package stackimp;

import java.util.Collection;

import java.util.Iterator;

class Stack {

private int maxSize;

private int[] stackArray;

private int top;

public Stack(int size) {

maxSize = size;

stackArray = new int[maxSize];

top = -1;

}

public void push(int value) {

if (!isFull()) {

stackArray[++top] = value;

} else {

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

}

}

public int pop() {

if (!isEmpty()) {

return stackArray[top--];

} else {

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

return -1; // or throw an exception

}

}

public int peek() {

return stackArray[top];

}

public boolean isEmpty() {

return (top == -1);

}

public boolean isFull() {

return (top == maxSize - 1);

}

}

// Queue implementation

class Queue {

private int maxSize;

private int[] queueArray;

private int front;

private int rear;

public Queue(int size) {

maxSize = size + 1; // one extra space to differentiate between empty and full condition

queueArray = new int[maxSize];

front = 0;

rear = -1;

}

public void enQueue(int value) {

if (!isFull()) {

if (rear == maxSize - 1) {

rear = -1;

}

queueArray[++rear] = value;

} else {

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

}

}

public int deQueue() {

if (!isEmpty()) {

int temp = queueArray[front++];

if (front == maxSize) {

front = 0;

}

return temp;

} else {

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

return -1; // or throw an exception

}

}

public boolean isEmpty() {

return (rear + 1 == front || (front + maxSize - 1 == rear));

}

public boolean isFull() {

return (rear + 2 == front || (front + maxSize - 2 == rear));

}

}

public class StackImp {

public static void main(String[] args) {

// Test Stack

Stack stack = new Stack(5);

stack.push(1);

stack.push(2);

stack.push(3);

System.out.println(stack.pop()); // Output: 3

System.out.println(stack.peek()); // Output: 2

// Test Queue

Queue queue = new Queue(5);

queue.enQueue(1);

queue.enQueue(2);

queue.enQueue(3);

System.out.println(queue.deQueue()); // Output: 1

}

}