**Queue**

* Queue using Array

//queue using array

public class QueueB {

static class Queue {

static int arr[];

static int size;

static int rear;

Queue(int size) {

this.size = size;

arr = new int[size];

rear = -1;

}

public static boolean isEmpty() {

return rear == -1;

}

public static boolean isFull() {

return rear == size-1;

}

public static void add(int data) {

if(isFull()) {

System.out.println("Overflow");

return;

}

arr[++rear] = data;

}

//O(n)

public static 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–;

return front;

}

public static 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);

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

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

}

}

Circular queue using array

//circular queue using array

public class QueueB {

static class Queue {

static int arr[];

static int size;

static int front = -1;

static int rear = -1;

Queue(int size) {

this.size = size;

arr = new int[size];

}

public static boolean isEmpty() {

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

}

public static boolean isFull() {

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

}

public static void add(int data) {

if(isFull()) {

System.out.println("Overflow");

return;

}

//if it's the 1st element

if(front == -1) {

front = 0;

}

rear = (rear + 1)%size;

arr[rear] = data;

}

public static int remove() {

if(isEmpty()) {

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

return -1;

}

int res = arr[front];

//if only 1 element is present

if(front == rear) {

front = rear = -1;

} else {

front = (front+1)%size;

}

return res;

}

public static 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);

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

q.add(6);

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

q.add(7);

while(!q.isEmpty()) {

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

}

}

}

* Queue using Linked List

//queue using Linked List

public class QueueB {

static class Node {

int data;

Node next;

Node(int data) {

this.data = data;

next = null;

}

}

static class Queue {

static Node head = null;

static Node tail = null;

public static boolean isEmpty() {

return head == null && tail == null;

}

public static void add(int data) {

Node newNode = new Node(data);

if(isEmpty()) {

tail = head = newNode;

} else {

tail.next = newNode;

tail = newNode;

}

}

public static int remove() {

if(isEmpty()) {

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

return -1;

}

int front = head.data;

//single node

if(head == tail) {

tail = null;

}

head = head.next;

return front;

}

public static int peek() {

if(isEmpty()) {

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

return -1;

}

return head.data;

}

}

public static void main(String args[]) {

Queue q = new Queue();

q.add(1);

q.add(2);

q.add(3);

q.add(4);

q.add(5);

while(!q.isEmpty()) {

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

q.remove();

}

}

}

* Java Collection Framework
* //queue using Java Collection Framework
* import java.util.\*;
* public class QueueB {
* public static void main(String args[]) {
* //Queue<Integer> q = new LinkedList();
* Queue<Integer> q = new ArrayDeque();
* q.add(1);
* q.add(2);
* q.add(3);
* q.add(4);
* q.add(5);
* while(!q.isEmpty()) {
* System.out.println(q.peek());
* q.remove();
* }
* }
* }
* Queue using 2 stacks

//queue using 2 stacks

import java.util.\*;

public class QueueB {

static class Queue {

static Stack<Integer> s1 = new Stack<>();

static Stack<Integer> s2 = new Stack<>();

public static boolean isEmpty() {

return s1.isEmpty();

}

public static void add(int data) {

while(!s1.isEmpty()) {

s2.push(s1.pop());

}

s1.push(data);

while(!s2.isEmpty()) {

s1.push(s2.pop());

}

}

public static int remove() {

return s1.pop();

}

public static int peek() {

return s1.peek();

}

}

public static void main(String args[]) {

Queue q = new Queue();

q.add(1);

q.add(2);

q.add(3);

while(!q.isEmpty()) {

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

q.remove();

}

}

}