<https://www.youtube.com/watch?v=NjaM8edjUA0>

<https://www.youtube.com/watch?v=bbyQHBbSrYA>

Stack Implementation in Java Using Array

Stack Implementation in Java Using LinkedList

**Stack Implementation in Java Using Array**

**package** test;

**public** **class** Stack {

**private** **int** maxSize;

**private** **int**[] stackArr;

**private** **int** top;

**public** Stack(**int** maxSize) {

**super**();

**this**.maxSize = maxSize;

**this**.stackArr = **new** **int**[maxSize];

**this**.top = -1;

}

**public** **void** push(**int** data) {

**if** (isFull()) {

**throw** **new** RuntimeException("Stack is Full!!");

}

stackArr[++top] = data;

}

**public** **int** pop() {

**if** (isEmpty()) {

**throw** **new** RuntimeException("Stack is Empty!!");

}

**return** stackArr[top--];

}

// get top most item

**public** **int** peek() {

**if** (isEmpty()) {

**throw** **new** RuntimeException("Stack is Empty!!");

}

**return** stackArr[top];

}

**public** **boolean** isEmpty() {

**if** (top == -1)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isFull() {

**if** (top == maxSize - 1) {

**return** **true**;

} **else** {

**return** **false**;

}

}

**public** **int** size() {

**return** top + 1;

}

}

**package** test;

**public** **class** Tester {

**public** **static** **void** main(String[] args) {

Stack stack = **new** Stack(5);

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

stack.push(50);

//stack.push(60);

System.***out***.println(stack.isFull());

System.***out***.println(stack.isEmpty());

System.***out***.println(stack.peek());

System.***out***.println(stack.pop());

System.***out***.println(stack.size());

}

}

// Space Complexity push(data) : O(n)

// Time Complexity push(data) pop() peek() isEmpty() isFull() : O(1)

**Stack Implementation in Java Using LinkedList**

**package** test;

**import** java.util.EmptyStackException;

**public** **class** Stack {

**private** Node top;

**private** **int** size;

**private** **int** MAX\_SIZE = 5;

**class** Node {

**private** **int** data;

**private** Node next;

**public** Node(**int** data) {

**super**();

**this**.data = data;

**this**.next = **null**;// initially node points to null

}

}

**public** **void** push(**int** data) {

Node newNode = **new** Node(data);

newNode.next = top;

top = newNode;

size++;

}

**public** **int** pop() {

**if** (isEmpty())

**throw** **new** EmptyStackException();

**int** result = top.data;

top = top.next;

size--;

**return** result;

}

**public** **int** peek() {

**if** (isEmpty())

**throw** **new** EmptyStackException();

**int** result = top.data;

**return** result;

}

**public** **int** size() {

**return** size;

}

**public** **boolean** isFull() {

**if** (size == MAX\_SIZE)

**return** **true**;

**else**

**return** **false**;

}

**public** **boolean** isEmpty() {

**if** (size == 0)

**return** **true**;

**else**

**return** **false**;

}

**public** **void** displayStack() {

Node current = top;

**while** (current != **null**) {

System.***out***.print(current.data + " ");

current = current.next;

}

}

}

**package** test;

**public** **class** Tester {

**public** **static** **void** main(String[] args) {

Stack stack = **new** Stack();

System.***out***.println(stack.size());

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

stack.push(50);

stack.push(60);

// System.out.println(stack.isFull());

System.***out***.println(stack.size());

System.***out***.println(stack.isEmpty());

System.***out***.println(stack.peek());

System.***out***.println(stack.pop());

System.***out***.println(stack.size());

stack.displayStack();

}

}

// Space Complexity push(data) : O(n)

// Time Complexity push(data) pop() peek() isEmpty() isFull() : O(1)