CYDEO

Fast Track Day 05

Day 5 – Collection Framework

- Generics
- Collection Framework
- (I) Iterable
- (I) Collection
- (I) List: ArrayList, LinkedList, Vector, Stack
- (I) Set: HashSet, LinkedHashSet, TreeSet
- (I) Queue PriorityQueue, ArrayDeque
- (I) Map: HashMap, LinkedHashMap, TreeMap, HashTable

Generics

A generic allows you to define a parameter for your class

type parameters can be used to define a general type instead of the a normal object type
these are usually written as a single upper case character <T>

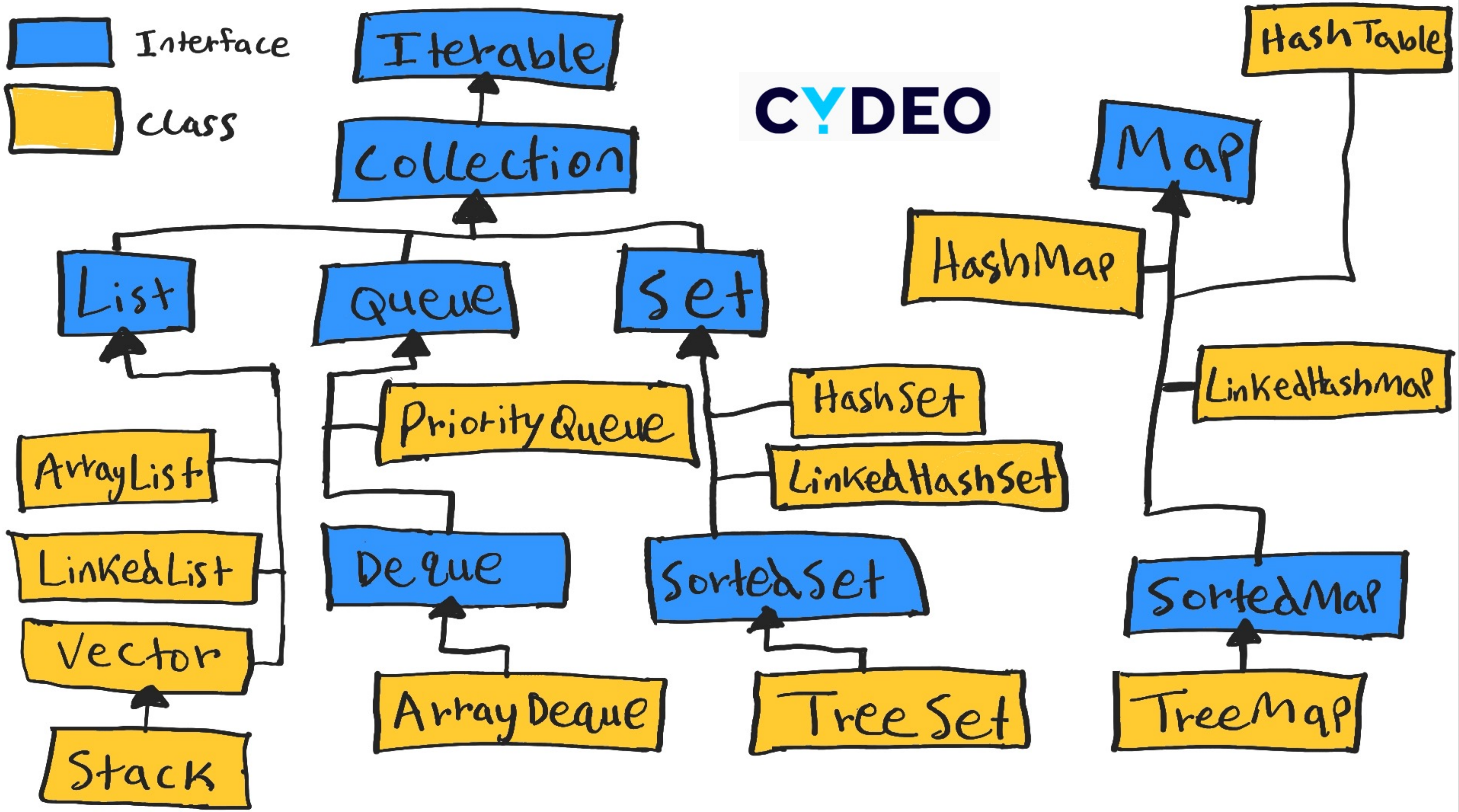
type parameters are used in the class and the actual object type that will be used for the type is defined when the class is used

The parameters and return types of methods can be flexible to work with multiple different types

Good use cases: Collection Framework, Functional Interface

Collection Framework

- The collections framework is built up of interfaces and classes that represent data structures with different algorithms to manipulate and handle data.
- The collections in the framework use different implementations to satisfy the algorithms being applied – So the programmer doesn’t need to implement them
- Only objects can be stored in these data structures
- We primarily need ArrayList & HashMap


Interface
class
Iterable
Collection
List
Queue
Set
ArrayList
LinkedList
Vector
Stack
PriorityQueue
Deque
ArrayDeque
HashSet
LinkedHashSet
SortedSet
TreeSet
HashTable
Map
HashMap
LinkedHashMap
SortedMap
TreeMap

Collection Interface

- The root is the Collection interface, and it inherits the Iterable interface
- This interface is implemented in all collection classes
- The methods declared in the the Collection interface are basic operations of the collections that implement them
- Some methods that are declared in Collection interface: add, remove, contains, size, toArray
- Constructors: no argument & Collection

List Interface

- Inherits the Collection interface
- This is an ordered collection, which allows full control of its elements. These elements are accessed by index numbers, starting from 0.
- This collection allows duplicate elements
- Classes that implement List interface: ArrayList, LinkedList, Vector

List Classes

ArrayList: This data structure acts as a resizable array. Uses arrays internally

LinkedList: This data structure uses nodes, which are objects that have data and reference to the next node. Also implements Deque interface (doubly linked).

Vector: This data structure is a legacy version of ArrayList. It is a resizable array, but it is synchronized

Synchronization – thread safety : multiple threads cannot access object at the same time

Stack: Class inherits the vector class and acts as a LIFO (last in first out) collection

Set Interface

- Inherits the Collection interface
- This collection is an unordered data structure, meaning the order of elements is not maintained and so there is no indexes
- This collection does not allow duplicate elements
- Classes that implement Set interface: HashSet, LinkedHashSet
- SortedSet interface inherits the Set interface but allows the ordering of the elements. These elements are automatically ordered by their natural order. TreeSet implements this interface.

Set Classes

HashSet: The algorithm used to implement this class is hash table.
Short version of hash table algorithm: fast and efficient approach to find elements

LinkedHashSet: Implementing the hash table and linked list this data structure will maintain the insertion order

TreeSet: This data structure is a naturally ordered Set that has additional methods to search for information. Does not allow null element

Queue Interface

- Inherits the Collection interface

- This data structure has additional features to insert, extract, and look for information. These method have two forms: one which will throw exception upon failure and and one which will return a value

  [add() == offer()], [remove() == poll()], [element() == peek()]

- Queues are usually FIFO (first in first out) meaning elements are added at the tail end of the data structure. BUT this is not always the case. It depends on the classes implementing this interface, they will have to define the insertion direction.

Queue Classes

PriorityQueue: This class implements Queue but does not follow FIFO. This data structure will store element using a priority algorithm which has some influence from the natural order. null is not a valid element

Deque: An interface which allows access to elements from the beginning or end

ArrayDeque: Class that implements Deque which enables it to access elements from both sides of the data structure. Does not allow null elements.

→ Faster than Stack if used as a stack type and faster than a LinkedList if used as a queue type

Iterable

- This interface is inherited to the Collection interface, so any class that implements the Collection interface would be implementing the Iterable interface

- Implementing the Iterable interface allows the objects to be used in the for each loop

- The Iterable interface defines a method that returns an iterator object. The collection can be iterated manually with that iterator object, (continually cycle through the elements if there still is one). The iterator also allows an element to be removed

For Each Loop

- Any data structure that implements the Collection interface, and as a result, the Iterable interface can be used in a for each loop to iterate through all the elements in that collection.

- Due to the design of the loop, it is only possible to iterate through the elements without making any changes. This is because the loop uses the iterator.
    - Attempting to make a change will result in a getting the ConcurrentModificationException

- It is also only possible to iterate through one collection at a time

Map Interface

- The map interface **does not** implement the Collection interface. Official documentation defines this interface to be part of the java collection framework

- A map is a data structure which works with a key/value format. Each key/value pair is called an entry.

- Every key is linked to a single value. Maps do not allow duplicate keys but have no issue with duplicate values.

- Classes that implement the map interface: HashMap and LinkedHashMap

- The Map interface is also implemented in the SortMap interface, which is implemented to the class TreeMap

Map Classes

HashMap: Like HashSet, HashMap is designed with the Hash table algorithm. The order of entries is not guaranteed over time

LinkedHashMap: This class implements the Map interface and inherits HashMap. The insertion order is maintained.

TreeMap: Implements the SortedMap interface. The data structure will have an ascending order for the keys. Cannot have a null key

HashTable: a synchronized legacy class. The order of entries is random and null is not allowed as a key or value

implements Map and extends Dictionary

Questions

- What is the difference of Array vs ArrayList?
  - Arrays have a fixed size
  - ArrayLists’ size is automatically adjusted
  - Arrays can hold both primitive and object types
  - ArrayList can hold only object types
  - Array is a build in data structure
  - ArrayList is implementing class of List interface in Collection framework

- What is the difference of List vs Set
  - List > Ordered and Indexed Collection, May contain duplicates
  - Set > Collection of Unique values, not ordered, no indexing

Questions

- What is the difference of ArrayList vs LinkedList?

  ArrayList is array based, internally uses array
  LinkedList is a doubly linked list
  LinkedList consists of nodes/values that are related to each other
  ArrayList and LinkedList both maintain ordering
  ArrayList and LinkedList both allow duplicates
  ArrayList is better to store and get information. LinkedList is better to manipulate information

- What is the difference of ArrayList vs Vector?

  ArrayList is not thread safe/not synchronized
  Vector is thread safe/synchronized
  ArrayList is faster than Vector
  Both allow duplicate values and keep ordering
  Both are implementations of List interface

Questions

- What is the difference of Iterator vs ListIterator

Iterator will cycle through the elements from beginning to end, but ListIterator is able to go in both directions.

Iterator can be used in any collection type which implements the collection interface, but ListIterator can only be used for the classes that implement List

There is a remove method for Iterator, but ListIterator has more operations available like add, set, remove

- How do you iterate though HashMap?

We can use the method keySet() to get a Set of the keys from the map and iterate through all those keys, which will allow us to iterator through each entry, values() method gives all the values, or iterate through the all the entries with entrySet() method

Questions

- Difference between HashMap and HashTable?

  HashTable is thread-safe/synchronized **|** HashMap is not thread safe and faster
  HashTable does not allow null key or value **|** HashMap allows one null key

HashTable is not recommended to use, for a thread safe map the ConcurrentHashMap is preferred

- Difference between HashMap and TreeMap?

  HashMap doesn’t maintain any order | TreeMap will store keys in ascending order
  HashMap can have one ’null’ key | TreeMap cannot have any ‘null’ key
  Neither HashMap nor TreeMap are synchronized

Questions

- Difference between HashSet and TreeSet?

  Both are implementing the Set interface
  HashSet performs better than TreeSet
  HashSet doesn’t guarantee any ordering, but TreeSet will maintain natural order

- How can you avoid the ConcurrentModificationException while iterating through collections?

  You shouldn’t change the collection during iteration, but if you need to you can use the Iterator object to manually iterate through the collection, or an object such as ConcurrentHashMap which is like HashMap but will allow you to make modification during iteration.

- Any combination of comparisons: Queue vs Stack, Stack vs Vector

Questions

- comparable vs comparator

  comparable is interface. should implement comapreTo() method in the class to compare each object

  comparator is implemented into a separate class which defines one exactly is going to be compared (year, rating, or name)

  Collections.sort(list, COMPARATOR HERE)

- Enumeration vs iterator

  Enumeration is a legacy interface which is used for traversing Vector, HashTable.

  Enumeration interface acts as a read only interface, one can not do any modifications to Collection while traversing the elements of the Collection.

  Enumeration does not have the remove() method