//Program restaurant simulation

package simulationtaverna;

import javax.swing.\*;

import java.io.\*;

import java.util.\*;

class Table {

public int groupSize;

public int departureTime;

}

//Simulation class

public class SimulationTaverna {

//Change only the bellow - make sure no negatives will result from your choices

//Everything is integer for convenience

static final int HOURS = 6;

static int MIN\_NUMBER\_OF\_TABLES = 6, MAX\_NUMBER\_OF\_TABLES = 15;

static int ARRIVAL\_INTERVAL\_MEAN = 10, ARRIVAL\_STD = 2;

static int GROUP\_SIZE\_MEAN = 4, GROUP\_SIZE\_STD = 1;

static int STAY\_TIME\_MEAN = 90, STAY\_TIME\_STD = 15;

//Don't change the bellow

static int MAX\_MINUTES = HOURS\*60;

static int ARRIVAL\_INTERVAL\_MIN = ARRIVAL\_INTERVAL\_MEAN - 3\*ARRIVAL\_STD,

ARRIVAL\_RANGE = 6\*ARRIVAL\_STD;

static int GROUP\_SIZE\_MIN = GROUP\_SIZE\_MEAN - 3\*GROUP\_SIZE\_STD,

GROUP\_SIZE\_RANGE = 6\*GROUP\_SIZE\_STD;

static int STAY\_TIME\_MIN = STAY\_TIME\_MEAN - 3\*STAY\_TIME\_STD,

STAY\_TIME\_RANGE = 6\*STAY\_TIME\_STD;

static Random r = new Random(Calendar.getInstance().getTimeInMillis());

//Produce a random number from a Gaussian distribution

//between 3 standard deviations around the mean

static double getNormalRandom() {

double x1, x2, w;

do {

x1 = 2\*r.nextFloat() - 1;

x2 = 2\*r.nextFloat() - 1;

w = x1\*x1+x2\*x2;

x1 \*= Math.sqrt(-2\*Math.log(w)/w);;

} while (w >= 1 || Math.abs(x1) >= 3);

return x1;

}

//Produce a random number for arrival time

static int getArrivalTime() {

return (int)((getNormalRandom()+3)\*ARRIVAL\_RANGE/6+ARRIVAL\_INTERVAL\_MIN);

}

//Produce a random number for Group Size

static int getGroupSize() {

return (int)((getNormalRandom()+3)\*GROUP\_SIZE\_RANGE/6+GROUP\_SIZE\_MIN);

}

//Produce a random number for Departure Time

static int getDepartureTime() {

return (int)((getNormalRandom()+3)\*STAY\_TIME\_RANGE/6+STAY\_TIME\_MIN);

}

//----------------------------------------------------------------------------------------------------------------

static void runSimulation(int nTables) {

int arrivalTime,groupSize, customersServed = 0, customersNotServed = 0, iTable;

Table[] tables = new Table[nTables];

//Initialize tables

for (iTable=0; iTable<nTables; iTable++) tables[iTable] = new Table();

arrivalTime = getArrivalTime();

for (int time = 0; time < MAX\_MINUTES; time++){

//Check if anyone leaves at this time

for (iTable=0; iTable<nTables; iTable++)

if (tables[iTable].departureTime == time) {

tables[iTable].groupSize = 0;

tables[iTable].departureTime = 0; }

//Check next arrival

if (time == arrivalTime) {

//Retrive group size

groupSize = getGroupSize();

//Check if there is an available table

for (iTable=0; iTable<nTables; iTable++)

if (tables[iTable].groupSize == 0) {

tables[iTable].groupSize = groupSize;

tables[iTable].departureTime = time+getDepartureTime();

break; }

if (iTable < nTables) customersServed += groupSize;

else customersNotServed += groupSize;

//Get new arrival time

arrivalTime = time+getArrivalTime();

}

}

System.out.printf(" %2d %3d %3d\n", nTables, customersServed, customersNotServed);

}

//----------------------------------------------------------------------------------------------------------------

public static void main(String[] args) {

//Run simulation for various table sizes

for (int nTables = MIN\_NUMBER\_OF\_TABLES; nTables <= MAX\_NUMBER\_OF\_TABLES; nTables++) runSimulation(nTables);

}

}