```java
import components.queue.Queue;
import components.queue.Queue1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * Put a short phrase describing the program here.
 *
 * @author Put your name here
 *
 */
public final class NextWordSeperator {

    /**
     * Definition of whitespace separators.
     */
    private static final String SEPARATORS = " \t\n\r";

    /**
     * Token to mark the end of the input. This token cannot come from the input
     * stream because it contains whitespace.
     */
    public static final String END_OF_INPUT = "### END OF INPUT ###";

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private NextWordSeperator() {
    }

    /**
     * Returns the first "word" (maximal length string of characters not in
     * {@code SEPARATORS}) or "separator string" (maximal length string of
     * characters in {@code SEPARATORS}) in the given {@code text} starting at
     * the given {@code position}.
     *
     * @param text
     *           the {@code String} from which to get the word or separator
     *           string
     * @param position
     *           the starting index
     * @return the first word or separator string found in {@code text} starting
     *         at index {@code position}
     * @requires 0 <= position < |text|
     * @ensures <pre>
     * nextWordOrSeparator =
     *   text[position, position + |nextWordOrSeparator|)  and
     * if entries(text[position, position + 1)) intersection entries(SEPARATORS) = {}
     * then
     *   entries(nextWordOrSeparator) intersection entries(SEPARATORS) = {}  and
     *   (position + |nextWordOrSeparator| = |text|  or
     *    entries(text[position, position + |nextWordOrSeparator| + 1))
     *      intersection entries(SEPARATORS) /= {})
     * else
     *   entries(nextWordOrSeparator) is subset of entries(SEPARATORS)  and
     *   (position + |nextWordOrSeparator| = |text|  or
     *    entries(text[position, position + |nextWordOrSeparator| + 1))
     *      is not subset of entries(SEPARATORS))
     * </pre>
     */
    private static String nextWordOrSeparator(String text, int position) {
        String output = "";
        char c = text.charAt(position);
        int i = 0;
        boolean check = true;
        while (i < SEPARATORS.length() && check) {
            if (c == SEPARATORS.charAt(i)) {
                    check = false;
                }
                i++;
            }
            output = c + output;
            i = 1;
            int u = 0;
            while (check) {
                c = text.charAt(position + i);
                while (u < SEPARATORS.length() && check) {
                    if (c == SEPARATORS.charAt(i)) {
                        check = false;
                    }
                    u++;
                }
                i++;
                output = c + output;
            }
            return output;
        }

        /**
         * Tokenizes the entire input getting rid of all whitespace separators and
         * returning the non-separator tokens in a {@code Queue<String>}.
         *
         * @param in
         *            the input stream
         * @return the queue of tokens
         * @updates in.content
         * @requires in.is_open
         * @ensures <pre>
         * tokens =
         *   [the non-whitespace tokens in #in.content] * <END_OF_INPUT>  and
         * in.content = <>
         * </pre>
         */
        public static Queue<String> tokens(SimpleReader in) {
            Queue<String> words = new Queue1L<>();
            while (!in.atEOS()) {
                int pos = 0;
                words.enqueue(nextWordOrSeparator(in.nextLine(), pos));
            }
            words.enqueue(END_OF_INPUT);
            return words;
        }

        /**
         * Main method.
         *
         * @param args
         *            the command line arguments
         */
        public static void main(String[] args) {
            SimpleReader in = new SimpleReader1L();
            SimpleWriter out = new SimpleWriter1L();
            in.close();
            out.close();
        }

    }
```