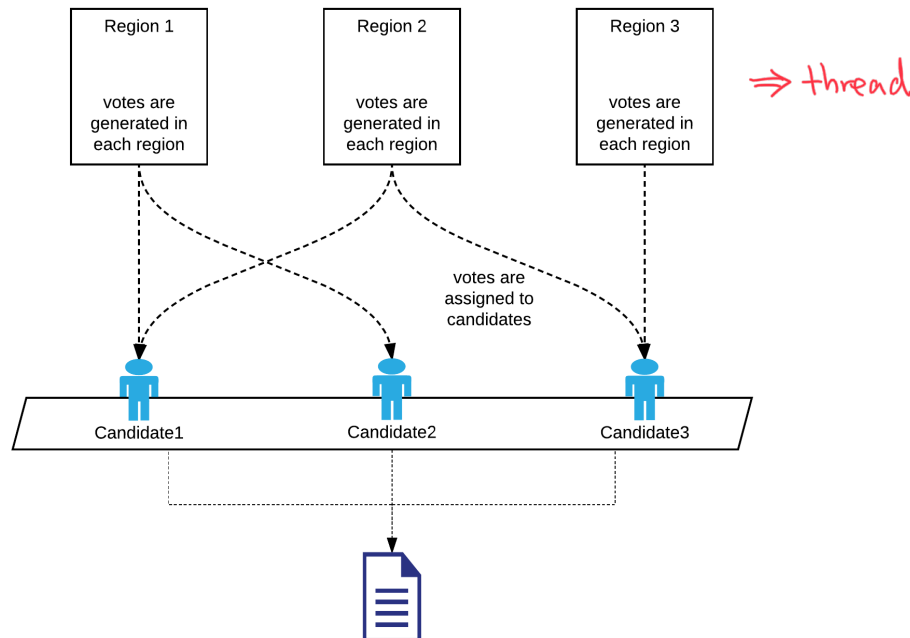


## Programming Assignment #3 Election Simulation

In this Assignment, we will simulate the results of a hypothetical election. Our elections simulator will allow us to “predict” the results of a very heated elections race.

Our elections simulation consists of Regions, Candidates, and Votes



1.

- Create a class named **Candidate** that will be used to store information about each candidate. The **Candidate** class will have three (3) instance variables.
  - **String name** : this is the name of the candidate.
  - **int numVotes**: this is the number of votes the candidate received.
  - **Votes[] votes**: this is an array of the votes a candidate received.
- Create a constructor that accepts **string name** and **int maxVotes**. The constructor should
  - set the name of the candidate
  - create an array of **Vote**. The size of the array is **maxVotes**.
- Implement **Comparable** on the **Candidate** class. Candidates are compared based on the **numVotes**.
- Create a **toString** method that returns a String in this form

```
-----Candidate-----
Name: John
Votes: 499
-----
```

- Create a class named **Vote**. The **Vote** class is written as an inner class of the **Candidate** class.
  - The class has one instance variable **int regionNum**.
  - Create a constructor for the **Vote** class that accepts a parameter **int regionNum**.
- Create an **addVote()** method in the candidate class. This method accepts a parameter **int regionNum**. It creates a new **Vote** object and adds the vote to the votes array of the candidate.

2.

- Create a class named **Region** that will be used to store information about each electoral region. The Region class is a **threaded** class. Therefore, you must implement **threading**. The Region class will have four (4) instance variables.
  - **String name** : this is the name of the region.
  - **int regionNum**: this is the number of the region.
  - **int population**: this is the population of the region.
  - **Candidate[] candidates**: this is an array of the candidates of the election.
- Create a constructor that accepts all the parameters and can create a Region object.
- Create a method **generateVotes()**. This method must do the following.
  - selects a number randomly between **0** and **number of Candidates**
  - calls the **addVote** method of the candidate object stored in the array at the random number location.
    - *{This simulates a vote from someone in that region.}*
- The Region thread should **run** the **generateVotes()** method.

3.

- Create a class named **ElectionSim** that will be used as the election simulator. This class must read input data from a text file, run the simulation and then write the output to another text file.
- The class has four (4) instance variables.
  - **string outputFile**: this is the path of the output file.
  - **int population**: this is the total number of votes.
  - **Candidates[] candidates**: this is the list of candidates.
  - **Region[] regions**: this is the list of regions.
- Create a **constructor** that accepts two (2) parameters
  - **String inputFile**: the path of the input file.
  - **String outputFile**: the path of the output file.
  - The body of the constructor **must** carry out the following tasks.
    - Set the outputFile instance variable
    - Read the input file and set the following
      - Set the population instance variable.
      - Create Candidate objects and add them to the **candidates** array.
      - Create Region objects and add them to the **regions** array.
- Create a method **saveData()**
  - This method must do the following
    - Sort the **candidates** array
    - Write the information in the **candidates** array to the output file.
- Create a method **runSimulation()**. This method will be used to start the simulation.
  - The method should call the start method on all the regions created.
  - **The method should wait until all threads end.** *{use an appropriate method of the thread class}*
  - The method should then call **saveData()** to save the simulation results to the output file.

## Sample Input

```
POPULATION 2500
CANDIDATES 8
Mickey
Mia
Anthony
Katy
Lewis
Ashley
Danny
Karen
REGIONS 3
Seoul 1 1500
Daegu 2 700
Daejeon 3 300
```

## Sample outputs

```
-----Candidate-----
Name: Anthony
Votes: 371

-----Candidate-----
Name: Katy
Votes: 369

-----Candidate-----
Name: Ashley
Votes: 367

-----Candidate-----
Name: Mia
Votes: 361

-----Candidate-----
Name: Danny
Votes: 340

-----Candidate-----
Name: Lewis
Votes: 333

-----Candidate-----
Name: Karen
Votes: 192

-----Candidate-----
Name: Mickey
Votes: 167
```

```
-----Candidate-----
Name: Katy
Votes: 394

-----Candidate-----
Name: Mia
Votes: 358

-----Candidate-----
Name: Lewis
Votes: 356

-----Candidate-----
Name: Anthony
Votes: 355

-----Candidate-----
Name: Danny
Votes: 347

-----Candidate-----
Name: Ashley
Votes: 333

-----Candidate-----
Name: Karen
Votes: 186

-----Candidate-----
Name: Mickey
Votes: 171
```

```
-----Candidate-----
Name: Danny
Votes: 378

-----Candidate-----
Name: Lewis
Votes: 372

-----Candidate-----
Name: Katy
Votes: 368

-----Candidate-----
Name: Mia
Votes: 358

-----Candidate-----
Name: Anthony
Votes: 348

-----Candidate-----
Name: Ashley
Votes: 335

-----Candidate-----
Name: Karen
Votes: 180

-----Candidate-----
Name: Mickey
Votes: 161
```

## Hints:

- The **addVote()** method of the candidate class is **"critical"**. This method should be *synchronized*.
- You may implement threading using **Runnable** or the **Thread** class
- Ensure the total number of votes for all candidates add up to the population
- Your inner class(es) should be private.
- This code can be used to test your work

```
public class simTest {

    private static final String INPUTFILE = "../../inputfile.txt";
    private static final String OUTPUTFILE = "../../outputfile.txt";

    public static void main(String[] args){

        ElectionSim eSim = new ElectionSim(INPUTFILE,OUTPUTFILE);
        eSim.runSimulation();

    }

}
```

## Submission :

You have to submit the source code **and** written documentation.

### Written Documentation

In the documentation, you should include a short description of the implementation methodology as well as an explanation of what you did. **Also, you should include print screen or screen shot or snapshot of your program's output in the document.**

You may include screenshots of the output of different input values that you chose.

### Online Submission

Submit your assignment's java code and document files via the **Assignment #3** on the LMS system.

### Submission deadline

Your submission is due at **midnight on June 10, 2022.**

- In case you do not meet the deadline,
  - 50% of your score will be deducted for a delay within 24 hours
  - 75% of your score will be deducted for a delay within 48 hours
  - 0 points will be given for a delay of more than 48 hours.
- This is an individual assignment. **Please do not copy another student's work.**