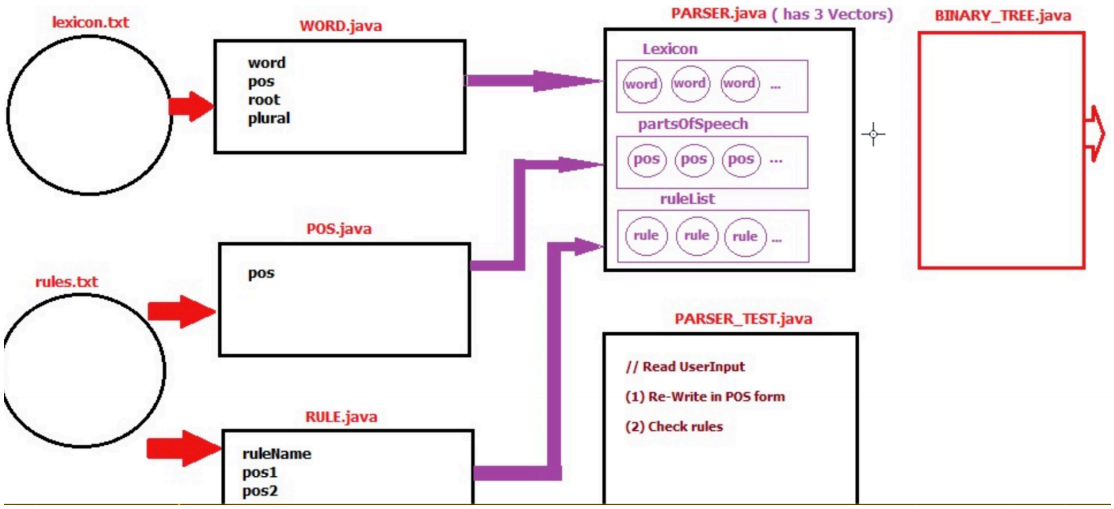
AHLT Assignment Brainstorm



lexicon.txt: Please refer to notes …should have a listing of each word or lexical item with 3 other elements….similar to notes: word, POS, root and number

rules.txt will have a list of all POS in first line and under this there will be a list of rules. Below is a sample but not a comprehensive listing of this. Please refer to tree bank POS categories in the notes for POS tags.

```
nn nns vbp vbz dt jj s np vp ap
s   np  vp
ap  jj np
np  dt  ap
np  nn  null
```

Information from these two files then needs to be transferred into memory in the form of a vector or arraylist or by any other means you wish.

Classes then must be written for these. This will convert them to objects and then these can be transferred into a vector or any data structure you wish.

You can name them as you wish. Rules.txt needs two classes, one for the first line, which supplied all the POS's and one for the rules. Lexicon.txt needs only one class.

On top of this you will need a parser class. The following is a sample of preparing a parser class using vectors. This is just an example. As I have advised previously…you may use any data structure you wish.

AHLT Assignment Brainstorm

```
class Parser{

  private Vector Lexicon;           // reads from lexicon.txt - holds words!!!
  private Vector partsOfSpeech;   // reads in the POSs
  private Vector ruleList;          // reads thre rest of the rules


  public Parser(String filename) throws Exception {
      partsOfSpeech = new Vector();
      ruleList      = new Vector();
      Lexicon       = new Vector();

      FileReader fr = new FileReader(filename);
      BufferedReader br = new BufferedReader(fr);

      //READ THE PARTS OF SPEECH FROM THE FILE
```

At the end of this stage everything should be in memory. The next step in the process is to test the rules you've created against a sentence entered into the programme. In the diagram above this is called ParserTest. Within this we instantiate parser and pass our string that the user has inputted to this.

We will then need to look up each word inputted by the user in our predefined lexicon vector and mark the word for POS.

At this point …if you wish you can create a tree.

Above we have a sample sentence and if this was the inputted sentence and we have found all the parts of speech from the lexicon we could now check that the sentence was correct by placing this into a data structure… this could be a list, queue, stack…. You can decide this… We could then iterate through this and check for grammatical correctness based on the rules we created…. From here you must decide how to output your results to the screen.