CSC103 Spring 2016
Sara Wexler

Assignment #4

Due date: **(Wednesday April 20 week 11)**

This assignment is to be done by a group of two persons

- **Do programming project 9 from chapter 7 on page 407**
- Create a **RunwaySimulation** class**,** where the Car Wash Simulation described on pages 353 - 365 provides a model upon which to base your program.
- In addition to the output specified in the book, your program should also:
  1. Show for each minute simulated how the runway is being used if the runway is (i.e. take-off, landing, crash landing or idle.)- look at output example
  2. After the simulated time period is over show a list of times at which planes crashed. This list is to be in **reverse** order of the crash occurrences (Hint: keep track of crash times using a stack.) .

You are to make use of the following components in constructing your program:

- the **BooleanSource** and **Averager** classes used in the Car Wash Simulation- with no changes!
  http://www.cs.colorado.edu/~main/edu/colorado/simulations/

- a **Runway class** modeled on the Washer class used in the Car Wash. The same Runway class should be used for takeoff and landing. Complete the class given below. Landing has a higher priority then take off. There is no need to set a priority queue, just pop from the landing queue before you pop from the take off queue.

- a Queue Class from the book **: LinkedQueue.java**
  a Stack Class from the book**: LinkedStack.java**
  a Node class from the book: **Node.java**
  All classes are on the M drive and in:
  http://www.cs.colorado.edu/~main/edu/colorado/collections

- create a **printTimeUnit** method in the RunwaySimulation class to print each TimeUnit of the simulation

- create another print method, to print all other statistics about the simulation

MAKE SURE TO HAVE :

- **CLEAR AND GOOD DOCUMENTATION**
- **CREATE A CLEAR OUTPUT**

The stack and queue store objects of the following class **Plane**.

```
Class Runway{
        private  int timeForLanding;
        private  int timeForTakeoff;
        private int runwayTimeLeft;
        private char operation;           // operation can be: I – Idle, L-Landing, T-takeoff

        public Runnway ( int time_takeofff, int time_landing){….}
        //set the time for landing, time for takeoff, and the operation to idle.

        public boolean isBusy() {…}

        public reduceRemainingTime(){…}

        public void startUsingRunway(char typeOfUse){…..}
        // if typeOfUse is 'T' - then the operation is take off  and set the runway time left
        // to the time it takes for take off.
        // if typeOfUse is 'L' - then the operation is landing and set the runway time left
        // to the time it takes for landing
        // if typrofUse is ‘I’ – then the runway is idle, set the runway time left to zero

        public char kindOf Operation() {…}
        // returns the type of operation the runway is used for.
        // returns  'L' if the runway is used for is landing.
        // returns  'T' if  the runway is used for taking off.
        // returns ‘I, if the runway is idle


class Plane {
        static private int  planeCount = 0;      // the plane number arrived to the queue
                                                // should be in incrementing order
        private int time;                       // the time the plane arrived to the queue
        private char operation;                 // the kind of operation the plane is doing 'L"
                                                // is  for landing 'T' is for taking off
       private int planeNo;                      // plane number

        public Plane( int aTime, char landingOrTakeOff)
        // operation  is the type of operation the plane is doing. If landingOrTakeOff is 'L' // it
        means the plane is landing, if landingOrTakeOff is 'T' it means the plane is
        // Takingoff.
        {
                time = aTime;
                 operation =  landingOrTakeOff;
                planeNo = ++planeCount;
        }
        public int getTime()
                {return time;}
        static public int getPlaneNo ()
                { return planeNo;}
        public char getOperation ()
                { return operation;}

       private static int getPlaneCount()
        {
                    return planeCount;
         }


    }
```

Example of output:

```
The time of simulation is:    30 minutes
The amount of time that is needed for one plane to take off is:    2 minutes
The amount of time that is needed for one plane to land is :        3 minutes
The average amount of time between arrival of planes to the takeoff queue is    7 minutes
The average amount of time between arrival of panes to the landing queue is    5 minutes
The maximum time a plane can stay in the landing queue before crashing is     9 minutes

No of  planes that came to the runway  for takeoff               :           6
No. of  plans that came to the runway for landing                :           8
No of  planes that crashed                                       :           3
The average time a plane spent on the takeoff queue is           :          15 min
The average time a plane spent on the landing queue is           :          8 min
```

<u>Description of use of the runway:</u>

```
min 1 :
          Arrived for Takeoff  :  Plane  #1
          Arrived for Landing :
           Runway : Plane #1  is taking off

min 2 :
          Arrived for Takeoff  :
          Arrived for Landing :
           Runway : Plane #1  is taking off  ( takeoff finished)




min 3 :
```

```
            Arrived for Takeoff  :
            Arrived for Landing :
             Runway :  Idle



min 4 :
            Arrived for Takeoff  :
            Arrived for Landing :      Plane #2
             Runway :  Plane #2  is landing

 min 5 :
            Arrived for Takeoff  :     Plane #3
            Arrived for Landing :      Plane #4
             Runway :  Plane #2  is landing

min 6 :
            Arrived for Takeoff  :
            Arrived for Landing :      Plane #5
             Runway :  Plane #2  is landing ( landing finished)

:
```

crashing planes:

plane 10 crashed at minute 25
plane 5 crashed at minute 20
plane 3 crashed at minute 15

Turned In :

- Use **closeable envelope / folder marked clearly outside with names, class section and assignment #**
- All source code written for the assignment and the javadoc document;
- **Highlight** all new code in the DoubleLinkedSeq class
- **Write your names at the top of the source code** and **the date**
- Hard copy of test cases run
- Disk with all files relevant to the Assignment, **including executable**.

Grading criteria

5 - if the project is working good, get correct output, and there are sufficient comments
4 - if the project is working good, but there are some problems with the code, or some of the output is incorrect or the comments are not sufficient
2 - if the code compile but does not run
1 - if the does not compile

Simulation Program: Write a simulation program for a small airport that has only one runway. There will be a queue of planes waiting to land and a queue of planes waiting to take off. However, only one plane can use the runway at a time, so there can be only one takeoff or one landing in progress at any one time. Assume that all takeoffs take the same amount of time. Assume that all landings take the same amount of time, but this need not be the same as the takeoff time. Assume that planes arrive for landing at random times but with a specified probability of a plane arrive during any given minute. Similarly, assume that planes arrive at the takeoff queue at random times but with a (possible different) specified probability of a departure. (Despite the fact that the takeoff and landings are scheduled, delays make this a reasonable assumption.) Since it is more expensive and more dangerous to keep a plane waiting to land than it is to keep a plane waiting to take off, landings will have priority over takeoffs. Thus, as long as some plane is waiting to land, no plane can take off. Use a clock that is an integer variable that counts the number of minutes simulated. Use a random number generator to simulate arrival and departure times of airplanes.

This simulation can be used, among other things for deciding when the air traffic has become so heavy that a second runway must be built. Hence, the simulation will simulate conditions that would be a disaster in which planes crash because they run out of fuel while waiting too long in the landing queue. By examining the simulated situation, the airport authority hopes to avoid real tragedy. Assume all planes can remain in the queue for the same amount of time before they run out of fuel. If a plane runs out of fuel, your simulation will not discover this until the simulated plane is removed from the queue; at that point, the fact that the plane crashed is recorded, the plane is discarded, and the next plane is processed. A crashed plane is not considered in the calculation of waiting time. At the end of the simulated time, the landing queue is examined to see whether any of the planes in the simulated queue have crashed. You can disregard the planes left in the queue at the end of the simulation, other than those that crashed for lack of sufficient fuel. Use the following input and output specifications:

Input: (1) The amount of time needed for one plane to land; (2) the amount of time needed for one plane to take off; (3) the average amount of time between arrival of planes to the landing queue; (4) the average amount of time between arrival of planes to the takeoff queue; (5) the maximum amount of time that a plane can stay in the landing queue without running out of fuel and crashing; (6) the total length of time to be simulated.

Output: (1) The number of planes that took off in the simulated time; (2) the number of planes that landed in the simulated time; (3) the number of planes that crashed because they ran out of fuel before they could land; (4) the average time that a plane spend in the takeoff queue; (5) the average time that a plane spend in the landing queue.