**Project Overview**

The penguins of Antarctica hold elections every twenty years. They use an unusual voting process where if there are *n* candidates in an electorate, each voter has *n* votes to distribute amongst them in any pattern they want. A voting paper is *formal* iff it contains exactly *n* marks (corresponding to the *n* candidates) each in the range *0..n* inclusive and adding up to no more than *n*. So for an election with four candidates, all of the following would be formal papers:

4,0,0,0

2,0,0,2

0,0,3,0

0,2,1,0

1,1,1,1

Obviously many other formal votes are possible. All of the following would be informal papers:

3,0,2,0       - total too high

3,0,2,-1     - negative votes not permitted

3,0,0         - too few marks

3,0,0,0,0    - too many marks

To determine the winner of the election, first of all the informal papers are discarded, then two ***separate*** totals are calculated for each candidate.

* The number of votes for a candidate is the sum of the votes given to that candidate on each formal paper. So given the nine papers above, the total vote for the first candidate would be 4+2+0+0+1 = 7. The total votes for the other candidates would be 0+0+0+2+1=3, 0+0+3+1+1=5, 0+2+0+0+1=3 respectively.
* The number of wins for a candidate is the number of formal papers on which they receive the highest number of votes, each one divided by the number of candidates who receive that highest number. So given the nine papers above, the total wins for the first candidate would be 1.0+0.5+0.0+0.0+0.25 = 1.75. The total wins for the other candidates would be 0.0+0.0+0.0+1.0+0.25=1.25, 0.0+0.0+1.0+0.0+0.25=1.25, 0.0+0.5+0.0+0.0+0.25=0.75 respectively.

The winner of the election is the candidate who has the highest number of votes; if there is a tie, the winner is the candidate amongst those tied who has the highest number of wins. So given the nine papers above, the first candidate would be the winner.

You are required to write a program that reads in a file of candidates' names and voting papers, conducts the election count, and determines the winner of the election.

**Reading in files of election info**

In this project, candidates' names and the voting papers will be provided in a text file. Have a look at *election1.txt* as an example. Every election file will have

* the candidates' names, one per line, followed by
* one blank line which separates the candidates from the votes and which should be discarded, followed by
* the voting papers, one per line.

You can read in these files very easily using the Java class FileIOwhich is provided. Simply declare a variable belonging to this class, and create an object with the name of the election file as the argument to the constructor. You can then access the contents of the file as separate lines using the appropriate accessor method. The project FileIOexample on the *Lecture Material* page shows a simple example of this in use.

**Project Materials**

Download the folder *Project1.zip*. This folder contains the following.

1. Skeletons for three Java classes.
   * VotingPaper represents one voting paper.
   * Candidate represents one candidate in the election.
   * Election represents the entire election process.

Collectively, these classes implement the election process. You are required to complete the constructors and methods for all three classes. Where the body of a constructor or method contains a comment TODO, delete the body and replace it with code which implements the required functionality, as described in the associated program comment. The numbers 1-23 in these comments suggest an order for you to perform these tasks. (Don't worry, many of these methods are very simple!)

1. A fully-written class FileIO, which will help you to perform file input operations. You do not need to change or submit this class.
2. Three JUnit test classes, which are provided for you to check your code. Note that the test cases are not complete; a method that gets all green ticks is not guaranteed to be completely correct. Additional test cases will be used for marking and it is your responsibility to thoroughly test your code.
3. Three sample input files *election\*.txt*, and three associated output files *election\*.out.txt*.

**Project Management Tips**

Hints and tips about the various methods may be uploaded here from time to time. Whenever that happens, the document version number will be updated.

1. For getStanding, there's a helpful method in the library class Math for rounding doubles.
2. In processFile, each line of the file is either a candidate's name or a voting paper, except the blank line that divides the two sections. So when you are looking at a particular line, how do you know whether it is a name or a paper?
3. For isCorrectLength, isLegalTotal, and anyNegativeMarks; each of these should care only about its own functionality, ***not*** about wider issues of whether the paper is formal. e.g. if marks is <3,0,-2>, the first two might still return true, even though the paper is clearly informal because of the -2.
4. processFile is causing some consternation. The important job here is to find out where is the first blank line in the file, then every line before that blank is a candidate's name, and every line after that blank is a voting paper.
5. Also with processFile: remember that candidates' names can contain digits too...