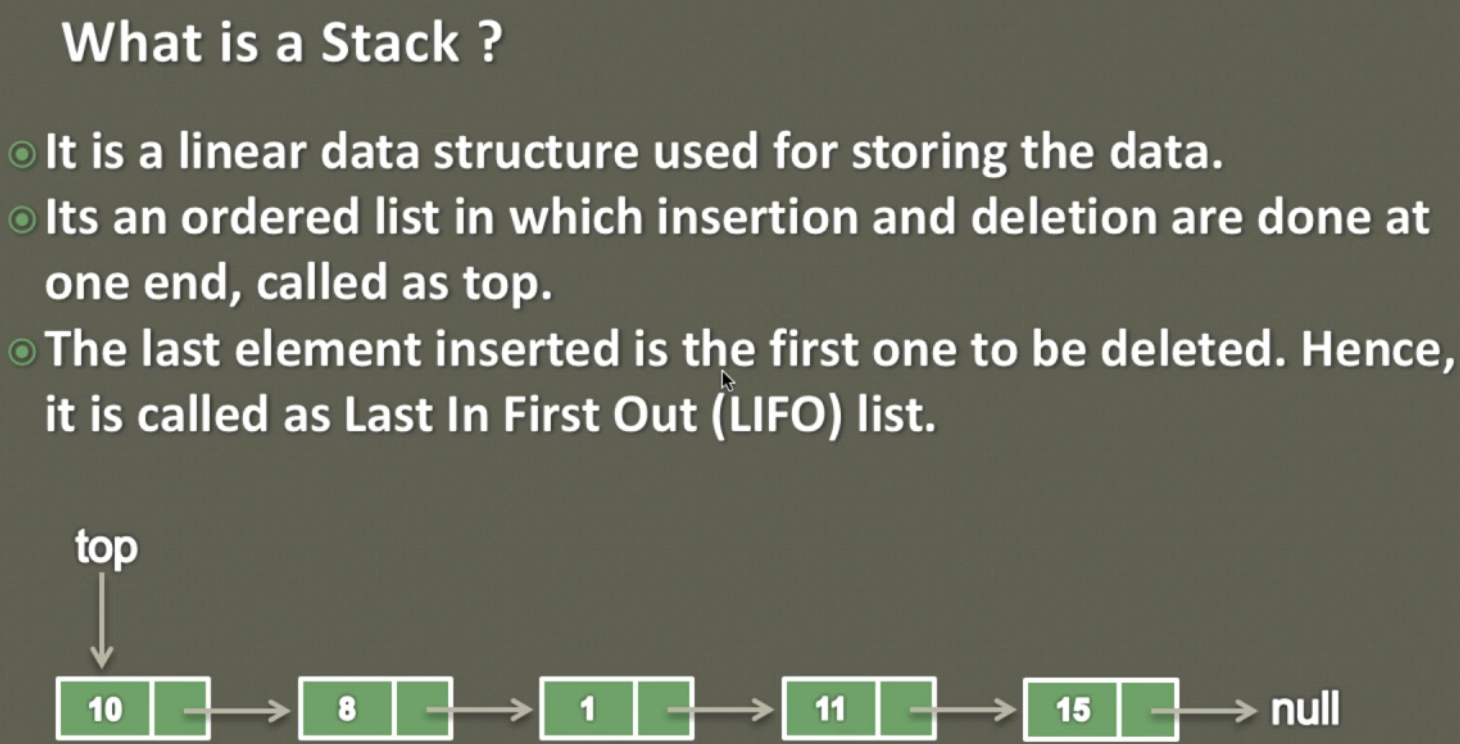
**80-How to represent a Stack in Java**



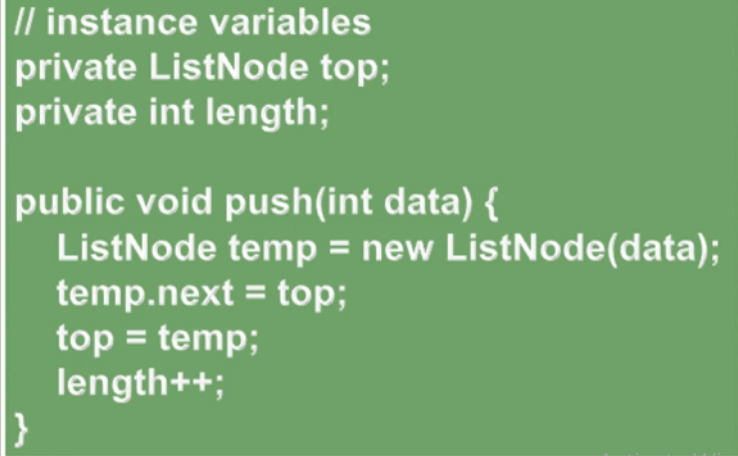
1. We call it as linear data structure because we can represent this as ArrayList or as an array where the nodes are adjacent to each other therefore it is linear data structure.

As per the above image we have stack where the first node is pointed by the top and we will have restriction here is that the node will be inserted or deleted from one end called top and this restrictiveness makes it LIFO in nature.

|  |  |  |  |
| --- | --- | --- | --- |
| As soon as we push this element it will become the top of the stack and top element (Last element) is supposed to delete first from the stack.    Now if we want to pop an element from the stack then there is one end top from where the nodes will be removed. So if we pop 20 then top will point to 15 and if we pop 15 then top will point to 10 and if pop 10 then top will point to null. |  | As soon as 15 gets removed now top start pointing to 10 | Stack is empty and now top is pointing to null |

**81-How to implement a Stack in Java - Stack in Data Structure**

So here we will implement stack using the linked list and as we know linked list is represented by list node where it contains two things, 1- data and 2- pointer to the next list node.

****

1. Since the first node of the list is always represent the top of the stack, we have created on variable of ListNode which represents top of the stack.
2. Create another variable which will count the length of stack as soon as node goes in or out of the stack.
3. Now here to insert node or element in the stack we would create one method with the name push which will take data as an input.
4. Inside push method:

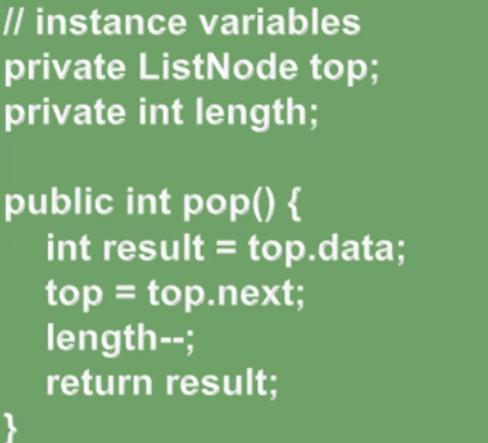
* Initially when stack is empty then its top node will point to null.
* Create a ListNode temp having data and since this temp node would be independent node so its data part will contain data and its next part will point null.
* Now to make this temp node as top node of the stack, just point [temp.next = top]. So now temp.next which earlier was pointing to null now will start pointing to top node.
* Finally assign the data value to the top node and increase length by one.
* In the same we way we can keep inserting the element in the stack and every time the last inserted node will point top of the stack and then increase the length of stack by one

|  |  |  |  |
| --- | --- | --- | --- |
| Push(10) |  |  | So finally we have create one node inside stack called top having data 10 and which next is pointing to null. |

|  |  |  |
| --- | --- | --- |
|  |  |  |

Now new node 15 got pushed into node and now which is nothing but the top of the stack.

**Implementing pop() of stack:**

****

1. Now the stack is full with nodes or element and we have to remove the node or element from the top, so let’s create one ListNode variable top
2. Create one int variable length to count the size of stack once we pop the element from the stack
3. Now let’s create one method which will return an integer. When we call the pop method of the stack then it removes the element from the top permanently

Inside the pop()

* First of all assign the top node data into one result variable
* Assign the top.next (which is currently pointing to next element of stack ) to the next node (below node) in the stack. So now the below node will start pointing as top. Because once the first node gets removed from the stack then another adjacent element start pointing to top
* Since we have removed the top data so now let’s decrease the size of stack by one.
* And finally return the result having the top node data.

|  |  |  |
| --- | --- | --- |
| Now length =2 | Now 15 is ready to be GC |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

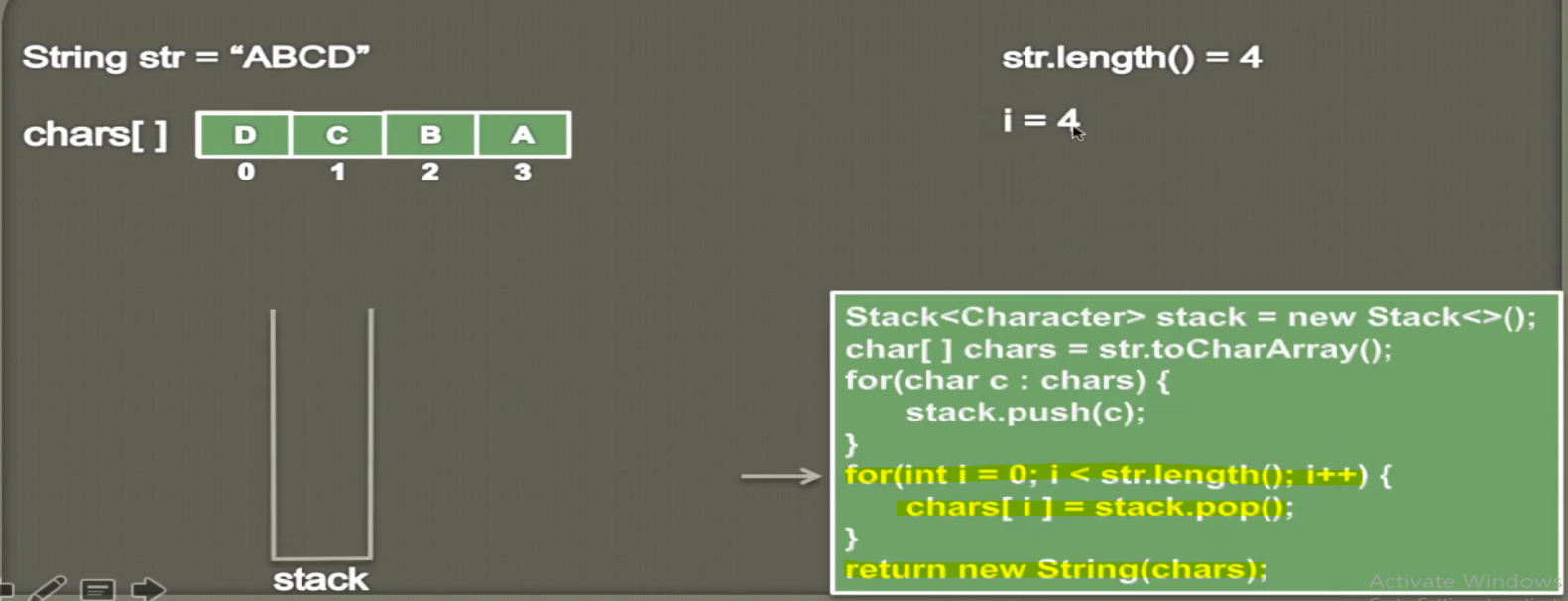
**Now top is pointing to null, it means now stack is empty.**

|  |
| --- |
| **package** com.linear.linked.list.stack;  **import** java.util.EmptyStackException;  **public** **class** Stack {  **private** ListNode top;  **private** **int** lenght;    **private** **class** ListNode{  **private** **int** data;  **private** ListNode next;    **public** ListNode(**int** data) {  **this**.data= data;  //this.next=null;  }  }  // Create an empty stack whose top will null and length =0  **public** Stack() {  **this**.top=**null**;  **this**.lenght=0;  }    **public** **int** length() {  **return** lenght;  }  // If stack length is 0 then return true  **public** **boolean** isEmpty() {  **return** lenght==0;  }  **public** **void** push(**int** data) {  ListNode temp = **new** ListNode(data);// Create a temp node having data value    temp.next=top; // Now point this temp next to top where top will point to null  top=temp; // finally assign the temp value to top  lenght++;  }  // Remove the last inserted value in the stack  **public** **int** pop() {  **if**(isEmpty()) {  **throw** **new** EmptyStackException();  }    **int** result = top.data; // assigning the top data value into result variable  top= top.next; // Assigning top to top.next (i.e. the just below node to the top  lenght--;  **return** result;  }  // peek () : This method just return the data of the last inserted node in the stack, and does not remove node from the stack  **public** **int** peek() {  **if**(isEmpty()) {  **throw** **new** EmptyStackException();  }  **return** top.data;  }  **public** **static** **void** main(String[] args) {  // Fist create an empty stack whose top will point to null and whose size = 0  Stack stack= **new** Stack();  stack.push(10);  stack.push(15);  stack.push(20);    System.***out***.println(stack.peek());// Only will display data of top node  stack.pop(); // will remove node from the stack permanently  System.***out***.println(stack.peek());  stack.pop();  System.***out***.println(stack.peek());  // stack.pop();  // System.out.println(stack.peek());  }  } |
| **Output:**  The top node value: 20  The top node value: 15  The top node value: 10 |

**82-How to reverse a String using a Stack in Java – Animation**

****

1. Convert the string into charArray()
2. Push each character into stack using push method in the for each loop
3. So the A > B > C > D

****

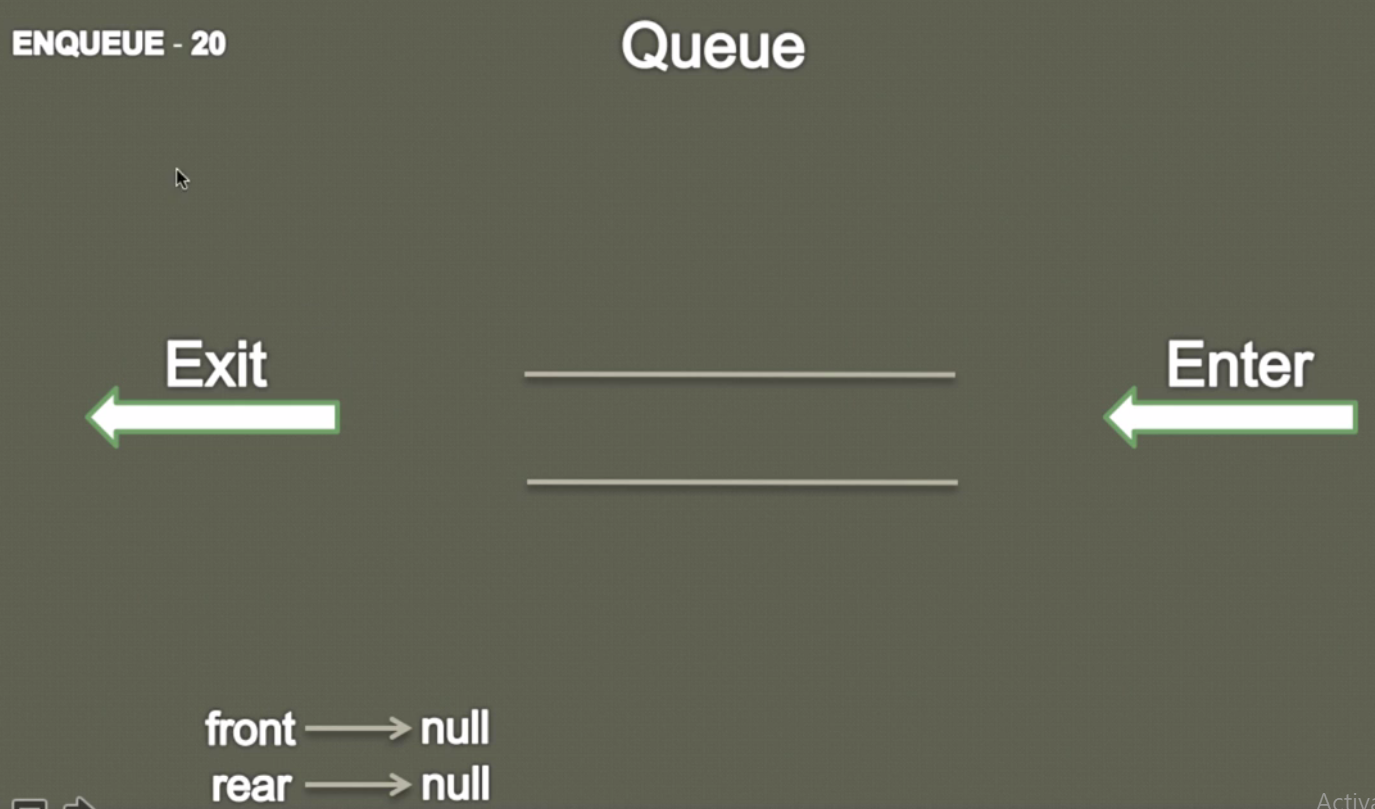
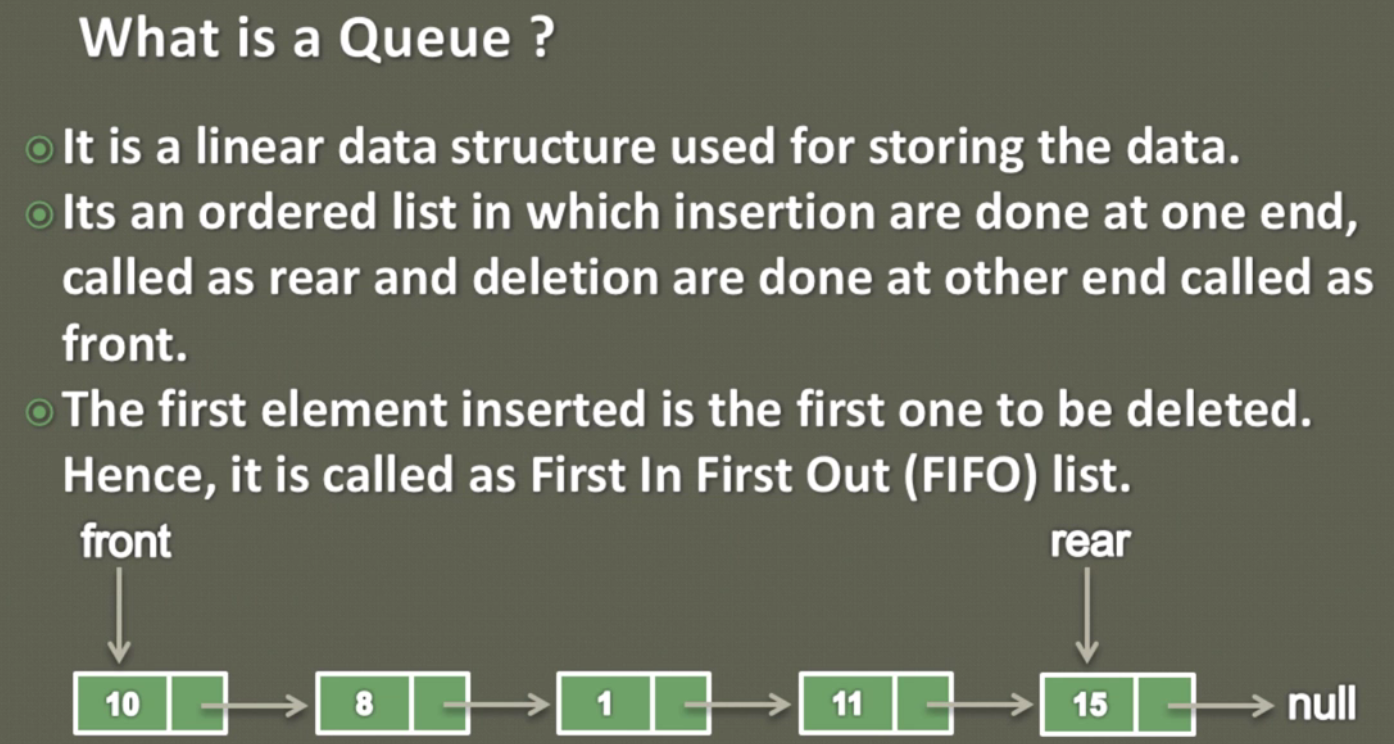
**Now**

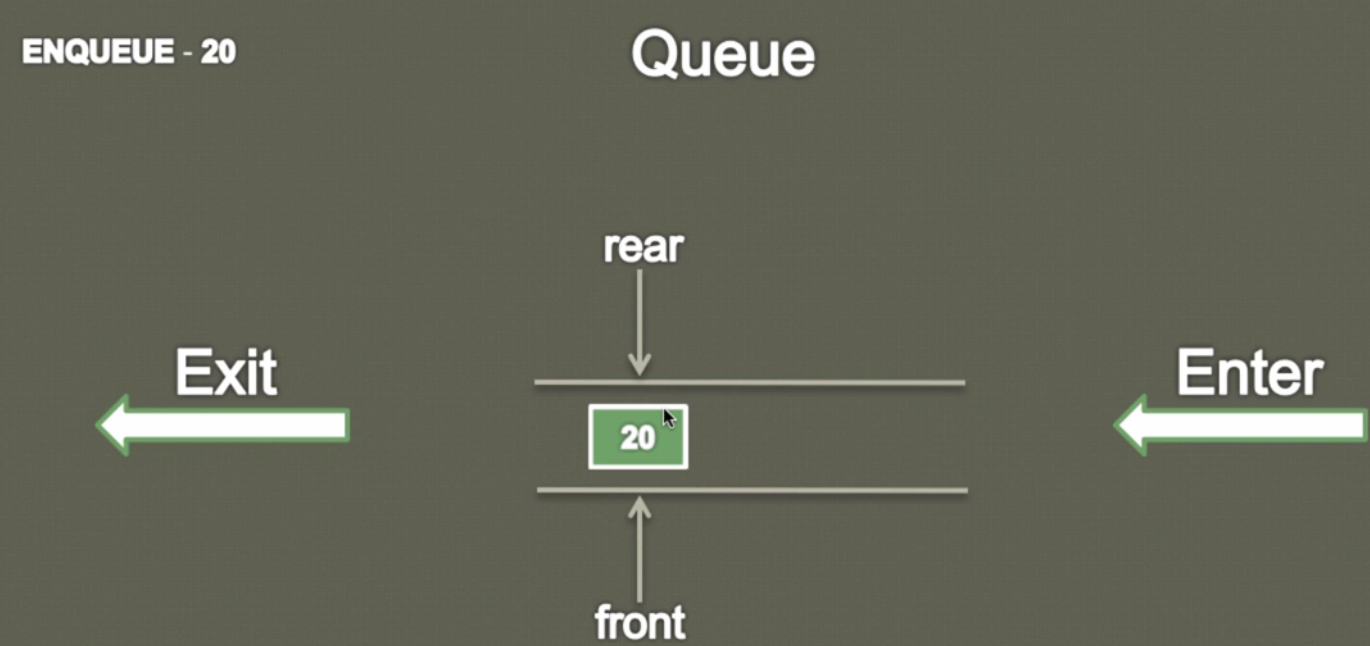
1. Travers the till length of the string and keep taking data from stack using pop method and store into char[] array
2. And finally return the same char array.
3. D > C > B > A

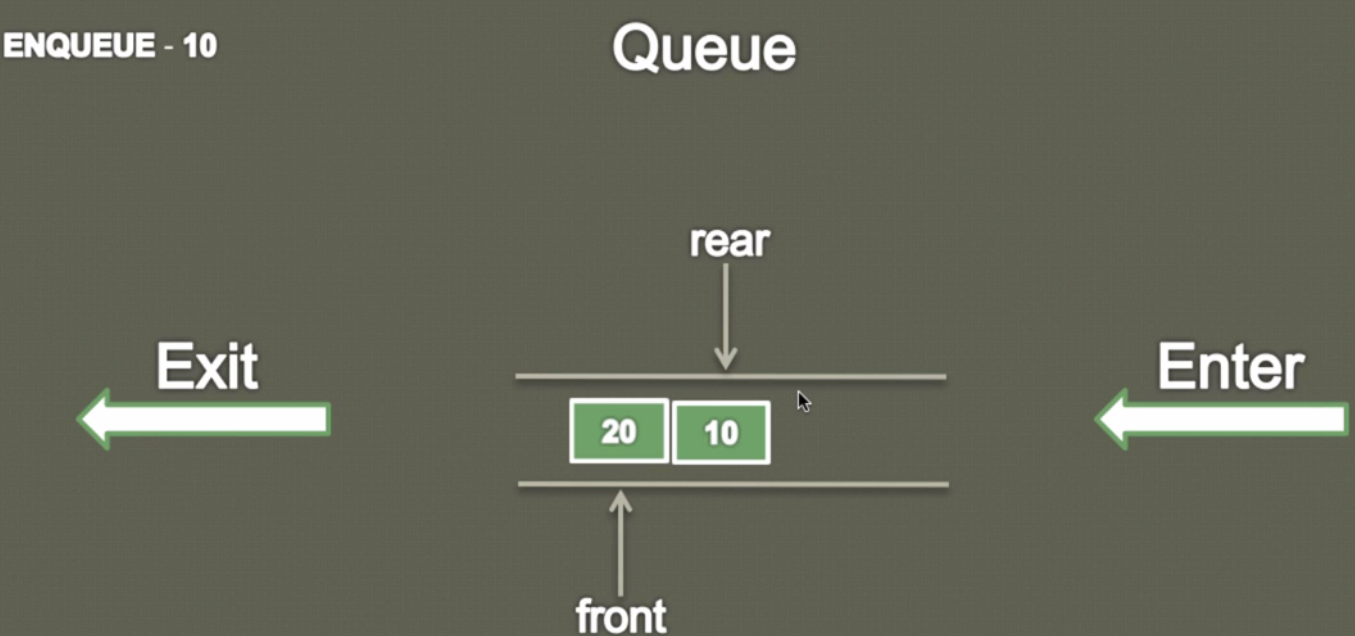
**In this way we can reverse the string using stack**

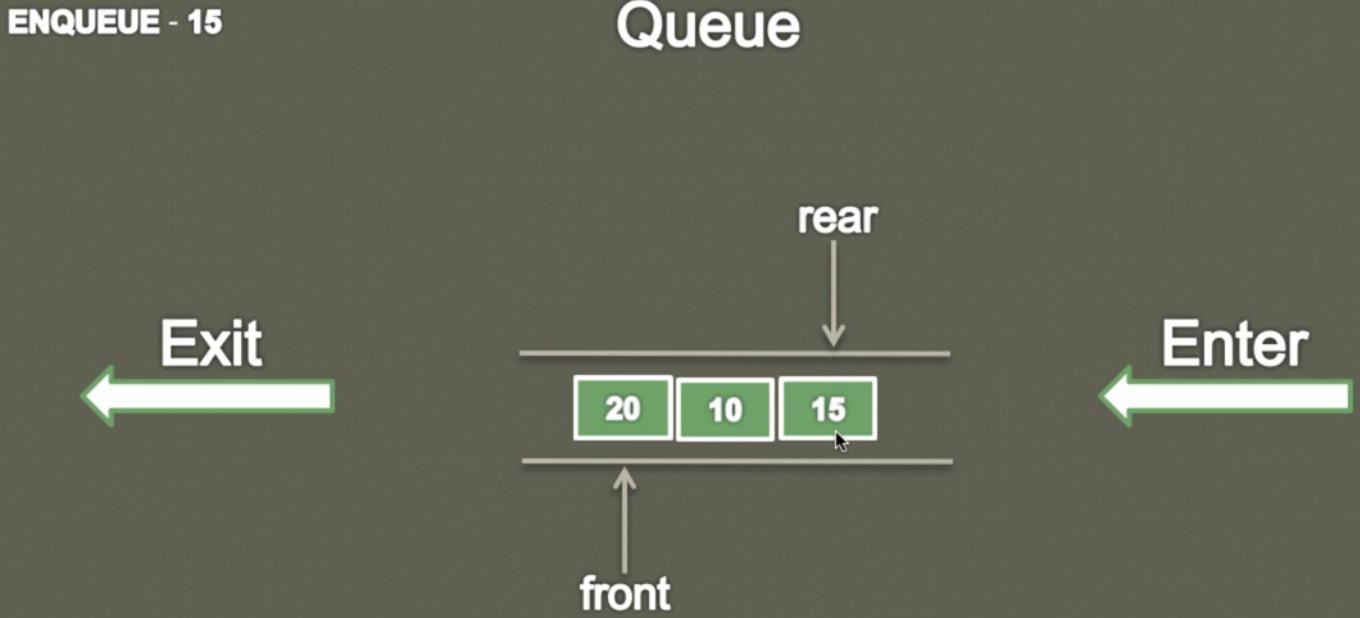
**Program:**

**84-How to represent a Queue in Java \_**

****

****

****

****