Talent Boost – Data Structures and Algorithms course

Hash Table, Hash Functions,

Hash Code

Vasilena Peycheva

Introduction

What is complexity?

The amount of resources required to run an algorithm.

How many types of complexity we use?

- Time complexity
- Space complexity

Which are the basic operations supported by most data structures?

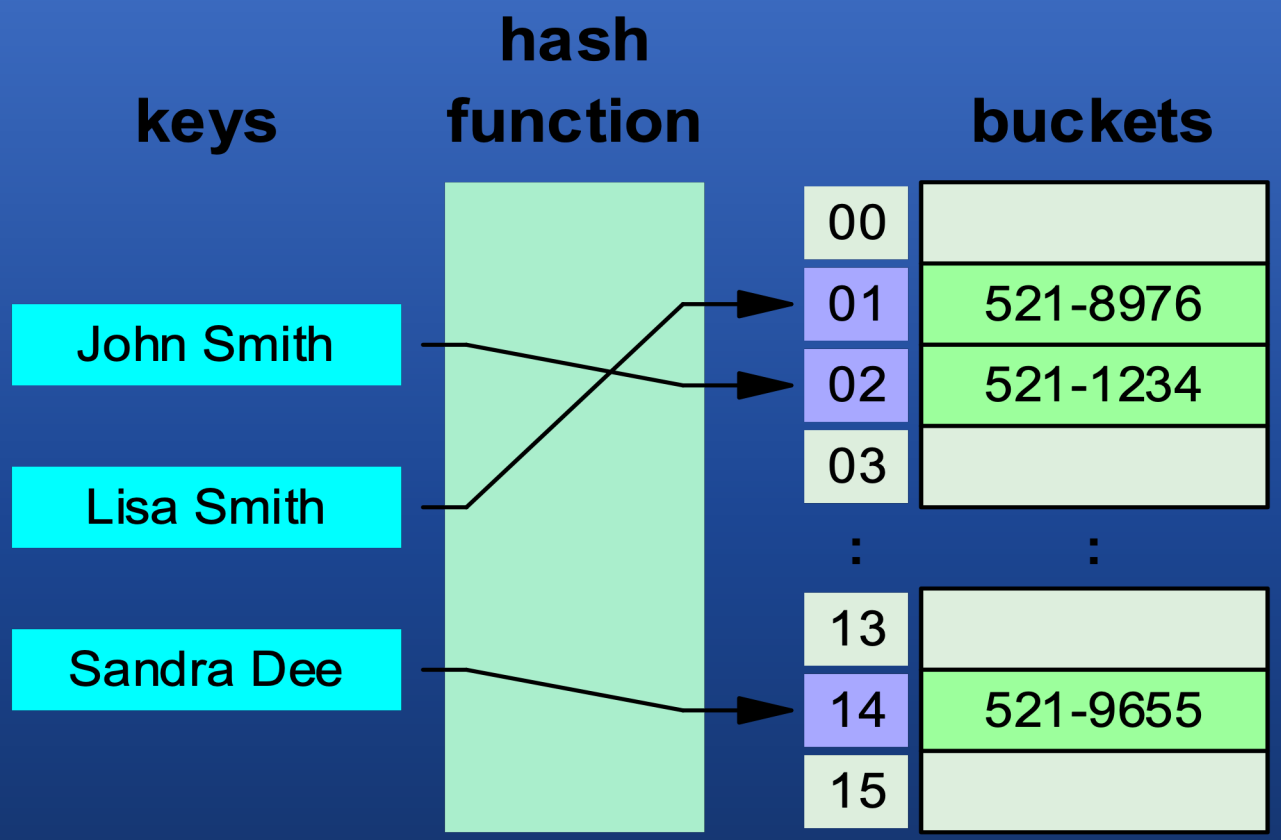
- *Create*
- *Search*
- *Delete*

What is the complexity of …

- Array
- Linked List
- Self-Balancing Tree

Can we do better?

Hash


keys
hash function
buckets
John Smith
Lisa Smith
Sandra Dee
00
01
02
03
13
14
15
521-8976
521-1234
521-9655

There are many things behind the term *Hash*

Hash Function

Hash Table

Hash code

What is Hash Table?

*Data structure with direct access to a desired element, **regardless** of the key type.*

How do we achieve this fast and direct access to a desired element in the table?

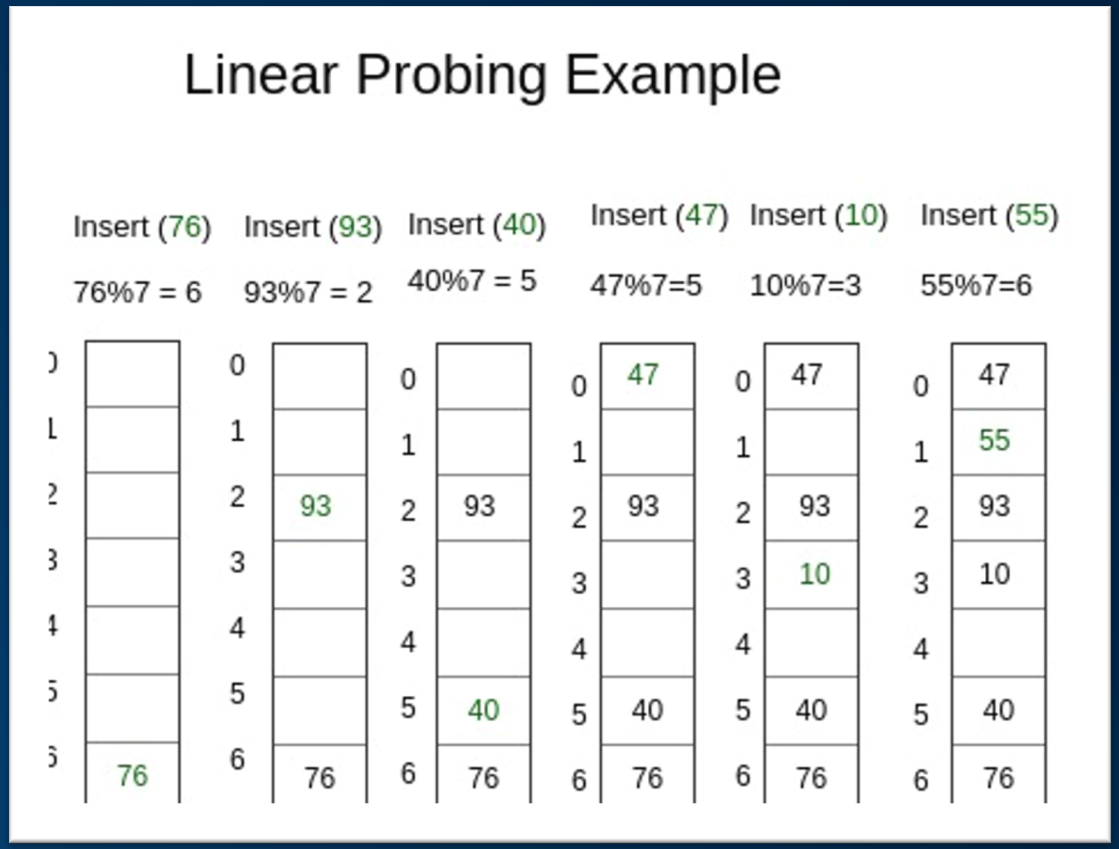
Hash Function - Function that maps for each element his index in the hash table uniquely using *Hash Code.*

What will happen if **two elements** have the **same** Hash Code?

Collision Handling

Open addressing

- Linear probing
- Quadratic probing
- Double Hashing

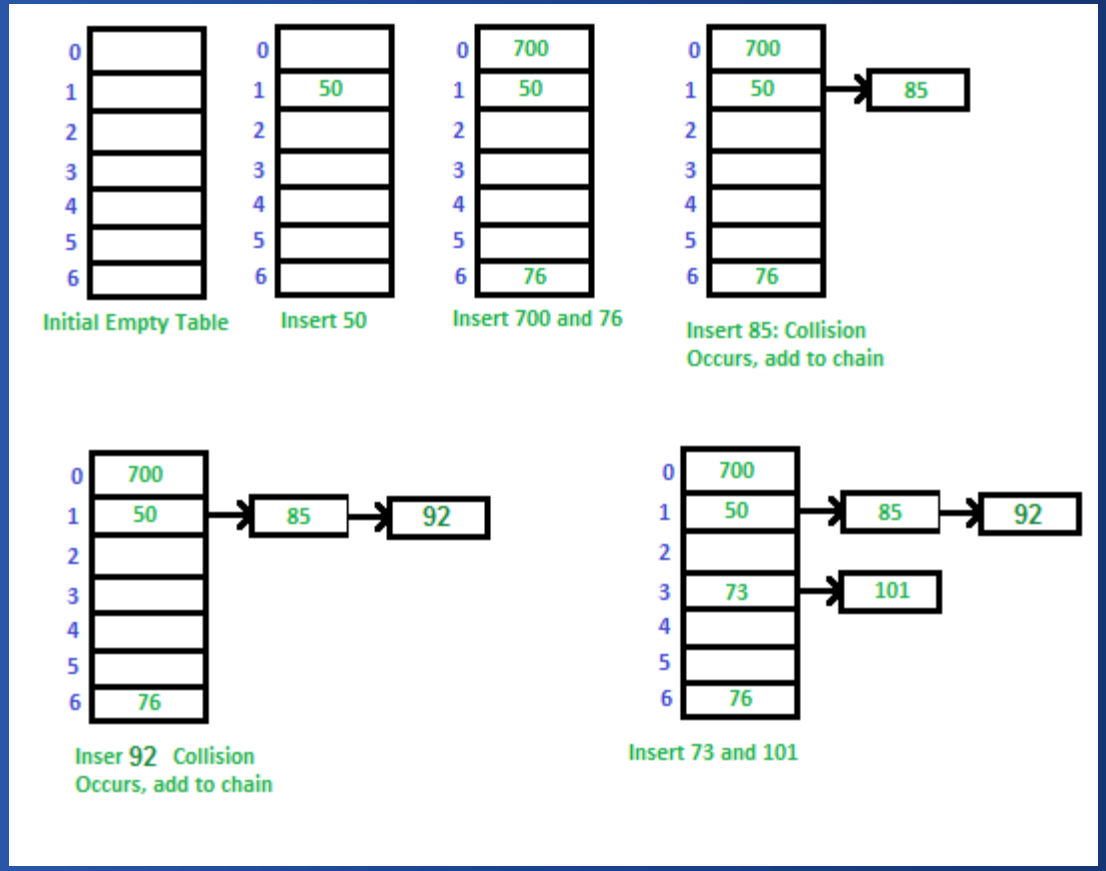


Collision Handling

Separate chaining

Implementation using Linked List

Implementation using Self-Balancing Tree



BREAK

The Design by Contract

The contract specifies what that component expects of clients and what clients can expect of it.

hashCode()

The general contract of hashCode is: Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in equals comparisons on the object is modified.

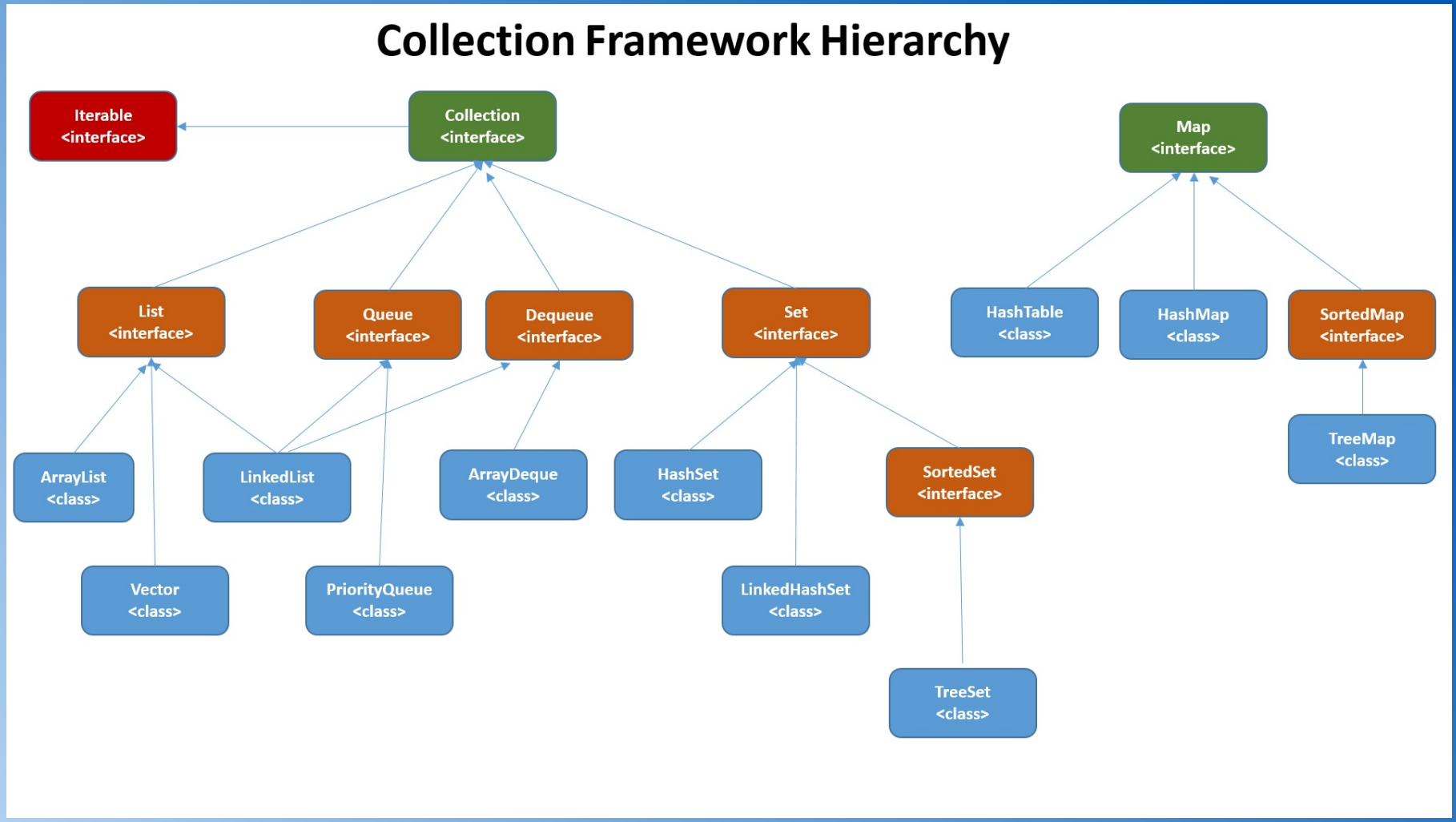
Equals()

If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.

Load Factor

The *load factor* is a measure of how full the hash table is allowed to get before its capacity is automatically increased.

Collections



Map

Hashtable

- is a legacy class
- doesn’t allow any null key or value

HashMap

- store values based on keys
- allows duplicate values
- contains unique keys

LinkedHashMap

- maintains the insertion order

BREAK

Set

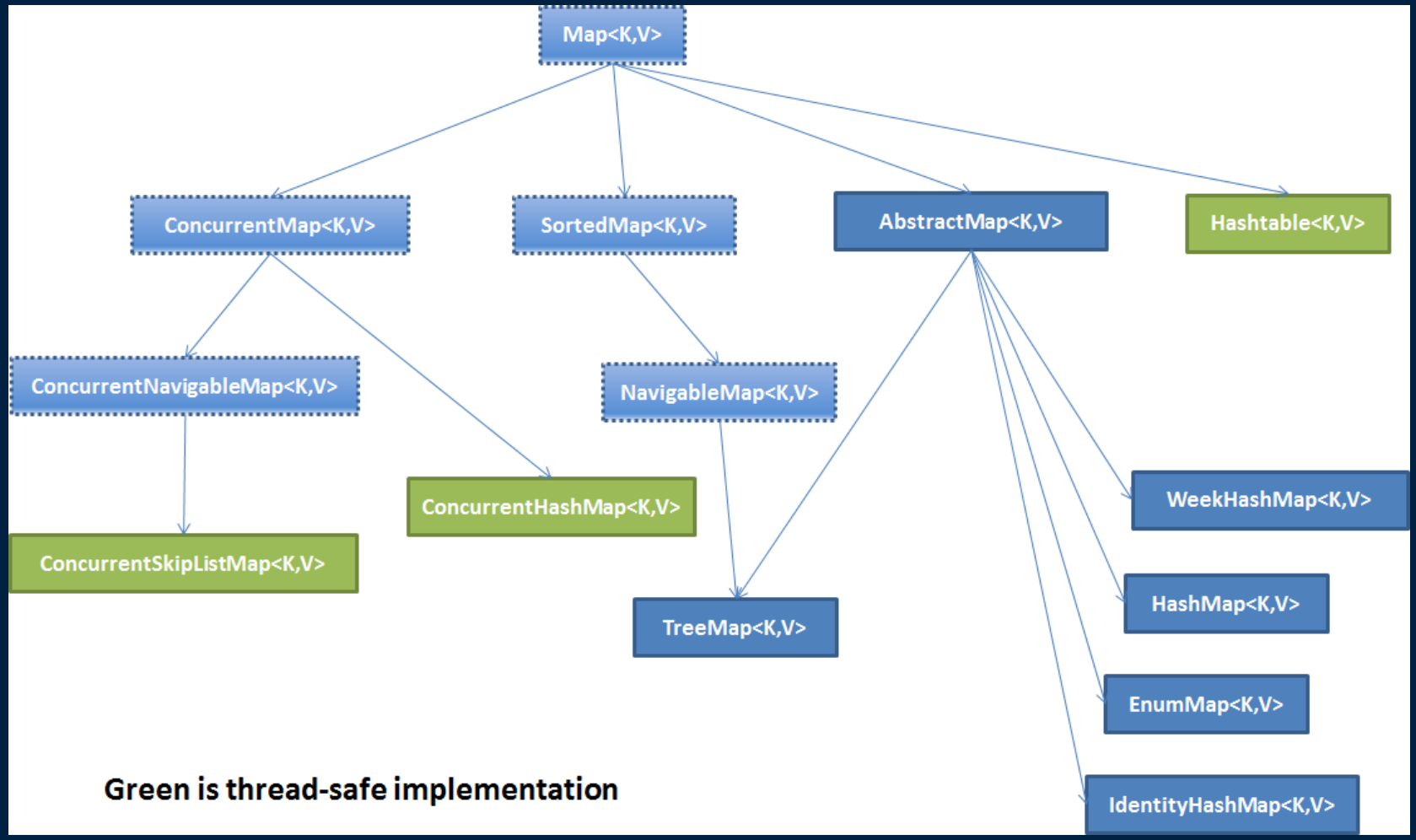
HashSet

- Internally uses data structure like a **hash table** for storage.
- Always contains unique items.

LinkedHashSet

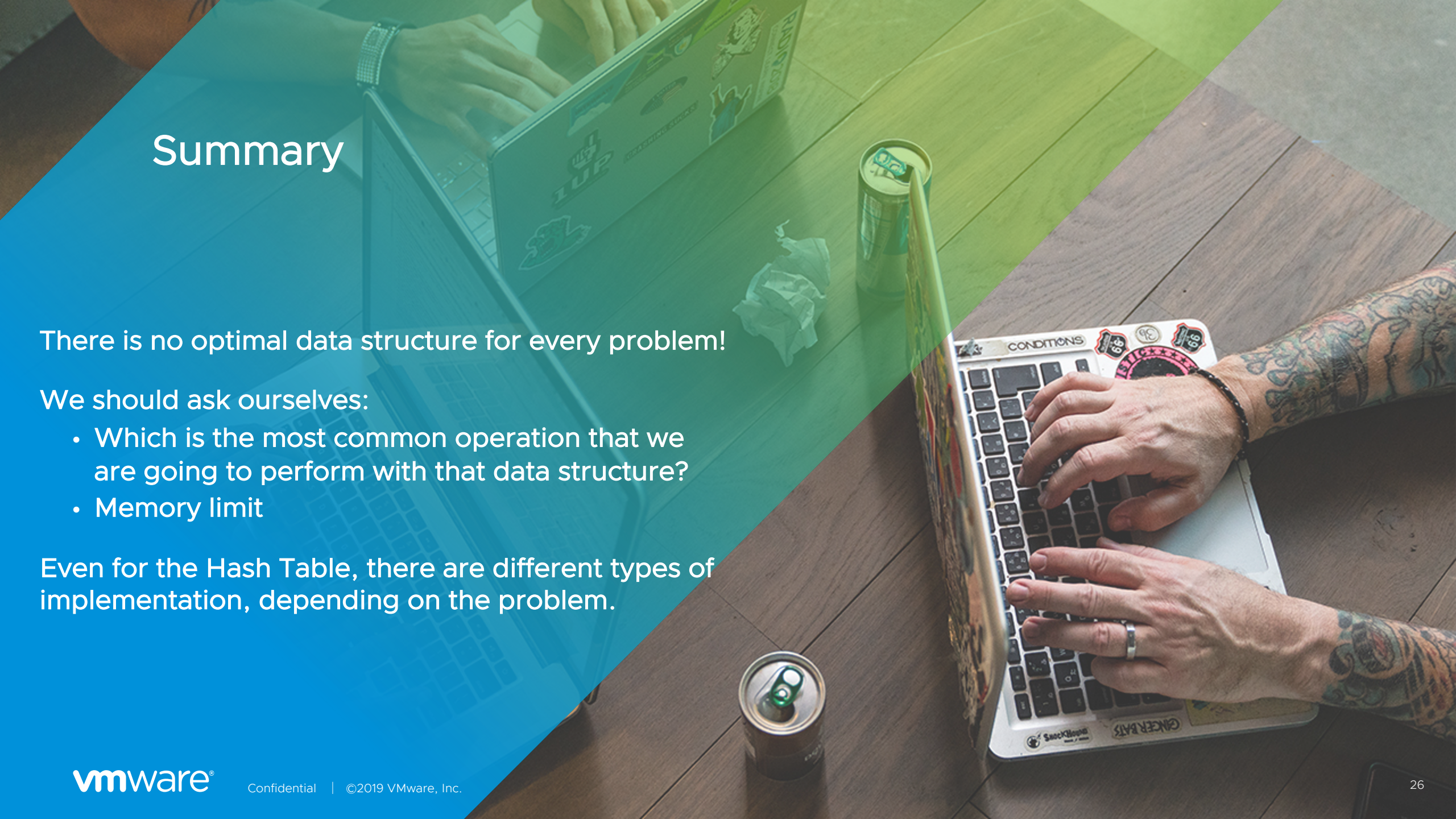
- Store items in LinkedList.
- Maintains the insertion order.
- Always contains unique items.

And many more ..



BREAK

Summary



There is no optimal data structure for every problem!

We should ask ourselves:

- Which is the most common operation that we are going to perform with that data structure?
- Memory limit

Even for the Hash Table, there are different types of implementation, depending on the problem.

Resources

- https://javapapers.com/core-java/java-hashtable/
- https://www.geeksforgeeks.org/hashing-data-structure/
- https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html
- https://docs.oracle.com/javase/8/docs/api/java/util/Hashtable.html
- https://facingissuesonit.com/2019/10/15/java-collection-framework-hierarchy/
- https://www.interviewcake.com/concept/java/lru-cache

Thank You