```
/*  StringsDemo.java
 *
 * This program implements a number of simple string-processing operations for
 * text files.  Upon startup, the user is prompted to choose a particular
 * operation, and then is prompted for the file specification for the input
 * file and for additional parameters as appropriate.  The output is written
 * to standard output.
 *
 * Copyright (c) 2000, 2004 - Russell C. Bjork
 */

import  java.io.*;
import  javax.swing.*;
import  java.awt.event.*;

public  class  StringsDemo
{
     /** Search the specified file for all occurrence of the specified
      *   pattern, printing each such line to standard output
      *
      *   @param source the file to search
      *   @param pattern the pattern to search for
      *
      *   This search is case sensitive
      */
     public  static  void  search(BufferedReader  source,  String  pattern)
               throws  IOException
     {
          System.out.println("Starting  results  of  search");
          String  line  =  source.readLine();
          while  (line  !=  null)
          {
               if (line.indexOf(pattern) >= 0)
               System.out.println(line);
               line  =  source.readLine();
          }
          System.out.println("Done");
          System.out.println();
     }

     /** Search the specified file for all occurrence of the specified
      *   pattern, printing each such line to standard output
      *
      *   @param source the file to search
      *   @param pattern the pattern to search for
      *
      *   This search is not case sensitive
      */
     public  static  void  searchIgnoreCase(BufferedReader  source,  String  pattern)
               throws  IOException
     {
          System.out.println("Starting  results  of  case  insensitive  search");
          pattern  =  pattern.toUpperCase();
          String  line  =  source.readLine();
          while  (line  !=  null)
          {
               if  (line.toUpperCase().indexOf(pattern)  >=  0)
               System.out.println(line);
               line  =  source.readLine();
          }
          System.out.println("Done");
          System.out.println();
     }
```

```
/** Substitute a specified replacement for all occurrences of a
 *  specified pattern within a specified file, writing the modified
 *  file to standard output
 *
 *  @param source the file to search
 *  @param pattern the pattern to search for
 *  @param replacement the text to replace it with
 *
 *  The search process is case sensitive
 */

public static void substitute(BufferedReader source,
                              String pattern,
                              String replacement)  throws IOException
{
    System.out.println("Starting results of substitute");
    String line = source.readLine();
    while (line != null)
    {
        // Replace each occurrence of pattern in the line

        int index = line.indexOf(pattern);
        while (index >= 0)
        {
            line = line.substring(0, index) +
                   replacement +
                   line.substring(index + pattern.length(), line.length());
            // Search for next occurrence of pattern, beginning at index +
            // replacement.length() in case pattern occurs in replacement
            index = line.indexOf(pattern, index + replacement.length());
        }

        // Output the line with any changes that may have been made

        System.out.println(line);

        // Move on to the next line

        line = source.readLine();
    }
    System.out.println("Done");
    System.out.println();
}
```

```
    /** Print all the individual words occurring in a specified file, one per
     *  line, writing the words to standard output
     *
     *  @param source the file to break up into words
     *
     *  Note: A word is considered to be a sequence of 1 or more alphabetic
     *  characters and/or digits.  Anything else (spaces, punctuation etc.)
     *  is considered to be separators and is not displayed.
     */

    public static void listWords(BufferedReader source)
            throws IOException
    {
        System.out.println("Starting list of words");
        String line = source.readLine();
        while (line != null)
        {
            // Iterate through line one word at a time.  start marks the
            // beginning position of a word, finish the first character after
            // its end

            int start = 0, finish = 0;
            while (start < line.length())
            {
                // Find start of next word (next letter, if any)

                while (start < line.length() &&
                          ! Character.isLetter(line.charAt(start)))
                    start ++;

                // Find finish of this word (next non-letter, if any)

                finish = start;
                while (finish < line.length() &&
                          Character.isLetter(line.charAt(finish)))
                    finish ++;

                // If we found a real word (not some spaces or punctuation at
                // the end of the line), then output it

                if (finish > start)
                    System.out.println(line.substring(start, finish));

                // Next word begins where this one left off

                start = finish;
              }

            // Move on to next line

            line = source.readLine();
        }
        System.out.println("Done");
        System.out.println();
    }
```

The code for the main method is omitted since it is not relevant to the issues in this handout

```
}
```