Data Structures

It is the way by which we can store the data in an efficient way.

(or)

A data structure is a specialized format for organizing, processing, retrieving and storing data.

(or)

Data Structure can be defined as the group of data elements which provides an efficient way of storing and organizing data in the computer so that it can be used efficiently.

Data Structures are the main part of many computer science algorithms as they enable the programmers to handle the data in an efficient way.

It plays a vital role in enhancing the performance of a software or a program as the main function of the software is to store and retrieve the user's data as fast as possible.

Basic Terminology

Data structures are the building blocks of any program or the software. Choosing the appropriate data structure for a program is the most difficult task for a programmer.

Following terminology is used as far as data structures are concerned.

Data: Data can be defined as an elementary value or the collection of values, for example, student's name and its id are the data about the student.

Group Items: Data items which have subordinate data items are called Group item, for example, name of a student can have first name and the last name.

Record: Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.

File: A File is a collection of various records of one type of entity, for example, if there are 60 employees in the class, then there will be 20 records in the related file where each record contains the data about each employee.

Attribute and Entity: An entity represents the class of certain objects. it contains various attributes. Each attribute represents the particular property of that entity.

Field: Field is a single elementary unit of information representing the attribute of an entity.

Need of Data Structures

As applications are getting complexed and amount of data is increasing day by day, there may arrise the following problems:

Processor speed: To handle very large amount of data, high speed processing is required, but as the data is growing day by day to the billions of files per entity, processor may fail to deal with that much amount of data.

Data Search: Consider an inventory size of 106 items in a store, If our application needs to search for a particular item, it needs to traverse 106 items every time, results in slowing down the search process.

Multiple requests: If thousands of users are searching for the data simultaneously on a web server, then there are the chances that a very large server can be failed during that process

to solve the above problems, data structures are used. Data is organized to form a data structure in such a way that all items are not required to be searched and required data can be searched instantly.

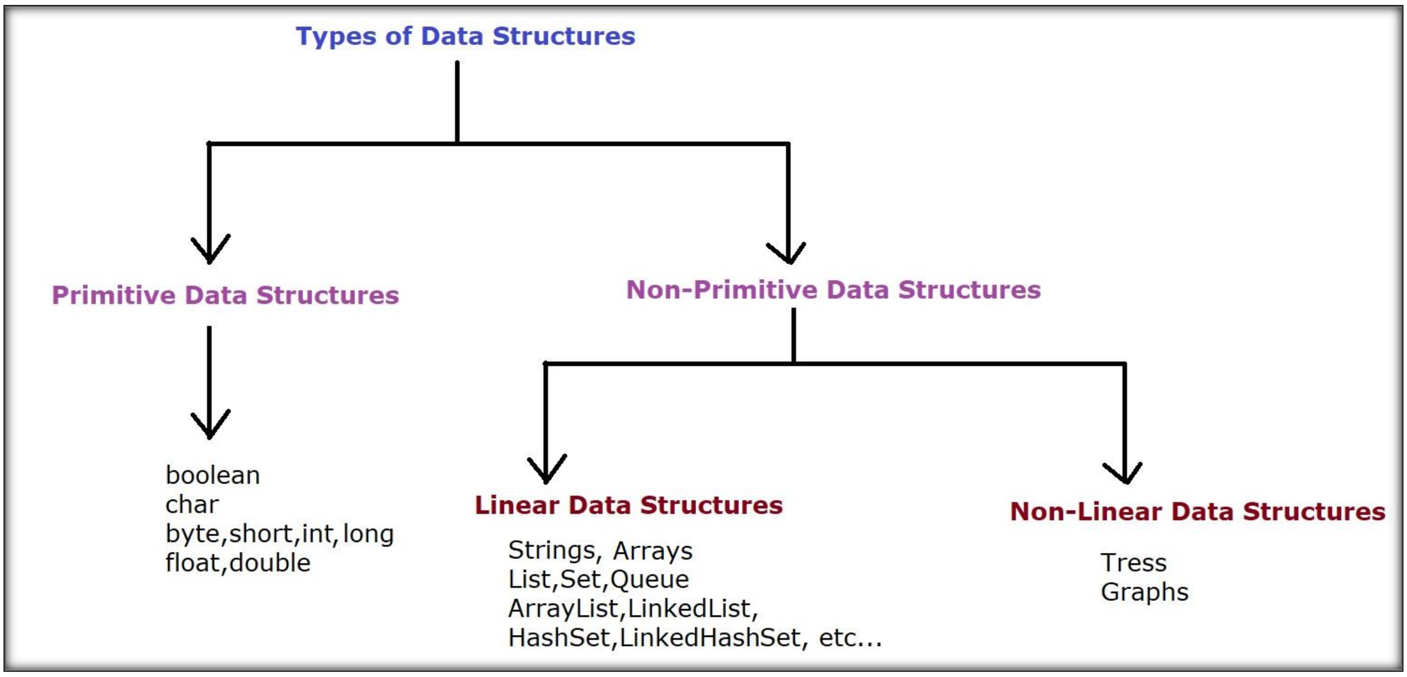
Advantages of Data Structures

Efficiency: Efficiency of a program depends upon the choice of data structures. For example: suppose, we have some data and we need to perform the search for a particular record. In that case, if we organize our data in an array, we will have to search sequentially element by element. hence, using array may not be very efficient here. There are better data structures which can make the search process efficient like ordered array, binary search tree or hash tables.

Reusability: Data structures are reusable, i.e. once we have implemented a particular data structure, we can use it at any other place. Implementation of data structures can be compiled into libraries which can be used by different clients.

Abstraction: Data structure is specified by the ADT which provides a level of abstraction. The client program uses the data structure through interface only, without getting into the implementation details.

Types of Data Structures



Example Applications by using Primitive Data Structures:

Simple Calculator, Snake Game, Currency Converter, Temperature Converter, etc.

Example Applications by using Non-Primitive Data Structures:

E-commerce, Social Networking, Banking, etc.

Collection Framework in Java

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

What is Collection in Java

A Collection represents a single unit of objects, i.e., a group.

What is a framework in Java?

- ✓ It provides readymade architecture.
- ✓ It represents a set of classes and interfaces.
- ✓ It is optional.

What is Collection framework

The Collection framework represents a unified architecture for storing and manipulating a group of objects. It has:

- ✓ Interfaces and its implementations, i.e., classes
- ✓ Algorithm

Hierarchy of Collection Framework

The java.util package contains all the classes and interfaces for the Collection framework.

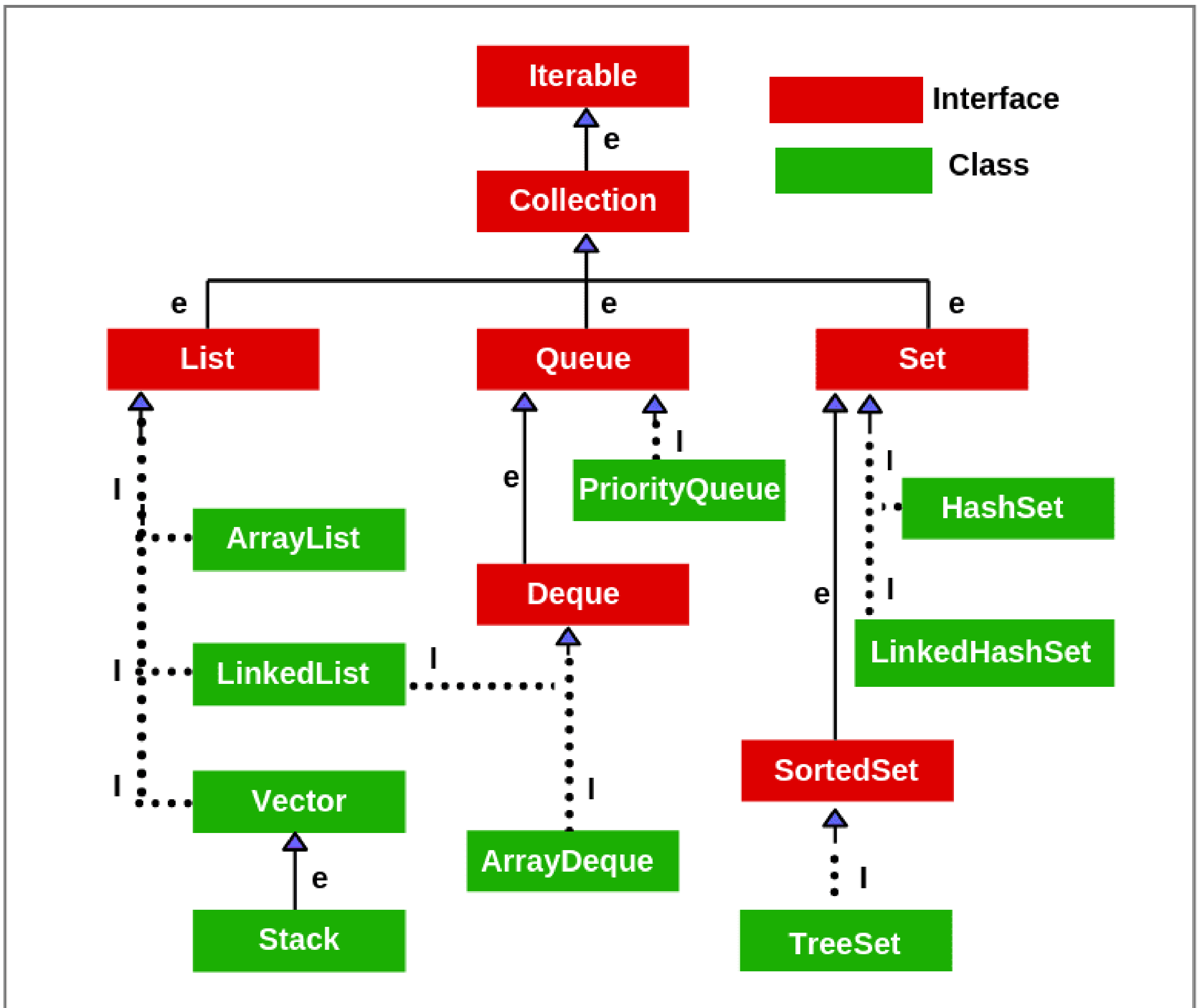
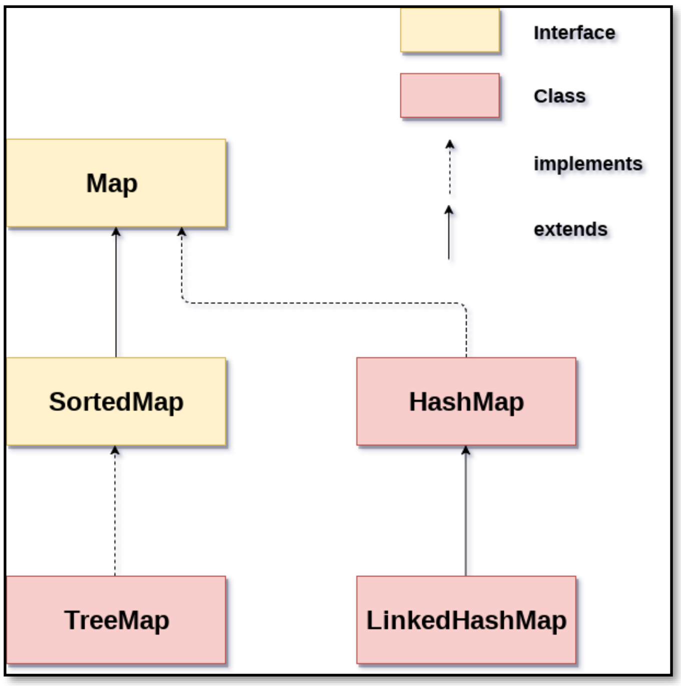


Fig: Collection Hierarchy in Java



Difference between Arrays and Collection in Java

Arrays	Collection
Arrays are fixed in size that is once we create an array we can not increased or decreased based on our requirement.	Collection are growable in nature that is based on our requirement. We can increase or decrease of size.
With respect to memory Arrays are not recommended to use.	With respect to memory collection are recommended to use.
With respect to performance Arrays are recommended to use.	With respect to performance collection are not recommended to use.
Arrays can hold only homogeneous data types elements.	Collection can hold both homogeneous and and heterogeneous elements.
There is no underlying data structure for arrays and hence readymade method support is not available.	Every collection class is implemented based on some standard data structure and hence for every requirement readymade method support is available being a performance. we can use this method directly and We are not responsible to implement these methods.
Arrays can hold both object and primitive.	Collection can hold only object types but primitive.

Parameter	Array	Collection
1)Size	Array is fixed in size	Collection is dynamic in size
2)Data	Array can store only homogeneous data only	Collection can store homogeneous as well as heterogeneous data
3)Data Storage	Array can store primitive as well as objects	Collections stores only object
4)Data Structure	Array does not have any standard data structure so no ready made methods available	Collection supports standard data structure so ready made methods are available
5)Memory	Memory wise array is worst choice	Memory wise collection is best choice
6)Performance	Performance wise array is good choice	Performance wise collection is worst choice

What is the Difference Between for loop and foreach Loop?

for Loop vs foreach Loop	
The for loop is a control structure for specifying iteration that allows code to be repeatedly executed.	The foreach loop is a control structure for traversing items in an array or a collection.
Element Retrieving	
A for loop can be used to retrieve a particular set of elements.	The foreach loop cannot be used to retrieve a particular set of elements.
Readability	
The for loop is harder to read and write than the foreach loop.	The foreach loop is easier to read and write than the for loop.
Usage	
The for loop is used as a general purpose loop.	The foreach loop is used for arrays and collections.

Normal for-loop	**Enhanced for-loop**
This for-loop is present from JDK1	This for loop is present from JDK5
In a normal for-loop, we can increase the counter as per our wish by using i=i+x(where x is any constant x=1,2,3…)	But enhanced for loop will execute in a sequential manner i.e counter will always increase by one.
Using this for loop we can iterate on any container object.	We can only iterate on that container by using this loop to implement the iterable interface.
In this for-loop, we can iterate in both decrement or increment order.	But in this for-loop, we can iterate only in increment order.
In this for-loop, we can replace elements at any specific index.	But in this for-loop, we don’t have access to the index, so we cannot replace elements at any specific index.
By using normal for-loop we can print array elements either in the original order or in reverse order.	But in the for-each loop**,** we can print array element only in the original order, not in reverse order

Example: Printing element in a 1D array
int[ ] x={1,2,3};
for(int i=0;i<x.length;i++)
{
System.out.println(x[i]);
}
	Example: Printing element in a 1D array using for-each loop
	int[ ] x={1,2,3};
	for(int a : x)
	{
	System.out.println(a);
	}

The following table summarizes the principal classes in Java collections framework for quick reference:

Main collection classes	D	O	S	TS
ArrayList	**Yes**	**Yes**	**No**	**No**
LinkedList	**Yes**	**Yes**	**No**	**No**
Vector	**Yes**	**Yes**	**No**	**Yes**
HashSet	**No**	**No**	**No**	**No**
LinkedHashSet	**No**	**Yes**	**No**	**No**
TreeSet	**No**	**Yes**	**Yes**	**No**
HashMap	**No**	**No**	**No**	**No**
LinkedHashMap	**No**	**Yes**	**No**	**No**
Hashtable	**No**	**No**	**No**	**Yes**
TreeMap	**No**	**Yes**	**Yes**	**No**

- **D:** Duplicate elements is allowed?
- **O:** Elements are ordered?
- **S:** Elements are sorted?
- **TS:** The collection is thread-safe?

From this table we can conclude the following characteristics of the main collections in Java Collection Frameworks:

- ➢ All lists allow duplicate elements which are ordered by index.
- ➢ All sets and maps do not allow duplicate elements.
- ➢ All list elements are not sorted.
- ➢ Generally, sets and maps do not sort its elements, except TreeSet and TreeMap – which sort elements by natural order or by a comparator.
- ➢ Generally, elements within sets and maps are not ordered, except for:
  - ✓ LinkedHashSet and LinkedHashMap have elements ordered by insertion order.

- ✓ TreeSet and TreeMap have elements ordered by natural order or by a comparator.
- ➢ There are only two collections are thread-safe: Vectorand Hastable. The rest is not thread-safe.

Difference between ArrayList and LinkedList in java

ArrayList

- ArrayList is implementation of list interface.
- ArrayList is not synchronized (so not thread safe)
- ArrayList is implemented using array as internal data structure. It can be dynamically resized .

LinkedList

- LinkedList is implementation of list and deque interface.
- LinkedList is not synchronized
- LinkedList is implemented using doubly linked list as internal data structure.

ArrayList vs LinkedList:

Parameter	**ArrayList**	**LinkedList**
Internal data structure	It uses dynamic array to store elements internally	It uses doubly Linked List to store elements internally
Manipulation	If We need to insert or delete element in ArrayList, it may take O(n), as it internally uses array and we may have to shift elements in case of insertion or deletion	If We need to insert or delete element in LinkedList, it will take O(1), as it internally uses doubly LinkedList
Search	Search is faster in ArrayList as uses array internally which is index based. So here time complexity is O(1)	Search is slower in LinkedList as uses doubly Linked List internally So here time complexity is O(n)

Interfaces	ArrayList implements List interface only, So it can be used as List only	LinkedList implements List,Deque interfaces, so it can be used as List,Stack or Queue

It actually depends on our need.

- If we have more insertion or deletion then we should use LinkedList.
- If we have less insertion or deletion and more search operations then we should use ArrayList.

When to use LinkedList and when to use ArrayList?

1) As explained above the insert and remove operations give good performance (O(1)) in LinkedList compared to ArrayList(O(n)). Hence if there is a requirement of frequent addition and deletion in application then LinkedList is a best choice.

2) Search (get method) operations are fast in Arraylist (O(1)) but not in LinkedList (O(n)) so If there are less add and remove operations and more search operations requirement, ArrayList would be your best bet.

Link: https://www.baeldung.com/java-arraylist-linkedlist#2-access-by-index-1