22AIE113 – Elements of Computing – 2

Assignment – 2

Tokenizer:

Code:

```
import re

class JackTokenizer:
    def __init__(self, input_file):
        with open(input_file, "r") as f:
            self.lines = f.readlines()
        self.tokens = []
        self.current_token = None
        self._tokenize()

    def _clean_line(self, line):
        line = re.sub(r"\/\/.*", "", line)
        line = re.sub(r"\/\*.*?\*\/", "", line)
        return line.strip()

    def _tokenize_line(self, line):
        line = self._clean_line(line)
        while line:
            if line[0] == '"':
                end_index = line.find('"', 1)
                token = line[: end_index + 1]
                line = line[end_index + 1 :]
            else:
                match = re.match(r"^\s*(\w+|\d+|.)", line)
                token = match.group(1)
                line = line[match.end() :]
            if token:
                self.tokens.append(token)

    def _tokenize(self):
        for line in self.lines:
            self._tokenize_line(line)

    def has_more_tokens(self):
        return bool(self.tokens)

    def advance(self):
        if self.has_more_tokens():
```

```
            self.current_token = self.tokens.pop(0)
        else:
            self.current_token = None

    def token_type(self):
        keywords = [
            "class",
            "constructor",
            "function",
            "method",
            "field",
            "static",
            "var",
            "int",
            "char",
            "boolean",
            "void",
            "true",
            "false",
            "null",
            "this",
            "let",
            "do",
            "if",
            "else",
            "while",
            "return",
        ]
        symbols = "{}()[].,;+-*/&|<>=~"
        if self.current_token in keywords:
            return "keyword"
        elif self.current_token in symbols:
            return "symbol"
        elif re.match(r"^\d+$", self.current_token):
            return "integerConstant"
        elif re.match(r'^".*"$', self.current_token):
            return "stringConstant"
        elif re.match(r"^[a-zA-Z_]\w*$", self.current_token):
            return "identifier"


    def to_xml(self):
        xml_output = "<tokens>\n"
        while self.has_more_tokens():
            self.advance()
            token_type = self.token_type()
            token_element = "<{0}> {1} </{0}>\n".format(token_type,
self.current_token)
            xml_output += token_element
```

```
        xml_output += "</tokens>"
        return xml_output


if __name__ == "__main__":
    input_file = "Main.jack"
    tokenizer = JackTokenizer(input_file)

    xml_output = tokenizer.to_xml()
    print(xml_output)
```

Input Jack file:

Main.jack

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/09/HelloWorld/Main.jack

/** Hello World program. */
class Main {
   function void main() {
      /* Prints some text using the standard library. */
      do Output.printString("Hello world!");
      do Output.println();      // New line
      return;
   }
}
```

Output:

```
[Running] python -u "c:\Users\Kaushik\OneDrive\Desktop\SEM 2\22AIE113 EOC-2\Projects\2\tokenizer.py"
<tokens>
<keyword> class </keyword>
<identifier> Main </identifier>
<symbol> { </symbol>
<keyword> function </keyword>
<keyword> void </keyword>
<identifier> main </identifier>
<symbol> ( </symbol>
<symbol> ) </symbol>
<symbol> { </symbol>
<keyword> do </keyword>
<identifier> Output </identifier>
<symbol> . </symbol>
<identifier> printString </identifier>
<symbol> ( </symbol>
<stringConstant> "Hello world!" </stringConstant>
<symbol> ) </symbol>
<symbol> ; </symbol>
<keyword> do </keyword>
<identifier> Output </identifier>
<symbol> . </symbol>
<identifier> println </identifier>
<symbol> ( </symbol>
<symbol> ) </symbol>
<symbol> ; </symbol>
<keyword> return </keyword>
<symbol> ; </symbol>
<symbol> } </symbol>
<symbol> } </symbol>
</tokens>
```