Data Structures and Introduction to Algorithms

Due: 9/18/2015 at noon (12:00pm)

"With this boy? Why, he is a common labouring-boy!"
I thought I overheard Miss Havisham answer—only it seemed so unlikely -
"Well? You can break his heart."
"What do you play, boy?" asked Estella of myself, with the greatest disdain.
"Nothing but beggar my neighbour, miss."
"Beggar him," said Miss Havisham to Estella. So we sat down to cards.
...
I played the game to an end with Estella, and she beggared me. She threw the cards down on the table when she had won them all, as if she despised them for having been won of me.

– Great Expectations by Charles Dickens.

1 Assignment 2: Beggar My Neighbour

1.1 Objectives

- *To get comfortable working with Linked Lists.*
- *To appreciate great 19th century literature.*

In Charles Dickens's classic *Great Expectations*, Pip and Estella play a card game called Beggar My Neighbor, a variant of War. You may know a more modern variant of this game called Egyptian Ratscrew, which involves slapping. (There will be no slapping in this assignment). Your task will be

to write a program that takes the order of cards in the deck as the input and outputs who will win the game.

1.2 The Rules of Beggar My Neighbour

The following rules are adapted from `https://en.wikipedia.org/wiki/Beggar-My-Neighbour`. A standard 52-card deck is divided equally between two players, and the two stacks of cards are placed on the table face down. The first player lays down his top card face up, and the opponent plays his top card, also face up, on it, and this goes on alternately as long as no ace or face card (King, Queen, or Jack) appears. These cards are called "penalty cards."

If either player turns up such a card, her opponent has to pay a penalty: four cards for an ace, three for a King, two for a Queen, or one for a Jack. When he has done so, the player of the penalty card wins the hand, takes all the cards in the pile and places them under her pack. The game continues in the same fashion, the winner having the advantage of placing the first card. However, if the second player turns up another ace or face card in the course of paying to the original penalty card, his payment ceases and the first player must pay to this new card. This changing of penalization can continue indefinitely. The hand is lost by the player who, in playing their penalty, turns up neither an ace nor a face card. Then, his opponent acquires all of the cards in the pile. When a single player has all of the cards in the deck in his stack, she has won. A player loses the game if he turns up his last card and that card is not a penalty card.

There are a couple Youtube videos explaining the rules and showing people playing: `https://www.youtube.com/watch?v=D9K8PehoCck` and `https://www.youtube.com/watch?v=iMTMXRaICvE`. They're not great, but may help you get the idea if the above description was insufficient.

1.3 Program Specification

Write a java program that reads from the standard input stream (System.in) the order of cards in a deck and outputs who will win as well as the number of rounds played. It is assumed that there are two players, Player 0 and Player 1. So a possible output is

```
Player 1 wins after 112 rounds.
```

After 5791 rounds, you should output:

```
Draw after 5791 rounds.
```

Incidentally, 5790 is the longest known game on a standard deck. If you find a longer one, make sure you have stored it, and **let me know**. However, we will work with some examples that do not use a standard deck.

The input is a list of strings corresponding to the cards in the deck. Because the suits do not matter, we will represent the cards as

```
2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 9 9 9 9 10
10 10 10 J J J J Q Q Q Q K K K K A A A A
```

We will not assume a standard 52-card deck. Cards are to be dealt out one at a time, alternating between player 0 and player 1 (starting with player 0), until all are dealt out.

Player 0 plays first.

For example, with an input like `J 2 3 J 4 5`, player 0 would have a hand `J 3 4` and Player 1 would have a hand `2 J 5`. Incidentally, this game would go on forever, so the output would be

```
Draw after 5791 rounds.
```

(Can you see why?)

For an even simpler example, consider the input `J 2 J 2`. In this case the output should be

```
Player 0 wins after 2 rounds.
```

You must implement and use your own linked list class. For this project, using any of the classes in the Java collections framework is not allowed.

1.4 Tips

The java.util.Scanner class, reviewed in Section 1.6 of the textbook, provides a simple way to parse the input. In the submission system, the input stream is redirected from a file. When testing the program on your computer, the standard input is normally associated with the Java console, and you will need to type control-Z on Windows or control-D on Unix at the console window to signal that the end of the input stream has been reached.

1.5 Submission Filename and Formatting

Your main file should be in a class called `BeggarMyNeighbour` in a file called `BeggarMyNeighbour.java`. Note the British spelling of "Neighbour". (It's a British game after all.)

We will be using the Mimir system to submit the assignment.

`https://app.mimirplatform.io`

You will have to submit your project as a `.zip` file. If you have never zipped a file before, this is a great chance for you to learn. **Please only submit the relevant java files. This means that you should not be exporting binaries, *.classpath*, or *.project* files.**

1.6 Other Resources

Timothy Gowers is a brilliant mathematician who wrote a nice blog entry on the question of bounding the expected number of rounds in a game. You can read it if you are interested: `https://gowers.wordpress.com/2008/04/05/open-problems-concerning-card-games/`.