LP4: Multi-dimensional search
G95

Team Members:

Jithendhiralal Ramlal - jgr150030
Ramakrishnan Sathyavageeswaran - rxs142530
Thiagarajan ramakrishnan - txr150430

In these project we have implemented the Multi-dimensional search using TreeMap and HashMap and LinkedList.

For Multi-dimensional search we have used different data structure with different key value pair.

Used TreeMap for these ID based search indexes for item.

```
TreeMap<Long, Item> itemTree = new TreeMap<>();
TreeMap<Double, LinkedList<Item>> priceTree = new TreeMap<>();
HashMap<Long, LinkedList<Item>> descriptionTree = new
HashMap<>();
```

Item ID based Index:

```
TreeMap<Long, Item> itemTree
Key: Long (Item's ID)
Value: Item
```

We used this TreeMap for the following operation:

1. Insert
2. Find
3. Delete
4. PriceHike

Item Description based Index:

```
HashMap<Long, LinkedList <Item>> descriptionTree
Key: Long (Item 's description )
Value: LinkedList <Item>
```

We used this TreeMap for the following operation

1. FindMinPrice
2. FindMaxPrice
3. FindPriceRange

Item's price based index:

```
TreeMap<Double, LinkedList <Item>> priceTree
Key: Double (Item's price)
Value: LinkedList<Item>
```

We used this TreeMap for the following operation

1. Range

We have created a Navigable Map out of price tree TreeMap which we used to find the range count.

SameSame Operation:

We have used modified version Hashmap for finding the Items.

```
HashMap<Item, Integer> samesameHash = new HashMap<>();
Key: Item (Item)
Value: Integer (count of same items)
```

In order to have Item itself as a key we need to modify both Hascode() and equals() method of the Hashmap.

Hashcode() : By discussing with other team, we got the idea of using XOR for improving the hashcode speed. We used XOR of descriptions to get a unique value in the hashcode.
Equals() : In equals method we have compared the description array of first object with the second object and returning true if they are equal and false if they are not equal.