AirPort Simulation:

Concepts and Requirements

write a program that simulates the operation of a busy airport that has only 2 runways to handle all takeoffs and landings. you may assume that each takeoff or landing *takes 15 minutes to complete. one runway request is made during each five-minute interval, and the likelihood of a* landing *request is the same as for a takeoff request. priority is given to planes requesting a* landing*. if a request cannot be honored, it is added to a takeoff or a* landingqueue*. your program should simulate 120 minutes of activity at the airport. each request for runway clearance should be time-stamped and added tto the appropriate* queue*. the outputs of your program should include the final* queue *contents, the number of landings and takeoffs completed, and the average number of minutes spent in each* queue*.*

Create a class called request storing all the data on the flight ie number time in queue etc
Create two queue collections [queue<dataType> = new LinkedList();]

Use that concept to build your program around.
I like the idea of this program, sounds good.

Airport Simulation: Part 1

Design and code a class called Airplane, and code a simulation of a small single-runway airport. Airplanes waiting to take off join a queue on the ground. Planes waiting to land join a queue in the air. Only one plane can use the runway at a time. All planes in the air must land before any plane may take off. At each time step, your simulation should "decide" whether to generate a new plane. If a new plane is generated, your simulation must "decide" whether that plane is taking off or landing. These "decisions" should appear to be random, yet you should seek to avoid ridiculous

scenarios. Your simulation should not have planes waiting a long time to take off, for example. Think carefully about how to avoid such things. Your simulation should run for what is its equivalent of one day. For example, if you decide to make each time step represent five minutes, then your simulation should run for 288 time steps. You may design and code a class called Clock, if you like.

Your simulation should make it clear to the user what is happening

at each time step. Below is some sample output to use as a guide.

The time is 12:05 PM.

There are 7 planes waiting to land.

There are 4 planes waiting to take off. Plane #23 is cleared to land.

Press <Enter> to continue.

The time is 12:10 PM.

There are 6 planes waiting to land.

There are 4 planes waiting to take off.

Plane #26 is cleared to takeoff.

Press <Enter> to continue.

Airport Simulation: Part 2

There is a small busy airport with only one runway. In each unit of time one plane can land or one plane can take off, but not both. Planes arrive ready to land or to take off at random times, so at any given unit of time, the runway may be idle or a plane may be landing or taking off. There may be several planes waiting either to land or to take off. Follow the steps given below to design the program.

1. Create two queues one for the planes landing and the other for planes taking off.
2. Get the maximum number of units for which the simulation program would run.
3. Get the expected number of planes arriving in one unit and number of planes ready to take off in one unit .

4. To display the statistical data concerning the simulation, declare following data members.
a. idletime - to store the number of units the runway was idle
b. landwait - to store total waiting time required for planes landed
c. nland - to store number of planes landed
d. nplanes - to store number of planes processed
e. nrefuse - to store number of planes refused to land on airport
f. ntakeoff - to store number of planes taken off
g. takeoffwait - to store total waiting time taken for take off
Initialize the queue used for the plane landing and for the take off
Get the data for , and from the user.

The process of simulation would run for many units of time, hence run a loop in main( ) that would run from towhere would be 1 and would be the maximum number of units the program has to be run. Generate a random number. Depending on the value of random number generated, perform following tasks.

1. If the random number is less than or equal to 1 then get data for the plane ready to land. Check whether or not the queue for landing of planes is full. If the queue is full then refuse the plane to

land. If the queue is not empty then add the data to the queue maintained for planes landing.

2. If the random number generated is zero, then generate a random number again. Check if this number is less than or equal to 1. If it is , then get data for the plane ready to take off. Check whether or not the queue for taking a plane off is full. If the queue is full then refuse the plane to take off otherwise add the data to the queue maintained for planes taking off.

3. It is better to keep a plane waiting on the ground than in the air, hence allow a plane to take off only, if there are no planes waiting to land.

4. After receiving a request from new plane to land or take off, check the queue of planes waiting to land, and only if the landing queue is empty, allow a plane to take off.

5. If the queue for planes landing is not empty then remove the data of plane in the queue else run the procedure to land the plane.

6. Similarly, if the queue for planes taking off is not empty then remove the data of plane in the queue else run the procedure to take off the plane.

7. If both the queues are empty then the runway would be idle.

8. Finally, display the statistical data As given below.

Total number of planes processed
Number of planes landed :
Number of planes taken off :
Number of planes refused use :
Number of planes left ready to land :
Number of planes left ready to take off :
Percentage of time the runway was idle :
Average wait time to land :
Average wait time to take off :

Airport Simulation: Part 3

In this project, you are asked to simulate airplane landing and takeoff at an airport. We will consider a small airport with only one runway. In each unit of time, one plane can land or one plane can takeoff, but not both. Planes arrive ready to takeoff or land at random times, so at any given moment of time, the runway may be idle or a plane may be landing or taking off, and there may be several planes waiting either to land or takeoff.

In simulating the airport, it will be useful to create a class Plane whose objects represent individual planes. This class will definitely need an initialization method and methods to represent takeoff and landing. You will also need to use a class Runway to hold information about the state and operation of the runway. This class will maintain members representing queues of planes waiting to land and takeoff.

In our simulation, we shall be especially concerned with the amounts of time that planes need to wait in queues before taking off or landing. Therefore, the measurement of time will be of utmost importance to our program. We shall divide the time period of our simulation into units in such a way that just one plane can use the runway, either to land or takeoff, in any given unit of time. The precise details of how we handle the landing and takeoff queues will be dealt with when you program the Runway class. Similarly, the precise methods of describing the operation of a Plane are not needed in the main program.

A key step in our simulation is to decide, at each time unit, how many new planes become ready to land and take off. Although there are many ways in which these decisions can be made, one of the most interesting and useful is to make a random decision. When the program is run repeatedly with random decisions, the results will differ from run to run, and with sufficient experimentation, the simulation may display a range of behavior not unlike that of the actual system being studied.

The Runway class needs to maintain two queues of planes, which we shall call landing and takeoff, to hold waiting planes. It is better to keep a plane waiting in the ground than in the air, so a small airport allows a plane to take off only if there are no planes waiting to land. Hence, our Runway method activity, which controls access to the Runway, will first service the head of the Queue of planes waiting to land, and only if the landing Queue is empty will it allow a Plane to take off. The aim of the simulation is to gather data about likely airport use. You are to use the class Runway itself to keep statistics such as the number of planes processed, the average time spent waiting, and the number of planes (if any) refused service. These details should be reflected in the various data members of the following Runway class definition.

The Plane class needs to maintain data about particular Plane objects. This data must include a flight number, a time of arrival at the airport system, and a Plane status as either arriving or departing.

The user must supply the number of time intervals the simulation is to run, the expected number of planes arriving, the expected number of planes departing per time interval, and the maximum allowed size for runway queues. The program should perform a random simulation of the airport, showing the status of the runaway at each time interval, and prints out a summary of airport operation at the conclusion.