Artificial Intelligence Homework 6
Due date: 4/29 by 11:59pm

Decision Trees

For this assignment, you will implement a decision tree to classify whether a member of the House of Representatives is a democrat or a republican.

This is a programming assignment and you may work with a partner if you choose. Click here to download the starter code. You will notice that the code is **not** arranged in packages! Instead, there are only two files:

- The data file is voting-data.tsv. This file contains the voting record on 10 different issues for 430 members of the House of Republicans. Each row corresponds to one member and has the format:

  Rep-6 D -++---+++.

  where the first token is a label, the second token is the member's party affiliation (D for democrat and R for republican), and the last token is the voting record. A plus (+) means a vote of "yea", a minus (-) means a vote of "nea", and a period (.) means the person did not cast a vote. Each token is separated by a tab.

- The PartyPredictor.java file which contains a single private method for reading in the data from file.

Implementing the Decision Tree

For this assignment, you should fill in the main() method in PartyPredictor.java. Your main() method should perform the following two tasks when executed:

1. First, use all 430 examples to learn a decision tree. After the decision tree is learned, you should print it to the screen using a depth-first traversal where each level is indented (use tabs to indent). For example, a small decision tree might look like:

```
Issue 5:
	+ Issue 8:
		+ D
		- R
		. D
	- Issue 4:
		+ R
		- Issue 2:
			+ R
			- R
			. D
		. R
	. D
```

   where the first node checks issue 5. If the member voted "yea", then check issue 8. If the member voted "nea", then check issue 4. If the member did not cast a vote, then classify them as a democrat.

2. Next, your main() method should evaluate the accuracy of the learned decision tree by pulling out 10% of the examples to be a "test set" on which you will compute the accuracy. In particular, you should:

- Set aside 10% of the examples (i.e. 43 examples) to be a test set. Use the remaining 387 examples to learn the decision tree.

- Evaluate the accuracy of the decision tree on the test set. The accuracy is simply the number of correctly classified examples divided by the total number of examples. Thus perfect accuracy would be 43/43 = 100%

- Repeat the above process 10 times, each time pulling out the next set of 43 examples to be the test set. During the first iteration, your test set should consist of examples 1 through 43. During the second iteration, your test set should consist of examples 44 through 86, so on and so forth.

- Average the accuracy over all 10 test sets and print this as a percentage to the screen. *Please do not print the decision trees themselves! Just print the average accuracy.*

Feel free to create other methods and classes as necessary. Think carefully about the time/space complexity of your code. In particular, make sure you're not copying the actual data set again and again and again.

Also, notice that you will be learning decision trees again and again and again, each time on a different set of data. As such, it makes sense to create a method that takes in a set of data and returns the decision tree. You can then call this method multiple times.

Grading and Submission

Your assignment will be graded based upon correctness, efficiency (i.e. the time/space complexity of your implementation), as well as its adherence to the Java style guide.

To submit, rename your directory hw6_FirstName_LastName. Then zip your directory and upload it to Moodle. Please remember to rename your directory *before* you zip it.
