QUEUE

public class CircularQueuesUsingArrays

{ static class Queue

{ static int arr[];

static int size;

static int rear;

static int front;

Queue(int n) {

arr = new int[n];

size = n;

rear = -1;

front = -1;

}

public boolean isEmpty() {

return rear == -1 && front == -1;

}

public static boolean isFull() {

return (rear + 1) % size == front;

}

// add

public void add(int data) {

if (isFull()) {

System.out.println("queue is full");

return;

}

// add 1st element

if (front == -1) {

front = 0;

}

rear = (rear + 1) % size;

arr[rear] = data;

}

// remove

public int remove() {

if (isEmpty()) {

System.out.println("Empty queue");

return -1;

}

int result = arr[front];

// last el delete

if (rear == front) {

rear = front = -1;

} else {

front = (front + 1) % size;

}

return result;

}

// peek

public int peek() {

if (isEmpty()) {

System.out.println("Empty queue");

return -1;

}

return arr[front];

}

}

public static void main(String[] args) {

Queue q = new Queue(5);

q.add(1);

q.add(2);

q.add(3);

q.add(4);

q.add(5);

while (!q.isEmpty()) {

System.out.println(q.remove());

}

}

}

/\*

* Output:
* 1
* 2
* 3
* 4
* 5 \*/

2.

/\*

\* Deque [ Double ended queue ]

\*/

import java.util.\*;

public class DequeB {

public static void main(String[] args) {

Deque<Integer> deque = new LinkedList<>();

deque.addFirst(1);

deque.addFirst(2);

deque.addFirst(3);

deque.addLast(5);

System.out.println(deque); // [3, 2, 1, 5]

deque.removeFirst();

deque.removeLast();

System.out.println(deque); // [2, 1]

System.out.println("get first : " + deque.getFirst()); // get first : 2

System.out.println("get last : " + deque.getLast()); // get last : 1

}

}

/\*

\* Output:

\* [3, 2, 1, 5]

\* [2, 1]

\* get first : 2

\* get last : 1

\*

\*/

3.

/\*

\* Queues Using Arrays

\*/

public class QueuesUsingArrays {

static class Queue {

static int arr[];

static int size;

static int rear;

Queue(int n) {

arr = new int[n];

size = n;

rear = -1;

}

public boolean isEmpty() {

return rear == -1;

}

// add

public void add(int data) {

if (rear == size - 1) {

System.out.println("queue is full");

return;

}

rear = rear + 1;

arr[rear] = data;

}

// remove

public int remove() {

if (isEmpty()) {

System.out.println("empty queue");

return -1;

}

int front = arr[0];

for (int i = 0; i < rear; i++) {

arr[i] = arr[i + 1];

}

rear = rear - 1;

return front;

}

// peek

public int peek() {

if (isEmpty()) {

System.out.println("empty queue");

return -1;

}

return arr[0];

}

}

public static void main(String[] args) {

Queue q = new Queue(5);

q.add(1);

q.add(2);

q.add(3);

q.add(4);

while (!q.isEmpty()) {

System.out.println(q.peek());

q.remove();

}

}

}==