Basic Three Data Structures and Implementation in Java.

- Stack
- Queue
- Linked List

Stack

Stack is a data structure that works on LIFO (Last in First Out)

Stack maintains a pointer named as Top. Top points to top most element in the stack.

Stack has two operations Push and Pop.

Push means you are inserting element in Stack and Pop means you are getting or taking out an element from Stack.

When Push operation is performed top pointer increases by 1 and when Pop Operation is performed Top pointer decreases by 1.

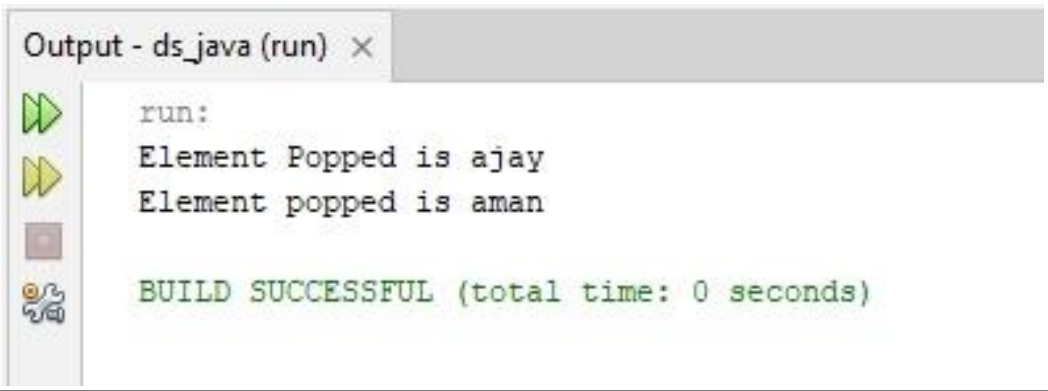
Stack can be used for Lexical Analysis or to create Lexical Analysers.

Following is the implementation of Stack in Java.

```
import java.util.Stack;


/**
 *
 * @author raman
 */
public class Stack_Impl {
   public static void main(String args[])
   {
       Stack a=new Stack();
       a.push("raman");
       a.push("aman");
       a.push("ajay");
       String e;
       e=(String)a.pop();
       System.out.println("Element Popped is " + e);
       e=(String)a.pop();
```

```
        System.out.println("Element popped is "+ e);

        System.out.println("");


    }

}
```

Output in Netbeans



Output - ds_java (run)

```
run:
Element Popped is ajay
Element popped is aman

BUILD SUCCESSFUL (total time: 0 seconds)
```

Next Data Structure is Queue

Queue is a Data Structure just like a Railway Ticket Queue in which the person standing in front of queue gets ticket first and person standing at the last of the queue gets ticket at the last.

Queue works on FIFO (First In First Out)

Queue maintains two pointers which are

- front
- rear

front pointer points to the beginning of the queue or we can say front points to the first element in the queue.

rear pointer points to the last of the queue or we can say queue points to the last element in the queue.

When performing insert operation in queue the rear pointer is increased by 1.

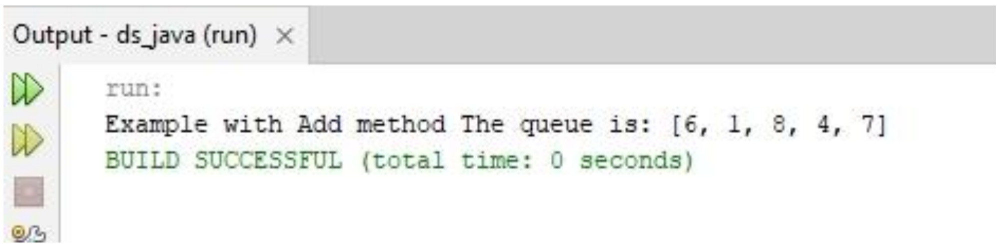
When performing remove operation in queue the front pointer is decreased by 1.

Following is the code for a Queue Data Structure in Java.

```
import java.util.LinkedList;

import java.util.Queue;


/**
```

```
 *
 * @author raman
 */
public class Queue_Impl {
   public static void main(String args[])
   {
    Queue<Integer> queueOne = new LinkedList<>();

queueOne.add(6); // add method to use insert element

queueOne.add(1);

queueOne.add(8);

queueOne.add(4);

queueOne.add(7);

     System.out.println("Example with Add method The queue is: " + queueOne);

   }
}
```

Output

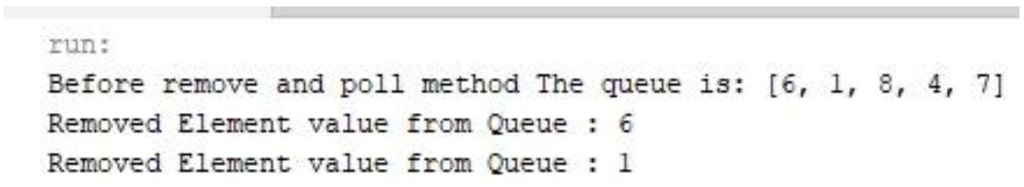


```
run:
Example with Add method The queue is: [6, 1, 8, 4, 7]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Following is the Java Code to remove elements from queue

```
import java.util.LinkedList;

import java.util.Queue;


/**

 *

 * @author raman

 */

public class Queue_Impl {

   public static void main(String args[])

   {

  // declaring variable


// Queue is a interface it has two methods to add elements


Queue<Integer> queueOne = new LinkedList<>();


queueOne.add(6); // add method to use insert element


queueOne.add(1);


queueOne.add(8);


queueOne.add(4);


queueOne.add(7);


System.out.println("Before remove and poll method The queue is: " + queueOne);


// poll method to remove top of the element in Queue
int positionOne = queueOne.poll();

System.out.println("Removed Element value from Queue : "+positionOne);

// remove method to remove top of the element in Queue

int positionTwo = queueOne.remove();

System.out.println("Removed Element value from Queue : "+positionTwo);



   }
}
```

Output in Netbeans



```
run:
Before remove and poll method The queue is: [6, 1, 8, 4, 7]
Removed Element value from Queue : 6
Removed Element value from Queue : 1
```

Linked List

A linked list is a linear data structure that stores a collection of data elements dynamically.

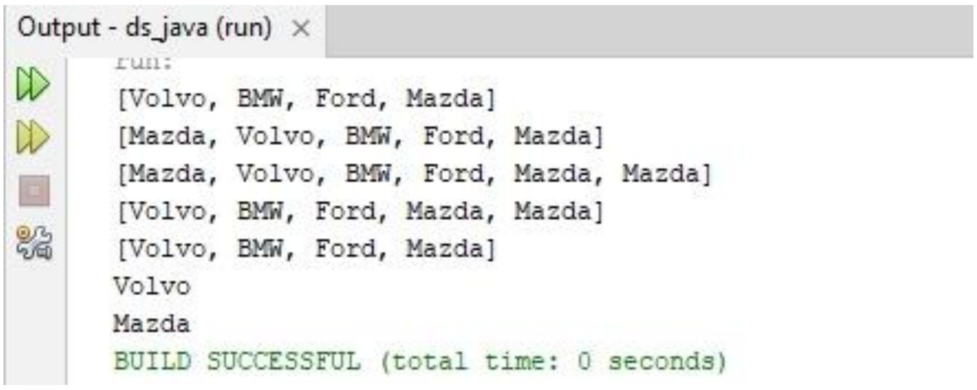
How Linked List works in Java

The `LinkedList` stores its items in "containers." The list has a link to the first container and each container has a link to the next container in the list. To add an element to the list, the element is placed into a new container and that container is linked to one of the other containers in the list.

Following code demonstrates Linked List Data Structure in Java

```
import java.util.LinkedList;

/**
 *
 * @author raman
 */
public class LinkedList_Impl {
     public static void main(String[] args) {
   LinkedList<String> cars = new LinkedList<String>();
   cars.add("Volvo");
   cars.add("BMW");
   cars.add("Ford");
   cars.add("Mazda");
   System.out.println(cars);
   cars.addFirst("Mazda");
   System.out.println(cars);
   cars.addLast("Mazda");
   System.out.println(cars);
   cars.removeFirst();
   System.out.println(cars);
   cars.removeLast();
   System.out.println(cars);
   // Use getFirst() to display the first item in the list
   System.out.println(cars.getFirst());
   // Use getLast() to display the last item in the list
   System.out.println(cars.getLast());


  }
}
```

Output in Netbeans



```
run:
[Volvo, BMW, Ford, Mazda]
[Mazda, Volvo, BMW, Ford, Mazda]
[Mazda, Volvo, BMW, Ford, Mazda, Mazda]
[Volvo, BMW, Ford, Mazda, Mazda]
[Volvo, BMW, Ford, Mazda]
Volvo
Mazda
BUILD SUCCESSFUL (total time: 0 seconds)
```