RE-BOOTCAMP

Collections

ARRAYS

- Arrays are a data structure that are used to store multiple values of the given datatype.
- These values are stored as elements that can be accessed though an index number. The index number starts at 0 for the first element.
- Key syntax for arrays is the [] symbols
- The size for arrays is fixed, so it cannot be changed.
- Declaring an array will give the element default values based on the datatype: 0 (integers), null (objects)

COLLECTIONS FRAMEWORK

- The collections framework is built up of interfaces and classes that builds data structures with different algorithms to manipulate data.

- The collections in the framework use different implementations to satisfy the algorithms being applied

- Only objects can be stored in these data structures

COLLECTION INTERFACE

- At the highest level is the Collection interface, inherits the Iterable interface
- This interface is implemented in all collection classes
- Some method that are declared in Collection interface: add, remove, contains, size, toArray
- The methods declared in the the Collection interface are basic operations of the collections that implement them.

ITERABLE INTERFACE

- This interface is inherited by the Collection interface, so any sub class of the Collection interface also inherits the Iterable interface
- Implementing the Iterable interface allows the objects to be used in the for each loop
- The Iterator interface is an iterator that can be used manually on a collection to enumerate, (continually cycle through the elements if there still is one) except 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 hides 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

LIST INTERFACE

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

ARRAYLIST, LINKEDLIST, VECTOR - CLASSES

- ArrayList: This data structure works similar as an array, but the size is expanded whenever needed

- LinkedList: This data structure has nodes, which are objects that have data and reference to another node. These nodes link for the list collection. Also implements Deque interface.

- Vector: This data structure is a legacy version of ArrayList. It is a resizable array, but it is synchronized
    - 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 elements are added is not maintained
- 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.

HASHSET, LINKEDHASHSET, TREESET - 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.

PRIORITYQUEUE, DEQUE, ARRAYDEQUE - CLASSES

- PriorityQueue: This class implements Queue but does not follow FIFO. This data structure will store element with their natural order. null is not a valid element

- Deque: An interface which allows elements to be used from the beginning or end

- ArrayDeque: Class that implements Deque which enables it to access elements from both sides of the data structure. This collection works faster than ArrayList and Stack

MAP INTERFACE

- The map interface **does not** implement the collections interface and thus is not a part of the 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

HASHMAP, LINKEDHASHMAP, TREEMAP - 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