CSE 8B Homework 2:
Building and Testing Classes

Learning goals:

- Write code to build classes in Java including:
  - constructors
  - methods
- Write code that uses a `while` loop in Java
- Write test cases to test your methods
- Reflect on ethical considerations of developing and using election software

NOTE: Starting with this programming assignment, we will grade your coding style, so please follow this style guideline: CSE 8B Coding Style Guidelines. If you have any questions in regards to the style guidelines, then feel free to ask on Piazza.

Part 1: Create and test the Candidate class (Pair Programming allowed) (4 points, class and tests)

Create a class named `Candidate` in a file named `Candidate.java`.

This class should have the following *private* member variables:

- `String name`
- `String party`
- `int voteCount`

This class should contain the following *public* methods. You must name your methods *exactly* as specified below:

One constructor:

```
public Candidate(String candidateName, String candidateParty)
```

This constructor should initialize the member variable `name` to the value of the parameter `candidateName`, initialize the member variable `party` to the value of the parameter `candidateParty`, and initialize the member variable `voteCount` to 0.

Three getter methods:

```
public String getName()
```

This method returns the candidate's `name`.

```
public String getParty()
```

This method returns the candidate's `party`.

`public int getVotes()`
This method returns the number of votes the candidate currently has (`voteCount`).

Two setter methods:
`public void setParty(String newParty)`
This method sets the candidate's `party` to the value of the parameter `newParty`.

`public void incrementVotes()`
This method increments the number of votes the candidate currently has (`voteCount`) by 1.

IMPORTANT: You should compile your code often. For example, after writing a method, run `javac Candidate.java`. Furthermore, you will not be able to run your code until you write your tester below. We recommend writing (and running) your tester as you write your Candidate class.

Test your Candidate class

In the same folder as `Candidate.java`, create a *separate* class named `ElectionTester` in a file named `ElectionTester.java`. In this new class's `main` method, write code to test your Candidate class as follows:

1. Create two different `Candidate` objects with two different names and parties. The names and parties are up to you.
2. Print each candidate's `name`, `party`, and `voteCount` on its own line. Similar to our example below
3. Change the party of one candidate, and increment that candidate's votes twice.
4. Print that candidate's `name`, `party`, and `voteCount` again.

To compile and run:

1. Open up your command prompt or Terminal.
2. Change your current directory to the directory where the files `Candidate.java` and `ElectionTester.java` are located.
3. Compile both files together (Hint: you can list both filenames after `javac`, or you can just compile the file with the `main` method. Alternatively, you could run `javac *.java` to compile all Java files in your current directory.).
4. Run `ElectionTester` (Note: you cannot run your `Candidate` class as it has no `main` method) and verify that your `Candidate` class works as expected.
5. Fix any bugs and repeat as necessary.

Part 2: Create and run the Election class (Pair Programming allowed) (5 points, class and tests)

In the same folder as `Candidate.java` and `ElectionTester.java`, create the class `Election` in the file `Election.java`.

This class should have the following private member variables:

- `Candidate candidate1`
- `Candidate candidate2`
- `Candidate candidate3`

These will represent the three candidates in the election.

This class should also have the following methods:

Constructor:
`public Election()`
This constructor takes no arguments. It assigns each member variable to a new `Candidate` object. The name and party of each candidate is up to you, but they should all be different.

Other method:
`public void runElection(Scanner scan)`
This method, which takes a `Scanner` as an argument, runs the election via the command line. This method should behave as follows:

1. This method should print out the three candidates' names and parties in a numbered list. See the example below.
2. Then, this method should loop until the user enters -1. In each loop iteration, this method should:
    a. Prompt the user for their vote.
    b. Read in the user's vote and increment the appropriate candidate's vote count.
3. After the loop, this method should print the total votes earned by each candidate as well as the name of the candidate that won the election*. If there is a tie or no winner, then print an appropriate message.

Note: Scanner should be closed using the `close()` method after use.

***A note on calculating the winner:**
You can use any simple voting strategy you like. For example, you could use plurality voting (the candidate with the most votes wins) or majority voting (a candidate only wins if they get the majority of the votes, otherwise the top two candidates are selected for a runoff election. You do not need to implement the runoff, though you can if you want!.)

Below is an example that uses plurality voting. User's input is in bold.

```
Welcome to the election.  We have 3 candidates:
```

```
1. Heather Sanchez from the Sixth party
2. Sandra Hui from the Cool party
3. Jasper Jones from the Birthday party

Enter the next vote (by candidate number).  Enter -1 if there are
no more votes.
2
Enter the next vote (by candidate number).  Enter -1 if there are
no more votes.
1
Enter the next vote (by candidate number).  Enter -1 if there are
no more votes.
3
Enter the next vote (by candidate number).  Enter -1 if there are
no more votes.
1
Enter the next vote (by candidate number).  Enter -1 if there are
no more votes.
1
Enter the next vote (by candidate number).  Enter -1 if there are
no more votes.
2
Enter the next vote (by candidate number).  Enter -1 if there are
no more votes.
-1

The vote count is as follows:
Heather Sanchez: 3
Sandra Hui: 2
Jasper Jones: 1

The winner is Heather Sanchez!
```

Notes:

- **Your output does not have to exactly match ours. Feel free to get creative.**
- You can assume that the user's input will always be an integer and that the user's input will always be -1, 1, 2, or 3.
- If you want to implement a complex voting strategy (*e.g.* automatic runoff) for a star point, you are welcome to do so. You may add variables or even use arrays to assist with this implementation, but TA/tutors will not be able to provide support with strategies that go well beyond the basic approach - that's why it's a star point!

Test your Election class

1. In the main method of the `ElectionTester` class, *below* the tests of your Candidate class (do NOT erase any previous tests), add code that does the following:
    - Creates **two** new `Election` objects and stores them in two different variables.
    - Calls `runElection` once on each `Election` object.

2. Compile all three classes together.

3. Run the `ElectionTester` class. In your interaction, you must ensure that there are two different outcomes of the two different elections, and at least one should result in no single winner (either a tie or a runoff).

4. Finally, **take a screenshot** that clearly shows the complete results of running your program, **including the output of the Candidate tests from Part 1 and the two full elections.** You might need more than one screenshot. Paste your screenshot into a document that you can save as a PDF.

Part 3: Ethical issues related to election software (individual) (1 point)

This week, you developed software to simulate voting in an election. While your program was simple, electronic elections present many important real-world considerations. Identify one ethical consideration pertaining to developing and using software to support voting/elections. Describe the issue and why it is important to consider. Include at least one online source for your information. You will enter your answer and your source on gradescope.

Grading information

- Parts 1 and 2:
    - Code correctness (auto + manually graded): 6.5 points
    - Code style: 0.5 point
    - Tests: 2 points
- Part 3: 1 point

Submission Instructions

Please follow ALL of the instructions below to completely submit your assignment.

- Submit your code from Parts 1 and 2 (`Candidate.java`, `Election.java` and `ElectionTester.java`) on Gradescope using the "PA2 - Code" submission
    - When you submit to gradescope you will receive feedback about whether your code compiled correctly (with our tester code).

    - You will also see the output of our run of your runElection method with our input, and it's up to you to verify that this method worked correctly.
    - Finally, we will run additional tests of your Candidate and Election classes for grading, but we will not show you these tests or their results until after your assignment is graded and returned to you.
- Submit the PDF containing the screenshot(s) demonstrating your tests of Candidate and the functionality of your `runElection` method to Gradescope using the "PA2 - Testing" submission
- Enter the description of your ethical issue from Part 3 on Gradescope using the "PA2 - Part 3" submission