Election Day

It's almost election day and the election officials need a program to help tally election results. There are two candidates for office—Polly Tichen and Ernest Orator. The program's job is to take as input the number of votes each candidate received in each voting precinct and find the total number of votes for each. The program should print out the final tally for each candidate—both the total number of votes each received and the percent of votes each received. Clearly a loop is needed. Each iteration of the loop is responsible for reading in the votes from a single precinct and updating the tallies. A skeleton of the program is in the file *Election.java*. Open a copy of the program in your text editor and do the following.

1. Add the code to control the loop. You may use either a while loop or a do...while loop. The loop must be controlled by asking the user whether or not there are more precincts to report (that is, more precincts whose votes need to be added in). The user should answer with the character y or n though your program should also allow uppercase repsonses. The variable *response* (type String) has already been declared.
2. Add the code to read in the votes for each candidate and find the total votes. Note that variables have already been declared for you to use. Print out the totals and the percentages after the loop.
3. Test your program to make sure it is correctly tallying the votes and finding the percentages AND that the loop control is correct (it goes when it should and stops when it should).
4. The election officials want more information. They want to know how many precincts each candidate carried (won). Add code to compute and print this. You need three new variables: one to count the number of precincts won by Polly, one to count the number won by Ernest, and one to count the number of ties. Test your program after adding this code.

Finding Maximum and Minimum Values

A common task that must be done in a loop is to find the maximum and minimum of a sequence of values. The file *Temps.java* contains a program that reads in a sequence of hourly temperature readings over a 24-hour period. You will be adding code to this program to find the maximum and minimum temperatures. Do the following:

1. Save the file to your directory, open it and see what's there. Note that a *for* loop is used since we need a count-controlled loop. Your first task is to add code to find the maximum temperature read in. In general to find the maximum of a sequence of values processed in a loop you need to do two things:

    - You need a variable that will keep track of the maximum of the values processed so far. This variable must be initialized before the loop. There are two standard techniques for initialization: one is to initialize the variable to some value *smaller* than any possible value being processed; another is to initialize the variable to the first value processed. In either case, after the first value is processed the maximum variable should contain the first value. For the temperature program declare a variable *maxTemp* to hold the maximum temperature. Initialize it to -1000 (a value less than any legitimate temperature).
    - The maximum variable must be updated each time through the loop. This is done by comparing the maximum to the current value being processed. If the current value is larger, then the current value is the new maximum. So, in the temperature program, add an if statement inside the loop to compare the current temperature read in to maxTemp. If the current temperature is larger set maxTemp to that temperature. NOTE: If the current temperature is NOT larger, DO NOTHING!

2. Add code to print out the maximum after the loop. Test your program to make sure it is correct. Be sure to test it on at least three scenarios: the first number read in is the maximum, the last number read in is the maximum, and the maximum occurs somewhere in the middle of the list. For testing purposes you may want to change the HOURS_PER_DAY variable to something smaller than 24 so you don't have to type in so many numbers!

3. Often we want to keep track of more than just the maximum. For example, if we are finding the maximum of a sequence of test grades we might want to know the name of the student with the maximum grade. Suppose for the temperatures we want to keep track of the time (hour) the maximum temperature occurred. To do this we need to save the current value of the *hour* variable when we update the *maxTemp* variable. This of course

requires a new variable to store the time (hour) that the maximum occurs. Declare *timeOfMax* (type int) to keep track of the time (hour) the maximum temperature occurred. Modify your *if* statment so that in addition to updating *maxTemp* you also save the value of *hour* in the *timeOfMax* variable. (WARNING: you are now doing TWO things when the *if* condition is TRUE.)

4. Add code to print out the time the maximum temperature occurred along with the maximum.

5. Finally, add code to find the minimum temperature and the time that temperature occurs. The idea is the same as for the maximum. NOTE: Use a separate *if* when updating the minimum temperature variable (that is, don't add an *else* clause to the *if* that is already there).

Counting Characters

The file *Count.java* contains the skeleton of a program to read in a string (a sentence or phrase) and count the number of blank spaces in the string. The program currently has the declarations and initializations and prints the results. All it needs is a loop to go through the string character by character and count (update the *countBlank* variable) the characters that are the blank space. Since we know how many characters there are (the *length* of the string) we use a count controlled loop—*for* loops are especially well-suited for this.

1. Add the *for* loop to the program. Inside the for loop you need to access each individual character—the *charAt* method of the String class lets you do that. The assignment statement

```
ch = phrase.charAt(i);
```

   assigns the variable *ch* (type char) the character that is in index i of the String *phrase*. In your for loop you can use an assignment similar to this (replace i with your loop control variable if you use something other than i). NOTE: You could also directly use phrase.charAt(i) in your if (without assigning it to a variable).

2. Test your program on several phrases to make sure it is correct.

3. Now modify the program so that it will count several different characters, not just blank spaces. To keep things relatively simple we'll count the a's, e's, s's, and t's (both upper and lower case) in the string. You need to declare and initialize four additional counting variables (e.g. *countA* and so on). Your current *if* could be modified to cascade but another solution is to use a *switch* statement. Replace the current *if* with a switch that accounts for the 9 cases we want to count (upper and lower case a, e, s, t, and blank spaces). The cases will be based on the value of the *ch* variable. The switch starts as follows—complete it.

```
switch (ch)
{
   case 'a':
   case 'A':  countA++;
              break;

   case ....

}
```

   Note that this switch uses the "fall through" feature of switch statements. If *ch* is an 'a' the first case matches and the switch continues execution until it encounters the *break* hence the countA variable would be incremented.

4. Add statements to print out all of the counts.

5. It would be nice to have the program let the user keep entering phrases rather than having to restart it every time. To do this we need another loop surrounding the current code. That is, the current loop will be nested inside the new loop. Add an outer while loop that will continue to execute as long as the user does NOT enter

the phrase *quit*. Modify the prompt to tell the user to enter a phrase or *quit* to quit. Note that all of the initializations for the counts should be inside the while loop (that is we want the counts to start over for each new phrase entered by the user). All you need to do is add the while statement (and think about placement of your reads so the loop works correctly). Be sure to go through the program and properly indent after adding code—with nested loops the inner loop should be indented