package CircularQueueProject;

import java.util.Arrays;

public class CircularQueue\_BOOKED {

private int front;

private int rear;

private String queueArray[];

CircularQueue\_BOOKED(int maxSize) {

queueArray = new String[maxSize + 1];

front = maxSize; // (maxSize + 1) - 1

rear = maxSize; // (maxSize + 1) - 1

}

// Add passenger's name to the queue

public void enqueue(String name) {

if(rear == queueArray.length - 1) {

rear = (rear + 1) % queueArray.length;

queueArray[rear] = name;

} else {

rear++;

queueArray[rear] = name;

}

}

// Changes the front variable

public void dequeue() {

if(front != rear) {

if(front == (queueArray.length) - 1)

front = 0;

else

front++;

}

else {

front = queueArray.length - 1;

rear = queueArray.length - 1;

}

}

// Returns the value of each element, front to rear, in the array

public String[] front() {

String[] passengers = new String[queueArray.length];

int dequeuedFront = -1;

int counter = 0;

int x = front;

while(x != rear) {

dequeuedFront = (x + 1) % queueArray.length;

if(x == (queueArray.length - 1)) {

passengers[counter] = (queueArray[dequeuedFront]);

x = 0;

}

else {

passengers[counter] = (queueArray[dequeuedFront]);

x++;

}

counter++;

}

return passengers;

}

// Check if the array is empty

public boolean isEmpty() {

if(front == rear)

return true;

return false;

}

// Check if array is full

public boolean isFull() {

if((rear + 1) % queueArray.length == front)

return true;

return false;

}

} // end class

package CircularQueueProject;

import java.util.Arrays;

public class CircularQueue\_WAITING {

private int front;

private int rear;

private String queueArray[];

CircularQueue\_WAITING(int maxSize) {

queueArray = new String[maxSize + 1];

front = maxSize; // (maxSize + 1) - 1

rear = maxSize; // (maxSize + 1) - 1

}

// Add passenger's name to the queue

public void enqueue(String name) {

if(rear == queueArray.length - 1) { // Wrap the rear of the queue to the front

rear = (rear + 1) % queueArray.length;

queueArray[rear] = name;

} else { // Add the name to the queue

rear++;

queueArray[rear] = name;

}

}

// Changes the front variable

public void dequeue() {

if(front != rear) {

if(front == (queueArray.length) - 1)

front = 0;

else

front++;

}

else {

front = queueArray.length - 1;

rear = queueArray.length - 1;

}

}

// Returns the value of each element, front to rear, in the array

public String[] front() {

String[] passengers = new String[queueArray.length];

int dequeuedFront = -1;

int counter = 0;

int x = front;

while(x != rear) {

dequeuedFront = (x + 1) % queueArray.length;

//System.out.println(dequeuedFront);

if(x == (queueArray.length - 1)) {

passengers[counter] = (queueArray[dequeuedFront]);

x = 0;

}

else {

passengers[counter] = (queueArray[dequeuedFront]);

x++;

}

counter++;

}

return passengers;

}

// Check if the array is empty

public boolean isEmpty() {

if(front == rear)

return true;

return false;

}

// Check if array is full

public boolean isFull() {

if((rear + 1) % queueArray.length == front)

return true;

return false;

}

} // end class

package CircularQueueProject;

import java.util.Scanner;

public class main {

public static void main(String[] args) {

final int MAX\_NUMBER\_IN\_QUEUE = 3;

int userChoice = 0;

String name = "";

String[] passenger = new String[MAX\_NUMBER\_IN\_QUEUE + 1];

Scanner keyboard = new Scanner(System.in);

CircularQueue\_BOOKED passBookings = new CircularQueue\_BOOKED(MAX\_NUMBER\_IN\_QUEUE);

CircularQueue\_WAITING passWaiting = new CircularQueue\_WAITING(MAX\_NUMBER\_IN\_QUEUE);

do

{

System.out.println("Menu");

System.out.println("========");

System.out.println("1. Add Passenger");

System.out.println("2. Delete Passenger");

System.out.println("3. Show Passengers");

System.out.println("4. Exit");

System.out.print("Enter choice: ");

userChoice = keyboard.nextInt();

switch(userChoice) {

// Add a passenger

case 1:

if(!passBookings.isFull()) {

System.out.print("Enter name: ");

name = keyboard.next();

passBookings.enqueue(name);

}

else if(passBookings.isFull() && (!passWaiting.isFull())) {

System.out.print("Enter name: ");

name = keyboard.next();

passWaiting.enqueue(name);

System.out.println("Sorry. Plane fully booked. Adding " + name + " to waiting list");

} else

System.out.println("Sorry. Plane and waiting list fully booked. Try later");

break;

// Delete a passenger

case 2:

if(passBookings.isFull() && !passWaiting.isEmpty()) {

passBookings.dequeue();

passenger = passWaiting.front();

passBookings.enqueue(passenger[0]);

passWaiting.dequeue();

System.out.println("Adding " + passenger[0] + " from waiting list");

}

else if(!passBookings.isEmpty() && passWaiting.isEmpty())

passBookings.dequeue();

else

System.out.println("No passengers to delete");

break;

// Show passengers

case 3:

if(!passBookings.isEmpty() && passWaiting.isEmpty()) {

System.out.println("Booked Passengers");

System.out.println("=================");

passenger = passBookings.front();

for(int x = 0; x < MAX\_NUMBER\_IN\_QUEUE + 1; x++) {

if(passenger[x] == null)

continue;

else

System.out.println(passenger[x]);

}

System.out.println("No passengers on waiting list");

}

else if(!passBookings.isEmpty() && !passWaiting.isEmpty() ){

System.out.println("Booked Passengers");

System.out.println("=================");

passenger = passBookings.front();

for(int x = 0; x < MAX\_NUMBER\_IN\_QUEUE + 1; x++) {

if(passenger[x] == null)

continue;

else

System.out.println(passenger[x]);

}

System.out.println("Waiting list");

System.out.println("=================");

passenger = passWaiting.front();

for(int x = 0; x < MAX\_NUMBER\_IN\_QUEUE + 1; x++) {

if(passenger[x] == null)

continue;

else

System.out.println(passenger[x]);

}

}

else

System.out.println("No passengers");

break;

// End of program

case 4:

System.exit(0);

default:

System.out.println("Invalid entry");

break;

} // Switch Statement

} while(userChoice != 4);

} // end main

} // end class



