Sort And Search Instructions

The purpose of this assignment is to practice using a selection sort along with a recursive binary search in Java. For this program, all of your code (including a 'main' method) should be contained in a single file named "SortAndSearch.java", and you must use the template/code shown below. With the exception of filling in the missing code in the indicated locations, you may not rearrange any of the code, add any code to the template, or modify the existing code in any way.

This program has three main parts:

1) Read a list of strings (words) from a user-chosen data file
2) Use a selection sort to order the words alphabetically
3) Use a recursive binary search to look for user-entered words

As you can see from the template below, your 'main' method consists mainly of calls to other methods. The program contains only one global variable, an ArrayList of Strings which will contain the words read from the user-supplied data file. In the program you may not have any other global variables, and you may not use any additional ArrayLists, standard arrays, or other Collections.

When alphabetizing the words from the disk text file, you must use a selection sort and ignore case. You should also ignore case when searching for words entered by the user. Your 'findWords' method must contain a loop that allows the user to keep searching for words until ENTER is pressed by itself.

Your program should function in a manner identical to what is shown in the example program run below, although the wording of your output can vary. You do not need to perform any error trapping. Assume that the disk file of words exists, contains at least one word, and does not contain any duplicate entries.

As always, you must do your own work. That is, this program must be created entirely by you. You may, however, use code from previous Java classworks that you have written. For example, you may choose to use parts of the disk-reading and data-sorting routines from your previous assignments.

Template for 'SortAndSearch' Class

```java
import java.util.*;
import java.io.*;

public class SortAndSearch
{
   // Create a global ArrayList ('myWords') of Strings
   private static ArrayList<String> myWords = new ArrayList<String>();

   public static void main(String[] args)
   {
      readWords();

      System.out.println("\nOriginal List of Words:");
      showWords();

      sortWords();

      System.out.println("\nSorted List of Words:");
      showWords();

      findWords();
   }

   // Ask the user to enter a complete filename, and then read all of the
   // Strings from that disk text file into the 'myWords' ArrayList
   public static void readWords()
   {
      // [Put Your Code Here.]
   }

   // This method displays, in their current order, all of the Strings in the 'myWords' ArrayList
   public static void showWords()
   {
      for (int i = 0; i < myWords.size(); i++)
         System.out.printf("  %-5d%-15s%n", (i + 1), myWords.get(i));
   }

   // This method uses a selection sort to alphabetize an ArrayList of Strings, ignoring case
   public static void sortWords()
   {
      // [Put Your Code Here.]
   }

   // This method swaps the words at positions 'spotA' and 'spotB' in an ArrayList of Strings
   private static void swap(int spotA, int spotB)
   {
      String spotC = myWords.get(spotA);
      myWords.set(spotA, myWords.get(spotB));
      myWords.set(spotB, spotC);
   }

   // This method has the user enter a word, and then calls a binary search to look for that String
   // in the 'myWords' ArrayList; if the word is found, then the method tells the user at what spot
   // the word is located in the sorted list; if not found, the user is told so; this method uses a
   // loop so that the user can keep entering words until the ENTER key is pressed by itself
   public static void findWords()
   {
      // [Put Your Code Here.]
   }

   // This method uses a recursive binary search to look in the ArrayList 'myWords' for a specific
   // String equal to the user-supplied word (ignoring case); the method returns the index of the
   // word if it is found, or -1 if not found; the parameter 'first' is the starting position (index)
   // of the search range in the ArrayList, and 'last' is the ending index of the range
   private static int binarySearch(String wordToFind, int first, int last)
   {
      // [Put Your Code Here.]
   }
}
```

Sample Input and Output from Run of Program

```
Enter a filename containing a list of words:  words.txt

Original List of Words:
  1    crab
  2    dog
  3    Fish
  4    shark
  5    elephant
  6    Ant
  7    monkey
  8    rhinoceros
  9    Eel
  10   aardvark
  11   porcupine
  12   Chipmunk

Sorted List of Words:
  1    aardvark
  2    Ant
  3    Chipmunk
  4    crab
  5    dog
  6    Eel
  7    elephant
  8    Fish
  9    monkey
  10   porcupine
  11   rhinoceros
  12   shark

Enter a word to find (ENTER by itself to quit):  ant
The word "ant" is in the sorted list at spot #2.

Enter a word to find (ENTER by itself to quit):  ANT
The word "ANT" is in the sorted list at spot #2.

Enter a word to find (ENTER by itself to quit):  dog
The word "dog" is in the sorted list at spot #5.

Enter a word to find (ENTER by itself to quit):  Dog
The word "Dog" is in the sorted list at spot #5.

Enter a word to find (ENTER by itself to quit):  shark
The word "shark" is in the sorted list at spot #12.

Enter a word to find (ENTER by itself to quit):  aardvark
The word "aardvark" is in the sorted list at spot #1.

Enter a word to find (ENTER by itself to quit):  giraffe
The word "giraffe" is not in the list.

Enter a word to find (ENTER by itself to quit):  crab
The word "crab" is in the sorted list at spot #4.

Enter a word to find (ENTER by itself to quit):  chipmunk
The word "chipmunk" is in the sorted list at spot #3.

Enter a word to find (ENTER by itself to quit):
```