CS1101C
AY1998/9 Semester 1 Practical Examination
(Question modified to Java setting.)

Grading an NOI Task – Dominoes.

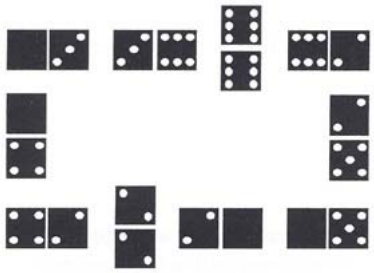
The first National Olympiad in Informatics (NOI) (for secondary school and junior college students) was organized by the School of Computing and held earlier this year. The following is the simplified version of one of the tasks in the final round of the competition. This is **not** your PE question.

Dominoes are small tiles traditionally carved from ivory or bone with small, round pips of inset ebony. The number of pips on a half-tile is in the range 0 through 6 inclusively. For example, the tile 2-4 (same as tile 4-2) is shown below.

You will be given a set of tiles, like the following:

0-2, 0-3, 4-0, 4-2, 2-5, 2-6, 2-2, 0-5, 6-6, 6-3

Your task is to arrange the tiles in a ring so that dominoes touch at matching numbers like the following. Assume that the given tiles can be arranged in a ring, and note that there can be more than one arrangement.



Your input file, called "**domino.in**", will be formatted as follows. On the first line will be the number of tiles, *N*, which is at most 100. Next, the tiles are listed one per line, where the tile with numbers *x* and *y* is represented as "*x y*". See the sample "domino.in" file below.

Your output file, "**domino.ans**", should have the following format. Output the tiles one after another so that the second number on each tile matches the first number on the next one (and the second number on the last tile matches the first number on the first tile). See the sample "**domino.ans**" file below.

Sample "domino.in"

```
10
0 2
0 3
4 0
4 2
2 5
2 6
2 2
0 5
6 6
6 3
```

Sample "domino.ans"

```
0 3
3 6
6 6
6 2
2 5
5 0
0 2
2 2
2 4
4 0
```

Your PE task begins here.

Now, let's turn back the clock and imagine that you are part of the team entrusted with the jobs of ensuring the smooth running of the event. Just hours before the grading is to begin, to everybody's horror, the hard disk that contained the grading program crashed, and the chief programmer who wrote the grading program is being confined at home, down with chicken-pox. You are the only hope. With just less than two hours to go, you are to write a grading program that would read the **domino.in** file, and the contestant's **domino.ans** file, and determine if the contestant has got her answer right. This is a real test of your programming skills; everybody's eyes are on you, and you cannot afford to fail.

Wiping the sweat off your forehead, you know you must act fast, but not hastily. You remember the lessons learned in your programming class – design comes before coding.

1. You are to define a class **Tile** that contains two integer data members: **left** and **right**.

   You decide that the data in the two files should be read into two arrays with elements of type **Tile**, one array to store the dominoes in the **domino.in** file, and the other to store the dominoes in the **domino.ans** file. Write the methods **readInfile** and **readAnsfile** to do the above.

2. You determine that the contestant's answer could be checked in stages. In the first stage, if the number of dominoes in **domino.ans** is different from the number of dominoes in **domino.in**, the contestant surely must have got it wrong.

3. Next, if the contestant's dominoes are not arranged in the proper order, that is, the second number of a tile does not match the first number of the next tile, then the contestant must have got it wrong too. To do this, you need to write a method **isAdjacent(Tile[] tile)** that returns true if the dominoes are arranged in a ring, or false otherwise.

4. Finally, in the last and the most elaborate stage, if the contestant's dominoes are different from those given in the file **domino.in**, then the contestant's answer must be wrong. Write appropriate methods to perform this task.
5. Call the above methods you have written in an appropriate manner in your main method. The following self-explanatory messages are to appear on the screen depending on the outcome of your checking.

```
Contestant's answer is correct!
Wrong! The number of tiles do not tally!
Wrong! Tiles cannot form a ring!
Wrong! Tiles are different from those in input file!
```

Case 1: Number of tiles do not tally if contestant's answer is shown in box 1 below.

Case 2: Tiles cannot form a ring if contestant's answer is shown in box 2 below.

Case 3: Tiles are different from those in input file if contestant's answer is shown in box 3 below.

```
0 3
3 6
6 6
6 2
2 5
5 0
0 2
2 2
2 4
```

Box 1

```
0 3
6 3
6 6
6 2
2 5
5 0
0 2
2 2
2 4
4 0
```

Box 2

```
0 3
3 6
6 6
6 2
2 5
5 0
0 3
3 2
2 4
4 0
```

Box 3

(If you want to hear real stories about NOI, talk to me. You may visit the NOI's website at http://www.comp.nus.edu.sg/~noi/)

After PE exercise:

For those who like challenges, solve the actual NOI problem. ☺