CS 202 Lab/Programming Assignment 4
Due: Before the Class on March 25, 2015

This program is worth **150 points**. You are to implement a **queue application** discussed in **§8.5** (**pages 434-444**) of your Data Abstraction book.

An important application of the FIFO queue structures is the simulation of a situation in which one must wait in line for some service. Some real-life examples are waiting at a bank, post office, or supermarket checkout. In operating systems, users competing for the CPU of a multi-user systems must also wait for his/her turn. Computer simulation can provide us with valuable information, such as the average wait time, so we can improve the service. In this assignment, you will write a program that uses FIFO queue(s) to simulate the traffic at a bank.

The problem to be solved by your program is a simplified version of a problem given at the High School Programming Contest in 1993. The problem states as follows:

WNE is a small but busy bank with **only one** teller. (The original problem has 3 tellers, and we simplify it to have only 1 teller.) A customer entering the bank will go to a waiting line. If the teller is free, the customer will proceed to the teller. If the teller is busy, the customer has to wait until he/she moves up to the front of the line and the teller is free. Your program should calculate and display all activities of the bank and computes the average waiting time of all customers.

You should run your program on **Lab4Data.txt** which is available on Kodiak. The first line of the input is a positive integer n, $1 \le n \le 30$, indicating the number of customers. Each of the following n lines contains the time that a new customer arrives at the bank and the time required for his/her transaction. The arrival time is in the form HH:MM XM, where HH is the hour, MM is the minute, and XM is either AM or PM. Note that HH and MM contains exactly two digits. So, if the hour or the minute is less than 10, it will have a leading zero. The transaction time is an integer representing the number of minutes. The input is arranged in ascending order of arrival time. If two or more customers enter the bank at the same time, their order in the waiting line will be determined by the order they appear in the input data. Your program should print five values for each customer: the customer number (1, 2, 3, ...) , the time the customer enters the bank, the time the customer gets service, the time the customer completes the transaction, and the time (in minutes) that he/she has to wait to get called by the cashier. All times must be printed in the form of HH:MM XM as described above for input data. Print an appropriate heading, then print each customer on a separate line. Finally, print the average waiting time in minutes, with two decimal places.

You may assume that all customers will complete their transactions on the same day before the bank closes.

To calculate and display time properly, you will first need to write a class that represents the time (**do not use the Java's Time class**):

```
class Time
{
     private int     hour;
     private int     minute;
     private boolean isMorning;

     public Time()
     {
          // constructor to initialize the time to default value
          ...
     }

     public Time(int, h, int m, boolean morning)
     { ... }

     public void setTime(int, h, int m, boolean morning)
     { ... }

     public int getHour()
     { ... }

     public int getMinute()
     { ... }

     public boolean isAM()
     { ... }

     public String toString()
     {  // return the time string in the format of HH:MM XM
      ... }

     public Time plus(int n)
     {  // create and return a Time object which is n minutes
        // from this time
      ... }

     public int timeDifference(Time t)
     {  // how many minutes (a nonnegative integer) between Time t
        // and this time?
      ... }

     public int compareTo(Time t)
     {  // return an integer that is < 0, = 0 or > 0 depending on
        // whether this time is < (earlier than), = or > (later than)
        // the parameter Time t
      ... }

     private int totalMinutes()
     {  // return an integer that is total amount of minutes since
        // 12:00 am of this time.
        // This method is called internally by timeDifference
        // and compareTo.
      ... }
}
```

We also need a class to hold the information about each customer.

```
class Customer
{
      int  ID;
      Time arriveTime;
      Time serviceTime;    // the time that the customer get served
      Time completeTime;   // the time that the transaction completes
      int  transaction;    // the amount of time (in minutes) required
                           // for the transaction
}
```

Sample Input

```
11
11:20 AM 8
11:22 AM 15
11:25 AM 3
11:25 AM 4
11:30 AM 20
11:40 AM 30
11:42 AM 26
12:00 PM 7
01:35 PM 145
02:05 PM 65
03:00 PM 1
```

Sample Output

```
Customer       Arrival       Service       Completing      Time
 Number         Time          Time           Time         Waited
------------------------------------------------------------------
   1          11:20 AM      11:20 AM      11:28 AM          0
   2          11:22 AM      11:28 AM      11:43 AM          6
   3          11:25 AM      11:43 AM      11:46 AM         18
   4          11:25 AM      11:46 AM      11:50 AM         21
   5          11:30 AM      11:50 AM      12:10 PM         20
   6          11:40 AM      12:10 PM      12:40 PM         30
   7          11:42 AM      12:40 PM      01:06 PM         58
   8          12:00 PM      01:06 PM      01:13 PM         66
   9          01:35 PM      01:35 PM      04:00 PM          0
  10          03:55 PM      04:00 PM      05:05 PM          5
  11          03:56 PM      05:05 PM      05:06 PM         69

The average waiting time =  26.64 minutes
```

Submit your program source code **.java** to Kodiak. Be sure to **comment your code** and put **your name** and a **brief description** at the beginning of your program.

Hand in the **output** and printout of source code **.java files**. Staple the output on top of Lab 4 source code printout. On the output, be sure to put **your name**, and **``Output from Lab4''** on top of the printout.

Note that in order to print the average time in two decimal digits, you may use the **C**-style **printf** function, **format** method, or the Formatter class. Here is a program that shows you various formats you can use to print an integer and a real number using printf:

```
//  FormatTester.java
//  This is a test program for printf.

public class FormatTester
{
    public static void main(String[] a)
    {
        double x = 15.123456789012;
        int    k = 123;

        System.out.println(
                    "x = 15.123456789012 printed in various formats:\n");
        System.out.printf
               ("    printed in 'f' format      = %f\n\n", x);
        System.out.printf
               ("    printed in '20f' format    = %20f\n\n", x);
        System.out.printf
               ("    printed in '20.3f' format  = %20.3f\n\n", x);
        System.out.printf
               ("    printed in '.3f' format    = %.3f\n\n", x);
        System.out.printf
               ("    printed in '20.15f' format = %20.15f\n\n", x);
        System.out.printf
               ("    printed in '20.20f' format = %20.20f\n\n", x);
        System.out.printf
               ("    printed in '20.12f' format = %20.12f\n\n", x);
        System.out.printf
               ("    printed in '.12f' format   = %.12f\n\n", x);
        System.out.println
               ("\nk = 123 printed in various formats:\n");
        System.out.printf
               ("    printed in 'd' format    = %d\n\n", k);
        System.out.printf
               ("    printed in '2d' format   = %2d\n\n", k);
        System.out.printf
               ("    printed in '5d' format   = %5d\n\n", k);
        System.out.printf
               ("    printed in '05d' format  = %05d\n\n", k);
    }
}
```

Here is the output from the program:

```
x = 15.123456789012 printed in various formats:
    printed in 'f' format      = 15.123457
    printed in '20f' format    =            15.123457
    printed in '20.3f' format  =               15.123
    printed in '.3f' format    = 15.123
    printed in '20.15f' format =   15.123456789012000
    printed in '20.20f' format = 15.12345678901200000000
    printed in '20.12f' format =      15.123456789012
    printed in '.12f' format   = 15.123456789012

k = 123 printed in various formats:
    printed in 'd' format    = 123
    printed in '2d' format   = 123
    printed in '5d' format   =   123
    printed in '05d' format  = 00123
```

What other data structures do you need and how do you plan to solve this problem?

(**Hint**: To solve the problem, use the **event-driven simulation as described on pages 434-444 of your Data Abstraction book.**)