DEPARTMENT OF INFORMATION TECHNOLOGY

HD in Software Engineering (IT114105) 2020/2021

ITP4510 DSA: Concepts & Implementation

ASSIGNMENT
Deadline: 11 Apr 2021 (Sun) 11:59pm

A ***Cross-Reference Map*** for a program is a list of all the **identifiers** employed in the program, along with the line numbers on which they appear. It is often useful that a programmer has this information on his or her hand during the debugging stage of programming.

Identifiers are words used in programs to name variables, classes, methods, or labels and are subjected to strict rules. **None of the Java keywords/reserved words may be used as identifiers**. An identifier can **begin with** a letter, a dollar sign ($) or an underscore character (_). The following is a list of Java keywords/reserved words for your reference. They are extracted from https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html.

abstract	*continue*	*for*	*new*	*switch*
assert	*default*	*goto*	*package*	*synchronized*
boolean	*do*	*if*	*private*	*this*
break	*double*	*implements*	*protected*	*throw*
byte	*else*	*import*	*public*	*throws*
case	*enum*	*instanceof*	*return*	*transient*
catch	*extends*	*int*	*short*	*try*
char	*final*	*interface*	*static*	*void*
class	*finally*	*long*	*strictfp*	*volatile*
const	*float*	*native*	*super*	*while*

Note: You may find that the list of Java keywords/reserved words from other sources of information can be a bit different from the above list. However, you should use the above for the purpose of this assignment.

You are required to develop a **Java program** (***XRef.java***) that will generate such a cross-reference map for a Java source file. You do not need to handle comments, therefore assume no comments in the input java source file. In the cross-reference map, the identifiers should be listed in alphabetical order, and the line numbers on which each identifier appears should be listed in ascending order.

The following is a sample execution of the program with the given **Sample.java** as the test file:

```
C:> java XRef Sample.java
```

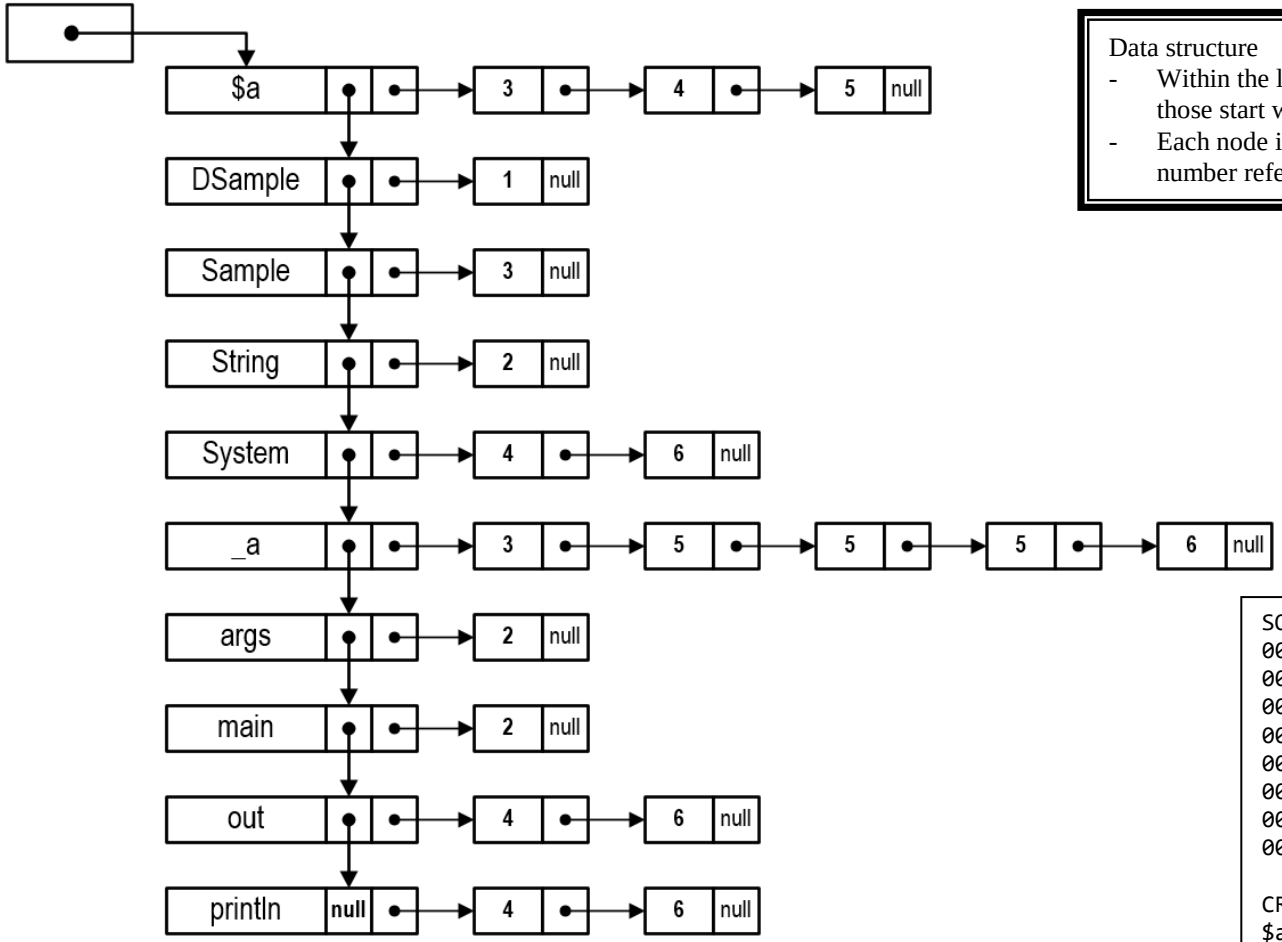
```
X R e f   v 1
=========================================
Program created by CHAN Tai Man. 31-01-2021

SOURCE FILE: Sample.java
0001 | import java.util.*;
0002 | public class Sample {
0003 |  public static void main(String [ ] args) {
0004 |          Scanner keyboard = new Scanner(System.in);
0005 |          int num, den;
0006 |          double result;
0007 |          System.out.print("Numerator? ");
0008 |          num = keyboard.nextInt();
0009 |          System.out.print("Denominator? ");
0010 |          den = keyboard.nextInt();
0011 |          try {
0012 |                  result = quotient(num, den);
0013 |                  System.out.println("Answer: " + result);
0014 |          }
0015 |          catch (DivideByZeroException e) {
0016 |                  System.out.println("DivideByZeroException caught!");
0017 |          }
0018 |          catch (ArithmeticException e) {
0019 |                  System.out.println("ArithmeticException caught!");
0020 |          }
0021 |          catch (Exception e) {
0022 |                  System.out.println("Exception caught!");
0023 |          }
0024 |  }
0025 |  public static double quotient(int num, int den) {
0026 |          if (den==0)
0027 |                  throw new DivideByZeroException();
0028 |          return (double) num / den;
0029 |  }
0030 | }
0031 |
0032 | public class DivideByZeroException extends ArithmeticException {
0033 | }

CROSS REFERENCE:
ArithmeticException    : [ 18 32 ]
DivideByZeroException  : [ 15 27 32 ]
Exception              : [ 21 ]
Sample                 : [ 2 ]
Scanner                : [ 4 4 ]
String                 : [ 3 ]
System                 : [ 4 7 9 13 16 19 22 ]
args                   : [ 3 ]
den                    : [ 5 10 12 25 26 28 ]
e                      : [ 15 18 21 ]
in                     : [ 4 ]
java                   : [ 1 ]
keyboard               : [ 4 8 10 ]
main                   : [ 3 ]
nextInt                : [ 8 10 ]
num                    : [ 5 8 12 25 28 ]
out                    : [ 7 9 13 16 19 22 ]
print                  : [ 7 9 ]
println                : [ 13 16 19 22 ]
quotient               : [ 12 25 ]
result                 : [ 6 12 13 ]
util                   : [ 1 ]

XRef v1 normally terminated.
```

Data Structure – DSample.java scenario



Data structure

- Within the linked list, identifiers are arranged in ASCII order including those start with dollar sign $ and underscore _.
- Each node in the linked list has an associated linked list to keep line number references.

```
SOURCE FILE: DSample.java
0001 | public class DSample {
0002 |    public static void main(String[] args) {
0003 |       int $a = 0, _a = 1, Sample;
0004 |       System.out.println($a);
0005 |       _a = _a*_a - $a;
0006 |       System.out.println(_a);
0007 |    }
0008 | }

CROSS REFERENCE:
$a          : [ 3 4 5 ]
DSample     : [ 1 ]
Sample      : [ 3 ]
String      : [ 2 ]
System      : [ 4 6 ]
_a          : [ 3 5 5 5 6 ]
args        : [ 2 ]
main        : [ 2 ]
out         : [ 4 6 ]
println     : [ 4 6 ]
```

IMPLEMENTATION REQUIREMENTS:

- Employ the techniques discussed in the ITP4510 DSA teaching materials to develop the linked list package (use of interface, comparator and list exceptions are recommended).
- Use an array to hold the Java keywords/reserved words in ascending or descending order.
- Devise an efficient searching algorithm (linear or binary search) for the ordered keyword array. The searching is used to determine if a given token/word is a java keyword or an identifier.
- Tokenize the Java source using the code provided in the Tokenizer section.
- Implement exception handlers for your program. Specifically, you may want to define custom exception to catch illegal identifiers that are extracted by the Tokenizer.
- Format the program output as close as possible to the sample outputs given. The first part of the output is the source labelled with line number and the second part is the cross-reference map.

Tokenizer

```
private static final String
    DELIMITER = "\"(?:\\\\\"|[^\"])*?\"|[\\s.,;:+*/|!=><@?#%&(){}\\-\\^\\[\\]\\&&]+";

public static String[] tokenizer(String javaStmt) {
    String[] tokens = javaStmt.split(DELIMITER);
    return tokens;
}
```

This tokenizer will extract Java program tokens from the string parameter based on the DELIMITER. It returns an array of strings/tokens. Sample program elements recognized by the tokenizer are identifier, keyword, numeric literal, and character literal. It will also return invalid elements like unclosed string. Refer to the note "2 – Exception" for an example on split method.

INSTRUCTIONS:

1. This assignment is an individual assignment and each student submits his/her own work. Plagiarism is a serious offence and any assignments that involve any plagiarism will be given ZERO marks. The award of Zero marks will apply to all parties, regardless of whether a student is the original author or the plagiarist. Further disciplinary action will follow.

2. You are asked to include the following documentation at the top of your program.

```
/**
 * class XRef - Cross Reference Map
 *
 * I understand the meaning of academic dishonesty, in particular plagiarism, copyright infringement
 * and collusion. I am aware of the consequences if found to be involved in these misconducts. I hereby
 * declare that the work submitted for the "ITP4510 Data Structures & Algorithms" is authentic record
 * of my own work.
 *
 * @Name : Your full name in English
 * @StdID: 20XXXXXXX
 * @Class: IT114105/1X
 * @2021-XX-XX
 */
```

3. You are **NOT ALLOWED** to use data structure classes provided by Java API. You must use the linked list (and/or other data structures) developed in this module for the assignment. However, it is at your discretion on extending the classes with attributes/methods to suit your case.

4. Your programs must follow the coding standard stated in Java coding standard maintained by Oracle (https://www.oracle.com/java/technologies/javase/codeconventions-contents.html). Marks will be deducted if the coding standard is not followed.

5. Program, class and method documentation are essential. Inline comments should be placed as appropriate. User-defined names should be descriptive as much as possible. Marks are given based on correctness, readability, style and quality.

6. All work is to be submitted to Moodle on/before **11th April 2021 11:59pm**. Late submission without valid reasons will be given ZERO mark.

7. You are required to hand in:
   a) All classes (programs) written and well commented by you for this assignment.
   b) The evidence of testing - You should submit at least 4 dump screens (similar to the sample output given above) to show the run results with your own input files (that means you should design your own test cases).

8. The weighting of this assignment is 40% of EA.

9. Marks allocation:

Implementation	
Linked list	15%
Keyword search	15%
Exception	11%
Solution design	33%
Evidence of testing	12%
Coding Standard	6%
Program documentation	8%

References:

Linear Search -
https://www.tutorialspoint.com/data_structures_algorithms/linear_search_algorithm.htm

Binary Search -
https://www.tutorialspoint.com/data_structures_algorithms/binary_search_algorithm.htm