Java APIs

Array

- Arrays.binarySearch(arr, target) -> O(nlogn)
- Arrays.sort()
- Arrays.toString()
- Arrays.asList(int[] arr) -> O(1) only accept object
- Arrays.copyOf(originalArray, newLength) -> O(1)
- Arrays.copyOfRange(originalArray, start, end) -> O(n)
- comparable -> compareTo(secondNum)

ArrayList

size-adjustable [ArrayList]: `List<X> list = new ArrayList<>();`

- add(E, e)
- add(int index, E e) // random access
- addAll(Collection c) // append whole to list
    - E get(int index)
    -
    - contains(val) // return true if find
    - indexOf(val) // return -1 if not find
    -
    - remove(int index)
    - remove(E e)
    - clear() // faster than removeAll
    -
    - set (int index, E e)
    - int size()
    - toArray() -> Object[]
- Collections.sort(list, new myComparator())
- COllections.reverse(arrayList)
- ArrayList.subList(start, end)
- list.forEach(k -> sb.append(k));
- Comparator -> compare(item1, item2)

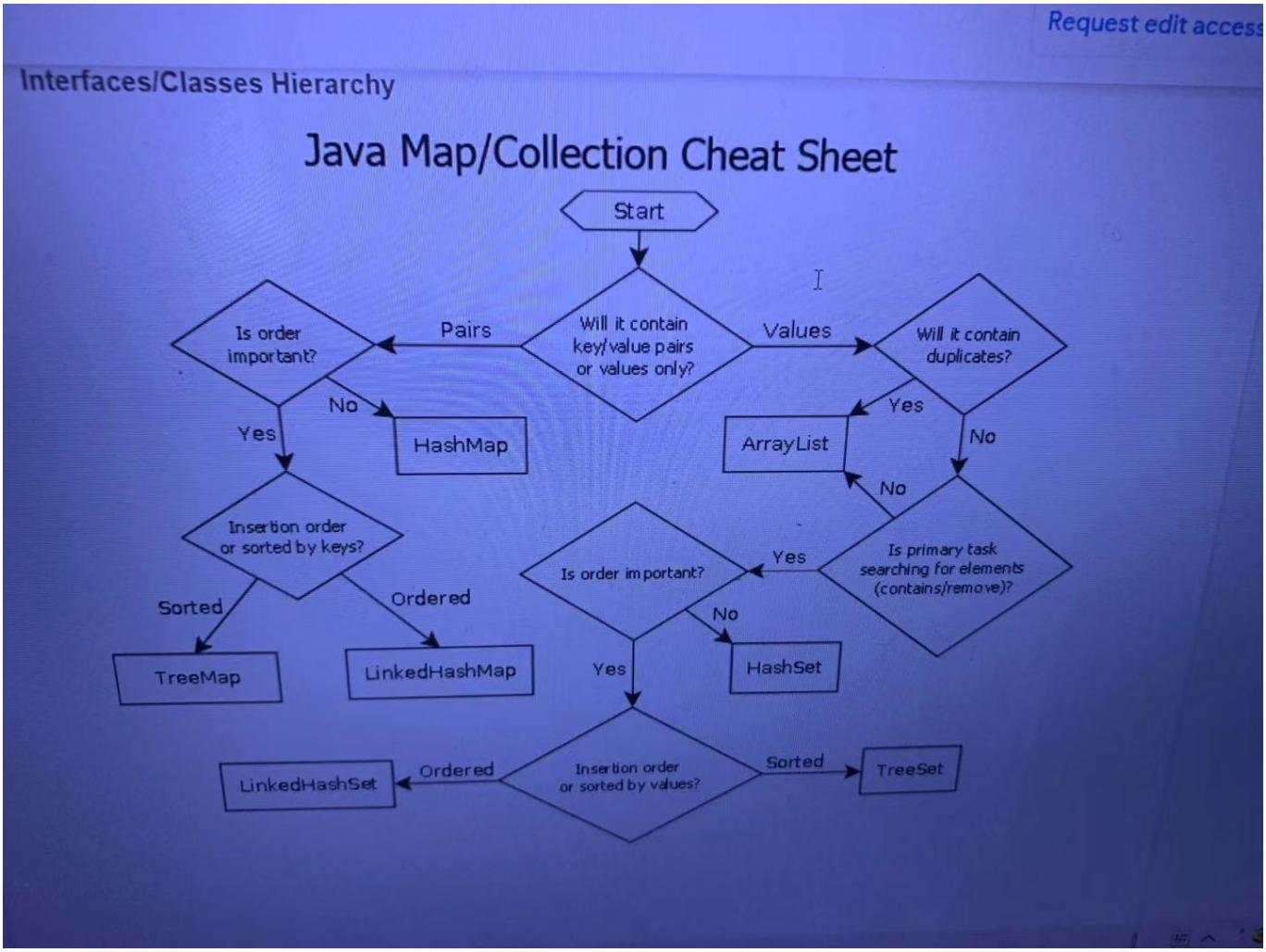
Deque

two end [queue]: `Deque<X> dq = new ArrayDeque<>();`

- offerFirst()/ offerLast();
    - pollFirst() / pullLast();
    - peekFirst()/ peekLast();
    - isEmpty();
    - size();

HashMap

[HashMap]: `Map<String, Integer> map = new HashMap<>();`
`ImmutableMap<String, Integer> map = ImmutableMap.of(...);`



> - ordered map: LinkedHashMap
> - sorted map: SortedMap<String, Integer> map = new TreeMap<>()

- map.get(key);
- map.getOrDefalut(key, defaultValue); // value or defaultValue
-
- map.put(key, value);
- map.putIfAbsent(key, value); // return null or current value
-
- map.containsKey(key);
- map.remove(key)
-
- map.keySet()
- map.values()
- map.entrySet()
-

for (Map.Entry<String, Integer> entry : map.entrySet()) {
String key = entry.getKey();

```
Integer value = entry.getValue()
}

public void iterateUsingLambda(Map<String, Integer> map) {
map.forEach((k, v) -> System.out.println((k + ":" + v)));
}
```

HashSet

[HashSet] `Set<Integer> set = new HashSet<>()`

> - ordered (by insertion): LinkedHashSet
> - sorted (object need comparator): TreeSet

- set.add(key)
- set.contains(key)
- set.remove(key) -> true/false
- set.toArray()

Integer

- Integer.parseInt(String)
- Integer.toString(number)

PriorityQueue / Heap

- min Heap [PriorityQueue]: `Queue<X> pq = new PriorityQueue<>();`
- max Heap: PriorityQueue maxHeap = new PriorityQueue<>(Collections.reverseOrder());
- pq.contains(key)

Queue

use a [queue]: `Queue<X> queue = new ArrayDeque<>();` `Queue<X> queue = new LinkedList<>();`

- throw exception add() element() remove()
- return special value offer() poll() peek()

Stack

use a stack: `Deque<X> stack = new ArrayDeque<>();`
~~Stack stack = new Stack<>();~~

- push(E)
- pop()
- peek()

LinkedList

- getFirst() / peekFirst()
- getLast() / peekLast()

- addFirst(e) / offerFirst(e)
- removeFirst() / pollFirst()
- List<> list = new LinkedList<>();

String:

- s.length()
- s.isEmpty()
- s.equals(s2)
- s.contains("word") -> true/false
- s.indexOf(char)
- s.lastIndexOf(char)
- s.startsWith(str)
- s.endsWith(str)
- s.substring(start)
- s.substring(start, end)
- s.toCharArray()
- String s = new String(char[])
- s.split("/") -> String[]
- s.split(",");
- s.trim()
- s.replaceAll("[^A-Za-z\d]+", "").toLowerCase();
- String.valueOf(num); //int i=10; Now it will return "10"
- Integer.toString(number)
- String.join(",", char[]); "a,b,c"

StringBuilder

[StringBuilder] `StringBuilder sb = new StringBuilder()`

- contructor: (int capacity/ CharSequence seq / String str)
- sb.append('a' / char[] / int / boolean)
- sb.deleteCharAt(sb.length() - 1);
- sb.size()
- sb.insert(int index, char ch) -> O(n)
- sb.setLength(int newLength)
- sb.setCharAt(int index, char ch)
- CharSequence sb.subSequence(int start, int end)
- StringBuilder(sb).reverse().toString();

Character

- Character.toLowerCase(c)
- Character.isLetterOrDigit(c)

char

- string.charAt(index) - 'a' -> ASCII 128 char

- string.charAt(index) - '0' // char to int
- Character.toString(char c) // char -> String
- String.valueOf(char c) // char -> String
- (char) int + '0' // int to char

Collection

- Collections.binarySearch(collection, target)
- Collections.sort(list, new myComparator())
- Collections.reverse(arrayList)

Class Creation

Comparator

```
PriorityQueue<Integer> queue = new PriorityQueue<>
((a, b) -> b - a); // biggest pop first, 30 20 10
```

```
private class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer i1, Integer i2) {
      if (i1.equals(i2)) {
        return 0;
      }
      return i1 > i2 ? -1 : 1;
    }
  }
```

```
Pair<Integer, String> pair = new Pair<>(1, "One");
  Integer key = pair.getKey();
  String value = pair.getValue();

private Object[] getPair() {
    // ...
    return new Object[] {key, value};
}
```