Go to the project Gutenberg website (https://www.gutenberg.org/) and download a book in plain text (.txt/UTF-8) format. You're going to write a program that asks two data-science oriented questions about the book.

(1) How frequently is a given word used?

(2) What are the most frequently used words in the entire text?

Part 1

Write a program `TextAnalytics.java` that answers question one for a given book. In order to answer this question, you should use a `HashMap`. Do not import it, instead write it yourself in `ObjectHashMap.java`. It should extend the provided `AbstractHashMap` and make use of the provided `Entry` object. You should implement a chaining hash-map. Note that the starting and ending points of the book are marked in the file with text similar to this:

"*** START OF THIS PROJECT GUTENBERG EBOOK..."

-and-

"*** END OF THIS PROJECT GUTENBERG EBOOK..."

Use this special marker text to exclude text that is not part of the book itself. Since this marker text can vary slightly (download a couple of books to find the variance), try to make your checking for the start and end robust to these differences.

Below is a list of a few tools you can use to make your life easier.

```
// Converts a given string to lowercase
str.toLowerCase();

// Removes all special characters from a given string
// This is called a "regular expression"
str.replaceAll("[^a-z]", "");

// Splits a string based on spaces (returns an array of strings)
str.split("\\s+");

// Some functions of the HashMap work with Object
// instead of String or Integer.  This makes the HashMap
// flexible / generic.  You can use casting to convert
// to the object types you want if you know the type actually is
Integer val = (Integer)map.get(w);

// There is a handy ArrayList or LinkedList method
// that converts a list to an array of Objects
Object[] arr = list.toArray();

// You can also get back an array of any type you want
// if you know the list already contains that type.
Entry[] arr = new Entry[list.size()];
arr = list.toArray(arr);
```

Note: `ObjectHashMap` will be used for the `TextAnalytics` program. However, you should write it in a way that is generic. From the outside it should function identically to the `java.util.HashMap` or the python `dictionary` structure. It should not require any specific type for the keys or values (that's why they are both of type `Object` in the provided `Entry` class.). After finishing part 1 your program should look like this:

```
javac TextAnalytics.java
java TextAnalytics book1.txt

Type a word or type 'q' to quit: hello
The word 'hello' is not present.

Type a word or type 'q' to quit: hi
The word 'hi' occurs 4 times.

Type a word or type 'q' to quit: the
The word 'the' occurs 8016 times.

Type a word or type 'q' to quit: q
```

Note: The book is passed in on the command line (`book1.txt`) so the program should work with any project Gutenberg book.

Part 2

Modify the program `TextAnalytics.java` so that it answers the second question. It should print the top five most frequently used words in the text before going into the user interaction. To do this you will use the `getEntries()` method in `ObjectHashMap` to get an array of all the entries. Then, you will need to sort that array in order to show the top five words. Implement and use the insertion sort algorithm in a method:

```
private static void insertionSort(Entry[] arr);
```

When you're done the program should look like this when you run it.

```
javac *.java
Note: ObjectHashMap.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
java TextAnalytics book1.txt
--Top 5 Most Frequent Words--
1.) 'the'   8016 uses.
2.) 'and'   4926 uses.
3.) 'of'   4013 uses.
4.) 'to'   3463 uses.
5.) 'a'   2919 uses.

Type a word or type 'q' to quit: hello
The word 'hello' is not present.
```

```
Type a word or type 'q' to quit: hi
The word 'hi' occurs 4 times.

Type a word or type 'q' to quit: the
The word 'the' occurs 8016 times.

Type a word or type 'q' to quit: q
```

Part 3

Go back and write comments for any methods / segments of code that are `O(n)` or greater time complexity where `n` is the number of words in the text file you choose.