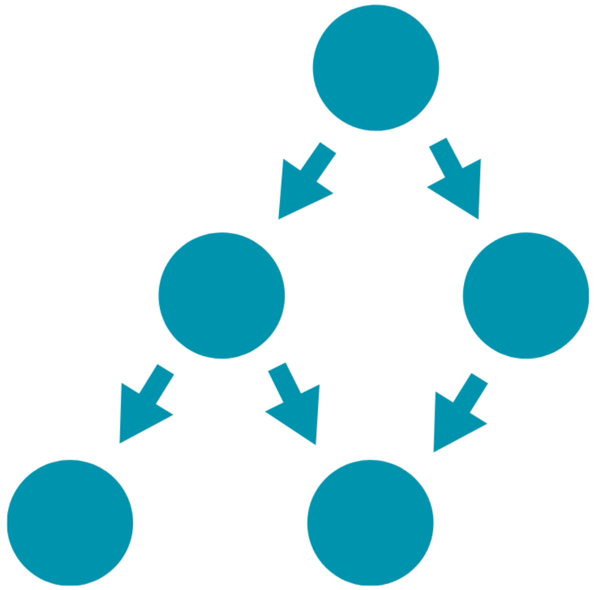


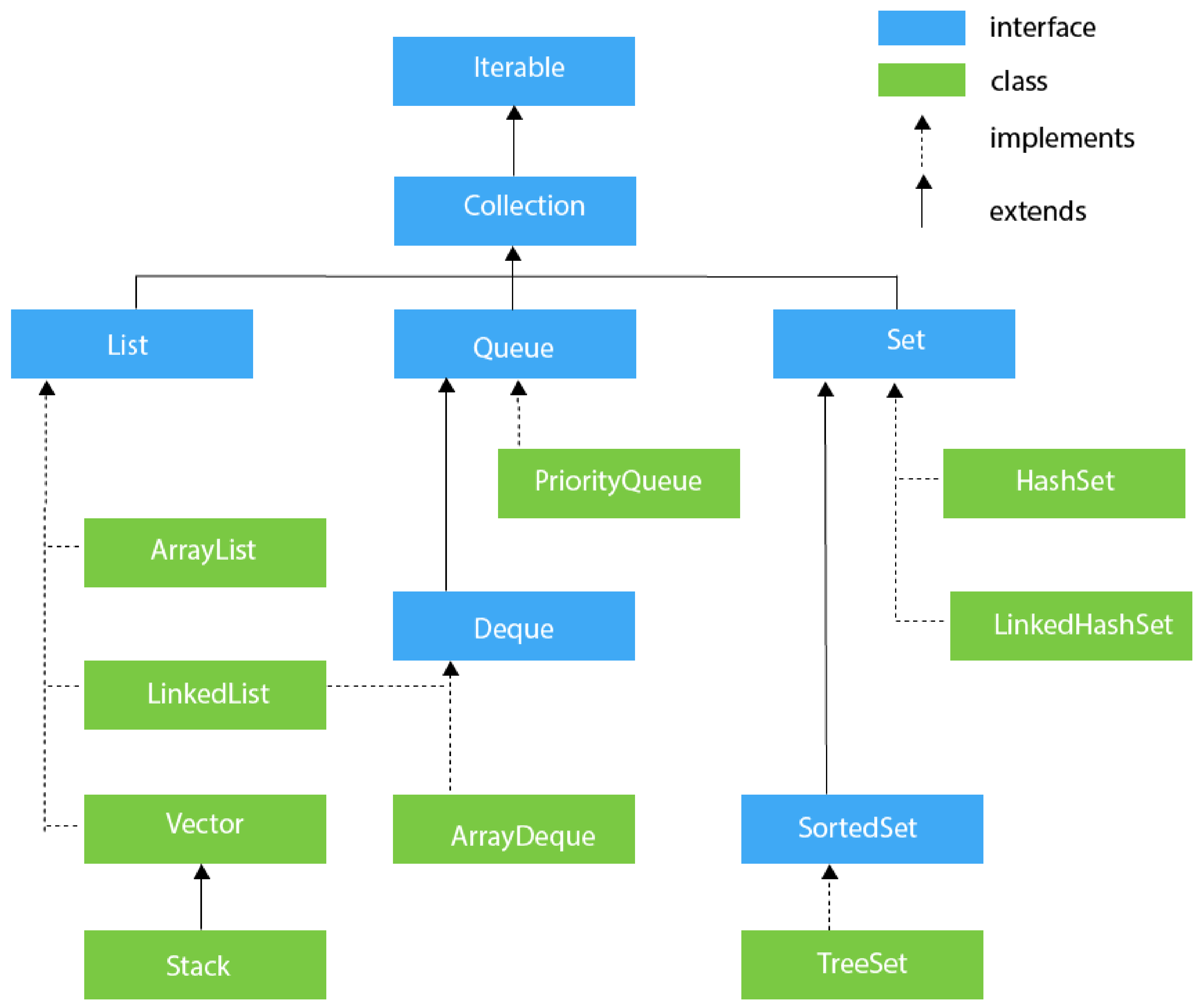
VERİ YAPILARILARI VE ALGORİTMALAR

Zafer CÖMERT

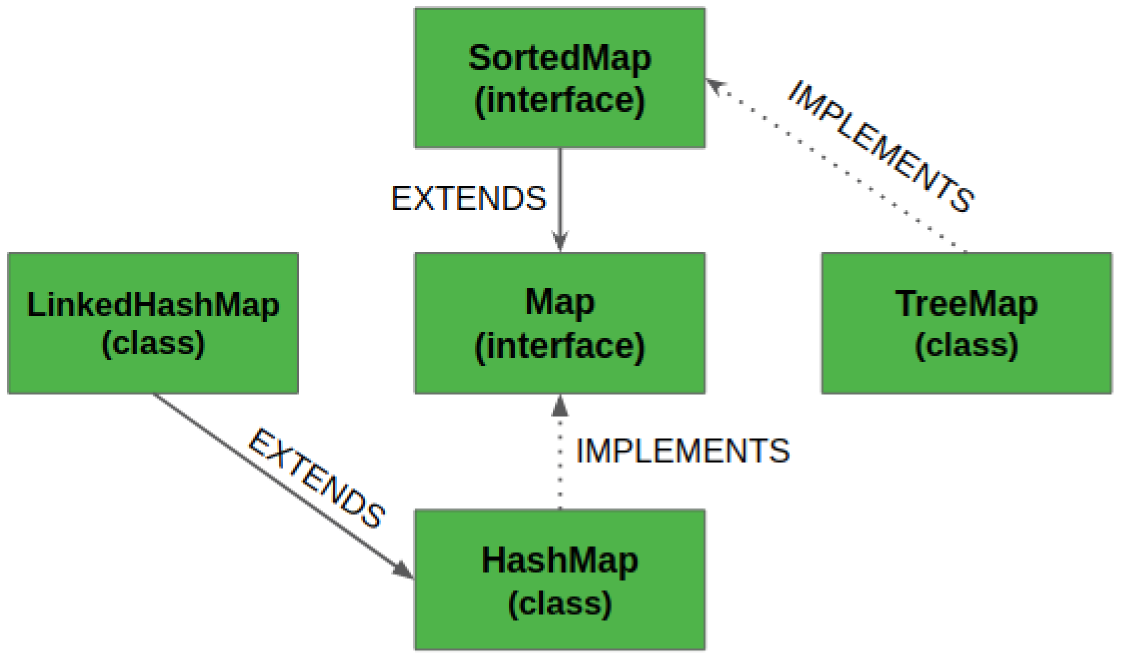
Giriş

Data Sturcutres and Algorithms in Java

https://senoritadeveloper.medium.com/data-structures-when-to-use-which-79e2006d4ee1



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



MAP Hierarchy in Java

https://www.geeksforgeeks.org/linkedhashmap-class-in-java/

Array

- Tuple (finite ordered list of elements)
- Sequential
- Linear Arrays — fixed size
- Dynamic Arrays — reserved space for additional elements. If full, it copies its content to a larger array

```
- Optimal for indexing
- Bad at searching, inserting, deleting (excluding the end)
```

Complexity

```
indexing = O(1)
search = O(n)
optimized search = O(logn)
insertion = (for dynamic) O(n)
```

ArrayList

- implements *List*
- Internally uses an Object[] of default size 10 (if not declared).
- When you add an item, it checks if there is any space left for the new element.
  If space is not a problem, the new item is added at the next empty space.
  If not; a larger array of 50% (using right shift operator to calculate) more the initial size is created and the current array is copied to the new one (using Arrays.copyOf).
- When you remove an element, elements are shifted (using Arrays.copyOf).

```
- add
- addAll
- clear
- clone
- remove
- subList
- toArray
```

```
append / get : O(1)
add / remove / indexOf / contains : O(n)
```

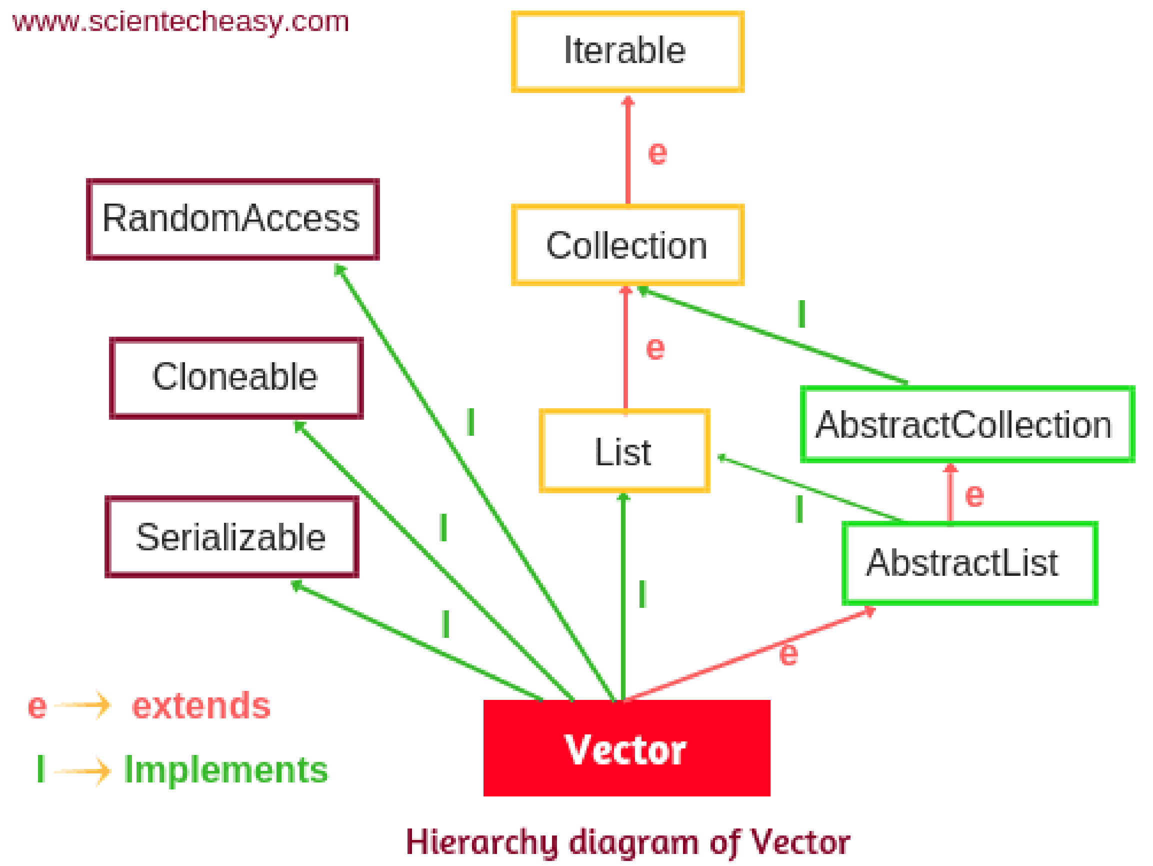
LinkedList

- Chain of nodes
- A node holds data of its own and a reference to its next node.
- *Doubly Linked List* — reference to previous and next nodes
  Circularly Linked List — head&tail linked
  Stack — LIFO, most commonly with LinkedLists (head is the only place for insertion and removal) but also with Arrays
  Queue — FIFO, implemented with LinkedLists (a doubly linked list that only adds to tail and removes from head) or Arrays

```
indexing = O(n)
search = O(n)
optimized search = O(n)
append = O(1)
prepend = O(1)
insertion = O(n)
```

```
- Optimized for insertion/deletion
- Slow at indexing/searching
```

Stack



Hierarchy diagram of Vector

```
- empty
- peek
- pop
- push
- search
```

PriorityQueue

- implements *Queue*
- When new elements are inserted into the PriorityQueue, they are ordered (and retrieved later) based on their natural ordering or by a defined "*Comparator*" provided when we construct the PriorityQueue.
- The internal working of the PriorityQueue is based on the Binary Heap.
- not thread-safe

```
- offer
- poll
- peek
```

```
- enque / deque : O(log(n))
- retrieval : O(1)
- contains: O(n)
```

LinkedList(Java)

- implements *List* and *Deque* interfaces
- List implementation (doubly linked list)
- null elements are allowed.
- not good at iteration; best at removing the current element during the iteration
- There is a static "*Node*" class.
- LinkedList class holds "*first*" and "*last*" variables.
- When you add the very first item, both the "*first*" and "*last*" point to the new "*Node*". They get updated according to the operation type.

```
- add
- addFirst
- addLast
- remove
- removeFirst
- removeLast
```

```
- append : O(1)
- add / get / remove / contains : O(n)
```

Map

- Hash Table or Hash Map
- data as key-value pairs
- hashing (a key and its unique output — beware of hash collisions)
- associative arrays, database indexing

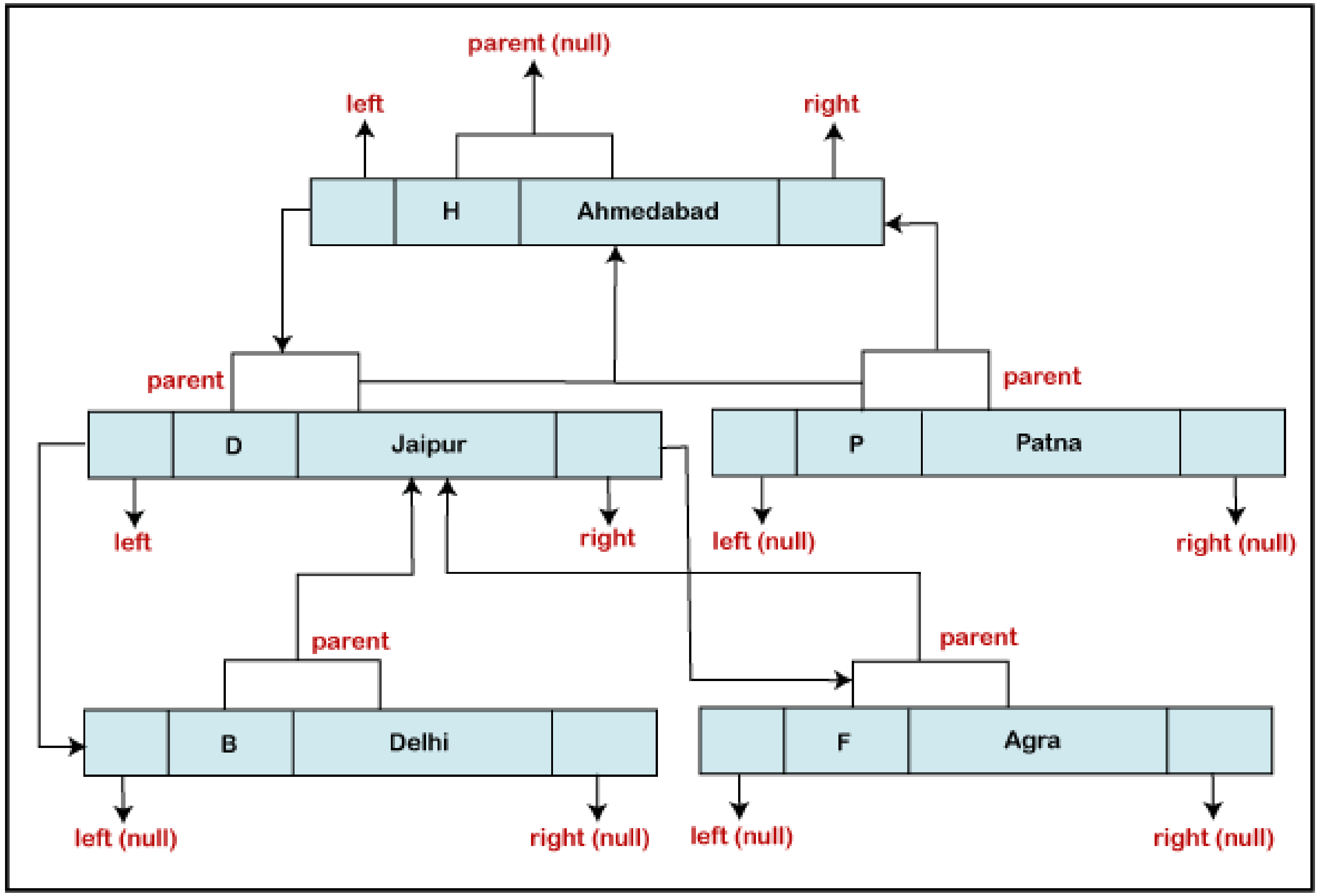
```
 - designed to optimize searching, insertion, deletion
```

```
indexing = O(1)
search = O(1)
insertion = O(1)
```

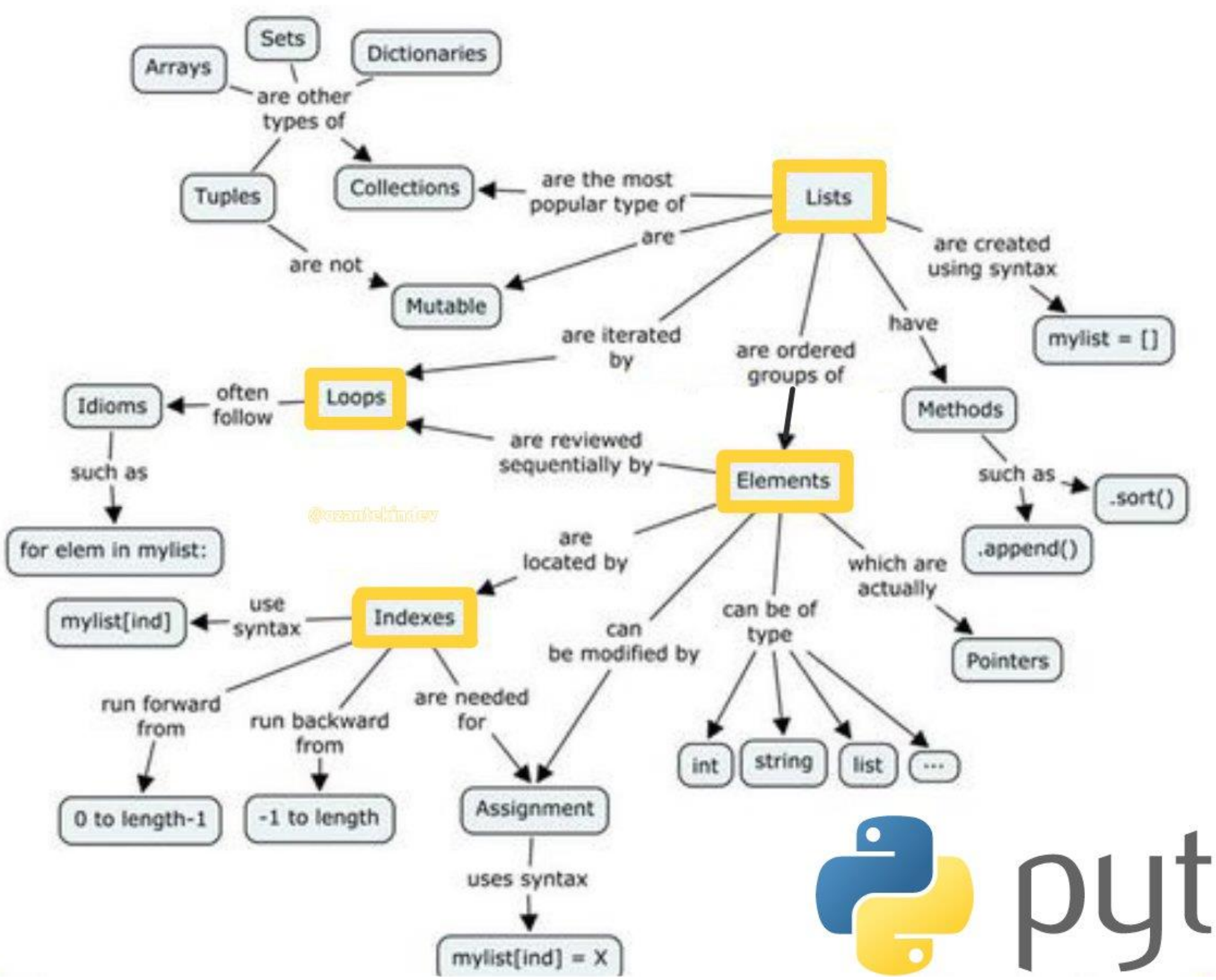
HashMap (Java)

- extends *AbstractMap*
- contains an array of "*Node*" which has "*hash*", "*key*", "*value*", "*next*" (points to the next node in the same bucket of array table)
- *Hashing* = process of converting an object into integer form by using the method "*hashCode*".
- A *bucket* is one element of HashMap array, used to store nodes.
  A single bucket can have more than one node (depending on *hashCode*);
  using link list to connect the nodes.
- Buckets are different in capacity.
- capacity = number of buckets * load factor

TreeMap (Java)



TÜM VERİ YAPILARI



python

Veri Yapıları ve Algoritmalar

ZAFER CÖMERT

Öğretim Üyesi