GENERATOR EXERCISE

The purpose of this exercise is to develop a toolset for the compression of text files. It exercises your knowledge of

(1) Java file io
(2) Java utility classes
(3) Java serialisation
(4) Make files

The objective is to have a pair of programs comp and decomp. The command

```
comp foo
```

will cause a compressed file foo.cmp to be created, which should hopefully, be shorter than the original.

The command

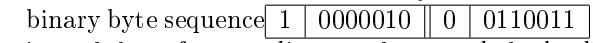
```
decomp foo.cmp
```

will regenerate the original text file from the compressed file.

Compression technique to be used. A lexicon base compressor is to be used. Each word of plaintext is to be assigned a unique code. The codes will be chosen so that the following obtains

(1) The codes 1..127 (00..7F hex) will designate the most common 127 words.
(2) The codes 128 to 16383 will designate the next most common group of words
(3) Codes above 16384..1048575 will designate other words

A variable length representation of the codes will be used.

- Codes in the first group are simply stored as single bytes
- Codes in the second group are stored as a pair of bytes with the leading bit of the second byte set to 0 and the leading bit of the first byte set to 1. Thus the number 307 = binary100110011 would be represented as the binary byte sequence 

. Thus 7 bits are available in each byte for encoding numbers, and the leading bit is used to indicate if another byte follows.
- Codes in the 3rd group are stored as a triple of bytes using a similar scheme, with the last byte having a most significant bit of 0.

The consequence of this variable length representation is that the most common words are represented with a single byte and the majority of the rest by a pair of bytes, only a few uncommon words will require 3 bytes.

In order to encode and decode text files a codebook data structure is needed. This class must be able to map words to codes and also perform the reverse map from codes to words. The mapping from words to codes can be perfomed using a Java *HashTable* or *HashedMap* whilst the reverse mapping can be performed with a Java *Vector* class. All of these are availble in the java.util package.

Constructing the codebook. The codebook can only be constructed if we know the relative frequencies of words in the english languge. This can be ascertained by reading a large number of English text files and counting the frequency of occurence of words. This initial phase of discovery can be performed by a training program which then generates the codebook.

Dependencies. In a project like this there are a number of dependencies. Some of these are obvious. If the source code of any of the classes used in any of the 3 programs change, then the program itself must be recompiled. Similarly the codebook depends upon both the training program and the set of files that make up the training set.

The appropriate way to handle these dependencies is to use a make file for the project. This should capture all of the dependencies shown allowing the system to be rebuilt if the sources or the trainingset changes.

Parsing source files. The source files, both during training and during compression can be split into words using the *StringTokenizer* class. Note that if this is done, you must output spaces after each word that is read in.

Consider what you are going to do about punctuation in the source text.

It is your responsibility to find source files for the training program.

A convenient way of providing these to the traiing program might be to give the training program a file containing a list of file names or URLs which themselves refer to the training text.

Testing

You should test your compressor and decompressor programs using a text file which is not included in the training set that you use. Having done this, estimate the sixes of the compressed relative to the uncompressed files.

Submission. You should submit

(1) The make file
(2) The source files
(3) Either your training files, or a file containing a list ofURLs that you used for training.
(4) A report that shows your class structure and gives an estimate of the compression ratio you achieved.
(5) Submit on the last week of term. This exercise will be 50% of your lab marks.