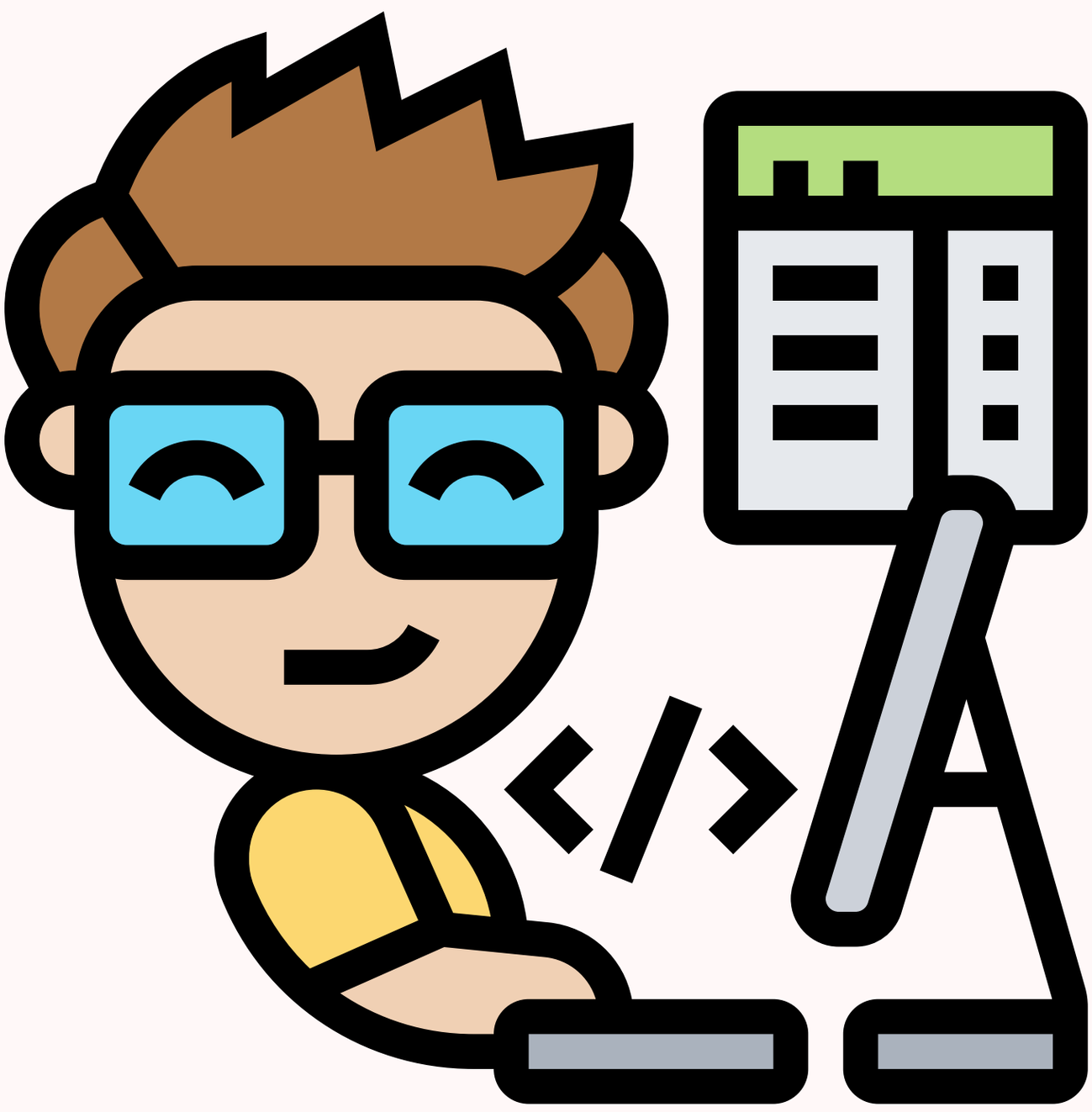
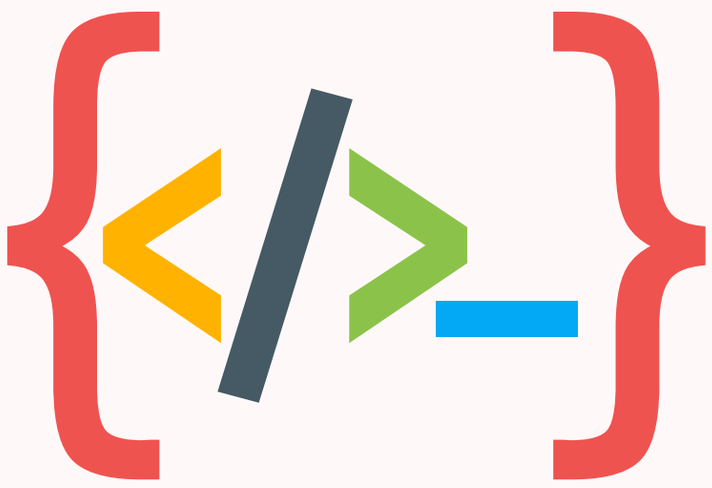
JAVA COMPLETE COLLECTION FRAMEWORK





linkedin.com/in/soumyadip-chowdhury/

youtube.com/println

@s_oumyadip

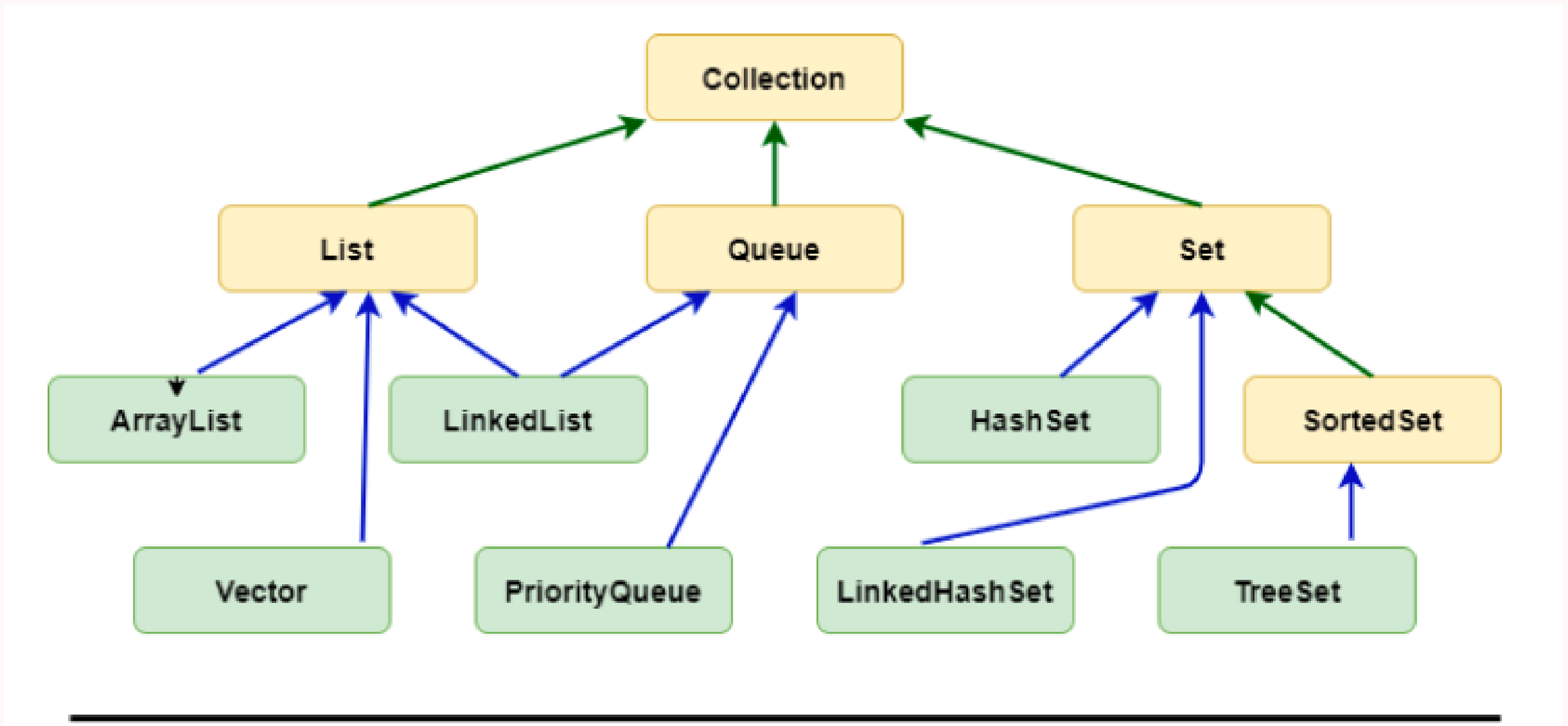
INTRODUCTION

Java's Collection framework provides a way to store and manipulate groups of objects. Java Collections can be used for searching, sorting, inserting, manipulating, and deleting data.

In Java, a collection is a single unit of objects. ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet, and LinkedList are all classes of Java Collection framework.

/soumyadip-chowdhury

Hierarchy of Collection Framework



/soumyadip-chowdhury

List Interface

The List interface is a child interface of the Collection interface. We cannot store an ordered collection of objects in a list-type data structure. Values can be duplicated.

Arraylist

The list is implemented by the ArrayList class. It uses a dynamic array to store the duplicate element of different data types.

The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed.

/soumyadip-chowdhury

ARRAYLIST Snippet

```java
Import java.util.*;

class TestJavaCollection1{

public static void main(String args[]){

    ArrayList<String> list=new ArrayList<String>();
    list.add("SOUMYADIP");/
    list.add("CODEWITH007");
    list.add("GOOGLE");
    list.add("LINKEDIN");

   }
}
```

/soumyadip-chowdhury

LinkedList

The LinkedList class implements the Collection interface. Internally, the elements are stored in a doubly linked list. Duplicate elements can be stored.

There is no synchronization and i maintains the order of insertion. In LinkedList, the manipulation is fast because no shifting is required.

/soumyadip-chowdhury

LinkedList Snippet

```java
import java.util.*;

public class SoumyadipJava{

public static void main(String args[]){
      LinkedList<String> al=new LinkedList<String>();
      al.add("Soumyadip");
      al.add("Codewith007");
      Iterator<String> itr=al.iterator();
      while(itr.hasNext()){
      System.out.println(itr.next());
      }
    }
}
```

Vector

Vector stores the data elements in a dynamic array. It resembles an array list.

It is synchronized, though, and has a lot of methods that are not included in the Collection framework.

/soumyadip-chowdhury

Vector Snippet

```java
import java.util.*;
public class SoumyadipJava{

public static void main(String args[]){
      Vector<String> v=new Vector<String>();
      v.add("Soumyadip");
      v.add("Java");
      Iterator<String> itr=v.iterator();
      while(itr.hasNext()){
         System.out.println(itr.next());
         }
      }
}
```

/soumyadip-chowdhury

Stack

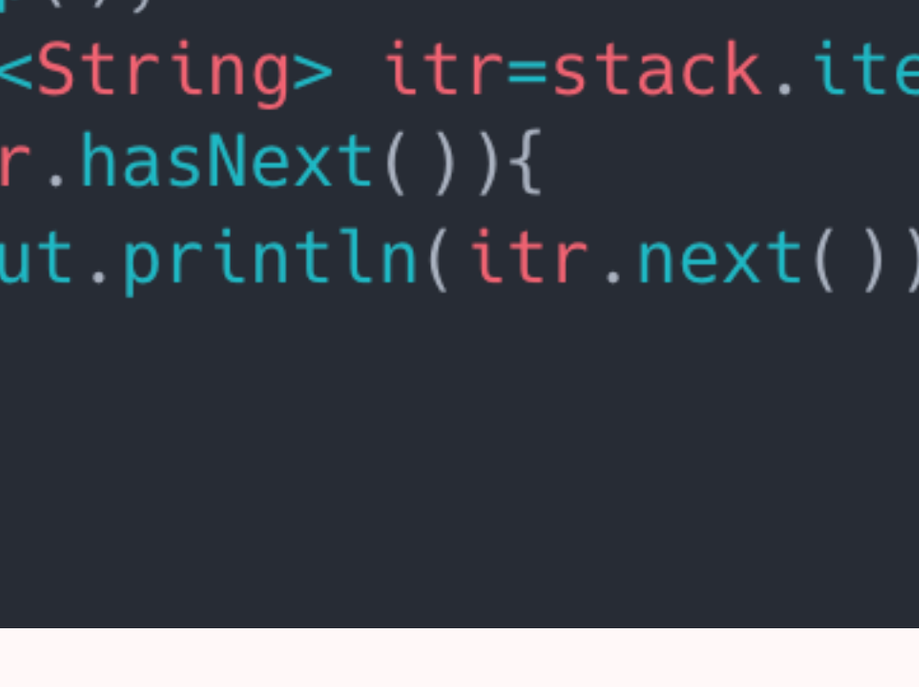
A subtype of the vector is the stack. It uses the stack data structure, or last-in, first-out.

The stack offers the methods of the Vector class, including boolean push(), boolean peek(), and boolean push(object o), which specify the class's properties.

/soumyadip-chowdhury

Stack Snippet

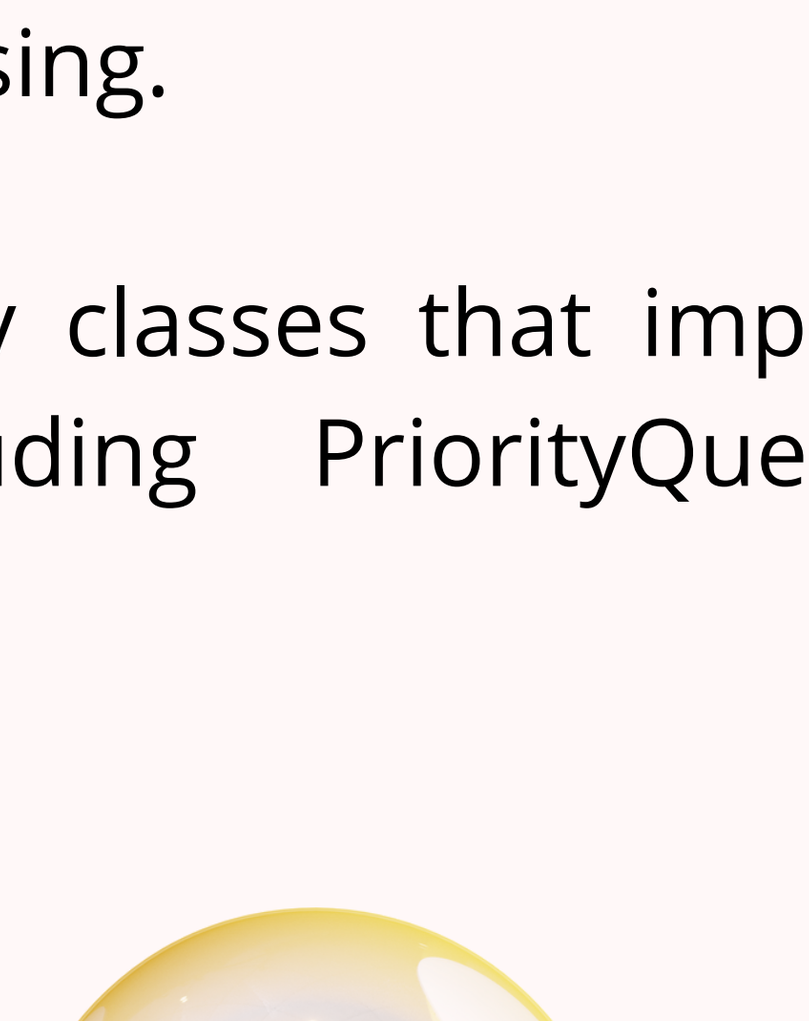
```java
import java.util.*;
public class Soumyadip{

public static void main(String args[]){
        Stack<String> stack = new Stack<String>( );
        stack.push("Soumyadip");
        stack.push("Println");
        stack.pop( );
        Iterator<String> itr=stack.iterator( );
        while(itr.hasNext( )){
        System.out.println(itr.next( ));
        }
    }
}
```



Queue

The order of first-in, first-out is maintained through the queue interface. It can be characterized as an ordered list used to store pieces that are scheduled to undergo processing.

There are many classes that implement the Queue interface, including PriorityQueue, Deque, and ArrayDeque.



/soumyadip-chowdhury

Queue Snippet

```
Queue<String> q1 = new PriorityQueue( );
Queue<String> q2 = new ArrayDeque( );
```

/soumyadip-chowdhury

PriorityQueue

The Queue interface is implemented by the PriorityQueue class.

It contains the substances or things that must be handled according to their priority. Null values cannot be saved in the queue using PriorityQueue.

/soumyadip-chowdhury

Priority Queue Snippet

```java
import java.util.*;

public class SoumyadipJava{

public static void main(String args[]){

    PriorityQueue<String> queue=new PriorityQueue<String>( );
    queue.add("Amit Sharma");
    queue.add("Vijay Raj");
    queue.add("JaiShankar");
    queue.add("Raj");
    }

}
```

/soumyadip-chowdhury

Deque Interface

The Queue interface is expanded by the Deque interface. In Deque, we have the ability to remove and add parts from either side.

A double-ended queue known by the abbreviation DEQUE allows us to conduct operations at both ends.

/soumyadip-chowdhury

ArrayDeque

ArrayDeque class implements the Deque interface. It facilitates us to use the Deque. Unlike queue, we can add or delete the elements from both ends.

ArrayDeque is faster than ArrayList and Stack and has no capacity restrictions.

/soumyadip-chowdhury

ArrayDeque Snippet

```java
import java.util.*;

public class SoumyadipJava{

public static void main(String[] args) {

        Deque<String> deque = new ArrayDeque<String>( );
        deque.add("Gautam");
        deque.add("Karan");
        deque.add("Ajay");

        for (String str : deque) {
        System.out.println(str);
        }

    }
}
```

/soumyadip-chowdhury

Set Interface

The Java.util package contains the Set Interface. The Collection interface is expanded by it.

It stands for an unordered set of elements that prevents us from storing duplicate stuff. In Set, there can only be one null value stored.

HashSet, LinkedHashSet, and TreeSet all implement Set.

/soumyadip-chowdhury

Set Snippet

```
Set<data-type> s1 = new HashSet<data-type>();
Set<data-type> s2 = new LinkedHashSet<data-type>();
Set<data-type> s3 = new TreeSet<data-type>();
```

/soumyadip-chowdhury

HashSet

Set Interface is implemented by HashSet class. It stands for the collection that stores data using hash tables.

The components of the HashSet are stored using hashing. It has special stuff in it.

/soumyadip-chowdhury

Set Snippet

```java
import java.util.*;

public class SoumyadipJava{

public static void main(String args[]){

      HashSet<String> set=new HashSet<String>( );
      set.add("Soumyadip");
      set.add("CodeWith007");
      set.add("Println");

      Iterator<String> itr=set.iterator( );

      while(itr.hasNext( )){
      System.out.println(itr.next( ));
      }

      }
}
```

/soumyadip-chowdhury

LinkedHashSet

The LinkedList implementation of the Set Interface is represented by the LinkedHashSet class.

It implements the Set interface and extends the HashSet class. It has distinctive components just like HashSet. It permits null elements and preserves insertion order.

/soumyadip-chowdhury

LinkedHashSet Snippet

```java
import java.util.*;

public class SoumyadipJava{

public static void main(String args[]){

      LinkedHashSet<String> set=new LinkedHashSet<String>( );
      set.add("Soumyadip");
      set.add("Println");
      set.add("Codewith007");

      Iterator<String> itr=set.iterator( );

      while(itr.hasNext( )){
          System.out.println(itr.next( ));
      }

    }
}
```

SortedSet Interface

The alternative to the Set interface, known as SortedSet, offers a complete ordering of its items.

The SortedSet's elements are organized in ascending (increasing) order. The SortedSet offers extra methods that prevent the elements' default sorting.

/soumyadip-chowdhury

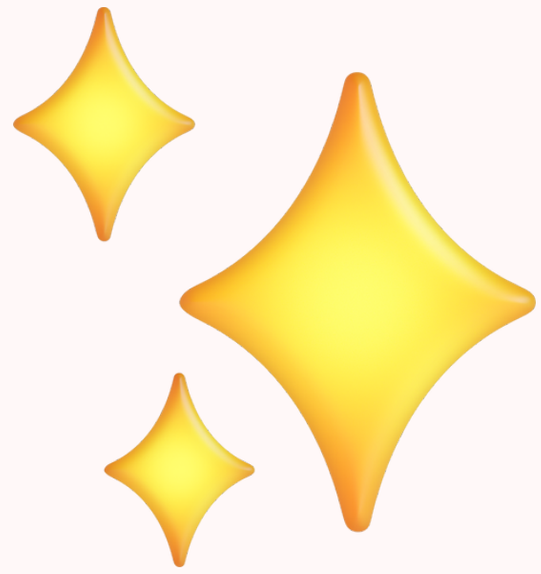
TreeSet Interface

The Set interface, which uses a tree for storage, is implemented by the Java TreeSet class.

TreeSet has distinct components, just as HashSet. However, TreeSet has fairly quick access and retrieval time. The TreeSet stores its elements in ascending order.

/soumyadip-chowdhury

TreeSet Snippet



```java
import java.util.*;

public class SoumyadipJava{

public static void main(String args[]){

      TreeSet<String> set=new TreeSet<String>( );
      set.add("Soumyadip");
      set.add("Java");
      set.add("Println");
      set.add("Codewith007");

      Iterator<String> itr=set.iterator( );
      while(itr.hasNext( )){
         System.out.println(itr.next( ));
      }

   }
}
```

/soumyadip-chowdhury

And for amazing stuff you can follow me

Soumyadip Chowdhury

soumyadip-chowdhury

@s_oumyadip

@println

References:

https://www.javatpoint.com/collections-in-java