Comp 2140 Lab Assignment 1: Regular expression in Java (8 points)

January 12, 2023

1 Due Date and Submission

In your lab section during the week of January 20. To be submitted 15 minutes before the end of your lab section. You can also submit it in the labs one week earlier. This assignment (and all the other assignments in this course) is an individual assignment and should be done independently. Make sure that you follow Senate Bylaws 31.

2 Objectives

Get familiar with regular expressions. Learn to read the grammar of Tiny language. Understand the wide application of regular expressions.

3 Assignment specification

Your task is to pick up identifiers in programs written in our Tiny language.

Note 3.1: Identifier

An identifier consists of a letter followed by any number of letters or digits. The following are examples of identifiers:

- x, x2, xx2, x2x,
- End, END2.

Note that End is an identifier while END is a keyword in the Tiny language. The following are not identifiers:

- IF, WRITE, READ, ... (keywords are not counted as identifiers)
- 2x (identifier can not start with a digit)
- Strings in comments are not identifiers.

You will write a method that picks out the identifiers from a text file. For example, given a sample input:

```
INT f2 ( INT x , INT y )
BEGIN
    z := x * x - y * y ;
    RETURN z ;
END
INT MAIN F1 ( )
BEGIN
    INT x ;
    READ( x , "A41 . i n p u t " ) ;
    INT y ;
    READ( y , "A42 . i n p u t " ) ;
    INT z ;
    z := f 2 ( x , y ) + f 2 ( y , x );
    WRITE ( z , "A4 . output " ) ;
END
```

Your code should return a set of identifiers that consists of f_2, x, y, z, F_1. Please note that in this sample input program, the following are not counted as identifiers:

- A41, input, output: they are quoted hence they are not treated as identifiers;
- INT, READ etc.: They are keywords used in our Tiny language hence they should not be picked up.

Here are a few test cases for the assignment: case 1, case 2, case 3, case 4, case 5, case 6. For those cases, their corresponding ID counts are 5, 4, 6, 7, 8, 9, respectively. To make your task simpler, In this assignment, you can suppose that there are no comments in the input program. You will write two different programs to do this as specified below.

4 A11: coding from scratch

The first approach is the accomplish the task from scratch without using any tools. This approach also motivates the introduction of DFA in Assignment 2. Program A11.java is not supposed to use regular expressions, not regex package, not any methods involving regular expression in String class or other classes. Your program should use the most primitive method, i.e. look at characters one by one, and write a loop to check whether they are quoted strings, identifiers, etc.

A simplified version of the algorithm can be depicted by Algorithm 20. It gets a set of identifiers from an input string x. The algorithm starts with the initial ("INIT") state and scans the characters one by one. Once it sees a letter, it goes to the "ID" state. In the "ID" state, it expects to see more letters or digits until it sees a character other than a letter or digit. At this point, it exits the "ID" states, and goes back to the initial state,"INIT". The algorithm needs to be expanded to deal with quoted strings and keywords. For quoted strings, you can remove them first before you pick the identifiers. For keywords, you can check whether a token belongs to the keyword set before adding into the identifiers set.

Algorithm 1 The algorithm for obtaining identifiers from an input string

```
Input An input string x.
Output a set of identifiers in x
state="INIT";
token="" ;
identifiers= ;
while (c=nextChar())!=end of string x  do
    if c isLetter then
        state="ID";
        token =token+c;
    end if
    if state is "ID" then
        if c is letter or digit then
            state="ID";
            token=token+c;
        else if
             thenadd token to identifiers;
            token="";
            state="INIT";
        end if
    end if
end while
```

We provide the starter code for A11 as follows. You need to expand it to deal with quoted strings and keywords.

Note 4.1: How to Run Java in Command Line

You will have much more control if you run java in command line instead of an IDE. To compile your java program in command line, you type

javac A11 . java

You may see a warning message like Note: Recompile with -Xlint:unchecked for details. You can simply ignore it. Then, you can run your program by typing:

java A11

There may be a class not found error. In this case, check whether you have a package declaration. If there is one, remove the package declaration. If class not found error is still there, try the following command:

java -classpath . A11

-classpath . tells the JVM to look for the class under the current directory. The dot (.) here means current directory.

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.util.Set;
import java.util.HashSet;

public class A11 {
        static boolean isLetter(int character) {
```

```
        return (character >= 'a' && character <= 'z') || (character >= 'A' && character <= 'Z')
    }

    static boolean isLetterOrDigit(int character) {
            return isLetter(character) || (character >= '0' && character <= '9');
    }

    public static Set<String> getIdentifiers(String filename) throws Exception{
        String[] keywordsArray = { "IF", "WRITE", "READ", "RETURN", "BEGIN",
                        "END", "MAIN", "INT", "REAL" };
        Set<String> keywords = new HashSet();
        Set<String> identifiers = new HashSet();
             for (String s : keywordsArray) {
                     keywords.add(s);
             }
             FileReader reader = new FileReader(filename);
             BufferedReader br = new BufferedReader(reader);
             String line;
        while ((line = br.readLine()) != null) {
                int i=0;
                while (i < line.length()) {
                        if (line.charAt(i)=='\"'){
                        // throw away quoted strings
                             if (isLetter(line.charAt(i))){
                             // get the identifier
                         }

                             return identifiers;
                     }
                }
        }

    public static void main(String[] args) throws Exception{
            Set<String> ids=getIdentifiers("A1.tiny");
            for (String id :ids)
                    System.out.println(id);
    }
}
```

5 A12

Program A12.java will use regular expressions in java.util.regex. Here is a tutorial for Java regex. There are many ways to solve the problem. One approach is described in the starter code below– it read the code line by line. In each line, it finds quoted strings and replaces them with empty strings. Then find identifiers and put them into a set if they are not keywords.

```
import java.io.*;
import java.util.HashSet;
import java.util.Set;
import java.util.regex.*;
public class A12 {
    public static Set<String> getIdRegex(String filename) throws Exception{
            String[] keywordsArray = { "IF", "WRITE", "READ", "RETURN", "BEGIN","END", "MAIN",
```

```
                        "INT", "REAL" };
                Set<String> keywords = new HashSet();
                     Set<String> identifiers = new HashSet();
                     for (String s : keywordsArray)
                             keywords.add(s);

                     FileReader reader = new FileReader(filename);
                     BufferedReader br = new BufferedReader(reader);
                     String line;
                //Pattern idPattern = ......;
                //Pattern quotedStringPattern = .....;
                while ((line = br.readLine()) != null) {
                Matcher m_quotedString = quotedStringPattern.matcher(line);
                                String lineWithoutQuotedStrings = m_quotedString.replaceAll("");
                                Matcher m = idPattern.matcher(lineWithoutQuotedStrings);
                                while (m.find()) {
                                        String id = line.substring(m.start(), m.end());
                                        if (!keywords.contains(id))
                                                identifiers.add(id);
                                }
                }
                     return identifiers;
    }
    public static void main(String[] args) throws Exception{
            Set<String> ids=getIdRegex("A1.tiny");
            for (String id :ids)
                    System.out.println(id);
    }
}
```

Note 5.1: Hints to work on your program

Try to run the following code first. Then modified it into A12.

```
import java.util.regex.*;
public class RegexTest {
public static void main(String args[]) {
        String pattern = "\\d{4}-(0?[1-9]|1[012])-\\d{2}";
        String text = "final exam 2008-04-22, or 2008-4-22, but not        2008-22-04";
        Pattern p = Pattern.compile(pattern);
        Matcher m = p.matcher(text);
        while (m.find()) {
                System.out.println("valid date:"+text.substring(m.start(),  m.end()));
        }
        }
}
```