OoL #3: Simulation Refinement

For this Out of Lab, we'll be focusing on algorithms and alternate ways of doing things. I'll be posting a possible OoL #2 solution to Moodle next week for you to examine. You may use that possible solution OR your own OoL #2 solution as a starting point for OoL #3. The specification for OoL #3 will be broken into 7 parts: Update, Recursion, Sorting, Analysis, Simple AI, Analysis II.

YOU MAY WORK IN GROUP UP TO SIZE 4: Each group member MUST be the PRIMARY author (Note primary authorship in BOTH comments **AND** TLDDR) for AT LEAST 1 of these code sections:

-- Bipartisanship Update Section

-- Manual Sort Implementation subsection

-- Comparable/Comparator Implementation subsection

-- Multiple Authorship Update subsection

UPDATE SECTION:

- Multiple Authorship Update:
    - Allow bills to be authored by more than one author.
    - If all authors are part of the same party:
        1. Same party voting odds should increase by 60%. Examples:
            - 2 same party authors. .6 + ((1 - .6) *.6) = .84
            - 3 same party authors. .6 + ((1 - .6) *.6) + ((1 -((1 - .6) * .6)) *.6) = .936
        2. Different party voting odds should decrease by 10%. Examples:
            - 2 different party authors. .1 - (.1 *.1) = .09
            - 3 different party authors. .1 - (.1 *.1) – (.1 * .1 * .1) = .089
        3. Same party and different party odds should be calculated by taking the weighted average of the same party and different party calculations. Examples:
            - 2 same party and 2 different party
                - 2 same party authors. .6 + ((1 - .6) *.6) = .84
                - 2 different party authors. .1 - (.1 *.1) = .09
                - (.84 * 2) + (.09 * 2) / 4 = .465
            - 3 same party and 2 different party
                - 3 same party authors. .6 + ((1 - .6) *.6) + ((1 -((1 - .6) * .6)) *.6) = .936
                - 2 different party authors. .1 - (.1 *.1) = .09
                - (.936 * 3) + (.09 * 2) / 5 = .5976
        4. You **MUST** use **recursion** to calculate the above sets of adjusted odds
- Bipartisanship Update:
    - Need a variable to denote the approval rating of the government as a whole

- If the bill pass rate (# of bills passed at any level / # of votes called at any level) at the end of a day is < 1% or > 10%, approval rating should drop 1 percentage point
- If the bill pass rate is between 1% (inclusive) and 10% (inclusive), approval rating should increase by 1 percentage point
- If approval rating is above 65%:
    1. President's Party is 50% less likely to vote for different party bills
    2. Non-President's Parties are 50% more likely to vote for President's party bills
- If approval rating is below 45%:
    1. Presidents Party is 50% more likely to vote for different party bills
    2. Non-President's Parties are 50% less likely to vote for President's party Bills

- Summary (the output after the simulation is completed) Update [to be done IN ADDITION TO the summary data from OoL #2]:
    - Print "Average Approval Rating"
    - Print "Max Approval Rating"
    - Print "Minimum Approval Rating"
    - Print out data on number of days in HIGH approval rating (> 65%)
    - Print out data on number of days in LOW approval rating (< 45%)
    - Print out to 3 files:
        1. Bills in the LawBook
        2. Bills in the Trash
        3. Bills that were still in process (not yet made into Laws or Trashed)

SORTING SECTION: 2 Versions of this code will be made for OoL #3

You will need a new program to analyze your 3 output files. This program should allow the user to select information to print to the screen. Note, you may find that you need to add extra fields to the Bill class in order to make this work.

- All Bills in Chronological Order (Creation date)
- All Bills in Chronological Order (Day they were signed into Law)
- All Bills in Chronological Order (Day they were Trashed)
- All Bills of <Allow User to select a party> in Chronological Order (Creation Date)
- All Bills of <Allow User to select a party> in Chronological Order (Day they were signed into Law)
- All Bills of <Allow User to select a party> in Chronological Order (Day they were Trashed)
- All Bills authored by representatives from <Allow User to select a state> sorted by author in alphabetical order (multiple authored Bills should sort on first author)
- All Bills authored by representatives from <Allow User to select a state> sorted in Chronological Order (Creation Date) and then by author in alphabetical order (multiple authored Bills should sort on first author) to break ties
- All Bills authored by representatives from <Allow User to select a state> sorted in Chronological Order (Day they were signed into Law) and then by author in alphabetical order (multiple authored Bills should sort on first author) to break ties

- All Bills authored by representatives from <Allow User to select a state> sorted in Chronological Order (Day they were Trashed) and then by author in alphabetical order (multiple authored Bills should sort on first author) to break ties
- All Bills authored by <Allow User to select specific congress person> in order of Passed into Law → Still in Process → Trashed. In the case of ties, then sort sub groups by Chronological Order (Creation Date)
- Total number of Laws
- Total number of Trashed Bills
- Total number of Bills still in the system.
- Percentage of Bills that were still in the system
- Percentage of Bills which were Trashed
- Percentage of Bills which were made into Laws

Comparable/Comparator Implementation:

- All sorts for this program MUST be done using Java's built-in sorting (Arrays.sort(), Collections.sort(), etc….) using either/both compareTo() methods and/or Comparators

Manual Sort Implementation:

- All sorting must be done via manually implemented sorts. You must use at least 2 of the sorts we've talked about in class (Quick, Bogo, Bubble, Insert, or Selection) to do this.

ANALYSIS SECTION: Run your code enough times with the same input files to a get picture for how things play out. Think about and answer the following questions in your TLDDR. Make sure to include your thoughts on WHY those answers are what they are. (Include your number of runs and final result data which led you to make your claims as text files in your jar):

- What tends to be the approval rating of the Government at the end of the simulation?
- Does approval rate at the end tend to be same as the starting approval rate? Explain.
- About what percentage of Bills make it to each level? (Drafted, Committee, Floor, 2 Committees, 2 Floors, Signed, Vetoed, Overridden)

SIMPLE AI SECTION: Based on what you learned in the Analysis Section (and any other thoughts you might have), attempt to add new code to your program to try and meet each of the below goals. You may wish to target 1 goal at a time to decrease the number of variables being manipulated at once. You may adjust/add percentage thresholds, create new methods (Example: Maybe a method for backRoomDeal() to make deals for votes?), etc….

- Approval Rating stays HIGH most of the time
- Approval Rating stays in the middle and self-corrects if it gets HIGH or LOW
- Approval Rating stays LOW most of the time

- President's party passes most of their bills (where President's Party has majority in both legislative units)
- President's party passes most of their bills (where President's Party has majority in only 1 legislative unit)
- President's party passes most of their bills (where President's Party has majority in zero legislative units)

ANALYSIS II SECTION: Run your code enough times to get a feel for how your changes helped/hindered the goalthe current rules work and answer the below questions (include your number of runs and final result data which led you to make your claims as text files in your jar). NOTE, you MUST repeat this section and the Simple AI section at least 2 times. You may repeat as many additional times as you want. Just keep adding sections in your TLDDR as you go. I'm not expecting you to get to "perfect" data (whatever that means), I'm just interested in your logic, discussion and implementation towards the specific goals. In other words, I'm focusing more on your thought process and ability to write code which is CONSISTENT with your thought process. As you go, include in your TLDDR:

- What code changes you tried
- Why you tried those code changes
- What happened (outcome of the changes)
- What each outcome indicated

Grading:

• Analysis Section	20 points
• Simple AI Section and Analysis II Section	35 points*
• Summary Update	10 points
• JavaDoc	10 points**
• Your Primary Author Section	25 points

*With quality repetition and discussion of Simple AI and Analysis II Sections above and beyond my required expectations, you could earn up to 10 points extra credit.

**The 10 points are for JavaDocing the NEW methods. If you still have non-JavaDoc'd methods/classes from existing code, I may deduct up to 20 points for that.