Development of a Java Source Code Analyzer for Learning Support That Runs in a Web Browser

Tatsuyuki Takano, Kanto Gakuin University, Japan
Takashi Kohama, Tokyo Denki University, Japan
Osamu Miyakawa, Tokyo Denki University, Japan

The Asian Conference on Education 2023
Official Conference Proceedings

Abstract
Students make various mistakes in the process of practicing computer programming. For this reason, we have developed a source code analyzer, which evaluates the source code submitted by students from The tool can identify misspellings in the method names of source code and can judge the compiling and execution results. However, because the tool was developed to run on a teacher's PC, it does not easily fit into the format of general programming learning sites. The use of a programming learning site is advantageous in that it allows students to learn programming without having to build a programming environment. However, programming languages other than those that run in the client browser, such as JavaScript, must be compiled and evaluated on the server side, placing a heavy burden on the server side. Therefore, we decided to use a method of running the developed tools on the As a result, we confirmed that the basic functionality of the tool runs on the browser and outputs evaluation results. The basic functionality of the tool outputs the results of spelling errors in class names and method names, coding style, compilation results, and execution. The basic functionality of the tool outputs the results of spelling errors in class names and method names, coding style, compilation results, and execution results for source code written in Java.

Keywords: Programming Education, Source Code Analyze, Learning Support Tools

Introduction

In today's society, the role of ICT (Information and Communication Technology) is becoming more important as AI (Artificial Intelligence) is becoming an everyday part of our daily lives. Therefore, software development technology is one of the important elements for constructing information systems, and the importance of training programming engineers is increasing. In Japan, the Ministry of Education, Culture, Sports, Science and Technology (MEXT) is promoting education that incorporates programming in compulsory education (Ministry of Education, Culture, Sports, Science and Technology. 2021).

And in introductory programming education, knowing the learning elements necessary for beginning students to master programming is an important guideline for improving the effectiveness of education. It is also important to know the elements that cause stumbling blocks in learning programming, and learning programming requires practice that includes the experience of making mistakes (Martin, R.C., 2008).

Therefore, we have developed a source code analyzer that can perform unit testing from an educational perspective, considering spelling errors. This tool analyzes source code created by learners using both static analysis methods that evaluate coding styles and program definitions, and dynamic analysis methods that evaluate by executing unit tests. This tool was previously used as a core function of our system (Shin Hasegawa, et al. 2011) to evaluate learners' source code in real time during lectures. Currently, this tool is also used to evaluate programming assignments (Takano, T. et al. 2023).

A related study is AutoLEP (W. Tiantian, et al. 2009), a system for evaluating learners' source code. This system evaluates programs statically and dynamically and feeds back errors to students. The difference between this tool and AutoLEP is that AutoLEP evaluates program errors based on similarity to the correct program, whereas AutoLEP allows misspellings in definitions and dynamically evaluates the implementation through unit tests. However, since the tool was developed to run on a teacher's PC, it does not easily fit into the format of general programming learning sites. The use of a programming learning site is advantageous in that it allows students to learn programming without having to build a programming environment. However, programming languages other than those that run in the client browser, such as JavaScript, must be compiled and evaluated on the server side, placing a heavy burden on the server side of the programming learning site.

We decided to use Doppio (Vilk, John. at el. 2014), a Java Virtual Machine that runs as JavaScript in the browser, to run the developed tools on the client side.

Development Tool Overview

The Source Code Analyzer is intended for Java programs that create source code from new files. The tool is developed in Java and Apache Groovy (Figure 1).

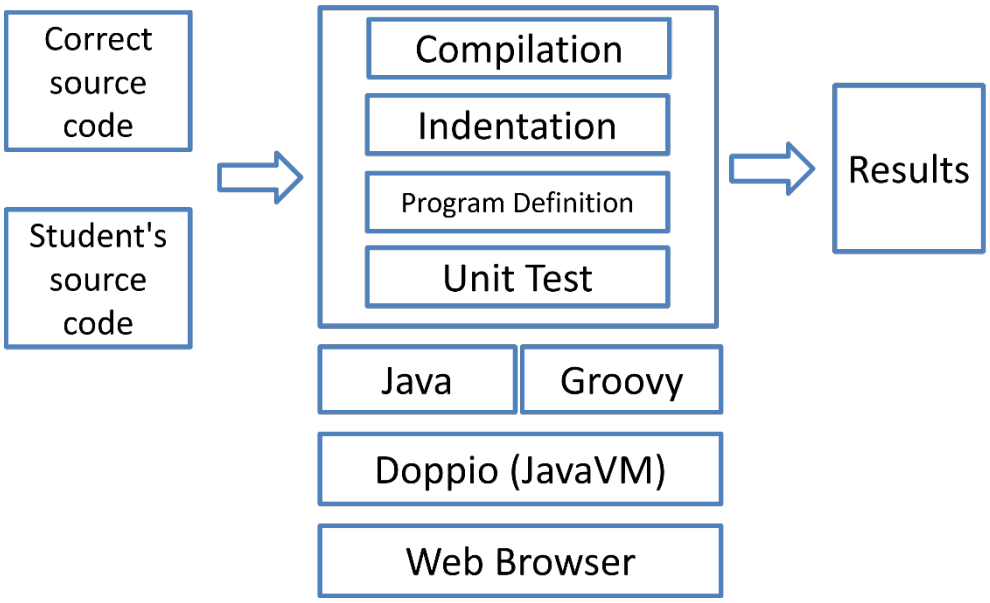

Figure 1: Overview of source code analyzer

The tool makes judgments on the main items: compilation, indentation, program definition, and unit testing. The tool also determines misspellings at various points before the learner completes the program. The following is a list of the main areas where spelling errors are detected.

- File name
- Class name
- Constructor name
- Field Name
- Method name

When determining misspellings, it is necessary to determine whether character strings are similar. One way to measure the similarity is to measure the distance between strings, and there are many algorithms for calculating this distance. This time, we use the Levenshtein distance, where the edit cost of a string is used as the distance. In this algorithm, the weighting of each of replacement, deletion, and addition used as the edit distance is set to 1. The normalized value is used as the similarity value based on the longest string compared by measuring the edit distance.

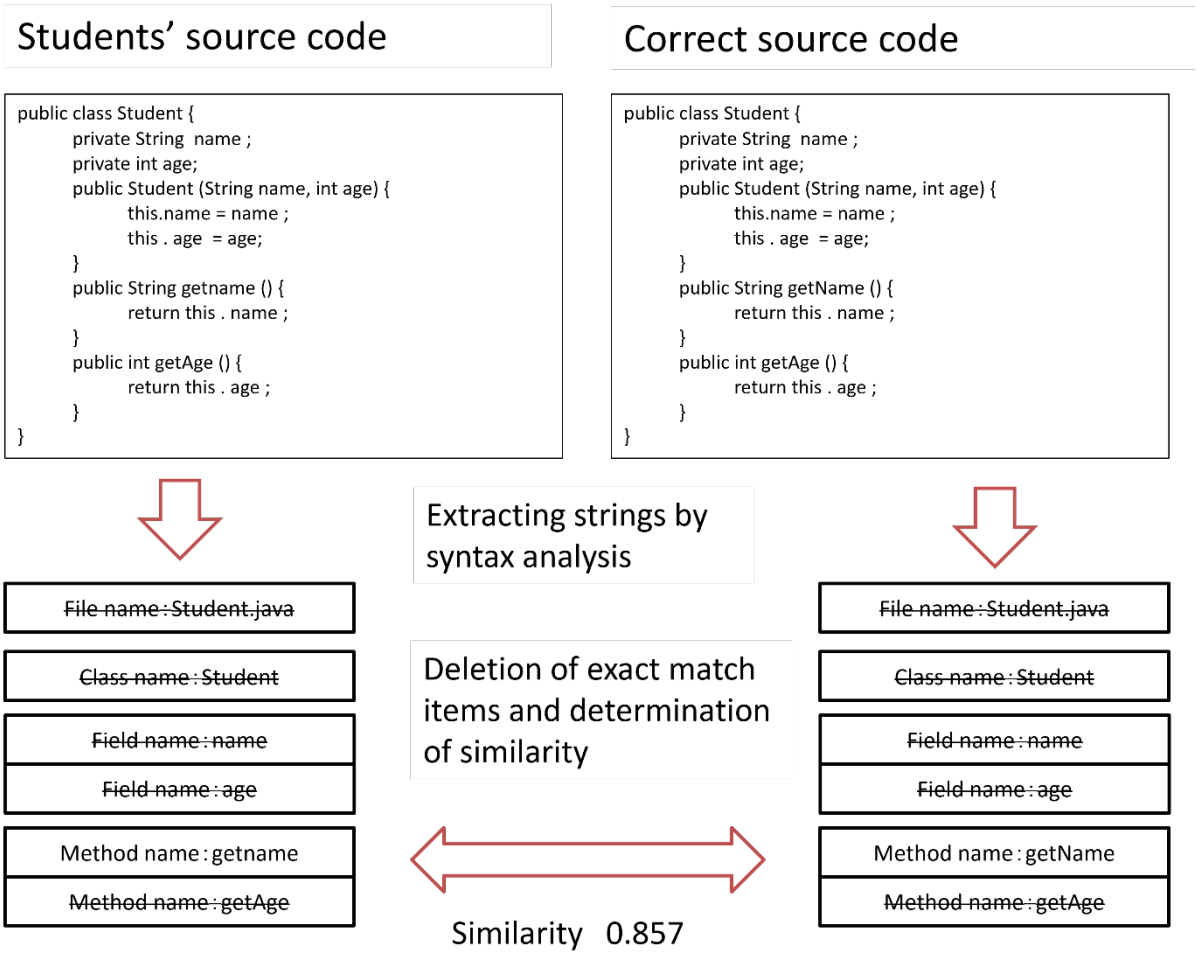

Figure 2: Example of algorithm for calculating similarity

Figure 2 shows an example of the algorithm for calculating the edit distance and similarity of the method names "getname" and "getName. First, a list of method names is created, and strings that match exactly are removed from the list. Then, misspelled strings are compared. Editing "getname" to "getName" requires a single edit, replacing "n" with "N". Since the strings compared have the same length of 7 characters, the similarity is 1 divided by 7 and the value subtracted from 1. In this example, the similarity is about 0.857. The Levenshtein distance was implemented using the API of Apache Lucene, a full-text search engine library.

The resulting misspelling information is then used to generate unit test source code for evaluation. Figure 3 shows how the unit test source code is generated using the spelling error information.

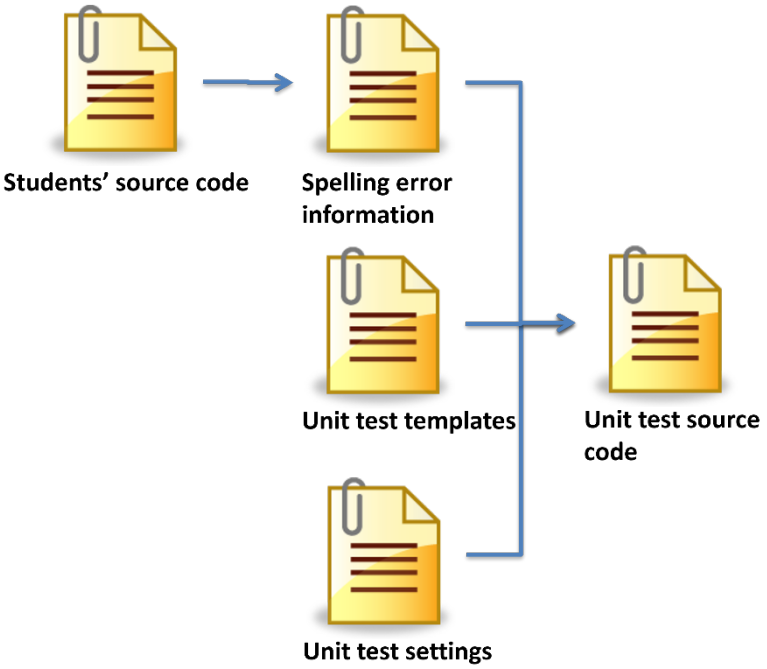

Figure 3: Unit Test Source Code Generation Methodology

Spelling error information is obtained from the program definition. The unit test configuration provides information on the conditions of the test and the values to be passed as arguments. The template that forms the framework of the unit test can then be used to generate the unit test code based on the misspellings. The template is processed using Groovy, and the unit test code dynamically generated by Groovy is used to execute the unit test. The class loader obtained from the compilation process is used to manage and execute the unit test code with a different namespace, even if the class names are the same.

Implementation on the Browser

In order to check the functionality of the source code analyzer in a browser, the tool was configured as a single JAR (Java ARchive) file. The scripting language Groovy was converted to Java bytecode, and libraries and code that did not work on Doppio were modified. BrowserFS, a related project of Doppio, was used for the file system on the browser. The source code analyzer has functions such as unit testing on a sandbox using Java Security and determining the character encoding of files, but these functions were omitted from this implementation.

Evaluating Behavior on the Browser

A source code analyzer was used to evaluate one issue on a browser. Google's Chrome was used as the browser. Figure 4 shows the actual operation of the tool, and it was confirmed that the source code was evaluated and the evaluation results were output as a CSV file without any visual change from the PC operation. It was also confirmed that the source code analyzer program was not loaded into the class loader in some of the unit tests.

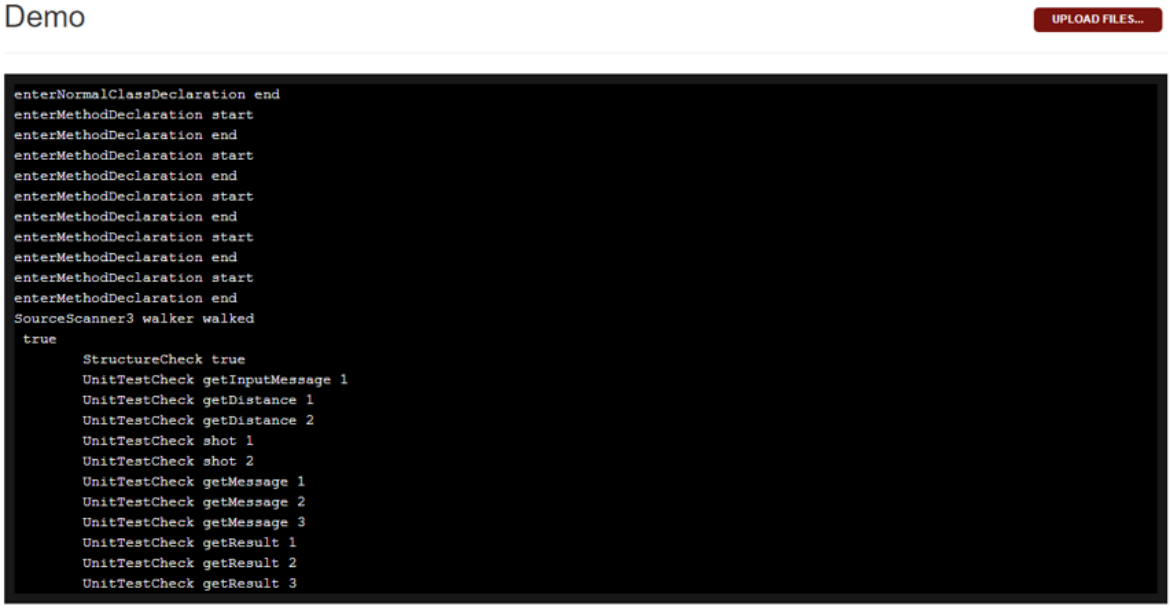

Figure 4: Evaluating behavior on a browser

Conclusions

With the increasing importance of software in society, it is increasingly important to train engineers to develop software. We have developed a source code analyzer to identify errors in the source code of novice programmers. We then applied the tool to run on JavaVM, which is created in JavaScript, to make it easier to use on a browser. As a result, we confirmed that the functions of the tool worked on the browser with some exceptions. In the future, we plan to fix class loader-related problems and improve performance, and to build a learning site for programming.

Acknowledgements

This study was supported by JSPS KAKENHI (grant number: JP21K02809).

References

Martin, R.C. (2008). *Clean Code: A Handbook of Software Craftsmanship*. 1st edition. Prentice Hall.

Ministry of Education, *Culture, Sports*, Science, and Technology. *Reiwa 3 Annual White Paper of the Ministry of Education, Culture, Sports, Science and* Technology. https://www.mext.go.jp/b_menu/hakusho/html/hpab202001/1420041_00010.htm

Shin Hasegawa, et al. (2011). A Real-time Instruction Support System for Introduction to Computer Programming Education. *Processing Society of Japan*.

Takano, T. et al. (2023). Development of a Tool to Analyze Source Code Submitted by Novice Programmers and Provide Learning Support Feedback with Comment. *Education & International Development 2023 Official Conference Proceedings.*

W. Tiantian, et al. (2009), AutoLEP: An Automated Learning and Examination System for Programming and its Application in Programming Course, *2009 First International Workshop on Education Technology and Computer Science*, Wuhan, China, 43-46.

Vilk, John. at el. (2014). Doppio: Breaking the Browser Language Barrier. *sigplan Not.* 49, 6, .508-518. acm.

Contact email: takano@kanto-gakuin.ac.jp