Quiz 2: Tuesday October 18th (Exception + Case studies)

LinkedList: Collections of values (they are allowed to have duplicates) that implements the List interface
Set: HashSet (elements are not ordered) and TreeSet (elements of the tree set are ordered in an ascending order). They cannot store duplicate values.
Map: group of (key, value) pairs. Keys of a map have to be unique. HashMap and TreeMap (Keys are ordered).

Example#1: ConcordancesDemo
the quick brown fox jumps   over the lazy dog

the: 2
quick: 1
brown: 1
....
HashMap<K, V> K and V are generic data types that I can replace with class names.
Processing steps:

```
int count = map.get("the");
map.put("the", count + 1)
```

HashMap
the: 2

Comparator interface is a built-in
Iterator: java.util.Iterator ---> hasNext, next. Examples of classes implementing it: Scanner
Comparable: java.lang.Comparable ---> compareTo. Examples of classes implementing it: String

Comparator<T> interface offers method called compare

```
int compare(T o1, T o2) {
        return o1.getSalary() - o2.getSalary();
}
```

- Queue: Priority Queue to efficiently sort a list of values
Original list (Size = n): 10   3   5   2   1

Priority queue: collection of prioritized elements
insert
removeMin: remove the element having the highest priority

Priority Queue sorting algorithm:
1. Insertion step:
Insert elements of the original list into the priority queue by means of n insert operations
2. Selection step:
Remove the elements from the priority queue through n removeMin operations and put them back into the list.

1   2   3   5   10

- If we implement the priority queue using an unordered list:

10   3   5   2   1

1. Insertion step:

insert: is just putting the element at the end of a list O(1)
n x O(1) = O(n)

10   3   5   2   1

2. Selection step:
removeMin
n + n-1 + ... + 1 = n (n + 1) /2 = O(n^2)

Time complexity: O(n^2)
Selection sort is an algorithm where the bottleneck is the selection step
Space complexity: O(n)

- If we implement the priority queue using a sorted list:

10 3 5 2 1

1. Insertion step:
1 2 3 5 10
insert:
1 + 2 + 3 + ... + n = O(n^2)

2. Selection step: O(n)
removeMin

Time complexity: O(n^2)
Bottleneck step: insertion ===> insertion sort
Space complexity: O(n)

- If we implement the priority queue using a heap
Complete binary tree that satisfies a well-defined relational property

insert: proportional to the height of the heap O(nlog2(n))
removeMin: proportional to the height of the heap O(nlog2(n))

Height of the heap: O(log2(n))

ADT (Abstract Data Types)

heap-based sorting algorithm:
Time complexity: O(n log2(n))
Space complexity: O(n)

PriorityQueue class offered by Java: it is a heap-based priority queue.
Example#2: PriorityQueueSorting

Queue (FCFS) vs PriorityQueue