CYDEO

COLLECTION FRAMEWORK REVIEW

Today's

Agenda

- Why we need Collections Framework?
- Arrays and ArrayLists
- Basic Ops with ArrayList
- Sets in Collections
- Question with Sets
- Maps in Collections
- Question with Maps
- Problem Solving Pattern and an Example

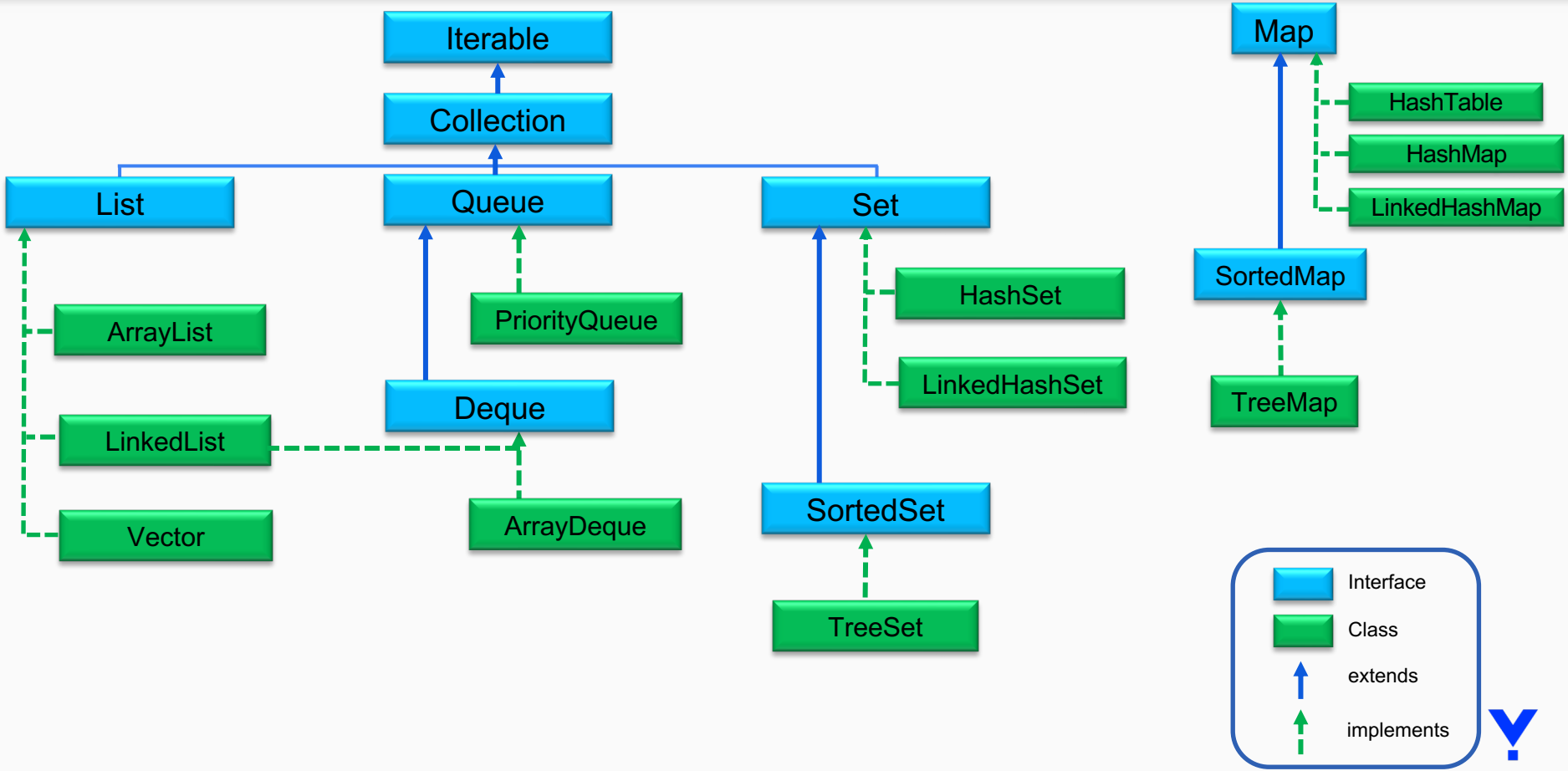
CYDEO

What is Collections Framework?

- The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.
- Collection is a set of implementations of most used Data Structures.
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
- Java Collection framework provides many **Interfaces** (Set, List, Queue, Deque) and

  Classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

What is Collections Framework?



How was the life before collections?

- Most popular programming language was C/C++.
- Collections came up with Java.
- The data structures implemented by collections had already existed well before Java.
- Developers/Enterprises had their own libraries for reusable code.
- What was the problem with this?

How was the life before Collections?

Developer 1

```
public abstract void insertLast(int number);
public abstract void removeFirst(int position);
public abstract void addInOrder(int number);
public abstract void put(int number);
public abstract void updateInfo(int number);
public abstract void sort();
public abstract void delete(int number);
public abstract void deleteFrom(int position);
```

Developer 2

```
public abstract void insert(int number);
public abstract void remove(int position);
public abstract void add(int number);
public abstract void put(int number);
public abstract void update(int number);
public abstract void sort();
public abstract void delete(int number);
public abstract void deleteFrom(int position);
```

Developer 3

```
public abstract void addAnElement(int number);
public abstract void deleteFrom(int position);
public abstract void insertInto(int position);
public abstract void insertInTheOrder(int number);
public abstract void put(int number);
public abstract void sortAscending();
public abstract void delete(int number);
```

Problems in here :

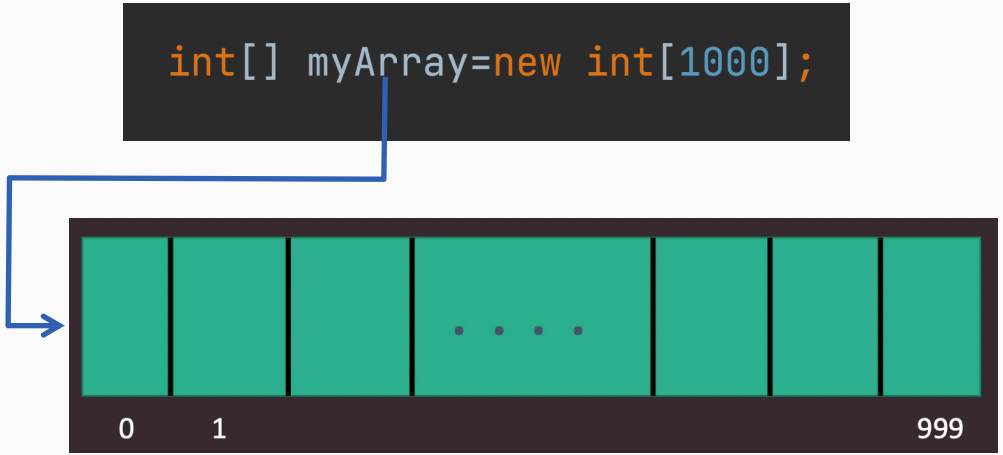
1. Not easy to understand
2. Increasing programming effort
3. No interoperability
4. No reuse

What is the advantage of Collections?

- **Reduces programming effort** by providing data structures and algorithms so you don't have to write them yourself.
- **Increases performance** by providing high-performance implementations of data structures and algorithms. Because the various implementations of each interface are interchangeable, programs can be tuned by switching implementations.
- **Provides interoperability** between unrelated APIs by establishing a common language to pass collections back and forth.
- **Reduces the effort required to learn**
- **Reduces the effort required to design** and implement by not requiring you to produce ad hoc collections.
- **Fosters software reuse** by providing a standard interface for collections and algorithms with which to manipulate them.

Array Data Structure

- An *array* is the basic mechanism for storing a collection of identically typed entities.
- Arrays use static memory allocation.



Limitations with Array

- Fixed in size
- There are no ready methods.

How can we overcome the limitations of Arrays?

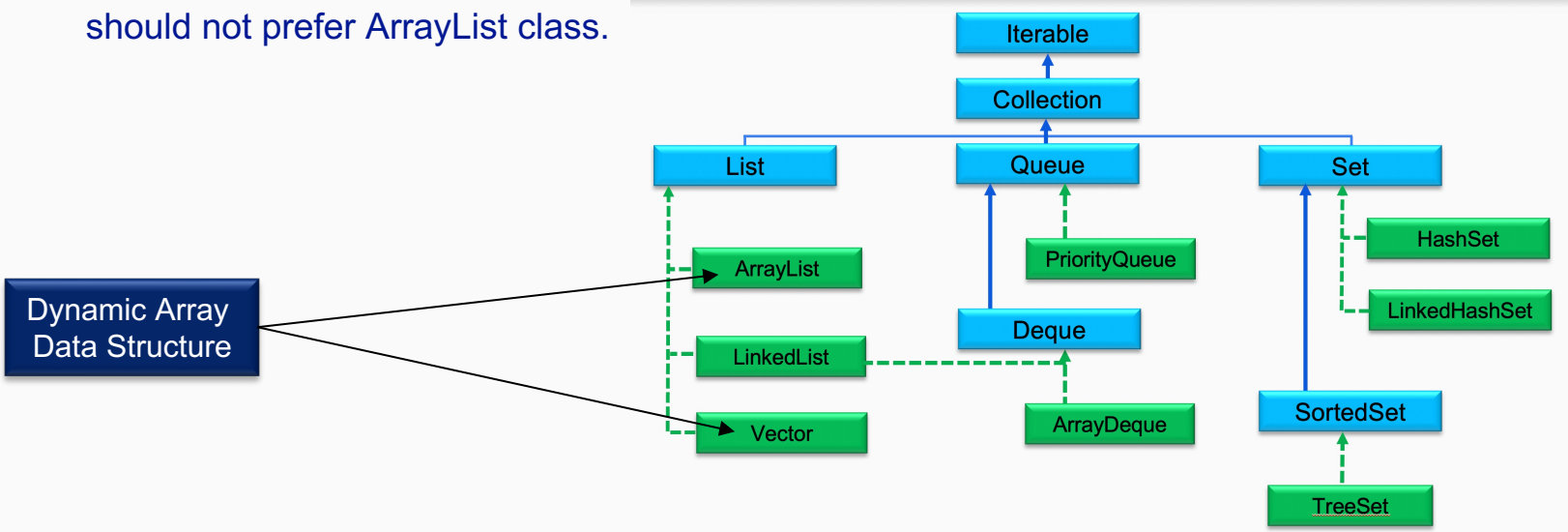
Dynamic Array

- Dynamic arrays are arrays that can grow automatically.
- Dynamic Arrays use dynamic memory allocation.

Do we have Dynamic Arrays in Collections

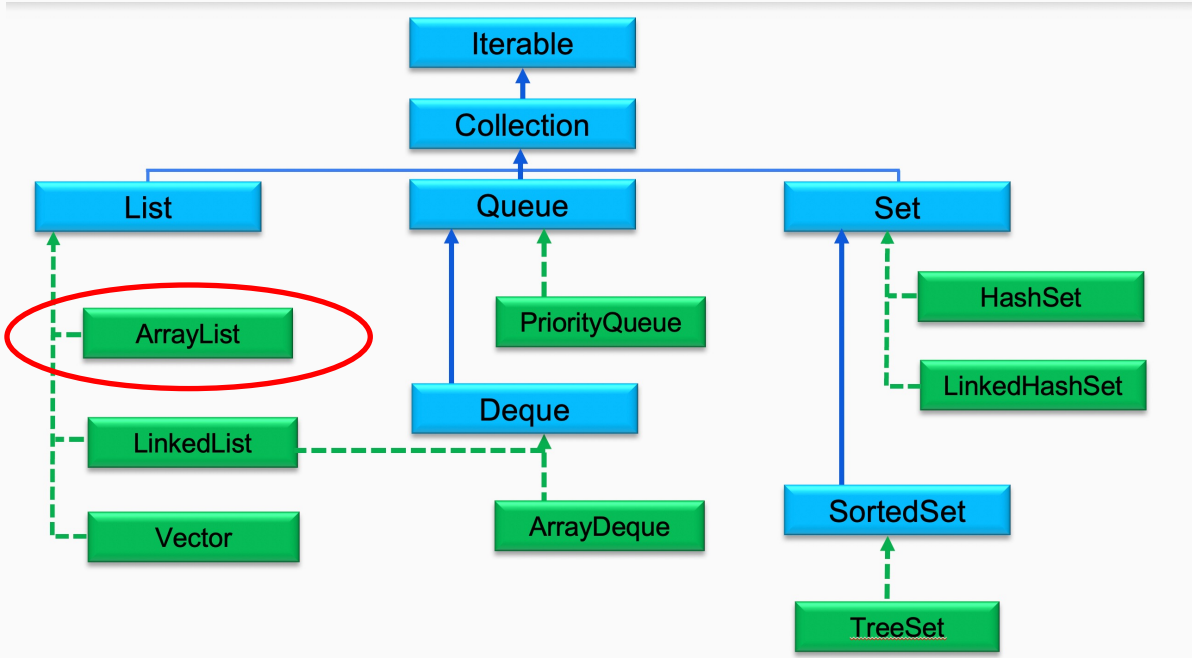
There two implementations of Dynamic Arrays in Java:

- Vector class : Size is increased by %100 if full. Synchronized (Only a single thread can access in multi-threaded environment)
- ArrayList class: Size is increased by % 50 if full. If you need multithreads access to data you should not prefer ArrayList class.



ArrayList

- Resizable-array implementation of the List interface.
- This class provides methods to manipulate the size of the array that is used internally to store the list.
- This class is roughly equivalent to Vector, except that it is unsynchronized.



ArrayList

- ArrayList internally uses an array to store the elements. Just like arrays, It allows you to retrieve the elements by their index.
- Java ArrayList allows duplicate values.
- Java ArrayList is an ordered collection. It maintains the insertion order of the elements.
- You cannot create an ArrayList of primitive types like int, char etc. You need to use boxed types like Integer, Character, Boolean etc.

ArrayList-Methods

Method	Description
add(E e)	Appends the specified element to the end of this list.
add(int index, E element)	Inserts the specified element at the specified position in this list
clear()	Removes all of the elements from this list.
clone()	Returns a shallow copy of this ArrayList instance.
contains(Object o)	Returns true if this list contains the specified element.
ensureCapacity(int minCapacity)	Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.
get(int index)	Returns the element at the specified position in this list.
indexOf(Object o)	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
isEmpty()	Returns true if this list contains no elements.
iterator()	Returns an iterator over the elements in this list in proper sequence.
lastIndexOf(Object o)	Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

ArrayList-Methods (cont.)

Method	Description
remove(int index)	Removes the element at the specified position in this list.
remove(Object o)	Removes the first occurrence of the specified element from this list, if it is present.
removeRange(int fromIndex, int toIndex)	Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive.
set(int index, E element)	Replaces the element at the specified position in this list with the specified element.
size()	Returns the number of elements in this list.
subList(int fromIndex, int toIndex)	Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.
toArray()	Returns an array containing all of the elements in this list in proper sequence (from first to last element).

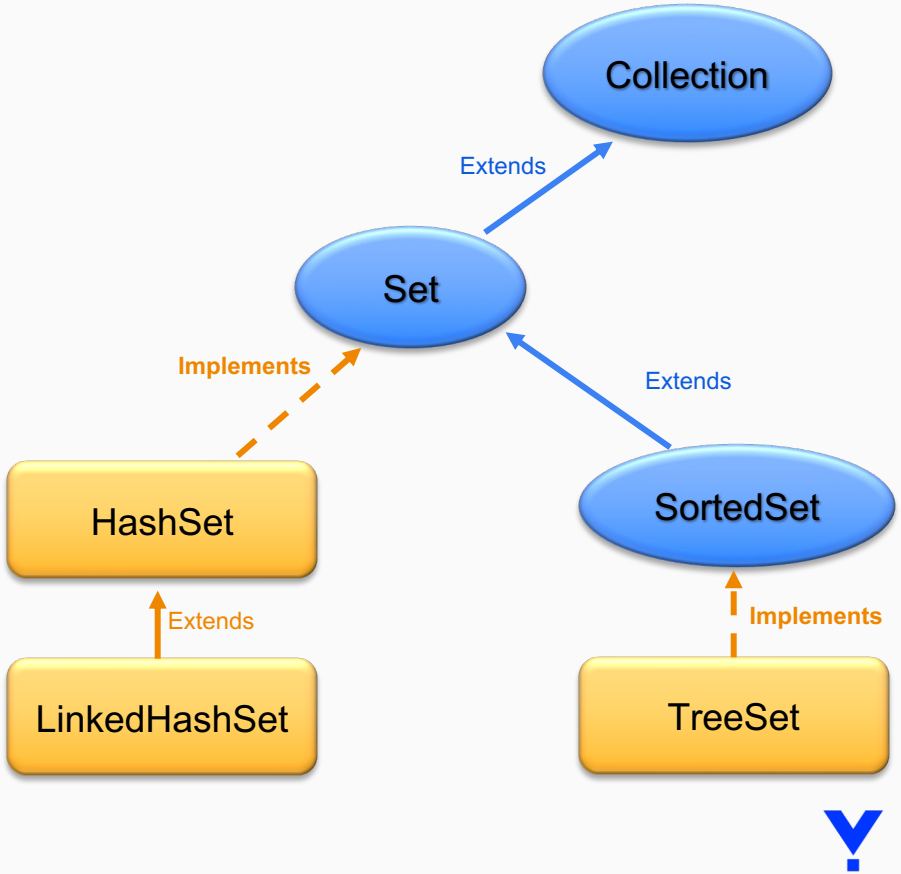
Loop Through Collection

1. For each loop
2. Any other loop(for, while, do while) by using get(index) method
3. Iterator
4. forEach method that came with java 8 (lambda expression)

ArrayList Example on IntelliJ

Set

- Set is child interface of Collection.
- Duplicates are NOT allowed.



HashSet

- HashSet class is used to create a collection that uses a hash table for storage.
- HashSet contains unique elements only.
- HashSet class is not synchronized.
- HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.

HashSet-Methods

Method	Description
add(E e)	Adds the specified element to this set if it is not already present. Returns boolean.
clear()	Removes all of the elements from this set.
clone()	Returns a shallow copy of this HashSet instance: the elements themselves are not cloned.
contains(Object o)	Returns true if this set contains the specified element.
isEmpty()	Returns true if this set contains no elements.
iterator()	Returns an iterator over the elements in this set.
remove(Object o)	Removes the specified element from this set if it is present.
size()	Returns the number of elements in this set (its cardinality).

Basic Set Operations

Lets switch to IntelliJ for Set review

Algorithm Problem (First Repeating Char in a String)

Find the first repeating char in a string.

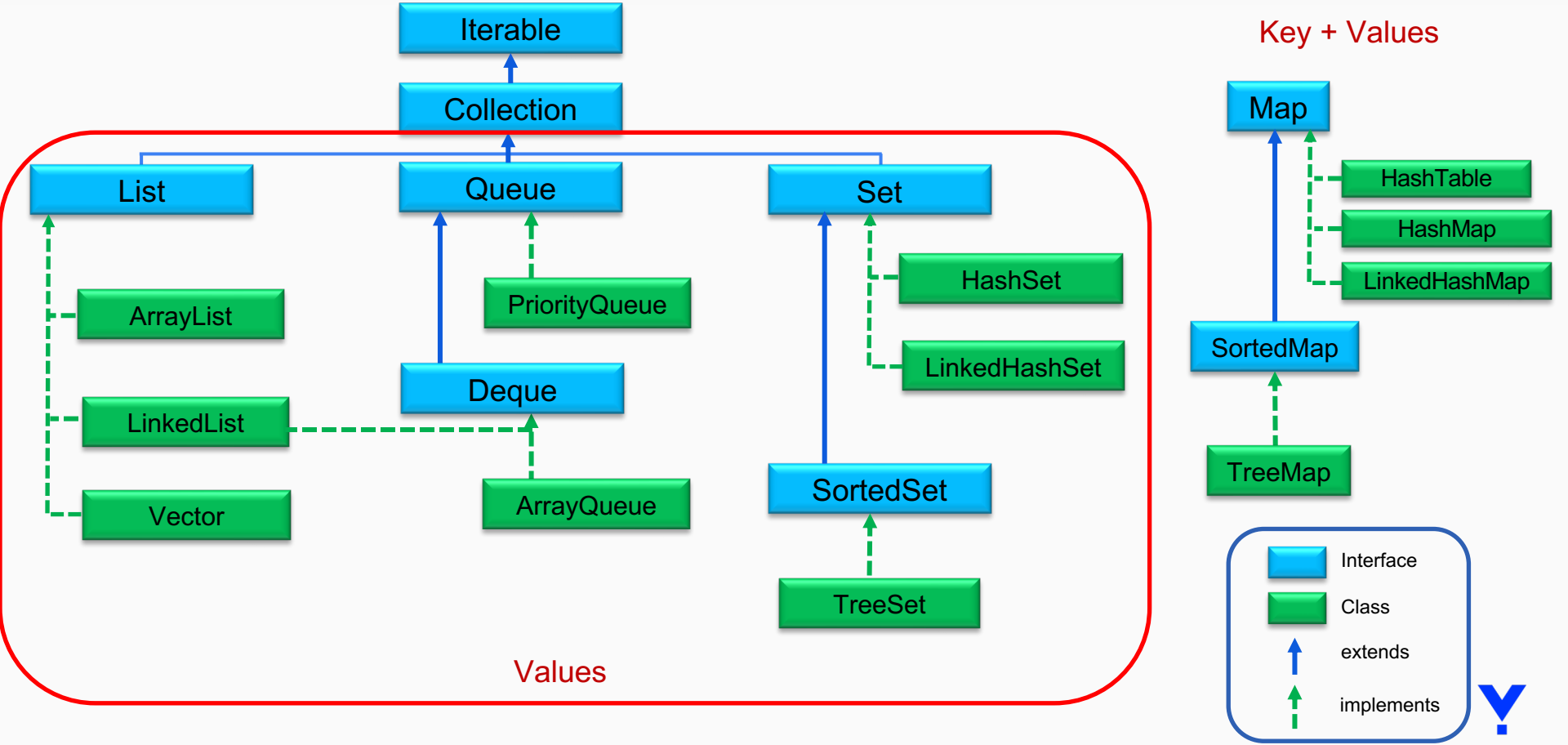
Example:

String="Java Developer"

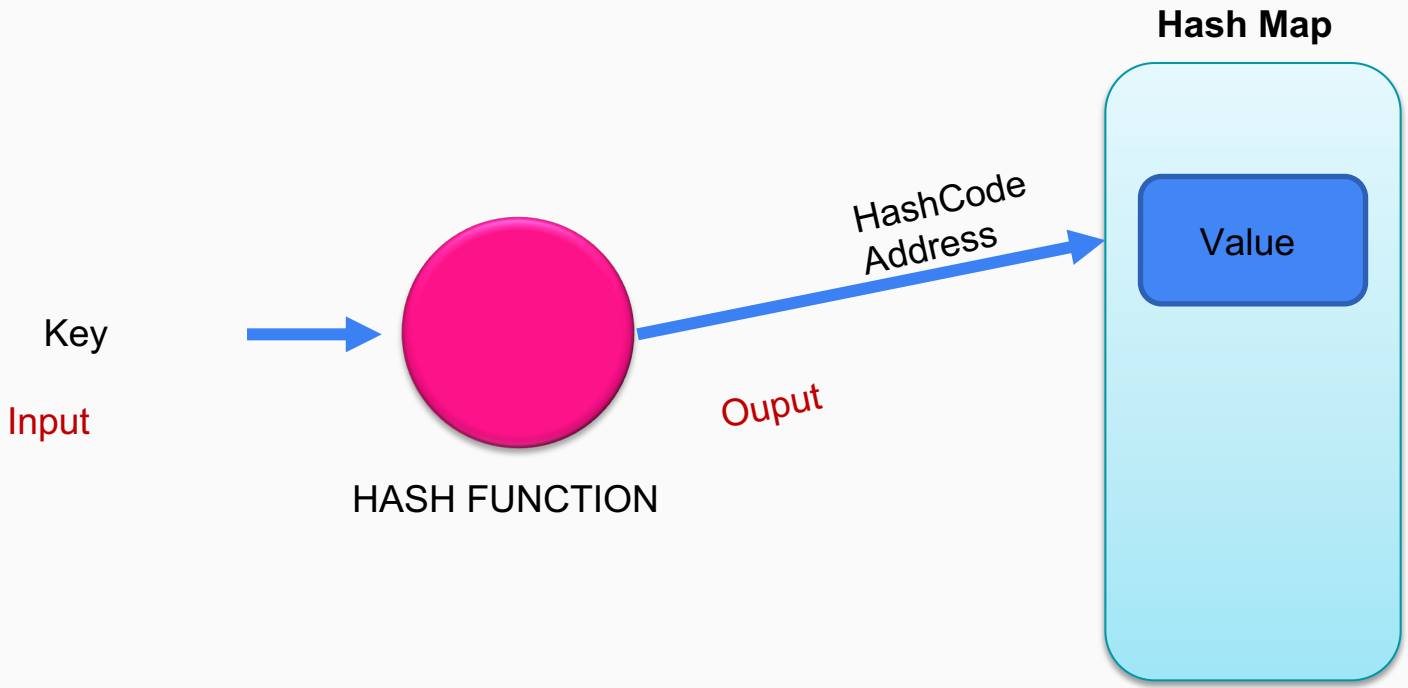
Output:

a

Collections with Values / Key+Values

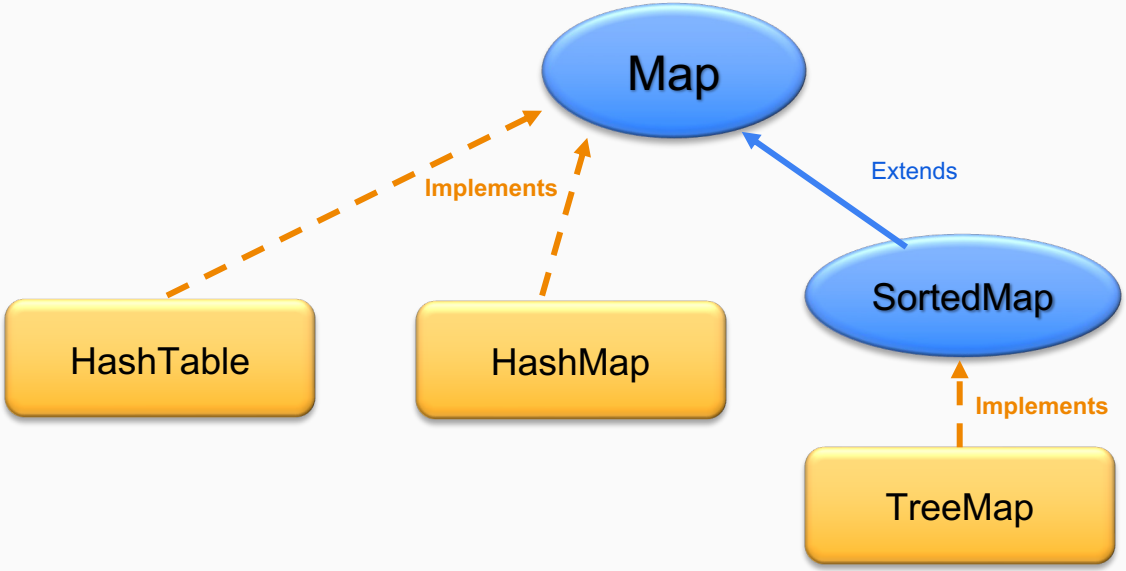


What is Hashing



Collection of Pairs : Map

- Data structure based on **key + value** pairs
- Map interface does not extend Collection interface



Map Basic Operations

Lets switch to IntelliJ for Map review

Algorithm Problem (First Non-Repeating Char in a String)

Find the first non-repeating char in a string.

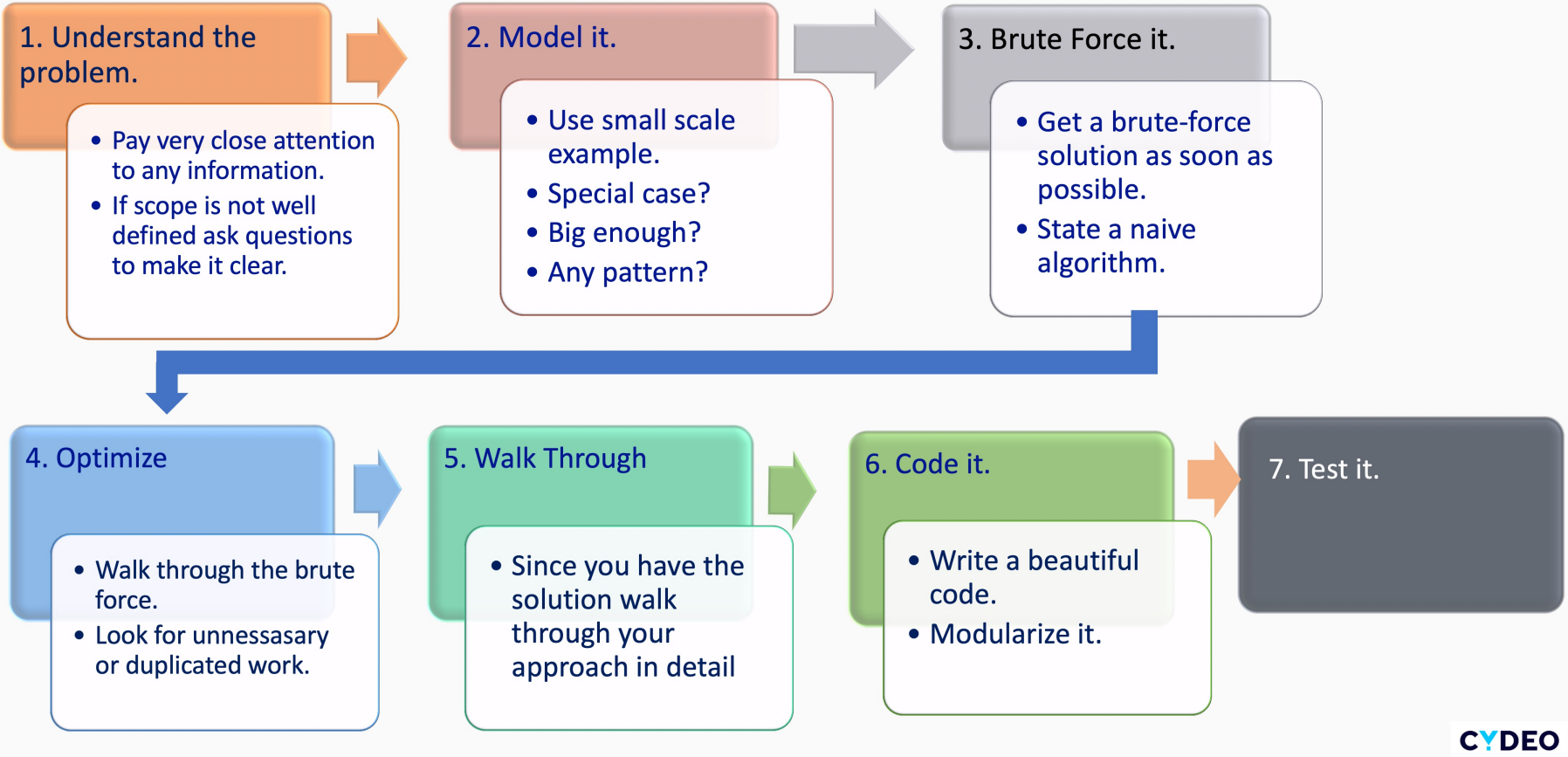
Example:

String="Java Developer"

Output:

J

Problem Solving Pattern



Problem Solving Using Collections

Problem: Two Sum

- Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*.
- You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.
- You can return the answer in any order.

Example:

Input: nums = [2, 7,11,15], target = 9

Output: [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

1. Understand The Problem

Ask questions to interviewers !

Is the array sorted?

- No

- Any duplicated values accepted

2. Model It

Looking for:

Such array[i]+ array[j] = target value;

Target value= 9

3. Find a Brute Force Solution

Lets switch to IntelliJ and write our first brute force solution.

4. Optimize the Brute Force Solution

Target value= 9

4. Optimize the Brute Force Solution

2	7	11	15

Target value= 9

1. Iterate array and put each (value & index) to a map.
2. Start iteration again;

```
for(int i=0;i<length();i++){
        if (Map.containsKey(target-array[i]))
        return new Array[i, Map.get(target-array[i])
}
```

HashMap

Key	value

5. Walk Through Optimal Solution

2	7	11	15

Target value= 9

1. Iterate array and put each (value & index) to a map.
2. Start iteration again;

```
for(int i=0;i<length();i++){
if (map.containsKey(target-array[i]))
return new int[] {i, map.get(target-array[i])}
}
```

i	Target-array[i]
0	9-2=7

HashMap

Key	Value
2	0
7	1
11	2
15	3

6. Code your solution

Lets switch to IntelliJ and write our first optimized solution.