**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\_\_":

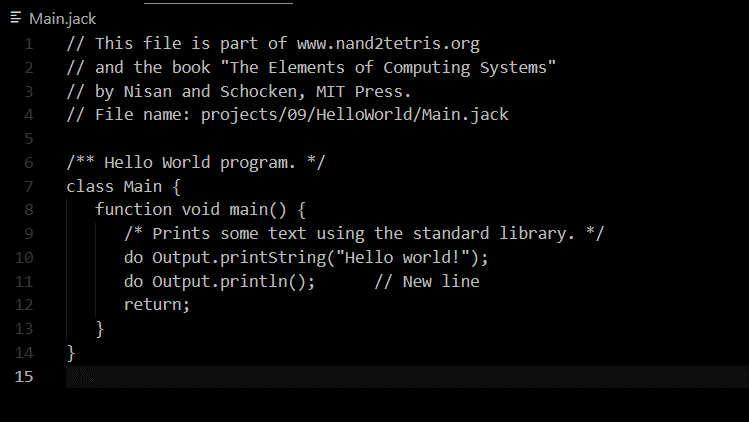
    input\_file **=** "Main.jack"

    tokenizer **=** JackTokenizer(input\_file)

    xml\_output **=** tokenizer.to\_xml()

**print**(xml\_output)

**Input Jack file:**



**Output:**

