***Dt : 14/11/2022***

***=>In realtime Stack<E> and Queue<E> are used in algorithmic***

***designs part of Vendor Engineering.(Product based Engineering)***

***============================================================***

***3.Queue<e>:***

***=>Queue<E> organizes elements based on the algorithm***

***First-In-First-Out/Last-In-Last-Out***

***=>The following are some important methods of Queue<E>:***

***public abstract boolean add(E);***

***public abstract boolean offer(E);***

***public abstract E remove();***

***public abstract E poll();***

***public abstract E element();***

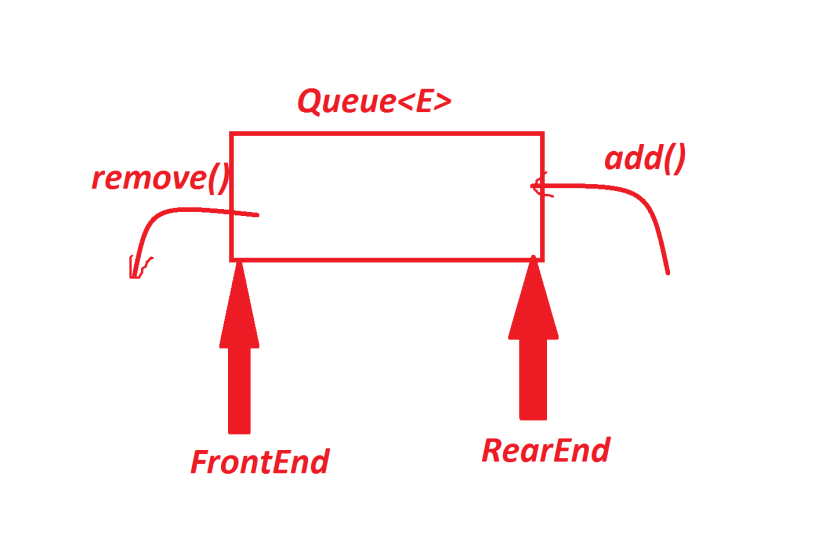
***public abstract E peek();***

***=>The following is the implementation of Queue<E>***

***=>PriotityQueue<E>***

***=>PriorityQueue<E> organizes elements based on elements***

***priority.***

******

***Ex-program : DemoQueue.java***

***package maccess;***

***import java.util.\*;***

***public class DemoQueue {***

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

***Queue<Integer> ob = new PriorityQueue<Integer>();***

***Scanner s = new Scanner(System.in);***

***try(s;){***

***try {***

***while(true) {***

***System.out.println("====Choice====");***

***System.out.println("1.add(E)\n2.remove()\n3.peek()\n4.element()\n5.poll()\n6.exit");***

***System.out.println("Enter the choice:");***

***switch(s.nextInt()) {***

***case 1:***

***System.out.println("Enter the ele:");***

***ob.add(new ~~Integer~~(s.nextInt()));***

***System.out.println(ob.toString());***

***break;***

***case 2:***

***if(ob.isEmpty()) {***

***System.out.println("Queue is empty...");***

***}else {***

***ob.remove();***

***System.out.println(ob.toString());***

***}***

***break;***

***case 3:***

***if(ob.isEmpty()) {***

***System.out.println("Queue is empty...");***

***}else {***

***System.out.println("Peek ele : "+ob.peek());***

***System.out.println(ob.toString());***

***}***

***break;***

***case 4:***

***if(ob.isEmpty()) {***

***System.out.println("Queue is empty...");***

***}else {***

***System.out.println("Element : "+ob.element());***

***System.out.println(ob.toString());***

***}***

***break;***

***case 5:***

***if(ob.isEmpty()) {***

***System.out.println("Queue is empty...");***

***}else {***

***System.out.println("poll : "+ob.poll());***

***System.out.println(ob.toString());***

***}***

***break;***

***case 6:***

***System.out.println("Queue Operations Stopped...");***

***System.exit(0);***

***default:***

***System.out.println("Invalid Choice..");***

***}//end of switch***

***}//end of loop***

***}catch(Exception e) {e.printStackTrace();}***

***}//end of try***

***}***

***}***

***o/p:***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***1***

***Enter the ele:***

***11***

***[11]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***12***

***Invalid Choice..***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***1***

***Enter the ele:***

***11***

***[11, 11]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***1***

***Enter the ele:***

***12***

***[11, 11, 12]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***1***

***Enter the ele:***

***13***

***[11, 11, 12, 13]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***2***

***[11, 13, 12]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***3***

***Peek ele : 11***

***[11, 13, 12]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***4***

***Element : 11***

***[11, 13, 12]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

***5***

***poll : 11***

***[12, 13]***

***====Choice====***

***1.add(E)***

***2.remove()***

***3.peek()***

***4.element()***

***5.poll()***

***6.exit***

***Enter the choice:***

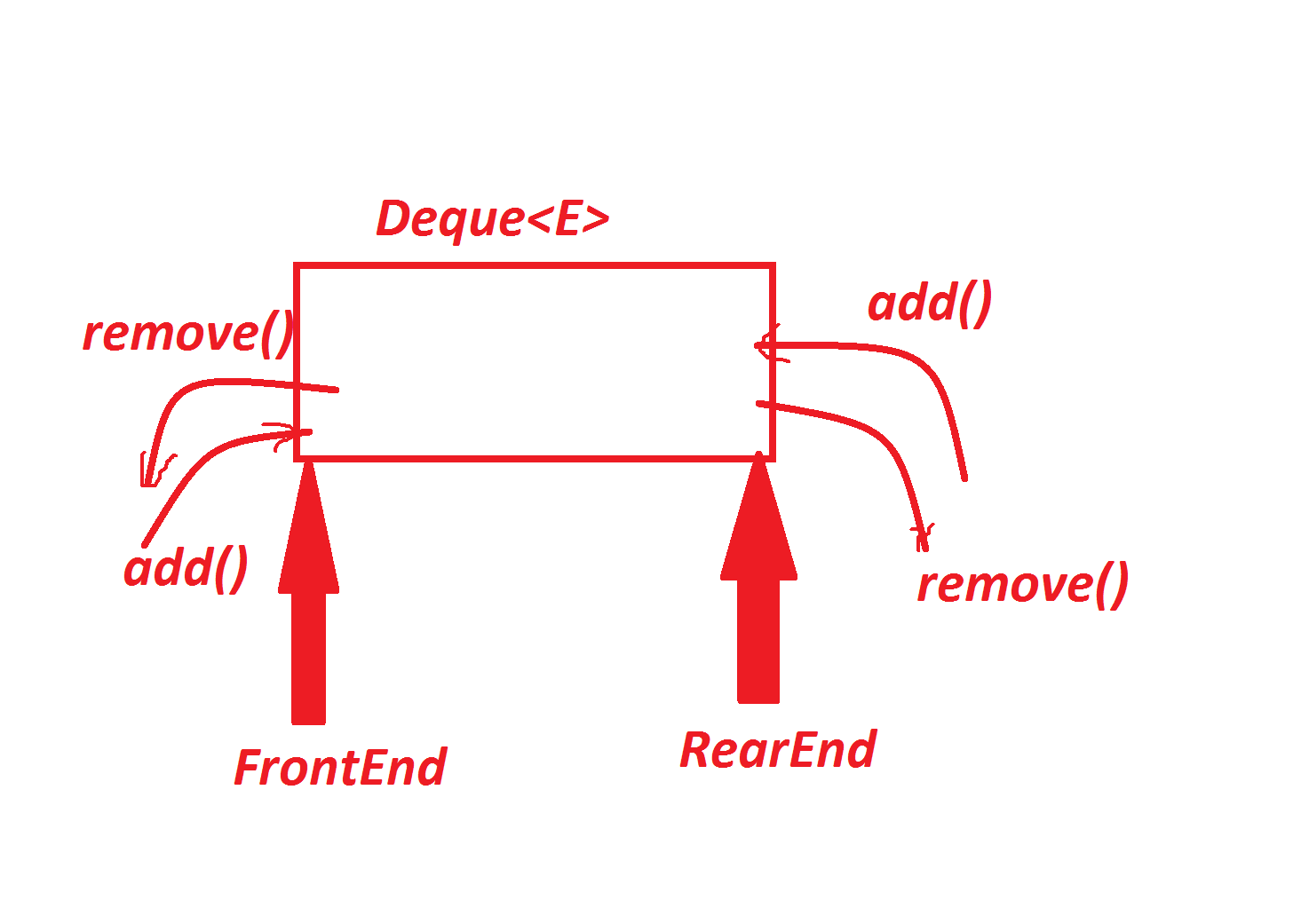
***========================================================***

***define Deque<E>?***

***=>Deque<E> is a Child-Interface of Queue<E> and which***

***organizes elements on both ends.***

***=>Deque<E> means Double-Ended-Queue***

******

***=>The following are some important methods of Deque<E>:***

***public abstract void addFirst(E);***

***public abstract void addLast(E);***

***public abstract boolean offerFirst(E);***

***public abstract boolean offerLast(E);***

***public abstract E removeFirst();***

***public abstract E removeLast();***

***public abstract E pollFirst();***

***public abstract E pollLast();***

***public abstract E getFirst();***

***public abstract E getLast();***

***public abstract E peekFirst();***

***public abstract E peekLast();***

***public abstract boolean removeFirstOccurrence***

***(java.lang.Object);***

***public abstract boolean removeLastOccurrence***

***(java.lang.Object);***

***-----------------------------------------------***

***=>The following are implementation classes of Deque<E>:***

***(i)ArrayDeque<E> - Sequence***

***(ii)LinkedList<E> - NonSequence***

***====================================================***