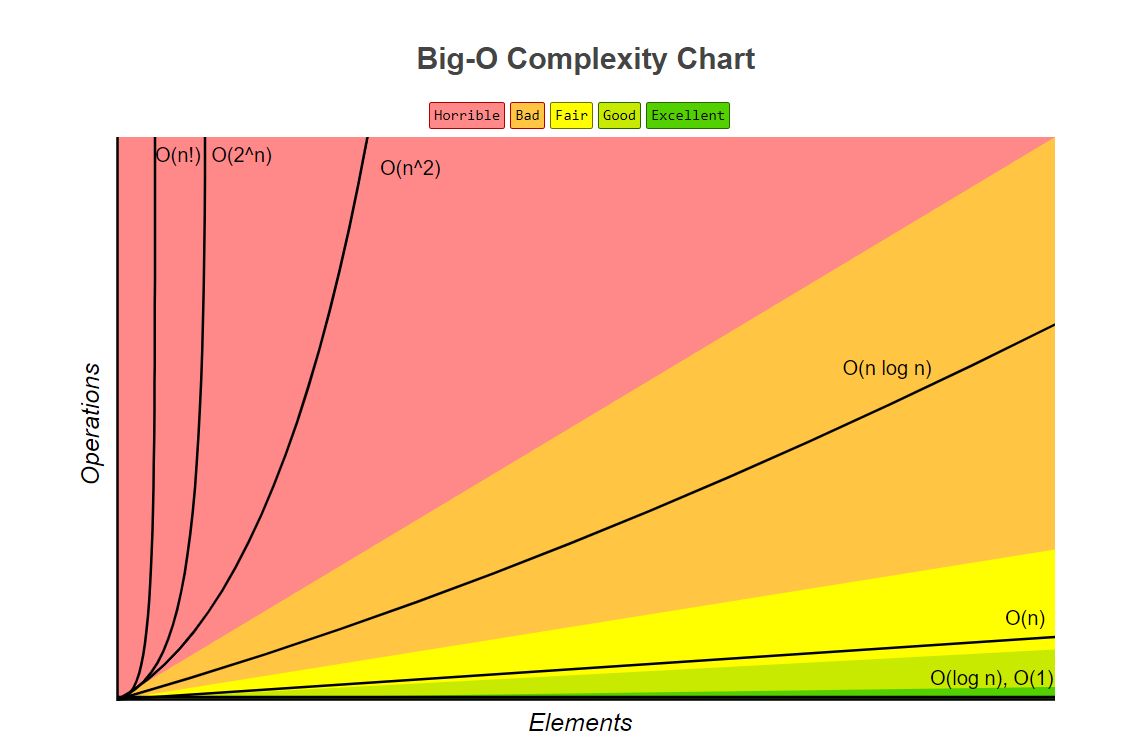
# 

# 7.Count comparision



**Performance and References to Data Structures**

# Binary search tree



Source: <http://bigocheatsheet.com/>

Cost of search and insert BST.

Time Complexity of BST in **o(log n)** which means it’s a linear function that corresponds to that the time complexity won’t change even if the number of nodes increase.

**Problem.**

Worst-case search and insert are proportional to N. If nodes in order, tree degenerates to linked list.

Cost of search and insert Hash Table.

Theorem.

The Complexity is **o(1)** is according to the code written.

Problem.

But *O* (1) is achieved only if load-factor is less than 1.  
Worst Case is always (*n*), you can go about looking-up all the elements in the sist

# hashtabel

It is very important to note that hash tables have amortized constant complexity i.e. on an average case the complexity will be O(1). In **worst** case, if too many elements were hashed into the same key, it can have a time complexity of O(n).

# Priority Queue

Time complexity for the methods offer & poll is O(log(n)) and for the peek() it is Constant time O(1) of java priority queue. NOTES: In Java programming, Java Priority Queue is implemented using Heap Data Structures and Heap has O(log(n)) time complexity to insert and delete element.

# Collection framework

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

# 

# Conclusion

A Simple Mobile management system is created by using Binary Search Tree, Priority Queue and Hash table as data structures, values are iterated using Binary Search Tree and different design patterns are applied. Learning outcomes derived from the project helped to discover new features of data structure that will serve for future projects as an avid input.

# References

<https://en.wikibooks.org/wiki/Data_Structures/Hash_Tables>

[BST Complexity](https://www.youtube.com/watch?v=tEoyeoHmqlk)

<https://stackoverflow.com/>

<https://www.javatpoint.com/collections-in-java>