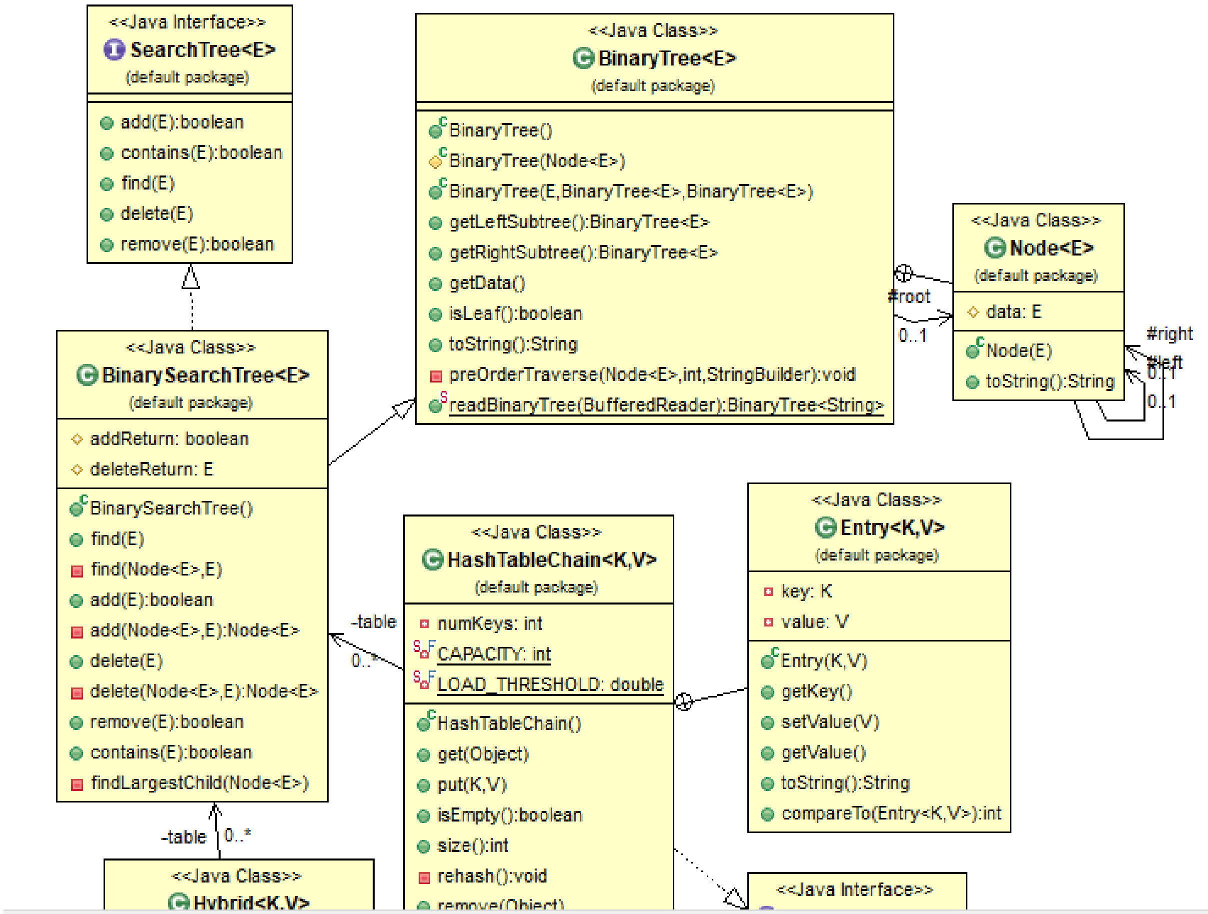
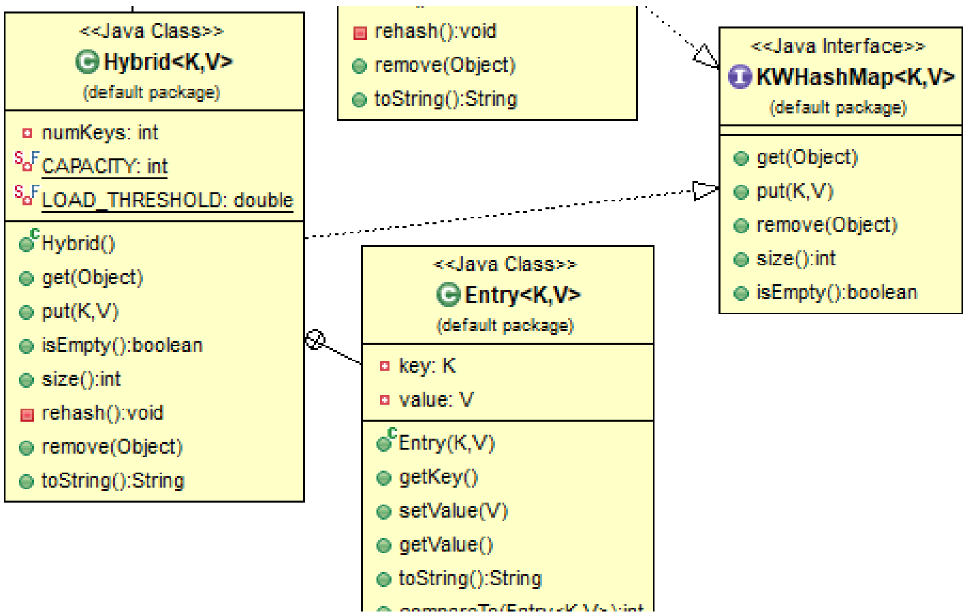
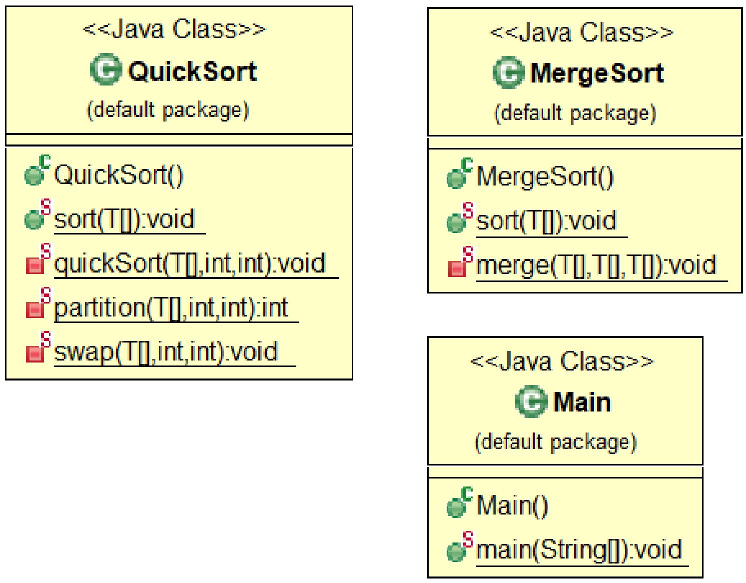
GIT Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 6 Report

Tuba Toprak
161044116

1. CLASS DIAGRAMS







Question 1:

1a) Implemented the chaining technique for hashing in book.

1b)

Coalesced Hashing Advantages and Disadvantages: Coalesced hashing is a collision avoidance technique when there is a fixed sized data. It is a combination of both Separate chaining and Open addressing. It uses the concept of Open Addressing(linear probing) to find first empty place for colliding element from the bottom of the hash table and the concept of Separate Chaining to link the colliding elements to each other through pointers. The hash function used is h=(key) % (total number of keys).

Advantages: It is better than chaining hashing because it inserts the colliding element in the memory of hash table only instead of creating a new linked list as in chaining hashing. The coalesced hashing method is one of the faster searching methods known today. Coalesced Hashing outperforms several well-known methods , chaining, linear probing, and double hashing.

Disadvantages: As in open addressing, deletion from a coalesced hash table is awkward and potentially expensive, and resizing the table is terribly expensive and should be done rarely, if ever. The method is especially suited for applications with a constrained amount of memory or with the requirement that the records cannot be relocated after they are inserted.

Double Hashing Advantages and Disadvantages: Double hashing is a collision resolving technique in open Addressed Hash tables. Double hashing uses the idea of applying a second hash function to key when a collision occurs.

Advantages: is that it is one of the best form of probing, producing a uniform distribution of records throughout a hash table.This technique does not yield any clusters.It is one of effective method for resolving collisions. Although the computational cost may be high, double hashing can find the next free slot faster than the linear probing approach.

Disadvantages: The double hash is more difficult to implement than the others. Double hash can cause a crash.

Question 2 :

Firstly, theoretical and empirical experiments were done after merge sort and quicksort implementation. Arrays were created to hold random numbers for merge sort and Quick sort. In the empirical experiment, the whole code was looped to run 1000 experiments. Experiments of different sizes were performed with the two sorting algorithms and output was obtained.

	Theoretical			Empirical		
	100	1000	10000	100	1000	10000
Quick Sort	1438300 ns	1590900 ns	6977900 ns	42820 ns	395445 ns	4480538 ns
Merge Sort	987500 ns	1281000 ns	7813100 ns	12494 ns	142674 ns	1873295 ns

	Quick Sort		Merge Sort
Worst case complexity :	O(n2)	-	O(n log n)
Works well on :	It works well on smaller array	-	It operates fine on any size of array.

According to the results obtained, there seems to be a difference between the empirical experiment and the theoretical results. In both experiments, merge sort was found to work faster.

2. RUNNING AND RESULTS

Question 1 :

```
------------Chaining technique for hashing (100)------------
Time of Put Method: 149400 nano
Time of Remove Method: 1634600 nano
------------Chaining technique for hashing (1000)------------
Time of Put Method: 2106200 nano
Time of Remove Method: 9937800 nano
------------Chaining technique for hashing (10000)------------
Time of Put Method: 9561800 nano
Time of Remove Method: 235289400 nano
```

```
"C:\Program Files\Java\

----Hash Table----
Key: 3   Value: 7
  null
  null


----Hash Table----
Key: 12   Value: 4
  null
  null
Key: 3   Value: 7
  null
  null
```

```
----Hash Table----
Key: 12   Value: 4
  null
  null
Key: 3   Value: 7
  null
  Key: 13   Value: 4
    null
    null


----Hash Table----
Key: 12   Value: 4
  null
  null
Key: 3   Value: 7
  null
  Key: 13   Value: 4
    null
    null
Key: 25   Value: 8
  null
  null
```

```
----Hash Table----
Key: 12   Value: 4
  null
  null
Key: 3   Value: 7
  null
  Key: 13   Value: 4
    null
    Key: 23   Value: 8
      null
      null
Key: 25   Value: 8
  null
  null
```

```
----Hash Table----
Key: 51   Value: 6
  null
  null
Key: 12   Value: 4
  null
  null
Key: 3   Value: 7
  null
  Key: 13   Value: 4
    null
    Key: 23   Value: 8
      null
      null
Key: 25   Value: 8
  null
  null
```

```
----Hash Table----
Key: 51   Value: 6
  null
  null
Key: 12   Value: 4
  null
  null
Key: 3   Value: 7
  null
  Key: 13   Value: 4
    null
    Key: 23   Value: 8
      null
      null
null
```

Question 2:

```
----Theoretical Result ----
Quick Sort (100): 1438300 nano
Merge Sort (100): 987500 nano
Quick Sort (1000): 1590900 nano
Merge Sort (1000): 1281000 nano
Quick Sort (10000): 6977900 nano
Merge Sort (10000): 7813100 nano

----Empirical Result ----
 Merge Sort (100): 42820 nano
 Quick Sort (100): 12494 nano
 Merge Sort (1000): 395445 nano
 Quick Sort (1000): 142674 nano
 Merge Sort (10000): 4480538 nano
 Quick Sort (10000): 1873295 nano


Process finished with exit code 0
```