Collections

Collections in Java is a framework that provides an ability to store and manipulate a group of objects. Any operation that you can perform on data, such as searching, sorting, insertion, manipulation, and deletion can be accomplished through Java Collections.

Not be even more confusing, inside of Collections, can be found Java Collection. This simply means a single unit of objects. Within Java Collection, you can find several useful classes such as:

- ArrayList
- LinkedList
- Queue
- HashMap
- HashSet

ArrayList
The ArrayList class implemented the List interface from Collections. This is a more flexible and robust option of normal array's. The biggest difference is the ability to list objects and the ability to resize the array. Adding and removing elements of an ArrayList are done simply by invoking the add/remove method… Here is an example:

```
listofcountries.add("Netherlands");
listofcountries.remove(5);
```

The list interface does not have the ability to sort an ArrayList, but this is easily accomplished using the sort method within the Collections interface. This is accomplished as shown below:

```
Collections.sort(listofcountries);
```

LinkedList
The LinkedList class is a doubly-linked implementation of the List and Deque interfaces. Here we have the ability to insert/remove, define values for the first/last items in the list, as well as get/set any value within the list.

Here is an example of adding first/last elements:

```
linkedlist.addFirst("First Item");
linkedlist.addLast("Last Item");
```

Here is an example of using the get/set methods:

```
String firstItem = linkedlist.get(0);
linkedlist.set(0, "Changed First Item");
```

Queue
The Queue interface uses two different implementations, LinkedList and PriorityQueue. Since Queue is an interface, we cannot instantiate it. We actually create instances of LinkedList or PriorityQueue and assign them to a Queue. Elements added to it are placed in at the end of the "queue" and removed from the beginning. This is similar to a "FIFO" approach with store inventory.

Queue does have specific methods that work differently than other lists. These methods include element, poll, and peek. The element method returns the head (or first) element of the queue. Poll removes the current head of the queue and returns the new head of the list. Peek does the same as element but returns a null value if the queue is empty. Here are some examples:

```
System.out.println("Head element: " + peopleQueue.element());
System.out.println("Removed element: " + peopleQueue.poll());
System.out.println("Head element: " + peopleQueue.peek());
```

HashMap
The HashMap is collection class implementing the Map interface and is best summarized as storing Key and Value pairs. This list does not return the key/value pairs in the same order they were inserted and the key/value pairs are not able to be sorted.

To display the contents of a HashMap, you need to use a different type of iterator to loop through the results. Here is an example

```
Set set = peopleMap.entrySet();
Iterator iterator = set.iterator();
while (iterator.hasNext()) {
      Map.Entry mapEntry = (Map.Entry)iterator.next();
      System.out.println("Key is: " + mapEntry.getKey() + " & value is: 
      " + mapEntry.getValue());
}
```

You can also easily retrieve specific elements of a HashMap with just a key value. Here is an example:

```
System.out.println("Specific Value: " + peopleMap.get(37));
```

HashSet
The HashSet is a collection class implementing the Set interface. It makes no guarantees about sort order (other classes in the Set interface do provide some sorting capabilities). HashSet does not allow for duplicates, duplicates would be overwritten with the same data.