Discussion-7: Collections: Hash Maps, Tree Maps, and Linked Lists

Discussion Topic:

What are the major differences between a list and a set in Java? How can you determine when to implement a list versus a set for an application? Provide an example code segment in Java that illustrates utilizing a list or a set. In response to your peers, include additional code segments of lists and sets.

My Post:

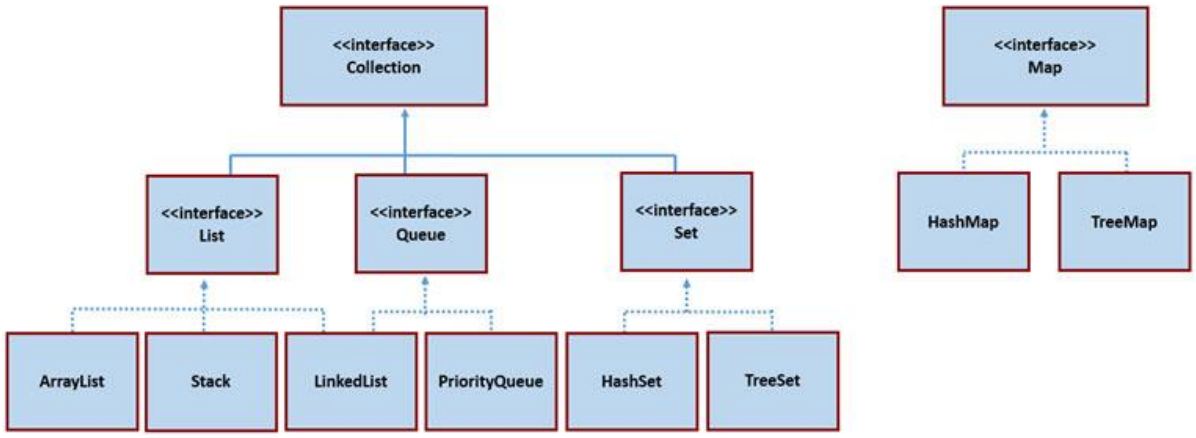
Hello class,

In linear algebra, lists are used to represent ordered collections of elements, typically in the form of vectors or matrices. On the other hand, set are used to represent unordered collections of elements. In Java, lists are ordered collections that allow duplicate elements, and the List interface is implemented by classes like ArrayList and LinkedList. Sets are unordered collections that do not allow duplicate elements, and the Set interface is implemented by classes like HashSet and TreeSet.

Note that in Java List and Set are interfaces that extend the Collection interface, this means that they share Collection interface methods. See the figure below for the Java collections framework.

Figure 1

Java Collections Framework



From: Module 7: Collections: Hash Maps, Tree Maps, and Linked Lists by CUS Global (n.d.).

The table below lists the difference between Lists and Sets:

Table 1

Lists vs Sets

List	Set
The List is an ordered sequence.	The Set is an unordered sequence.
List allows duplicate elements	Set doesn't allow duplicate elements.
Elements by their position can be accessed.	Position access to elements is not allowed.
Multiple null elements can be stored.	The null element can store only once.
List implementations are ArrayList, LinkedList, Vector, Stack	Set implementations are HashSet, LinkedHashSet.

From: List Interface in Java With Examples by GeeksforGeeks (2020)

As previously mentioned, in Java, List and Set are interfaces that extend the Collection interface. The program below is an example demonstrating some of the shared methods.

```
package main;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class Main {
   public static void main(String[] args) {
      List<String> fruitList = new ArrayList<>();
      Set<String> fruitSet = new HashSet<>();

      // Adding elements
      System.out.println("Adding elements:");
      System.out.println("List add 'Apple': " + fruitList.add("Apple"));
      System.out.println("List add 'Banana': " + fruitList.add("Banana"));
      System.out.println("List add 'Apple' again: " + fruitList.add("Apple")); // Allows duplicates

      System.out.println("Set add 'Apple': " + fruitSet.add("Apple"));
      System.out.println("Set add 'Banana': " + fruitSet.add("Banana"));
      System.out.println("Set add 'Apple' again: " + fruitSet.add("Apple")); // Doesn't allow duplicates

      System.out.println("List: " + fruitList);
      System.out.println("Set: " + fruitSet);
```

```
    // Size and contains
    System.out.println("\nSize and contains:");
    System.out.println("List size: " + fruitList.size());
    System.out.println("Set size: " + fruitSet.size());
    System.out.println("List contains 'Banana': " + fruitList.contains("Banana"));
    System.out.println("Set contains 'Banana': " + fruitSet.contains("Banana"));

    // Remove
    System.out.println("\nRemoving elements:");
    System.out.println("List remove 'Apple': " + fruitList.remove("Apple")); // Removes first occurrence
    System.out.println("Set remove 'Apple': " + fruitSet.remove("Apple"));

    System.out.println("List after remove: " + fruitList);
    System.out.println("Set after remove: " + fruitSet);

    // Clear
    fruitList.clear();
    fruitSet.clear();
    System.out.println("\nAfter clear:");
    System.out.println("List is empty: " + fruitList.isEmpty());
    System.out.println("Set is empty: " + fruitSet.isEmpty());

    // AddAll
    List<String> moreFruits = Arrays.asList("Cherry", "Date", "Cherry");
    fruitList.addAll(moreFruits);
    fruitSet.addAll(moreFruits);

    System.out.println("\nAfter addAll:");
    System.out.println("List: " + fruitList);
    System.out.println("Set: " + fruitSet);

    // Iterator (enhanced for loop)
    System.out.println("\nIterating:");
    System.out.print("List iteration: ");
    for (String fruit : fruitList) {
        System.out.print(fruit + " ");
    }
    System.out.print("\nSet iteration: ");
    for (String fruit : fruitSet) {
        System.out.print(fruit + " ");
    }
  }
}
```

Output:

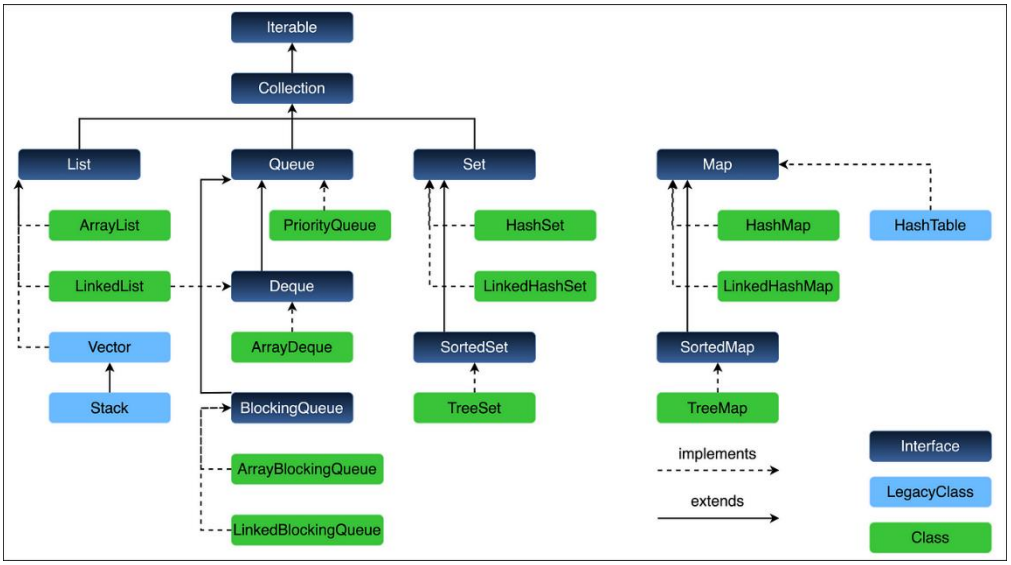
```
Adding elements:
List add 'Apple': true
List add 'Banana': true
List add 'Apple' again: true
Set add 'Apple': true
Set add 'Banana': true
Set add 'Apple' again: false
List: [Apple, Banana, Apple]
```

```
Set: [Apple, Banana]

Size and contains:
List size: 3
Set size: 2
List contains 'Banana': true
Set contains 'Banana': true

Removing elements:
List remove 'Apple': true
Set remove 'Apple': true
List after remove: [Banana, Apple]
Set after remove: [Banana]

After clear:
List is empty: true
Set is empty: true

After addAll:
List: [Cherry, Date, Cherry]
Set: [Cherry, Date]

Iterating:
List iteration: Cherry Date Cherry
Set iteration: Cherry Date
```

Note that both lists and sets can be iterated using an enhanced for loop. This is because both interfaces (List and Set) inherit from the Collection interface, which in turn inherits from the Iterable interface. See the figure below, which provides a detailed depiction of the Java class/interface framework.

Figure 2

Java Class/Interface Framework — Class Hierarchy



Note: The Map interface does not inherit from the Collection interface. From Java Collections Framework — Class Hierarchy by Gupta (2024).

-Alex

References:

CUS Global (n.d.). *Module 7: Collections: Hash maps, tree maps, and linked lists* [Interactive lecture]. In Colorado State University Global, CSC372: Programming II, Computer Science Department. Canvas. Retrieved July 21, 2024, from https://csuglobal.instructure.com/courses/94948/pages/7-dot-1-overview-of-the-java-collections-framework?module_item_id=4868915

GeeksforGeeks. (2020, June 25). *List interface in Java with examples.* GeeksforGeeks. https://www.geeksforgeeks.org/list-interface-java-examples/

Gupta, V. (2024, January 2). *Java collections framework — Class hierarchy*. Medium. https://levelup.gitconnected.com/java-collections-framework-class-hierarchy-latest-2024-51f9154f1f57