**Stack:**

Stack can be simply defined as Last In First Out (LIFO) data structure,i.e.,the last element added at the top of the stack(In) should be the first element to be removed(Out) from the stack.

Whatever are element added in last we can access first(LIFO).

**Queue:**

Java Queue interface orders the element in FIFO(First In First Out) manner. In FIFO, first element is removed first and last element is removed at last.

|  |
| --- |
| **publicclass**MyStackImpl {  **privateint**stackSize;  **privateint**[] stackArr;  **privateint**top;    /\*\*  \* constructor to create stack with size  \* **@param** size  \*/  **public**MyStackImpl(**int**size) {  **this**.stackSize = size;  **this**.stackArr = **newint**[stackSize];  **this**.top = -1;  }    /\*\*  \* This method adds new entry to the top  \* of the stack  \* **@param**entry  \* **@throws** Exception  \*/  **publicvoid**push(**int**entry) **throws** Exception {  **if**(**this**.isStackFull()){  **thrownew**Exception("Stack is already full. Can not add element.");  }  System.***out***.println("Adding: "+entry);  **this**.stackArr[++top] = entry;  }    /\*\*  \* This method removes an entry from the  \* top of the stack.  \* **@return**  \* **@throws** Exception  \*/  **publicint**pop() **throws** Exception {  **if**(**this**.isStackEmpty()){  **thrownew**Exception("Stack is empty. Can not remove element.");  }  **int**entry = **this**.stackArr[top--];    System.***out***.println("Removed entry: "+entry);  **return**entry;  }    /\*\*  \* This method returns top of the stack  \* without removing it.  \* **@return**  \*/  **publicint**peek() {  **return**stackArr[top];  }    /\*\*  \* This method returns true if the stack is  \* empty  \* **@return**  \*/  **publicboolean**isStackEmpty() {  **return** (top == -1);  }    /\*\*  \* This method returns true if the stack is full  \* **@return**  \*/  **publicboolean**isStackFull() {  **return** (top == stackSize - 1);  }    **publicstaticvoid**main(String[] args) {  MyStackImplstack = **new**MyStackImpl(5);  **try** {  stack.push(4);  stack.push(8);  stack.push(3);  stack.push(89);  stack.pop();  stack.push(34);  stack.push(45);  stack.push(78);  } **catch** (Exception e) {  System.***out***.println(e.getMessage());  }  **try** {  stack.pop();  stack.pop();  stack.pop();  stack.pop();  stack.pop();  stack.pop();  } **catch** (Exception e) {  System.***out***.println(e.getMessage());  }  }  } |

Bubble Sort:

|  |
| --- |
| **public** **class** BubbleSortProgram {  **public** **static** **void** main(String[] args) {  **int**[] array = {99, 88, 55, 77, 1, 66};  **int** temp;  **for**(**int** pass = 1; pass <array.length; pass++) {  **for**(**int** current=0; current < array.length-pass; current++) {  //System.out.println("I:J "+pass +": "+current);  **if**(array[current] > array[current+1]) {  // System.out.println(array[current] +" > "+array[current + 1]);  temp = array[current];  array[current] = array[current + 1];  array[current + 1] = temp;  }  }  }  System.***out***.println("After Sorting Elements");  **for**(**int** i : array) {  System.***out***.println(i);  }  }  } |