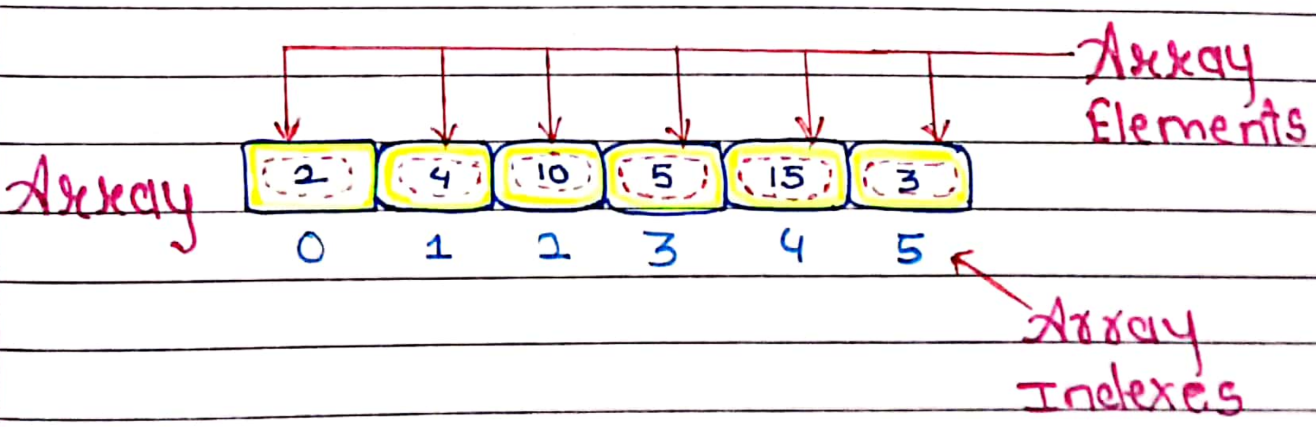
* Data Structure *

1] Arrays

@coders_notes

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array. (generally denoted by the name of the array).



Searching in Array

• Sequential Search :-

@coders_notes

In this, the list or array is traversed sequentially and every element is checked.

For example :-

Linear Search. Time Complexity is O(n).

• Interval Search :-

These algorithms are specifically designed for searching in sorted data-structures. These type of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half.

For example :-

Binary Search. Time Complexity is O(logn)

@coders_notes

2] ArrayList

An ArrayList, or dynamically resizing array, allows you to have the benefits of an array while offering flexibility in size. You won't run out of space in the arraylist since it's capacity will grow as you insert elements. An Arraylist is implemented with an array. When the array hits capacity, the Arraylist class will create a new array with double the capacity and copy all the elements over to the new array.

Important Features of Arraylist :

@coders_notes

1] Arraylist inherits AbstractList class and implements the List interface.

2] ArrayList is initialized by size. However, the size is increased automatically if the collection grows or shrinks if the objects are removed from the collection.

3] Java ArrayList allows us to randomly access the list.

4] Arraylist can not be used for primitive types, like int, char, etc. We need a wrapper class for such cases.

5] ArrayList in Java can be seen as a vector in C++.

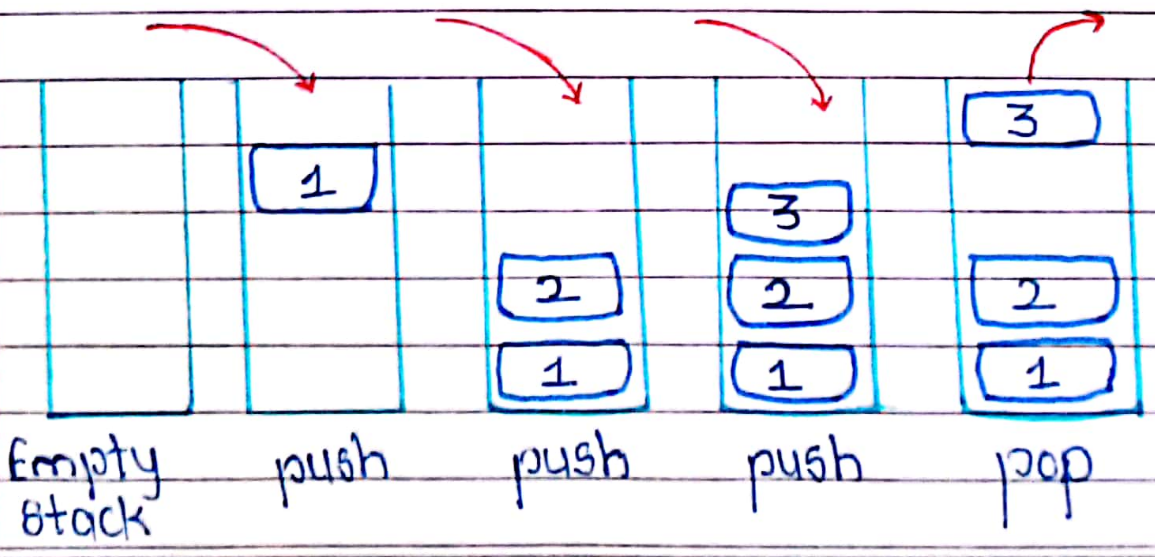
6] ArrayList is not Synchronized. Its equivalent synchronized class in Java is vector.

3] Stack

@coders_notes

Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).

There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO (Last In First Out) / FILO (First In Last Out) order.


1
2
1
3
2
1
3
2
1
Empty stack
push
push
push
pop