CMSC B113 - Computer Science 1
Fall 2020
Homework Assignment #2

Overview

In this assignment, you are asked to take on the role of a political campaign consultant who has been hired by a candidate to help them get a sense of how likely they are to win an upcoming election. To do this, you will write two Java programs that perform calculations and simulations related to the election.

For each program, first read the problem, then devise a solution. Clearly identify all the inputs and outputs. Write down the steps and calculations that will need to be performed. Code the steps into a Java program. Create, compile and run the program and test it on several inputs. Double-check the program's answers with the ones we have provided to confirm that the results are correct.

Note that in this assignment, you will also be asked to write up a brief report regarding your findings and observations. Be sure to allocate time for doing that write-up, in addition to programming and testing your solution.

Learning Outcomes

In completing this assignment, you will learn to:

- Use if-else statements in a Java program
- Use for-loops in a Java program
- Write code that performs data validation on runtime arguments

Part 1: Election Results

Your first task is to write a small program that determines the victor in an election and displays the winner's name and percentage of votes received.

Assume that three candidates ran for office in a constituency, and their votes were tallied. Write a program that takes the votes tally and prints out the following: The winner, total number of votes cast, and percentage of votes cast for the winner. For example:

```
$ java-introcs ElectionResults Nitisha 1450 Sarah 1783 Anna 1220
Sarah wins the election receiving 1783 of a total of 4453 votes.
Overall 40.040422187289465% votes were cast for Sarah.
Congratulations to Sarah!
```

Your program should only print results **only if the input data is valid** and should display an error message if the number of votes cast for a candidate is negative. For example,

```
$ java-introcs ElectionResults Alyssa 24782 Faith 16349 Judy -42167
Invalid data. The number of votes cannot be negative.
```

You may assume, though, that there are exactly three candidates and that all six runtime arguments are provided and are of the expected types, i.e. Strings for the candidates' names and integers for the number of votes. Additionally, you may assume that the candidates all receive distinct numbers of votes and that there are no ties.

Part 2: Estimating Probability of Victory

Your next task is to estimate your candidate's probability of victory in the election by considering the number of voters and how likely they are to vote for your candidate.

In this case, we'll assume that there are only two candidates, but there are three groups of voters:

- **BASE** refers to voters in your political party's base who are likely to vote for your candidate
- **OPP** refers to voters in your opponent's party's base who are likely to vote for your opponent
- **SWING** refers to voters who are not affiliated with either party

The number of voters in each group, and the likelihood of voting for your candidate, are as shown in the table below:

Group	Number of voters	Likelihood of voting for your candidate
BASE	1,200,000	70-80%
OPP	1,300,000	25-40%
SWING	2,500,000	35-60%

Using these numbers, write a Java program called **Victory** that calculates the probability that your candidate will win the election by simulating the outcome for the different possible values of the BASE, OPP, and SWING likelihoods of voting for your candidate.

That is, your program should consider all possible combinations of the likelihoods that the BASE, OPP, and SWING voters vote for your candidate, and for each combination, determine who the winner would be.

The probability that your candidate wins the election is the number of those combinations that result in victories for your candidate, divided by the total number of combinations.

To approach this:

- First, determine whether your candidate would win the election if 70% of the BASE voters, 25% of the OPP voters, and 35% of the SWING voters voted for your candidate. You can calculate the number of votes your candidate receives from each group by multiplying the number of voters times the likelihood, e.g. in this case your candidate would receive 0.70 * 1,200,000 = 840,000 votes from the BASE voters, and so on.

- Then determine whether your candidate would win the election if 70% of the BASE voters, 25% of the OPP voters, and **36%** of the SWING voters voted for your candidate
- Then determine whether your candidate would win the election if 70% of the BASE voters, 25% of the OPP voters, and **37%** of the SWING voters voted for your candidate
- And so on, up to 60% of the SWING voters
- Then determine whether your candidate would win the election if 70% of the BASE voters, **26%** of the OPP voters, and 35% of the SWING voters voted for your candidate
- And so on, until you have covered all combinations of values for the BASE, OPP, and SWING voters' likelihood of voting for your candidate
- Then, after doing so, print the likelihood that your candidate wins the election to the terminal using System.out.println.

To implement this program, you will need to use nested loops to iterate over all possible combinations of BASE, OPP, and SWING likelihoods; calculate the number of votes each candidate receives for the given combination; determine the winner of that election simulation; and keep track of the number of elections won by each candidate.

The following "pseudocode" is meant to give you an idea of how to structure your Java code. Some details have intentionally been omitted; you will need to determine how to "fill in the blanks" when implementing this in Java:

```
baseVoters = 1200000
oppVoters = 1300000
swingVoters = 2500000
for each value of baseLikelihood from 70 to 80
    for each value of oppLikelihood from 25 to 40
        for each value of swingLikelihood from 35 to 60
            baseVotes = baseVoters * baseLikelihood / 100
            oppVotes = …
            swingVotes = …

            ourVotes = baseVotes + oppVotes + swingVotes
            theirVotes = …

            if ourVotes > theirVotes then increment wins
            else if ourVotes < theirVotes then increment losses
            else increment ties

probability = wins / (wins + losses + ties) * 100
```

You may assume that the likelihood of voting for your candidate is always an integer, e.g. for BASE likelihood you need consider 70, 71, 72, … 79, 80 but do not need to consider 70.1, 70.2, etc.

Also, you should assume that all voters actually vote, and that there are only two candidates, i.e. that if a voter does not vote for your candidate, then they vote for your opponent.

If correctly implemented, and using the numbers provided above, your program should simulate a total of 4576 elections, and your candidate should win 2364 and lose 2206, with six ties, for an overall probability of 51.66%. **Do not continue with the rest of the assignment until your program produces this correct output!**

Once your program is producing the correct output, modify Victory.java so that it takes the name of the state and the number of BASE, OPP, and SWING voters as runtime arguments (in that order), performs the same simulation described above, and prints the win probability for that state.

For instance, using the data provided above, it should be possible to run the program like this:

```
$ java-introcs Victory Keystone 1200000 1300000 2500000
Keystone: 51.66083916083915%
```

As in Part 1, you may assume that all four runtime arguments are provided and are of the correct type, i.e. a String for the state name and integers for the numbers of voters, but should display an error message if the number of specified voters is negative:

```
$ java-introcs Victory Sunshine 2400000 -1750000 3800000
Invalid data. The number of voters cannot be negative.
```

Once you have modified the program, you can use it to predict outcomes in different states with different voter profiles. Assume the likelihood of voting applies universally to all states, i.e., in any given state your candidate is likely to receive 70-80% of votes from their base, etc. Determine the output of your program for the following states:

STATE	BASE voters	OPP voters	SWING voters
Garden	3,250,000	3,150,000	5,400,000
Empire	5,925,000	3,200,000	9,550,000
Nutmeg	1,450,000	1,200,000	2,300,000
Golden	11,250,000	10,800,000	3,200,000
Lone Star	3,500,000	7,250,000	6,400,000

Examine the results closely and write a short (1-2 paragraphs) report interpreting the data obtained that will be submitted to the campaign committee. In particular, your report should answer the following questions:

- What is the likelihood of victory for your candidate in each of the five states listed above?
- Assuming that the overall winner of the election is determined by the number of states won, in which state should your candidate focus on converting SWING voters to BASE voters?

Submitting Your Solutions

To submit your work, copy/paste the source code of your Java programs, as well as the report to the campaign committee from Part 2, into a *single* PDF, and submit the PDF in the “Assignment #2” subfolder of the Dropbox folder that your instructor created for you. You only need to submit your final version of Victory.java from Part 2.

Your submission should include sample inputs and outputs for all programs demonstrating that your programs run and produce the correct results.

To create the PDF, you can put the code into Microsoft Word and export/print it as a PDF, or put it into a Google Doc and then do File → Download → PDF.

Please be sure to put all of your code into a *single* document and please only submit a PDF, not a Microsoft Word .docx file or the .java source files.