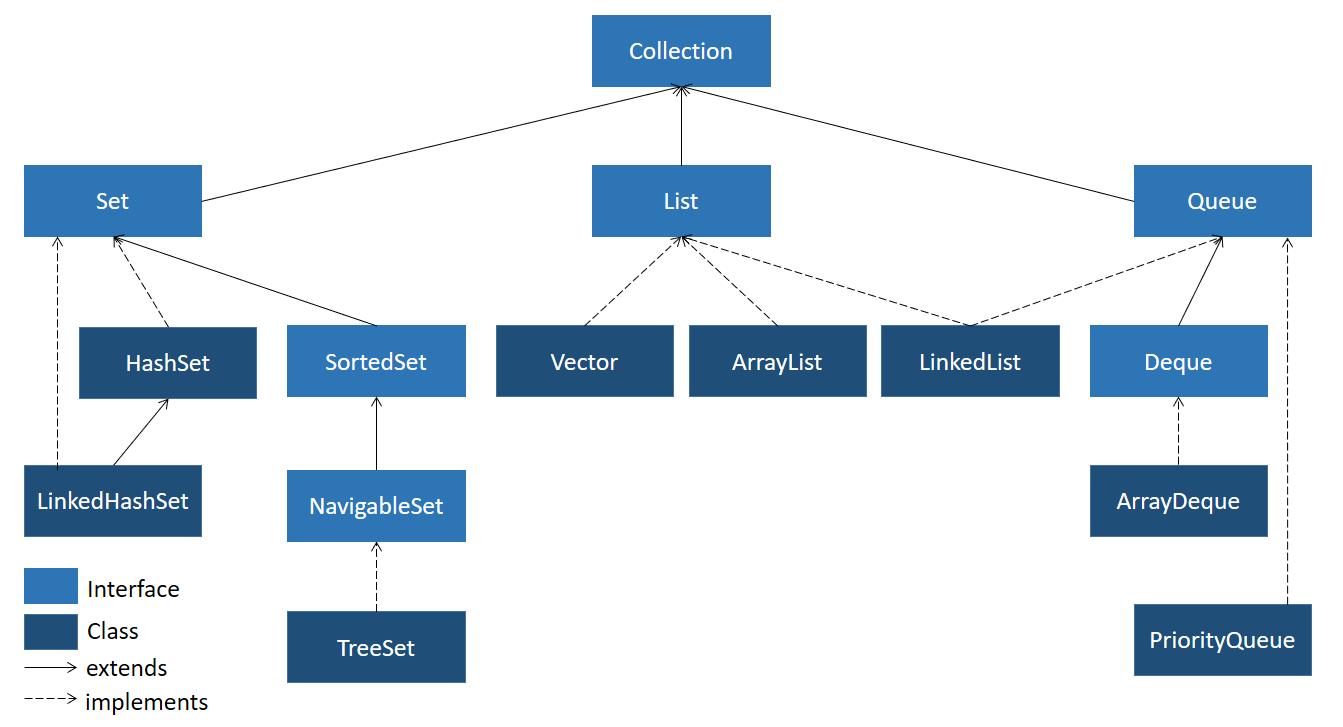
**Collections Framework and Generics:**

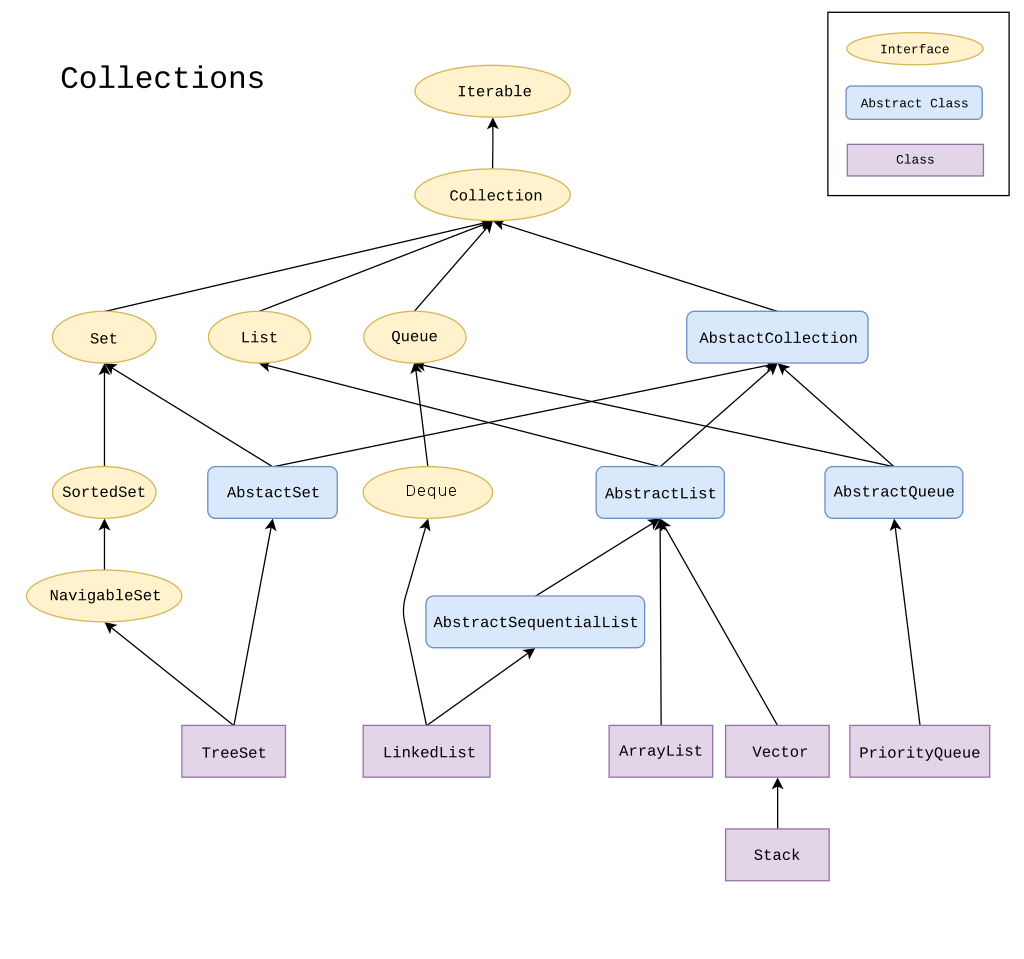
A collection is an object that groups multiple elements into a single unit. Collections are used to store, retrieve and manipulate data.

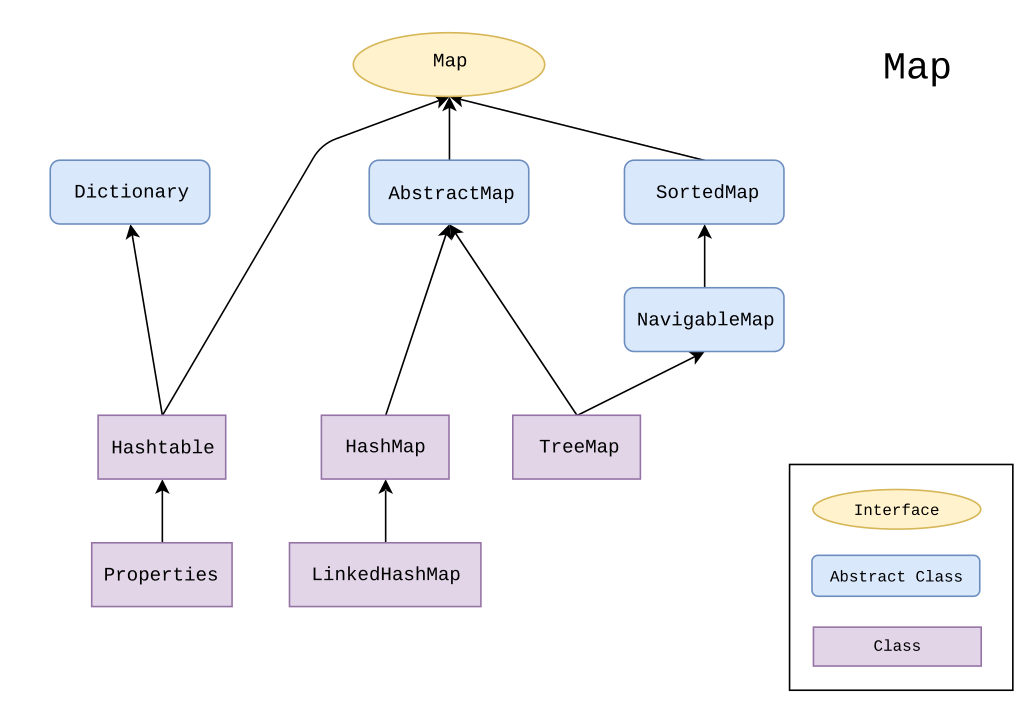
Collections framework in Java helps us in representing and manipulating collections.

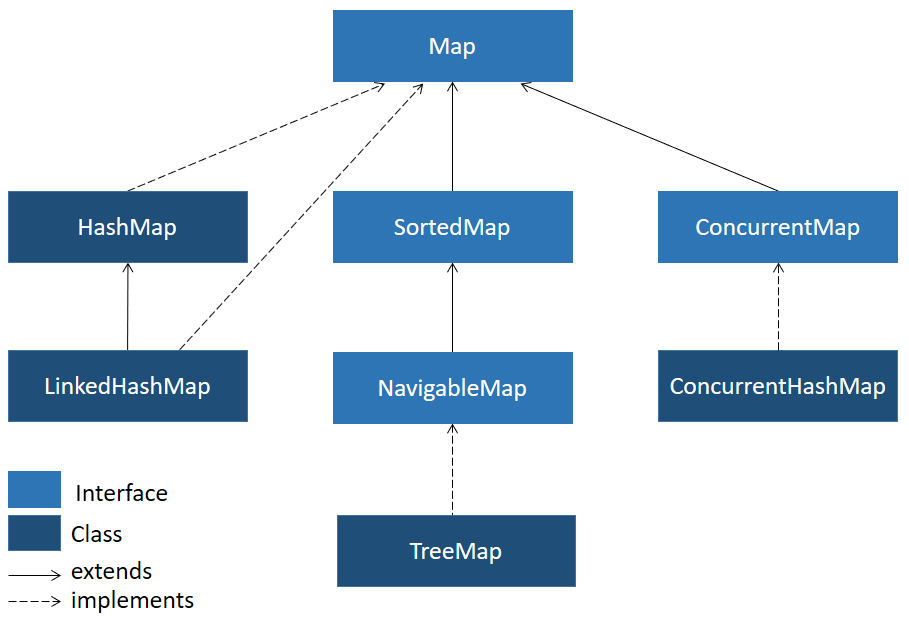
It is a collection of interfaces and classes which helps in storing and processing the data efficiently. Collections Framework standardizes the way we store and access the data from collections and is a part of the**java.util package.**

The diagram given below represents the hierarchy of the Java Collections framework.









**Introduction to Generics:**

Let us now look at the declaration of the Collection interface.

public interface Collection<E> ... {

}

The <E> syntax tells you that the interface is generic.

When you declare a Collection instance, you should specify the type of object contained in the collection. Specifying the type allows compiler to verify (at compile-time) the type of object you put into the collection, thus reducing the errors at runtime.

Generics makes the code more stable by making more bugs detectable at compile time.

Generics enable types (classes and interfaces) to be parameters while defining classes, interfaces and methods.

The variable E used in the above declaration is called as **type parameter**.

* + Like formal parameters used in method declarations, type parameters provide a way for you to re-use the same code with different inputs.
  + The difference is that the inputs to formal parameters are values, while the inputs to type parameters are types. The actual type that is substituted for the type parameter is called as a **type argument**.

E.g.

Collection<String> list = new ArrayList<String> ();

list.add("hello");

**Invoking and instantiating a Generic Type:**

**To reference** the generic Container class from within your code, you must perform a generic type invocation, which replaces T with some concrete value, such as Integer:

Container<Integer> integerContainer;

Generic type invocation can be considered as being similar to an ordinary method invocation but instead of passing an argument to a method, you are passing a type argument to the Container class itself — Integer in this case.

**To instantiate** this class, you can use the new keyword as usual, but you need to place <Integer> between the class name and the parenthesis:

Container<Integer> integerContainer = new Container<Integer>();

In Java SE 7 and later, you can replace the type arguments required to invoke the constructor of a generic class with an empty set of type arguments (<>).

For example, you can create an instance of Container<Integer> with the following statement:

Container<Integer> integerContainer = new Container<>();

**Type parameter Naming Conventions**

By convention, type parameter names are single, uppercase letters.

The most commonly used type parameter names are:

E - Element (used extensively by the Java Collections Framework)

K - Key

N - Number

T - Type

In addition to generic types, type parameters can also be **used to define generic methods**.

Generic methods are methods that introduce their own type parameters.

This is similar to declaring a generic type, but the type parameter's scope is limited to the method where it is declared. You can create static and non-static generic methods and generic constructors.

The syntax for a generic method includes a list of type parameters inside angle brackets which appears before the method's return type.

public static <E> void method() {

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Collection interface in Java:**

Collection is an interface present in java.util.package. It is used to represent a group of individual objects as a single unit.Any group of individual objects which are represented as a single unit is known as the collection of the objects. In Java, a separate framework named the *“Collection Framework”* has been defined in JDK 1.2 which holds all the collection classes and interface in it.

The Collection interface (**java.util.Collection**) and Map interface (**java.util.Map**) are the two main “root” interfaces of Java collection classes.

# **Collections Class in Java**:

**Collections** class is a member of the [Java Collections Framework](https://www.geeksforgeeks.org/collections-in-java-2/). The [java.util.Collections](https://www.geeksforgeeks.org/java-util-package-java/)package is the package that contains the Collections class. Collections class is basically used with the [static methods](https://www.geeksforgeeks.org/static-methods-vs-instance-methods-java/) that operate on the collections or return the collection. All the methods of this class throw the **NullPointerException** if the collection or object passed to the methods is null.

Collections is a utility class present in java.util.package. It defines several utility methods like sorting and searching which is used to operate on collection. It has all static methods. These methods provide much-needed convenience to developers, allowing them to effectively work with [Collection Framework](https://www.geeksforgeeks.org/collections-in-java-2/). For example, It has a method *sort()* to sort the collection elements according to default sorting order, and it has a method *min()*, and *max()* to find the minimum and maximum value respectively in the collection elements.

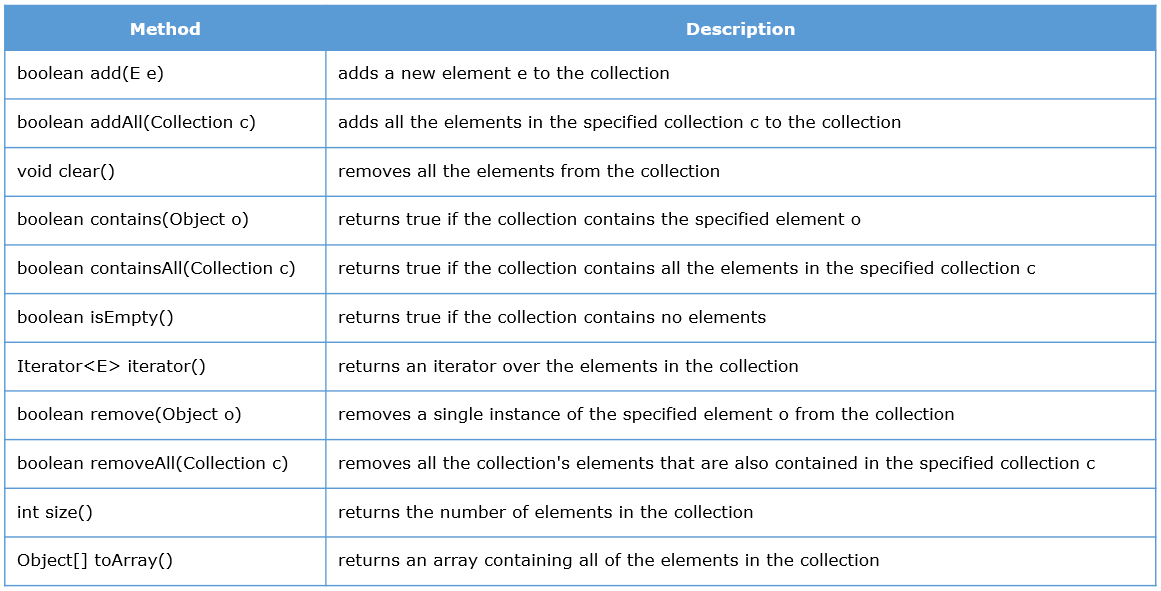
-------------------------------------------------------------------- OR ------------------------------------------------------------

**Collection Interface:**

Now that you have understood Generics, we will get into details of Collections framework.

The root of the Collection hierarchy is the **java.util.Collection<E>** interface.

It specifies the basic operations for manipulating elements in a collection. These operations are inherited and implemented in child interfaces and the classes implementing the interfaces.

Below are some of the important methods provided by the Collection interface:

* + toArray() method converts a collection into an array. If you want to convert an array into a list, you can use the Arrays.asList() method.
  + iterator() method allows traversing a collection. [LATER; noted down]

Iterator<String> iterator = orders.iterator();

System.out.println("Elements in the list: ");

while(iterator.hasNext()) {

    System.out.println(iterator.next());

}

Note: Iterators maintain a cursor to identify the location of elements.

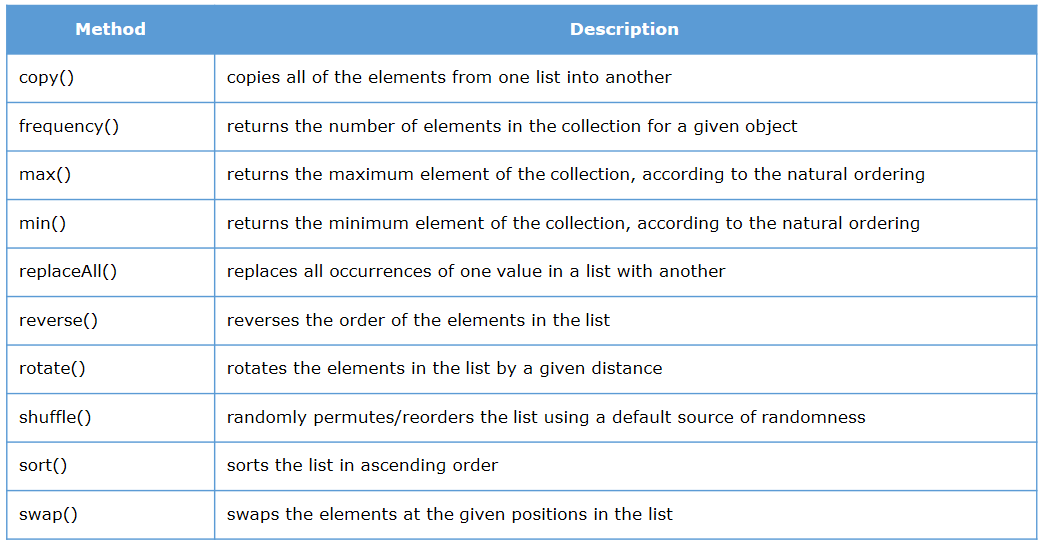
**Collections Class:**

Apart from the Collection interface, Java also has a Collections class.

The **java.util.Collections** class directly extends the Object class. This class consists of static methods that operate on or return collections.

The methods of this class throw a **NullPointerException** if the collections or class objects provided to them are null.

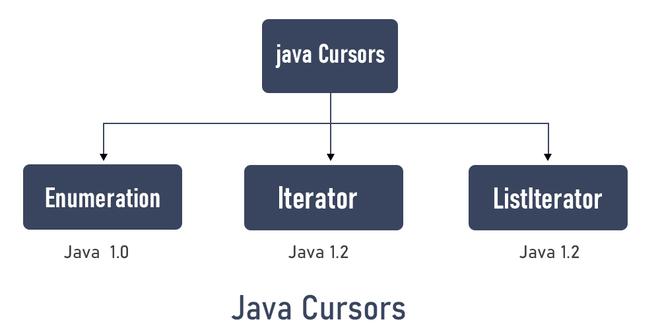
Below are some common methods of the Collections class that operate on collections:



# **Iterators in Java:**

A Java Cursor is an Iterator, which is used to iterate or traverse or retrieve a Collection or Stream object’s elements one by one. There are **three cursors in Java**.

1. Iterator
2. Enumeration
3. ListIterator



‘Iterator’ is an interface which belongs to collection framework.

Iterators in Java are used in the Collection framework to retrieve elements one by one. It is a **universal** iterator as we can apply it to any Collection object. By using Iterator, we can perform both read and remove operations.

Iterator () method is used to iterate through elements in a Collection. It is an implementation of the Iterator interface's iterate() method.

**Methods of Iterator Interface in Java:**

**hasNext()**, **next()** and **remove()** are important methods that are declared by the Iterator interface.

* + **hasNext():**Returns true if the iteration has more elements.
  + **next():**Returns the next element in the iteration.

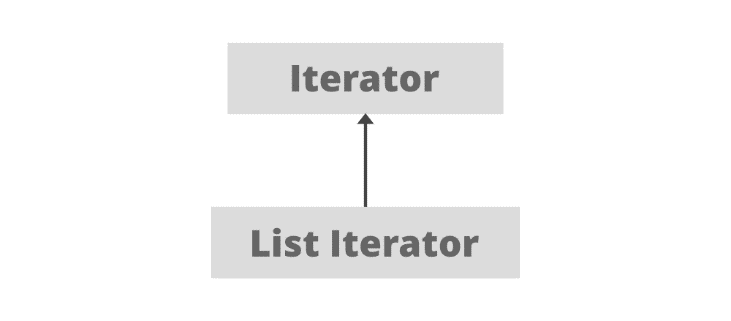
It throws **NoSuchElementException** if no more element is present.

* + **remove():**Removes the next element in the iteration. This method can be called only once per call to next().

# **ListIterator in Java:**

List interface exclusively provides the ListIterator which allows traversing a list in the reverse direction as well. ListIterator allows traversal in the reverse direction using the **hasPrevious()** and **previous()** methods.

It extends the[iterator](https://www.geeksforgeeks.org/iterator-interface-in-java/) interface.



* + **hasPrevious():** This method returns true when the list has more elements to traverse while traversing in the reverse direction
  + **previous():** This method returns the previous element of the list and shifts the cursor one position backwards.

**Overview**

# **Collections in Java:**

<https://www.geeksforgeeks.org/collections-in-java-2/?ref=lbp>

**Hierarchy**

**1. Iterable Interface:**

This is the root interface for the entire collection framework. The collection interface extends the iterable interface. Therefore, inherently, all the interfaces and classes implement this interface. The main functionality of this interface is to provide an iterator for the collections. Therefore, this interface contains only one abstract method which is the iterator. It returns the

*Iterator iterator();*

**2. Collection Interface:**

This interface extends the iterable interface and is implemented by all the classes in the collection framework.

This interface contains all the basic methods which every collection has like adding the data into the collection, removing the data, clearing the data, etc. All these methods are implemented in this interface because these methods are implemented by all the classes irrespective of their style of implementation.

And also, having these methods in this interface ensures that the names of the methods are universal for all the collections. Therefore, in short, we can say that this interface builds a foundation on which the collection classes are implemented.

**List Interface:**

This is a child interface **of the collection interface**. This interface is dedicated to the data of the list type in which we can store all the ordered collection of the objects.

This also allows duplicate data to be present in it.

This list interface is implemented by various classes like **ArrayList, Vector, Stack**, etc. Since all the subclasses implement the list, we can instantiate a list object with any of these classes.

*List <T> al = new ArrayList<> ();   
List <T> ll = new LinkedList<> ();   
List <T> v = new Vector<> ();*

*Where T is the type of the object*

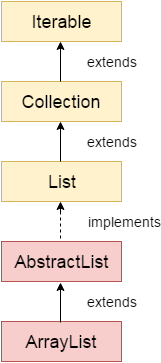
***The classes which implement the List interface are as follows:***

 **1. ArrayList:**

ArrayList provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. The size of an ArrayList is increased automatically if the collection grows or shrinks if the objects are removed from the collection. Java ArrayList allows us to randomly access the list.

ArrayList cannot be used for primitive types, like int, char, etc. We will need a wrapper class for such cases.

[1, 2, 3, 4, 5]

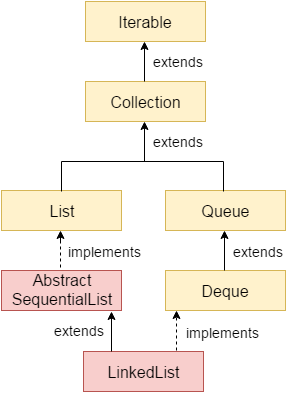


**2. LinkedList:**

LinkedList class is an implementation of the LinkedList data structure which is a linear data structure where the elements are not stored in contiguous locations and every element is a separate object with a data part and address part.

The elements are linked using pointers and addresses. Each element is known as a **node**.

[1, 2, 3, 4, 5]



**3. Vector:**

A vector provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed.

This is identical to ArrayList in terms of implementation. However, the primary difference between a vector and an ArrayList is that a Vector is synchronized and an ArrayList is non-synchronized.

[1, 2, 3, 4, 5]

**4.**[**Stack**](https://www.geeksforgeeks.org/stack-class-in-java/)**:**

Stack class models and implements the Stack data structure. The class is based on the basic principle of *last-in-first-out*. In addition to the basic push and pop operations, the class provides three more functions of empty, search and peek. The class can also be referred to as ***the subclass of Vector***.

[Geeks, For, Geeks, Geeks]

**Note: Stack** is ***a subclass of Vector*** and a legacy class. It is **thread-safe** which might be overhead in an environment where thread safety is not needed. An alternate to Stack is to use **ArrayDequeue** which *is not* ***thread-safe*** and ***has faster array implementation***.

[**Queue Interface**](https://www.geeksforgeeks.org/queue-interface-java/)**:**

A queue interface maintains the **FIFO***(First In First Out)* order similar to a real-world queue line. This interface is dedicated to storing all the elements *where the order of the elements matter*. For example, whenever we try to book a ticket, the tickets are sold on a first come first serve basis. Therefore, the person whose request arrives first into the queue gets the ticket. There are various classes like **PriorityQueue, ArrayDeque**, etc. Since all these subclasses implement the queue, we can instantiate a queue object with any of these classes.

*Queue <T> pq = new PriorityQueue<> ();   
Queue <T> ad = new ArrayDeque<> ();*

*Where T is the type of the object.*

**Deque Interface:**

This is a very slight variation of the queue data structure. **Deque**, also known as **a double-ended queue**, is a data structure *where we can add and remove the elements from both ends of the queue.*

This interface extends the **Queue interface**.

The class which implements this interface is **ArrayDeque**. Since ArrayDeque class implements the Deque interface, we can instantiate a deque object with this class.

*Deque<T> ad = new ArrayDeque<> ();*

*Where T is the type of the object.*

***The class which implements the deque interface is ArrayDeque.***

**1.** [**ArrayDeque:**](https://www.geeksforgeeks.org/arraydeque-in-java/)

ArrayDeque class which is implemented in the collection framework provides us *with a way to apply resizable-array*. This is a special kind of array that grows and allows users to add or remove an element from ***both sides of the queue***. Array deques have **no capacity restrictions** and they grow as necessary to support usage.

[10, 20, 30, 40, 50]

**Set Interface:**

A set is an *unordered collection of objects* in which *duplicate values cannot be stored*. This collection is used when we wish to avoid the duplication of the objects and wish to store only the unique objects. This set interface is implemented by various classes like **HashSet, TreeSet, LinkedHashSet**, etc. Since all the subclasses implement the set, we can instantiate a set object with any of these classes.

*Set<T> hs = new HashSet<> ();   
Set<T> lhs = new LinkedHashSet<> ();   
Set<T> ts = new TreeSet<> ();*

*Where T is the type of the object.*



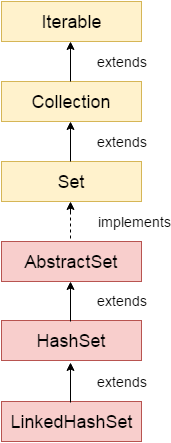
***The following are the classes that implement the Set interface:***

**1. HashSet:**

The HashSet class is an inherent implementation of the ***hash table data structure***. The objects that we insert into the HashSet *do not guarantee to be inserted in the same order*. The objects are *inserted based on their* ***hashcode***. This class also *allows the insertion of* ***NULL******elements****.*

**2.**[**LinkedHashSet**](https://www.geeksforgeeks.org/linkedhashset-in-java-with-examples/)**:**

A LinkedHashSet is very similar to a HashSet. The difference is that this **uses a doubly linked list to store the data** and *retains the ordering of the elements*.



[**Sorted Set Interface**](https://www.geeksforgeeks.org/sortedset-java-examples/)**:**

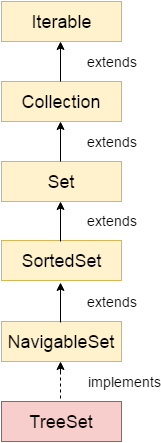
*This interface is very similar to the* ***set interface***. The only difference is that this interface *has extra methods that maintain the ordering of the elements*. The sorted set interface extends the set interface and is used to handle the data which needs to be sorted. The class which implements this interface is **TreeSet**. *Since TreeSet implements the SortedSet*, we can instantiate a SortedSet object with this class.

*SortedSet<T> ts = new TreeSet<> ();*

*Where T is the type of the object.*

***The class which implements the sorted set interface is TreeSet.***  
   
**1.** [**TreeSet:**](https://www.geeksforgeeks.org/treeset-in-java-with-examples/)

The TreeSet class uses a **Tree** for storage. The *ordering of the elements is maintained by a set using their natural ordering* whether or not an explicit comparator is provided. This must be consistent with equals if it is to correctly implement the Set interface. It can also be ordered by a Comparator provided at set creation time, depending on which constructor is used.



[**Map Interface**](https://www.geeksforgeeks.org/map-interface-java-examples/)**:**

A map is a data structure that supports the **key-value pair mapping** for the data. *This interface* *doesn’t support duplicate keys because the same key cannot have multiple mappings*. A map is useful if there is data and we wish to perform operations on the basis of the key. This map interface is implemented by various classes like **HashMap, TreeMap**, etc. Since all the subclasses implement the map, we can instantiate a map object with any of these classes. For example,

*Map<T> hm = new HashMap<> ();   
Map<T> tm = new TreeMap<> ();  
   
Where T is the type of the object.*

Java Map Hierarchy

***The frequently used implementation of a Map interface is a HashMap.***  
   
**1.** [**HashMap**](https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/)**:**

HashMap provides the basic implementation of the Map interface of Java*. It stores the data in (****Key, Value****) pairs*. To access a value in a HashMap, we must know its key.

HashMap uses a technique called **Hashing**. Hashing is *a technique of converting a large String to a small String that represents the same String so that the indexing and search operations are faster*.

**HashSet** also uses **HashMap** internally.



Flow Chart : -

Plan 2021-10-27 : Programming

Topic-wise links:

1. Arrays

-Bare Minimum

a. [https://www.geeksforgeeks.org/top-50-...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbEFtd3J0SmY1N2RkdzRZZjBoQlR2MEdJQTRQZ3xBQ3Jtc0ttQWtwSF9YeXBjaVQyUGZjZmFoUWVKU3FhZUo0ZVZLTkw3WERlMVlzdzgyRHN3UTRzUU9BTkxBdEJzX2FPTkM0N0g3dlgyYmJHNjl4RnFCRGE1Sm9KbVZLcktZUm4td2lpaXlfS3pCT24wendkeU1VRQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Ftop-50-array-coding-problems-for-interviews%2F)

-Bonus

a. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbWFfLXIxdWpldUlBc1doM0FRbXlDdloydUVyd3xBQ3Jtc0trWkxsaVRUYVlna0lpOXQxMmQ2Zld3MWJxQkpERXlSVUN4Wkx4OU1sMWlvRVQ2WWlNZnoyWGpLZWJtVVNBSVI3alVERnlPMG5TRUh3S3g2Mm90Q0tpSmtTcllMQ1JDbkQ4VVhqYmZoU2pnM0dzUjFWbw&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Farrays%2F)

b. [https://leetcode.com/tag/array/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbWtLaGEwcXlxdTlBZjNPNTRZRmJaZGVUS0ZBd3xBQ3Jtc0trcGVwcmhaaTd4SURwQVNGSjBZVUNDMmc3R19JM2N0U0tzdmJ6Z2dvUU1GQUs4Nk9VellScXg3ZU1neTVKTHlyTmoxdVhXWmZnSk0ybzM1Z190LVVIMzl6Nm8ycGRGQ1pwakFUaWRMMDlmX2JVdVNaTQ&q=https%3A%2F%2Fleetcode.com%2Ftag%2Farray%2F)

2. Strings

-Bare Minimum

a. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbmtPOVE0M1ZEVmpfeDBhNmtsakYwbUlJenMzZ3xBQ3Jtc0ttLXRBa29IYUd3aDg5REVTWV9QQ2lfMUI5N2VzekxDU2w5MHVyTUs0ZEtVUHFGQUlnMzVSZHRDRUc4YXd5RVFtWlNwOU1YX1NkY0hSbU9NdExhQlBBX1lwYjBndXR6N3gxTnE0SFZMWGlnYUMxYlpjYw&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Fstrings%2F)

-Bonus

a. [https://leetcode.com/tag/string/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa1ZsNU04Q0xoVWpFelFMWU1qWGV1SzkxLWRmZ3xBQ3Jtc0tsd1hDMzltLVBwNjBRM3hma2QzVnNtSHd3d1Z3dGU2bE5uNExqYWFNMmJGeHBSeWtHTEJQTWk4SS1GZE1MYnU4RVNQNmtvUmp4OWpOOGp0dTNjUUF5SEJ4ZkJwN3hCUTlHWEZuQ1JHMlNaekFFUVhQMA&q=https%3A%2F%2Fleetcode.com%2Ftag%2Fstring%2F)

b. [https://www.hackerrank.com/domains/al...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbHZwb2daOV9TLUtMTEU0VGpsb2J3VWhZdjhWZ3xBQ3Jtc0trUEpXTUt3SW9EdmpDUVluYlBQcHdjUWRaY1BNbzVRZlR2S000TTVmMGE1cmFKM0dtZkdlcjdoYm1kYl8wWWNkMV9YTTV4X2hqbm12YWxjMDhyQnBQUjRwaTI2Qlc0OVdJLW9ZRExhY3p5dmtncUtUbw&q=https%3A%2F%2Fwww.hackerrank.com%2Fdomains%2Falgorithms%3Ffilters%255Bstatus%255D%255B%255D%3Dunsolved%26filters%255Bsubdomains%255D%255B%255D%3Dstrings%26badge_type%3Dproblem-solving)

3. Linked Lists

-Bare Minimum

a. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbFQyRkZwb1BCeWI1dGpLbGNRSzY3Mlk0VnIxUXxBQ3Jtc0ttN1J6anRUTHEwTkFzazZfbEE4X3g0TmFlTmdBUFMtRWQyYmtoUWtIRXFpZ1NrQnBaQ2pWTmszaDNjSUEzWDc0MHlqWEE5N0RxT0NqMEotcjhrWUVKU2ZiSVhpaDFwTHU0UUZSZ0xGZnNGVFNaOEswVQ&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Flinked-lists%2F)

-Bonus

a. [https://leetcode.com/tag/linked-list/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbnE0NEhFZkVYVVEyMzVSME5qam5oWVdCYXYyUXxBQ3Jtc0tsck1ObkVSc2x1Rkktemhicm5tY3djWXZONVhOWUN4d3lSSUhFSlVEd1NRMDMwdFg5M2xVTktNUUtKUjlKX3pidTZhSVQ0UDZXWkRpUU1JdWtpWGtDaktXSmJVWjkyLXRxQ0tUV0JCNjl1TEpFcFVtcw&q=https%3A%2F%2Fleetcode.com%2Ftag%2Flinked-list%2F)

b. [https://www.geeksforgeeks.org/top-20-...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbmFQT2s5NUd0VU56YkV2RHFmMnFyV3paZUd4QXxBQ3Jtc0tuOHU4QzcwS0pMWEVGZnZ4V0hibjZtZmJkaHJKbzNZeFgtOC11ZG9ibnN2V3lUelhUYkdfMExHMDY3V2N2SHV2d1ZGc250Sm10aDhwSnQ2alFxWFVVVFRJM0hBVTIyWHJNUWdFOU5CTTk0aXZ1cFA0NA&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Ftop-20-linked-list-interview-question%2F)

4. Stacks and Queues

-Theory:

a. [https://www.geeksforgeeks.org/stack-d...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbFpWempOREVrczdKb1hYWkliNktZRkJSMWFXQXxBQ3Jtc0tuWXctTmlBUkZKNmQ0QlJ0ZVBMMUp1cFdEUlFpUXZIOUROMUtHS1o2VXZ0QTc4dTVhRWlISUluUUVPaVhLd0JlcHU2Y1QtSFhybXI0T3FZbWlYVHZUQWNSc3BqZWRqdnZtaDRTOU9DdDdQNmdfZVlJRQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fstack-data-structure-introduction-program%2F)

b. [https://www.geeksforgeeks.org/queue-s...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbk9uWGFNR09SaG9VLXA5eU1VWmM3Um9ValBwQXxBQ3Jtc0tueGtwQm9GOXl0ek1PWmxJUzBJSDh5eWRjZzA0RHpiaTh4b1NhR3BBVDRub0R4WF9paDl3RzNLem5pRkRXanNzbDhDbkhwMTBmRXc4YUpxNi1EYjRvZlNoZnVubzk0OEliV1E5WmxYcXk5c3RCUWJCdw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fqueue-set-1introduction-and-array-implementation%2F)

-Bare Minimum

a. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbVYzWS1pd0NvYjI5Z0J4N25Jdk93NjNpNUZ2UXxBQ3Jtc0trekZHOWpMUVEtSVVzODhkRFdJbjlZY1d0YXkzY0UzWEx3ZUxBclhsbV8zU25WcmZuZ3R5T3RTa2JMSHVXTEhwSlRqQzNCazUyNzdOX0ZsX1UzWTFCU0djNGdhVENyM0FlOTNjMUtoTnFUcEowY3Rudw&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Fstacks-and-queues%2F)

-Bonus

a. [https://leetcode.com/tag/stack/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbGJIbXhlRlJGM0IyTENEZEZNUF9yWnVRYUMyd3xBQ3Jtc0tsX184S2k0eVAtZVl2UTdGMWdvVFdjY29saGR1Z1I3dkRja3VqWW1Na2ZjYUM4N1RqQVo0bi1VOXFsX2JSYi1YQ011TUE1VVllODRwbVpSUlpYbm9VTFNZeEVHNUZaRG1mbEVNSm1PMUZKNVQ0Z2dIUQ&q=https%3A%2F%2Fleetcode.com%2Ftag%2Fstack%2F)

b. [https://leetcode.com/tag/queue/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbVN6N1hjUlYtdlRqbTlyS09oUWp3ekJUVHB6Z3xBQ3Jtc0ttOEZZNE9mbDJ3TERhX05TVG91bEFxWl9BMWpqakFwdnhRWWo0R1F5Nkc3NnBXS0ZwUTQ3NVBFRGlITk4yNEtrbDVNZUJkWFFHdHJTUVAxRnlhV1ItUDI2ZDNLd0hudXp2eTRBNzZMMXpKTXdZaTQxNA&q=https%3A%2F%2Fleetcode.com%2Ftag%2Fqueue%2F)

c. [https://www.geeksforgeeks.org/queue-d...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbTFGSThJMFVrZGhUbUVyMlpjcmFrQ0JzRnl2UXxBQ3Jtc0tsMUJzS2p2UnpGcGJNTEIxTmpOakplcU4wOXF3aTFGSE5WajF4UFM5TVppbEFKRTB4alpON3pTSTZabkwwbjJKU2ZnZEpkUmt2VWZRQmxNM09tanF2ZFB5T0tjSWlOR0t2Xzh5bFJPbHpVRWNiSlkxNA&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fqueue-data-structure%2F)

d. [https://www.geeksforgeeks.org/stack-d...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbTRNV2o1aTRXWUhOYVkwdG4xcTJ2cVJ6dDAwUXxBQ3Jtc0tscmlTclZaZ2FYWGZJek9oVjlQMmNvYk8zVDBEN3Jnd24xeXEyV2pTYkN2V3dWem9UWTJ3aDd0VnZnVHVVS0hGbVh4TmtIVWx1LWV2WXRENU5EX1VMVHNtbkVvbk1DdnFqVjZXbTJ3RkRmM2I2ek1PMA&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fstack-data-structure0)

5. Tree-based data structures:

-Theory:

a. [https://www.geeksforgeeks.org/binary-...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbURreDVjQkpaMXlqUmJwbW94ZVNhRWNRTE9ad3xBQ3Jtc0tsZDBFSFE3bWZjZzlqeFFqVXZDTUhHSlo5Z013MlY2N1RfX0Zrdl96SXNfRkQ2RWxyallzS1dRUUZYZExmaF9lQWljV2tmZDZQUzg1VGVTZUNqamJRU2JBX0ZBNVpwenA4S3Q3QkhqNjFZR0NLUEhrZw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fbinary-tree-data-structure%2F)

b. [https://www.geeksforgeeks.org/binary-...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbDFJODE3MTJZbVZfUnZUaHc1R05iWWJCU0Z1Z3xBQ3Jtc0tsdTRIYnRVRl83Z2JfNVprLURRcjlTUTFmbC03eC1ub1d0bDBOUnhSQjRfTjYtRF9YSTYta1FtYzVPTkYwU05Lci1BRVNyM2JsZV9WQXJyV3dVdVUtRTBEeGhIMVluUWpUUy02VlRqcmkxTi14WVhEOA&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fbinary-search-tree-data-structure%2F)

c. [https://www.geeksforgeeks.org/trie-in...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbW1lSWQtc3BZTGxiVk1LQmdsVFlOQmNmN19nZ3xBQ3Jtc0trWDk4bUw4ay1LY016TTBONmVrWjZub2NwOWRoZDVTc0NwY2xIRGhHU2VJUWJ5MFB1c1gzSnotT3N1cmVnYU4yU2JlQWo1STJ5bVAzbnVsMU1IUUM3YkM1azBFdXVaRHJZVnVGZGo0RlF1RDhmNm85VQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Ftrie-insert-and-search%2F)

d. [https://www.geeksforgeeks.org/heap-da...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUZjMm5KcUd0MGdGbE9aRXNVYnhhRFJBdjVaZ3xBQ3Jtc0tudmZGUms4NW9kUG9GUlIwSEY0eU0yeDZXYl95OTk1akk2eUpaU1FfQ0FMM210Q2xnY011allTalJSdzZsb1liazF4N1cwR3NaYXAtSEVsVTFMQVJhZm1SSFdMVVU5VjVUOTVfREwwczltTWlCZlRaOA&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fheap-data-structure%2F)

e. [https://www.geeksforgeeks.org/hashing...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbmpNU0Q2b1pPSmJPRFc0RWFiOUU1Wm0za21MUXxBQ3Jtc0tsTVZqN2V3Q0dINGdwODhjNmswUDZzb1BjT0t6M0JvUmVMcUVQYUN0TDB6YWN6MXIxT2wzVUhMUEtBX3BEdzZjazJyMzlfN2llNGpQemJPVWRfN0NTYmxvQklFUjdDRWFqbGp4aDA3Wm1ZRjF3dXlvMA&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fhashing-data-structure%2F)

-Bare minimum:

a. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbGZhNHJfa2dXVGtNWGtWUFFURG1scEhfbE1SUXxBQ3Jtc0ttWW00X1U3RjV3MnZDY0xuYjFaNjdNNEZUb2dQaldXcUhHZHVaeFBxY0dpSklEcEl2emtjbmVSN1NRaVBoZzFveGhCc2E4MmFBQXFpN0t3eFpvSllyamsyUEhhQzZ0TE94M3p3QWJlanNldUNESkZzcw&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Ftree-data-structure%2F)

b. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa3o4UXdaWXJtWFdCbHNwUVRQRWNkOHdrUUplZ3xBQ3Jtc0tsbHpwZEp5R2VSYTlmcnN5c0lOLXlNRmg3U1ZLTGR3LWloVjlkbjF1OTNFMzJJdWtHanVWS2I4eWVJVlI0YWV3TjB6U1FXNnZJUDRlS1NnQ3N2ZVVXZnpLV1NtYnhQQlBZT0NkTFc5S1Y5d0NHOWlHbw&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Fheaps-and-maps%2F)

c. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbm10S3ZGZkFjNU92clhBWkh3RG9WVjVvLWltZ3xBQ3Jtc0trRXhfZjN5QVJycGJQTkhvWk9IRmdnTzBrb1k3NFpFOWJiSEE5dU5uMGplSVJrUkRZWWZmZDZ4dTNFbFZmWjZoOHRsVUMyZDY5cERrMWhxV1BjbE1maHpNZmpBZXRfbFVMemZJaENGdmtSdUNCWF9ydw&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Fhashing%2F)

-Bonus

a. [https://leetcode.com/tag/tree/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbHFiOHVPeHlReXc0TW1uRERMRTNxWkdiTlpiZ3xBQ3Jtc0tsajNOUzlWU2VkZWIzZW82RVE0ekZSdERUZEVhVzhoMXhIUUZ3UEFYM1NTcDRISUgyQV9EWW5jMk5ZVGlucUJOYzNybEpOcnRka1MzemtBODhLVVRGSVpaQXFCX1BFQ0FQLVVqcEgtRDhpdjNIcFM0QQ&q=https%3A%2F%2Fleetcode.com%2Ftag%2Ftree%2F)

b. [https://leetcode.com/tag/heap/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa3BBMjZkSXM2cTNOeXZQUC1XMExNbThua1ZsUXxBQ3Jtc0tucHpkSmRYb1F3QW1LcGI3UXNwVE1Ha04zb3VrZXFFVWN6N1M5cV8xM1dHaHZRMEFLZkphTW9GTUxZdHI4OGdKTG1BVkl2MkNZZWpOSWlrSVVfdGozR0N0bFdsbmVRZ0RZaFgxM2hBSjB5akJONmFoQQ&q=https%3A%2F%2Fleetcode.com%2Ftag%2Fheap%2F)

c. [https://leetcode.com/tag/trie/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbHdqWGI0WS1jMDZvcFlWclJFOVpGdDlVUDBpUXxBQ3Jtc0ttdUx5aC1xNUdFX1h1OFdfblBsbFFtaWFfZWV3YTVfY1Z6WEhqSDhjOFBSbXVmVHViRUNYVWY5bldzVWItZjJpR3VNbVBiZ2kteDhSNDN6cDlzdDZhVnpFeTV4YkpkZ2VWZ3NfZXRjb1UxRDhPMUZLcw&q=https%3A%2F%2Fleetcode.com%2Ftag%2Ftrie%2F)

d. [https://leetcode.com/tag/hash-table/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbklmd01hYVVjYm9GUW0yMjVROGl1aHc0MW4wQXxBQ3Jtc0tsbFBUZ0VJQXdkQmFvZUJvNnM4bVZPTFhpbmRNTmlrOXl2UWZVREJiVThjakZHZnAtM0E4TElIbzhMeXlHc1JGa0FQVXhMRzZ3X09URjZ5Q2V6bVYyQ0pPTjhLWHNiMG53bWwweEdEU2laS2ZmcXU1TQ&q=https%3A%2F%2Fleetcode.com%2Ftag%2Fhash-table%2F)

6. Graphs: -Theory: a. [https://www.geeksforgeeks.org/graph-a...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbFdzaU8zekxGWTJrdGFrS1U3b0cydHJDUjFZd3xBQ3Jtc0tuTjYydDFpNWJ5YzNJSXZNM3NjWVZXY0NTb1FPS0FtMDdGYnBpbTdvRU5BNG9KbXhuUXdwaEhGbElHdnFGUUU1SVYwYUJiU2RITk5ld29JQUhwMDdjLXBpWnBBWVBJR2NtTmdBemp6RHdVaHJ1Ty1Mdw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fgraph-and-its-representations%2F) -Standard Algos: a. BFS - [https://www.geeksforgeeks.org/breadth...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbW45TlkxdzI4RlFnQTczNDJSbEhTMmZrVWN6d3xBQ3Jtc0tuZkMtZGJuMVYxcE5mMEpNcmNTem5RdXcyUkpTanBFeDB5Y1ZoOFNkQUVZY2xlNHowR2txWGM5TDRqMk1URnV4ZlVXS0tmTTQwdzRwTmoxandJNmdmcjhCOXljODFTSXg1clpVX0V0MHRzTnhOaWdabw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fbreadth-first-search-or-bfs-for-a-graph%2F) b. DFS - [https://www.geeksforgeeks.org/depth-f...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbWt3VUdsTEkyMmREN09QZG9tQW0yT2g4SE1Xd3xBQ3Jtc0tscGJyTlJ3T1Y1aWJ6S0l1a1BtWENOTUtwOTZYOWNCdVNGbUVCSmpPZ3JhRHZLZ01vdTV6YTRNckxFQms5b0RqX2R6anpwamtvRjU4OG5DejNTZXp3Yk9jdVlKTDdKNThpcUJFVnRBa2hoX09hRmNwRQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fdepth-first-search-or-dfs-for-a-graph%2F) c. Dijkstra - [https://www.geeksforgeeks.org/dijkstr...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbGl3QzdtMTBfQjBfZ3RHREp3UlVwemVHZS1pQXxBQ3Jtc0tseVlMLTRQdTd0bVNOZEVHeXAzNXJiSnRBOUJ3cjJBZUxWRkcwYmZfVGVPZ2JFYTBWQ1VBVWdRWDFBUXBTakFMWlE3YjlqMEVkMnhTbFVlVkF1bFl6cU5IZjhzOFJEcWYwY1dHTEJfRzdERjIyaUtxdw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fdijkstras-shortest-path-algorithm-greedy-algo-7%2F) d. Prim's - [https://www.geeksforgeeks.org/prims-m...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbXd2SXN5ZUJkZTAzallYSGhvTU1USWxRbFhad3xBQ3Jtc0ttb0FxbWV2Q1JYanJiNDloMXZpdGRJNTZKRzk4TEVRRGQyMURqUmo4dFhaZTZQeUtrRTAwcHhUNjJtVWRCaUs3VDdXb0dTelhVc2Y1LTVKRDR2QThqeWpTTTlBQlZHazl2QXo1d01GOEZxV3hXYzhMTQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fprims-minimum-spanning-tree-mst-greedy-algo-5%2F) e. Kruskal - [https://www.geeksforgeeks.org/kruskal...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa3ZXdkNBTm5oMkppV1lRbnNSUktLeXVEcG1Wd3xBQ3Jtc0tuajRtM3huWmhRNDMxMjRoTm8yeVJzaWlSelpwZjFNR3NEb1k2SnE3dUpDVEV6WmNKTlY5cDRXcl81SllDS2JzcHhRRHZ6cUM1V0NXbjZCTzVtU2xCNkFJVTJQZzBTSDhBcTVCSlk1UFdHS1NSd2laTQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fkruskals-minimum-spanning-tree-algorithm-greedy-algo-2%2F) f. Floyd-Warshall - [https://www.geeksforgeeks.org/floyd-w...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa09XR0Q4NHB3U0w0QUZ1SDExNFU3TklSaUV5UXxBQ3Jtc0trdFpCbm51b3NoU2NyVEFUUnlpV3ZGLXJRY3puajdNZWRDcnlfTW9Fc2Q3Q2h1UUxySnBKbFJLRkJVOWxpZ3pINVF1ank4YTZfT1BNZ3d4NG1LSk03SWZDNWE0UlBvRnA0dGNXd19td3lXY0xnbFBiZw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Ffloyd-warshall-algorithm-dp-16%2F) g. Union Find - [https://www.geeksforgeeks.org/union-f...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbS1JazVWWWFfOENUeUlKazBndW5iT1lNQkNiQXxBQ3Jtc0trSVVELVVaWXBIYmQ4Q25RTHVkOVo3Y0VqdTNPNmRkNVI4UWRSOXIwNWdBeG5LS0NQbTRDbnlfdFNuV3RfV3NNcmNIdkF0N3ZyNjJPZlhSUno2SUotTnBRcDNKZzEwQ2M0elgtNWZwRHlmT0VrRU45VQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Funion-find-algorithm-union-rank-find-optimized-path-compression%2F) -Bare Minimum: a. [https://leetcode.com/tag/graph/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbU44TGFHdHRoWHg1WjBacElvcnRheE1sNFhnZ3xBQ3Jtc0tuNWRlRHFKcVhmUGY0bmo1eHN5Z2NRSTRFOXA5SU1QRFVQbllQNVBmZlR5NHlWSmMxWHkxaU5TSE1laFlqZjlDRURXWUdrMmU2cGY4WEEtbEdMMVpMOE8tUDBDX3BLbDF1eDJwbU1RREJaT3lweG92QQ&q=https%3A%2F%2Fleetcode.com%2Ftag%2Fgraph%2F) (Easy and Medium) -Bonus: a. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbFgzeDJRMzBfVjFvTHoxZzY0MWhKVmxBaHNVZ3xBQ3Jtc0tuT2xreVhpSzhvZU9MV2c0ZHdBdk1HcktXbTVmR1N5eWRpdERQUHhTRGc3T0dYOGxjZmxXVWxPUHdaVFFkbUxPOGVPV1pybUt1N3hIdmNxbTMxcm5HbGZ4Y1RGNEhucmh6QzgwWGk3bExZbUdVZWNwTQ&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Fgraph-data-structure-algorithms%2F)

7. Dynamic Programming: -Video lectures: a. Lec 1 - [https://www.youtube.com/watch?v=OQ5js...](https://www.youtube.com/watch?v=OQ5jsbhAv_M&list=PLcDimPvbmfT8qAxD6JH_kmXiQwTNcoK78&t=0s) b. Lec 2 - [https://www.youtube.com/watch?v=ENyox...](https://www.youtube.com/watch?v=ENyox7kNKeY&list=PLcDimPvbmfT8qAxD6JH_kmXiQwTNcoK78&t=0s) c. Lec 3 - [https://www.youtube.com/watch?v=ocZMD...](https://www.youtube.com/watch?v=ocZMDMZwhCY&list=PLcDimPvbmfT8qAxD6JH_kmXiQwTNcoK78&t=0s) -Bare minimum (Standard problems): a. [https://www.geeksforgeeks.org/program...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbW05S3pCYXJnRVJfbHpUWWNHUVZMNmhVNWNjZ3xBQ3Jtc0tuaVZvVlpPMVhaVERwRXFMNWpjTGdaRWx0VXRReWpBa05Fb3dtMGZnTGJaSk9RM3Zsc0M1REwxSVZlc3lkTU5QNThVbDVXYlZ4Um1UeXB1cGtxd2tkMmx0V0N4VFZIT2V0VkgxdDYzZmNPQlNzV3FvYw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fprogram-for-nth-fibonacci-number%2F) b. [https://www.geeksforgeeks.org/0-1-kna...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa21ZLUFyMEFzVVN5cjdGVEJnbVF5T0xTUFFRd3xBQ3Jtc0tra3lRbEpLOUxSSVRzS09ENmFwdi1ac0RuN1JKMVROZE01Tk9pSXVzY2xTUjFid3BOZUtwYUJnd1ROaFV0ck9YWVJRdkhKdC1sUFNiTkdrSFhnTkVhTHgwenlaT2UyVXhSZ1V0cXFsaXhsUlUzbVk4aw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2F0-1-knapsack-problem-dp-10%2F) c. [https://www.geeksforgeeks.org/coin-ch...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbGlCVkhHQ2NSVFVkU1Rzbm1KOEVmUF93SDJqZ3xBQ3Jtc0trcTd5OUlKeGxySTZhMFdQYmh3Y1FGb09heGMtbVFJcU5TTnZzUnlyci1xb3ZBUkpRR1dEVmN6UHJTbHFxa0FXUHlKV3NvQnRESFRkOEFYenpPSFBGWWtLWFdETXJidHctTXNUV1p0bDJ3ajZTUVVoVQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fcoin-change-dp-7%2F) d. [https://www.geeksforgeeks.org/compute...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbEQya19odV9XLWpSLXlyNXFYWldiZDREMGRhQXxBQ3Jtc0tuc3FqM2pTZUxNQ3lIVmhYYlBkaHJkQ2VjUkZlZjBnbHktTndtNFpKYlBINkVtamJwOUFHSTFRaUs4MHRZT0l1U1BKdHJNQzVFLUlyREdHV1BhQlpyZkRmdnR5QzBTN0l4OWRWMzc1N1ZEZ1IyOFk0VQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Fcompute-ncr-p-set-1-introduction-and-dynamic-programming-solution%2F) e. [https://www.geeksforgeeks.org/longest...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUZReTdoS1d2Zmc2dkItbXRYUjNHcmo3UTZfUXxBQ3Jtc0ttOGlUT19FVl9wWlFwU0hSNVh1R0NJdkxoUkdWYmhwNGhqYktYbF9ySzRRYkFvSWVreUNoWkw1YVZHdmNHWi1rWkZYYTRUakNQdVQ5UENGNmdpdEdUakY0TXI2NFpOX2JJYVJkREk1eHdSaHBuZ0VGWQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Flongest-increasing-subsequence-dp-3%2F) f. [https://www.geeksforgeeks.org/longest...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUJwZmpKUHNIWGNTYmxWSnMwZ2dlVGtqc3BkUXxBQ3Jtc0tsWEhqdlBHYlFGNlVFeVZ2ZkVPeVFTWkFvSHlSQXFPVkN5cFZxXzV1TDNfd1ZqaW45bXdPU0dwSE0tQ1Z3cGdDNGpMWE95SlhPeUliYzZReXg4eUlISWFIYW9wUEY5dWl6Z1o0U2lUUkxZaEM0YXBNRQ&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Flongest-common-subsequence-dp-4%2F) g. [https://www.geeksforgeeks.org/longest...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbm9OMzZ1ZnV4VGRXXzRNUkdjUFVhUE0wNmVGQXxBQ3Jtc0tteW9TUDZVWjZSQ0Rva1l1UjB0YXhaaU1NWENsanRsUTdQT2x0cjg4YTVFMWVLWElJend3NFppRUxXUFlybzZaZ3hMLWVPenpiZDh2eVNMOXZOdDZJMGtqWjBvbXpZVjc0THdjYmktY0pUdE5GSVZHYw&q=https%3A%2F%2Fwww.geeksforgeeks.org%2Flongest-common-substring-dp-29%2F) -Bonus: a. [https://www.interviewbit.com/courses/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa0Itb2JHT0dIay0yR3lXbTIxcVhiajBUSlFKZ3xBQ3Jtc0tseHkzeW9NWEZZbmhqNlRDcnUyQ3ljUUVwT3BQUVludlpqWlpHbmU1LWotOEthcGxWWWRLWXhPcVdpVnRqVjkxNVR0NUl5MGx0QlN1OWphQUNLcDk1NzJYYnV3WVFfdkxYRXN2WkRjNHdxYmc4anBiYw&q=https%3A%2F%2Fwww.interviewbit.com%2Fcourses%2Fprogramming%2Ftopics%2Fdynamic-programming%2F) b. [https://leetcode.com/tag/dynamic-prog...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa05yclNVUTlpQVJ1bUoxSE95NWtuN3NodDdYd3xBQ3Jtc0tua0d1SkNzMThDbUp4aGx5dVdCdGZzMzBXYTR3dG9vc0JmSlgwV2VIWUM0V3FZZjZLUnlrUmZBb2Uyakc1Q3FHZ1dHeDlqTXdLbFJoRi1GV3BJMWxfQ0o5M1FpMzNqLUdJZzlJUjRzMFdSOVpFWFV2dw&q=https%3A%2F%2Fleetcode.com%2Ftag%2Fdynamic-programming%2F) Use my code THECODESKOOL to get a 10 percent discount on any GeeksForGeeks course. Happy learning!