```java
package proj5;

/**
 * Uses a thesaurus and word frequencies to replace overused words in a text document with random synonyms.
 *
 * "I affirm that I have carried out the attached academic endeavors with full academic honesty, in
 * accordance with the Union College Honor Code and the course syllabus."
 * author: Son Nguyen (Kyrie)
 * version: 6/3/2020
 */
public class GrammarChecker {

    // ASCII integers
    private int a_ASCII = 97;
    private int z_ASCII = 122;
    private int A_ASCII = 65;
    private int Z_ASCII = 90;

    // instance variable
    private Thesaurus thesaurus;
    private int threshold;
    private WordCounter wc;
    private LineReader paragraph;
    private String[] currentLine;

    /**
     * Non-default constructor.
     * @param thesaurusFile path to comma-separated file used to build a thesaurus
     * @param threshold a word is considered "overused" if it appears more than
     *                  (but not equal to) this many times in a text document
     */
    public GrammarChecker(String thesaurusFile, int threshold) {
        thesaurus = new Thesaurus(thesaurusFile);
        this.threshold = threshold;
        wc = new WordCounter();
        paragraph = null;
        currentLine = null;
    }

    /**
     * helper method to slice a word including punctuation to its content only
     * @param word
     * @return word's content
     */
    private String splitWord(String word) {
        String toReturn = word;
        char firstChar = word.charAt(0);
        if (!Character.isLetter(firstChar)) {
            if (toReturn.length() == 1) {
                return "";
            }
            toReturn = toReturn.substring(1);
        }

        int lastChar = word.charAt(word.length() - 1);
        if (!Character.isLetter(lastChar)) {
            if (toReturn.length() == 1) {
                return "";
            }
            toReturn = toReturn.substring(0, toReturn.length() - 1);
        }
        return toReturn;
    }

    /**
     * Given a text file, replaces overused words with synonyms.
     * Finished text is printed to the console.
     * @param textfile file with original text
     */
    public void improveGrammar(String textfile) {
        System.out.println(splitWord("--"));
        wc.findFrequencies(textfile);
        paragraph = new LineReader(textfile, " ");
        currentLine = paragraph.getNextLine();
        while (currentLine != null) {
            for (String rawWord: currentLine) {
                char firstChar = rawWord.charAt(0);
                char lastChar = rawWord.charAt(rawWord.length() - 1);
                String currentWord = splitWord(rawWord.toLowerCase());
                String replacement = "";
                int currentFrequency = wc.getFrequency(currentWord);
                if (currentFrequency > threshold) {
                    replacement = thesaurus.getSynonymFor(currentWord);
                }
                if (replacement != "") {
                    if (firstChar < a_ASCII || firstChar > z_ASCII) {
                        if (firstChar >= A_ASCII && firstChar <= Z_ASCII) {
                            replacement = Character.toUpperCase(replacement.charAt(0)) + replacement.substring(1);
                        }
                        else {
                            replacement = firstChar + replacement;
                        }
                    }
                    if (lastChar < a_ASCII || lastChar > z_ASCII) {
                        replacement += lastChar;
                    }
                }
                else {
                    replacement = rawWord;
                }
                System.out.print(replacement + " ");
            }
            currentLine = paragraph.getNextLine();
        }
        paragraph.close();
    }
}
```