DAY 14

public class QueueArray {

int SIZE = 100;

int[] queue = new int[SIZE];

int front = -1, rear = -1;

void enqueue(int x) {

if (rear == SIZE - 1) {

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

return;

}

if (front == -1) front = 0;

queue[++rear] = x;

}

void dequeue() {

if (front == -1 || front > rear) {

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

return;

}

System.out.println("Dequeued: " + queue[front++]);

}

void display() {

if (front == -1 || front > rear) {

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

return;

}

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

for (int i = front; i <= rear; i++)

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

System.out.println();

}

public static void main(String[] args) {

QueueArray q = new QueueArray();

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.display();

q.dequeue();

q.display();

}

}

2)

class Node {

int data;

Node next;

Node(int d) {

data = d;

next = null;

}

}

public class QueueLinkedList {

Node front = null, rear = null;

void enqueue(int x) {

Node newNode = new Node(x);

if (rear == null) {

front = rear = newNode;

return;

}

rear.next = newNode;

rear = newNode;

}

void dequeue() {

if (front == null) {

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

return;

}

System.out.println("Dequeued: " + front.data);

front = front.next;

if (front == null) rear = null;

}

void display() {

if (front == null) {

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

return;

}

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

Node temp = front;

while (temp != null) {

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

temp = temp.next;

}

System.out.println();

}

public static void main(String[] args) {

QueueLinkedList q = new QueueLinkedList();

q.enqueue(5);

q.enqueue(15);

q.enqueue(25);

q.display();

q.dequeue();

q.display();

}

}

3) import java.util.\*;

public class ReverseFirstKQueue {

public static Queue<Integer> reverseK(Queue<Integer> q, int k) {

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

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

stack.push(q.remove());

}

while (!stack.isEmpty()) {

q.add(stack.pop());

}

int size = q.size();

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

q.add(q.remove());

}

return q;

}

public static void main(String[] args) {

Queue<Integer> q = new LinkedList<>();

q.add(1);

q.add(2);

q.add(3);

q.add(4);

q.add(5);

int k = 3;

Queue<Integer> result = reverseK(q, k);

System.out.print("Queue after reversing first " + k + " elements: ");

while (!result.isEmpty()) {

System.out.print(result.remove() + " ");

}

System.out.println();

}

}