CSC120 2025S Lab No.10
Write-in Election

This lab aims to write a program, `Election`, that tallies votes in an election and finds the winner.

The votes are write-in, so there is no predetermined set of candidates. Whoever appears the most in the votes is the winner, where a tie can be broken arbitrarily. There are two ways to enter votes. One is to use the keyboard, where the user enters a name per line ending with CTRL-D. The other is to use a file where the votes appear one name per line. The program asks if the user wants to use a file to enter votes. If the answer starts with a `"y"`, the program receives a file name and instantiates a `File` object using the name as the path and then instantiates a `Scanner` object `in` with the file that has been chosen. To do this, the program uses a `boolean` variable `useFile` to specify whether or not a file will be used. Otherwise, the program instantiates `in` by simply assigning the `Scanner` object that the program is using for receiving the keyboard input and then presents the following instruction

```
##########################################
# Enter the votes, one vote per line.    #
##########################################
```

At the end of the main method, if the `useFile` is `true`, execute `in.close()` to close the `Scanner`.

The program tallies the votes using two arrays, a `String[]` variable named `names` and an `int[]` variable named`counts`. When receiving votes from the user, the two arrays will be the same length. Their lengths are equal to the number of unique names that have appeared in the votes so far. Thus, the initial length is 0 for both arrays.

To add a vote, we use three methods:

```
public static int find( String[] names, String name )
public static String[] addName( String[] names, String name )
public static int[] addNewCount( int[] counts )
```

The first one, `find`, scans the array, `names`, using a for-loop that iterates over all the valid index values `0 .. names. length - 1`. If the array has an entry equal to the second parameter, `name`, ignoring the case, the method returns the index at which the value equal to `name` has been found. If the for-loop completes without terminating, the method returns `-1` as the indication of not seeing `name`.

The second method, `addName`, creates a new `String` array from `names` by appending the value of `name` at the end and then returns the new array.

The third method, `addNewCount`, creates a new `int` array from `counts` by appending a new element of 1 at the end.

After necessary initialization (the two arrays and a `Scanner` object to receive input from the keyboard), the program prints a prompt to inform the user how to enter the votes.

After this, the program enters a while-loop to receive an indefinite number of votes from the user. The loop can be terminated in two possible ways. The condition for the while-loop should be set to terminate with CTRL-D; that is, the condition is:

```
while ( in.hasNext() )
```

In the body of this while-loop, the program receives input from `in` using `in.nextLine()` and stores it in a `String` variable, `name`. If `name` is equal to `""`, the loop terminates using `break`. If this does not occur, the method calls `find` to identify the position at which `name` occurs in `names`. If the returned value of `find` is nonnegative, the name already exists on the array, so the program increases the value of counts at the returned position by 1. Otherwise, the program executes `addName( names, name )` and `addNewCount( count )` to record that the name has appeared just once.

After terminating the loop, the program calls a method

```
public static void findWinner( names, counts )
```

that finds the winner based on the tally. The method `findWinner` first reports how many votes each person received, in the order of appearance in the array `names`, and then computes the winner. To compute the winner, it uses two `int` variables, `maxCount` and `theWinner`, whose initial values are 0. The method scans the array's contents using a for-loop with an iteration variable, `i`. During the scan, if `counts[ i ]` is greater than `maxCount`, `names[ i ]` is the winner for the moment, so the program updates `maxCount` with `counts[ i ]` and `theWinner` with `i`.

Here is an execution example of the code.

```
Use a file? y
Enter a file name: votes.txt
Draco received 20125 votes.
Dudley received 19883 votes.
Harry received 20135 votes.
Ron received 19799 votes.
Hermione received 20058 votes.
--------
The winner is Harry!
```

Here is another example.

```
Use a file? n
##########################################
# Enter the votes, one vote per line.    #
##########################################
Frodo
Sam
Pippin
Frodo
Frodo
```

```
Pippin
Pippin
Pippin
Sam
Sam
Pippin
Frodo
Frodo
Frodo
Sam
Pippin
Pippin
Pippin
Pippin
Sam
Sam
Pippin
Frodo received 6 votes.
Sam received 6 votes.
Pippin received 10 votes.
--------
The winner is Pippin!
```

The file `votes.txt` is also attached to this assignment.