HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

BBM203 PROGRAMMING LAB.
ASSIGNMENT 3

Subject　　　　　　　　　: Data Structures (Queue)
Programming Language : JAVA
Submission Date　　　　: 12.11.2014
Deadline　　　　　　　　: 26.11.2014 – 23:59 pm
Advisors　　　　　　　　: Asst. Prof. Dr. Mustafa EGE, Asst. Prof. Dr. Lale ÖZKAHYA, R.A. Gültekin IŞIK

1. INTRODUCTION/AIM

In this experiment, you are required to **design** and **implement** a JAVA program that generates queues according to the commands read from an input file.

2. EXPERIMENT

In this experiment, you are required to **design** and **implement** a JAVA program that generates queues according to the commands read from an input file.
Your application will simulate an airport landing and takeoff system. This airport will have three runways, **runway 0**, **runway 1** and **runway 2**. Each runway can handle one takeoff or landing at each time slot. **runway 0** and **runway 1** are used only for landing. **runway 2** is used only for takeoff except emergency landings (when plane runs out of fuel). There are 3 queues, two queues for landing (one for **runaway 0**, one for **runaway 1**) and the other for takeoff. Arriving plane, if it has enough fuel, will be ready for takeoff.

  ➢ **Arriving plane record**
Each plane record is stored in system with its;
  • Identification number (**id**) (your application must give unique integer **id** number for each arriving plane)
  • Name (**name**) (you read from input file)
  • Remaining fuel (**fuel**) (you read from input file)

  ➢ **Departing plane record**
Each plane record is stored in system with its;
  • Identification number (**id**) (your application must give unique integer **id** number for each departing plane)
  • Name (**name**)

2.1 EXECUTION

Your application gets some parameters from command line. These parameters are described below:

AirportSim <inputfile> <outputfile> <executiontime> <fuelconsump> <landingtime> <takeofftime> <queuetype>

  • **AirportSim** is your application's name.

  • **<inputfile>** is your input file name.

  • **<outputfile>** is your output file name.

  • **<executiontime>** describes how long your simulation will run.

  • **<fuelconsump>** describes plane's fuel consumption for each time slot.

  • **<landingtime>** describes each planes' landing time.

  • **<takefftime>** describes each planes' takeoff time.
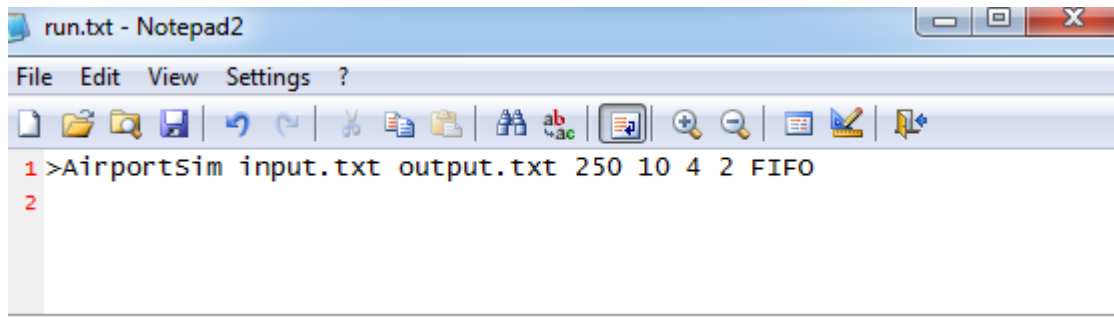
- **<queuetype>** describes type of queue that is used for simulation.
    - If **<queuetype>** is **FIFO**. You will use FIFO queue for your simulation.
    - If **<queuetype>** is **Priority**. You will use priority queue for your simulation.

Note 1: Plane that has less remaining fuel is prior for the **Priority** queue.

Note 2: No fuel consumption during waiting time at ground.

Note 3: If runway 0 and runway 1 are empty in the same time, arriving plane will land to the runway 0.

Running Example :

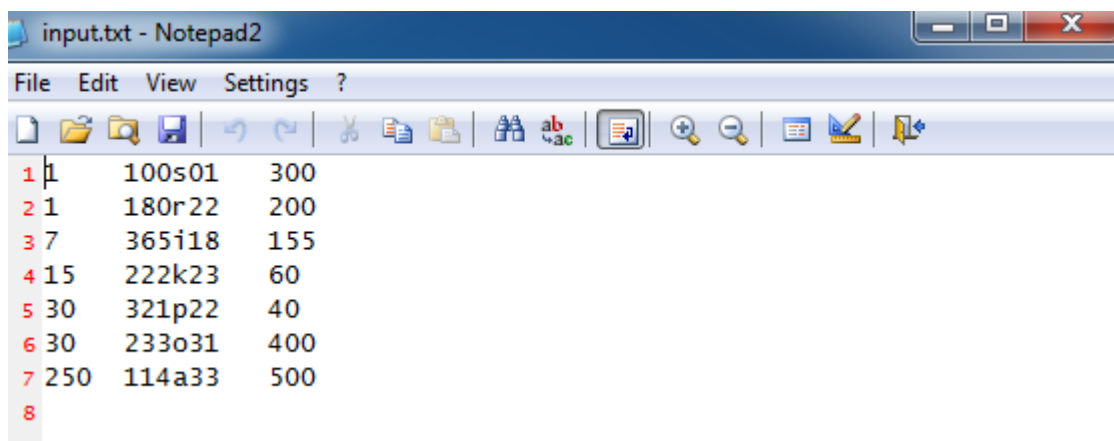## 2.2 INPUT/OUTPUT FORMAT

2.2.1 INPUT FILE FORMAT

The commands of input file are described below:

<time> <plane's name> <fuel>

<time> is plane's arriving time to the simulation.

Input File Example:

2.2.2 OUTPUT FILE FORMAT

When the commands of input file are read and processed successfully, program must write related information to the output file in the following format.

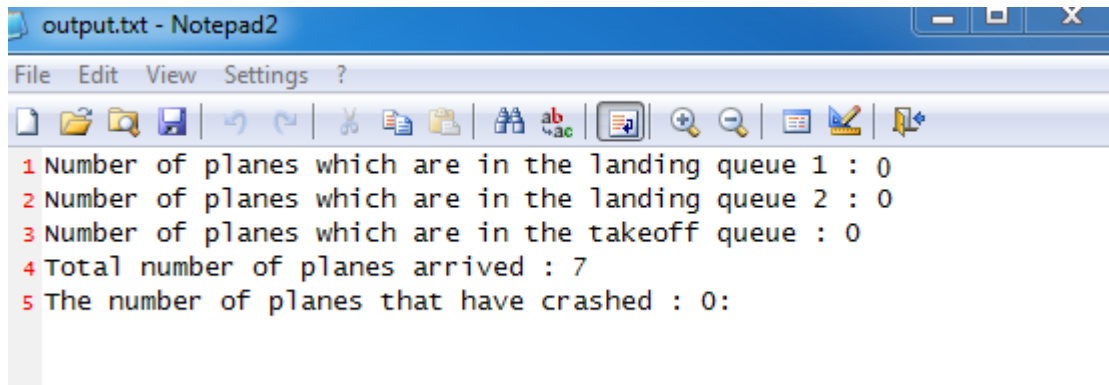**Number of planes which are in the landing queue 1 : <number_of_landing_queue_1 >**

Number of planes which are in the landing queue 2 : <number_of_landing_queue_2 >

Number of planes which are in the takeoff queue : <number_of_takeoff_queue >

Total number of planes arrived : <number_of_planes>

The number of planes that have crashed : <number_of_crashed_planes>:<planename1>,<planename2>,…

Output File Example :


```
1 Number of planes which are in the landing queue 1 : 0
2 Number of planes which are in the landing queue 2 : 0
3 Number of planes which are in the takeoff queue : 0
4 Total number of planes arrived : 7
5 The number of planes that have crashed : 0:
```

3. EVALUATION

3.1 REQUIRED FILES

You should create and submit a ZIP archive in the following structure for evaluation. An invalid structured archive will cause you partial or full score loss.

Directory	Files	Description
Source	Main.java, *.java, etc	Program source files
Report	report.pdf	Your report (Only pdf format is accepted)

3.2 REPORTS

You must write a report which is related to your program. (Report Format: ftp://ftp.cs.hacettepe.edu.tr/pub/dersler/genel/FormatForLabReports.doc).

LAST REMARKS:

- You can not use **Packages.**
- Give your report in necessary detail.
- Your algorithm, data structures and other things which you will use in application must be given in report.
- The output of your program will be graded automatically. Therefore, any difference of the output (even a smallest difference) from the sample output will cause an error and you will get 0 from execution. Keep in mind that a program that does not work 100% right is a program that works wrong.
- Regardless of the length, use **UNDERSTANDABLE** names to your variables, classes and functions.
- You should obey the JAVA Naming Conventions.
- Write **READABLE SOURCE CODE** block.
- You will use online submission system to submit your experiments. **https://submit.cs.hacettepe.edu.tr** Deadline is: 23:59 pm. No other submission method (such as diskette, CD or email) will be accepted.
- Do not submit any file via e-mail related with this assignment.

- **SAVE** all your work until the assignment is graded.
- The assignment must be original, **INDIVIDUAL** work. Duplicate or very similar assignments are both going to be punished. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms or source codes.
- You can ask your questions through course's web page and you are supposed to be aware of everything discussed in the newsgroup: **https://dersler.cs.hacettepe.edu.tr.**