Turnitin Report

by P W

Submission date: 06-May-2021 07:39AM (UTC-0400)
Submission ID: 1579517251
File name: A_Specific_Comparison_of_Scheme_Solution_with_Java_Code.docx (79.34K)
Word count: 833
Character count: 4714

A Specific Comparison of Scheme Solution with Java Code

Student's Name:

Course Title:

Institution:

Date:

A Specific Comparison of Scheme Solution with the Java Code

This paper's purpose is to highlight Java Code and Scheme Solution syntax differences, similarities, runtime speeds and comparing the paradigms of the specific codes shown below. In other words, the aim of this paper is to give an objective comparison of the Scheme solution to the Java Code Syntax.

A. Scheme Solution

```
1
2
3  (define max-equal-sublists (lambda (list sublist1 sublist2 sub1length sub2length)
4              (cond
5                  ((null? list) (if (and (= sublist1 sublist2)(= sub1length sub2length)) sub1length 0))
6                  ((> (mes (cdr list) (+ sublist1 (car list)) sublist2 (+ sub1length 1) sub2length)
7                      (mes (cdr list) sublist1 (+ sublist2 (car list)) sub1length (+ sub2length 1)))
8                   (mes (cdr list) (+ sublist1 (car list)) sublist2 (+ sub1length 1) sub2length))
9                  ((> (mes (cdr list) sublist1 (+ sublist2 (car list)) sub1length (+ sub2length 1))
10                     (mes (cdr list) sublist1 sublist2 sub1length sub2length))
11                  (mes (cdr list) sublist1 (+ sublist2 (car list)) sub1length (+ sub2length 1)))
12                 (else (mes (cdr list) sublist1 sublist2 sub1length sub2length))
13             )
14     )
15 )
```

Figure 1: Scheme Solution Code

The Scheme code shown in figure 1 above is incredibly simple and straightforward semantics and methods to structure expressions. Moreover, the Scheme code above is structural and employs one lexical setting for every variable, and examines the operator position of every function call in a similar manner as an operand position. Also, this Scheme solution code includes first-class escape protocols, which are used in this program to synthesize all pre-existing sequential control structures.

B. Java Code

Figure 2 below shows an object-oriented Java source code that is compiled into bytecode and then run by a Java interpreter. Since Java runtime environments and interpreters, such as Java Virtual Machines (VMs), are present for most versions of operating systems, the compiled version of this Java code can operate on most machines.

Figure 2. Java Code Snippet

```
1
2
3   private static int maxEqualSublists(Node node, int s1, int s2, int sl1, int sl2) {
4           if(node == null) {
5               return s1 == s2 && sl1 == sl2 ? sl1 : 0;
6           }
7           if(maxEqualSublists(node.next, s1 + node.data, s2, sl1 + 1, sl2)
8              > maxEqualSublists(node.next, s1, s2 + node.data, sl1, sl2 + 1)) {
9               return maxEqualSublists(node.next, s1 + node.data, s2, sl1 + 1, sl2);
10          } else if(maxEqualSublists(node.next, s1, s2 + node.data, sl1, sl2 + 1)
11                  > maxEqualSublists(node.next, s1, s2, sl1, sl2)) {
12              return maxEqualSublists(node.next, s1, s2 + node.data, sl1, sl2 + 1);
13          } else {
14              return maxEqualSublists(node.next, s1, s2, sl1, sl2);
15          }
16      }
```

Comparison between Java Code and the Scheme Solution

Compared to the Scheme Solution, the Java Code sample has an additional syntax that allows a programmer to navigate the more sophisticated features instantly conveniently, but this complicates the entire learning process, defeating the purpose of being simple to understand. Scheme solution, on the other hand, embraces closures and first-level functions. Furthermore, in the Scheme solution, more elaborate operations are represented using simplified operations, which simplify executing Scheme Solution. For example, see the Java code snippet below:

```
private static int maxEqualSublists(Node node, int s1, int s2, int sl1, int sl2) {
      if(node == null) {
         return s1 == s2 && sl1 == sl2 ? sl1 : 0;
      }
      if(maxEqualSublists(node.next, s1 + node.data, s2, sl1 + 1, sl2)
        > maxEqualSublists(node.next, s1, s2 + node.data, sl1, sl2 + 1)) {
         return maxEqualSublists(node.next, s1 + node.data, s2, sl1 + 1, sl2);
      } else if(maxEqualSublists(node.next, s1, s2 + node.data, sl1, sl2 + 1)
            > maxEqualSublists(node.next, s1, s2, sl1, sl2)) {
         return maxEqualSublists(node.next, s1, s2 + node.data, sl1, sl2 + 1);
      } else {
         return maxEqualSublists(node.next, s1, s2, sl1, sl2);
      }
   }
```

It would be expressed in Scheme Solution as:

```
(define max-equal-sublists (lambda (list sublist1 sublist2 sub1length
sub2length)
             (cond
                 ((null? list) (if (and (= sublist1 sublist2)(= sub1length
sub2length)) sub1length 0))
```

```
                ((> (mes (cdr list) (+ sublist1 (car list)) sublist2 (+
sub1length 1) sub2length)
                    (mes (cdr list) sublist1 (+ sublist2 (car list))
sub1length (+ sub2length 1)))
                 (mes (cdr list) (+ sublist1 (car list)) sublist2 (+
sub1length 1) sub2length))
                ((> (mes (cdr list) sublist1 (+ sublist2 (car list))
sub1length (+ sub2length 1))
                    (mes (cdr list) sublist1 sublist2 sub1length sub2length))
                 (mes (cdr list) sublist1 (+ sublist2 (car list)) sub1length
(+ sub2length 1)))
                (else (mes (cdr list) sublist1 sublist2 sub1length
sub2length))
            )
     )
)
```

Notably, most of the Scheme syntax is specified in the Scheme code; therefore, while the Java code defines subroutines, Scheme still enables the specification of new syntax and determines what new syntax to use at runtime. The apparent disadvantage of this, just as shown in the scheme solution code, is that there are a number of brackets in the Scheme source code, which may be frustrating at first. However, these parentheses are, on the other hand, valid (Min & Yeom, 2021). "The evidence of several practical attempts to articulate the full force of Scheme in a more traditional syntax that was never successful demonstrates that these brackets perform easier than they appear at first sight, and a majority of Scheme programmers agree that once someone understands it, it makes everything much simpler" (Desai, 2019).

Unlike the code, the Scheme solution is easy to understand since the language's core is minimalist. Furthermore, Scheme Solution has been implemented in a multi-paradigm manner, which means that it can be mutated if a programmer prefers either Object-Oriented Programming or functional programming. However, the Java code has been exclusively written in an Object-Oriented manner with no packages, pointers, and threads. Also, compared to Scheme Solution, Java code has a wealthy set of libraries, but the code is obviously wordy than the simple, precise scheme solution code.

Markedly, the Scheme solution captures the returned values of methods in both the execution and method invocation. Markedly, it implements execution handling by itself. Further, just like Java code, the Scheme solution supports Object-Oriented-like Abstraction and supports both static and dynamic time. Lastly, upon execution, the Scheme solution has a faster UI response and runtime speed compared to Java code.

References

Desai, S. (2019). The Scheme Programming Language. *International Journal of Computer Science Engineering and Information Technology Research*, *9*(1), 1-10. https://doi.org/10.24247/ijcseitrjun20191

Min, C., & Yeom, H. (2021). Efficient Java Full GC Compact Scheme for Multicores. *KIISE Transactions on Computing Practices*, *27*(3), 129-136. https://doi.org/10.5626/ktcp.2021.27.3.129

Turnitin Report

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		