//Assignment 14

//1.

**package** psr;

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

**public** **class** homework {

**private** **int**[] queue;

**private** **int** front, rear, size;

**private** **final** **int** capacity = 100;

**public** homework() {

queue = **new** **int**[capacity];

front = -1;

rear = -1;

size = 0;

}

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

**return** size == 0;

}

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

**return** size == capacity;

}

**public** **void** enqueue(**int** value) {

**if** (isFull()) {

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

**return**;

}

**if** (isEmpty()) {

front = 0;

}

rear = (rear + 1) % capacity;

queue[rear] = value;

size++;

}

**public** **int** dequeue() {

**if** (isEmpty()) {

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

**return** -1;

}

**int** result = queue[front];

front = (front + 1) % capacity;

size--;

**if** (size == 0) {

front = -1;

rear = -1;

}

**return** result;

}

**public** **void** display() {

**if** (isEmpty()) {

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

**return**;

}

**int** count = size;

**int** i = front;

System.***out***.print("Queue: ");

**while** (count > 0) {

System.***out***.print(queue[i] + " ");

i = (i + 1) % capacity;

count--;

}

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

}

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

homework q = **new** homework();

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.display();

System.***out***.println("Dequeued: " + q.dequeue());

q.display();

}

}

//2.

**package** psr;

**class** Node {

**int** data;

Node next;

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

**this**.data = data;

**this**.next = **null**;

}

}

**public** **class** homework {

**private** Node front, rear;

**public** homework() {

front = rear = **null**;

}

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

**return** front == **null**;

}

**public** **void** enqueue(**int** value) {

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

**if** (rear == **null**) {

front = rear = newNode;

**return**;

}

rear.next = newNode;

rear = newNode;

}

**public** **int** dequeue() {

**if** (isEmpty()) {

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

**return** -1; // or throw new NoSuchElementException("Queue is empty");

}

**int** result = front.data;

front = front.next;

**if** (front == **null**) {

rear = **null**;

}

**return** result;

}

**public** **void** display() {

**if** (isEmpty()) {

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

**return**;

}

Node temp = front;

System.***out***.print("Queue: ");

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

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

temp = temp.next;

}

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

}

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

homework q = **new** homework();

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.display();

System.***out***.println("Dequeued: " + q.dequeue());

q.display();

}

}

//3.

**package** psr;

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

**public** **class** homework {

**public** **static** **void** reverseFirstK(Queue<Integer> queue, **int** k) {

**if** (queue == **null** || k > queue.size() || k <= 0) {

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

**return**;

}

Stack<Integer> stack = **new** Stack<>();

**for** (**int** i = 0; i < k; i++) {

stack.push(queue.poll());

}

**while** (!stack.isEmpty()) {

queue.offer(stack.pop());

}

**int** size = queue.size();

**for** (**int** i = 0; i < size - k; i++) {

queue.offer(queue.poll());

}

}

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

Queue<Integer> q = **new** LinkedList<>(Arrays.*asList*(1, 2, 3, 4, 5));

**int** k = 3;

System.***out***.println("Original queue: " + q);

*reverseFirstK*(q, k);

System.***out***.println("After reversing first " + k + " elements: " + q);

}

}