Shih, J. L. et al. (Eds.) (2019). Proceedings of the 27th International Conference on Computers in Education.
Taiwan: Asia-Pacific Society for Computers in Education

Application of Programming Learning Support System to Object-Oriented Language

Satoru KOGURE[a*], Kento OGASAWARA[a], Koichi YAMASHITA[b], Yasuhiro NOGUCHI[a], Tatsuhiro KONISHI[a] & Yukihiro ITOH[c]
[a]*Faculty of Informatics, Shizuoka University, Japan*
[b]*Faculty of Business Administration, Tokoha University, Japan*
[c]*Shizuoka University, Japan*
*kogure@inf.shizuoka.ac.jp

Abstract: In the field of programming education, learners often cannot fully understand the behavior of algorithms and programs, because it is difficult for them to capture a realistic image of their behavior. Many researchers have proposed methods to visualize "the behavior of algorithm programs" as a solution to address this problem. Many programming learning support systems that incorporate these methods. We have developed a programming learning support system, TEDViT, that allows teachers to control the method of visualization. However, TEDViT is a programming language compatible with the C programming language, hence, the system cannot handle visualization in object-oriented languages. We developed and evaluated a prototype system that can visualize the proposed model.

Keywords: Programming learning support system, visualization, object-oriented programming language

1. Introduction

It is essential that beginning programmers are able to grasp and understand the program behavior correctly. However, it can be difficult for beginners to grasp concrete behavior. As a way to solve this problem, many program visualization (PV) systems have been proposed to visualize the program behavior (Pears, 2007, Moreno et al., 2014, Guo, 2013). These PV systems can only visualize the behavior of the program created by the learner based on specific methods decided by the tool developer. Therefore, it is not possible for teachers to freely depict what they want to explain in their teaching materials. To solve this problem, we have developed TEDViT (Teacher Explaining Design Visualization Tool), a C language behavior visualization system that can reflect the teacher's the explanation intention (Yamashita et al., 2018). We conducted many practical programming classes using TEDViT. The results obtained indicate that TEDViT can contribute to the understanding of the program behavior.

In this research, we propose Object-Oriented Conceptual Model (OOC Model) as a framework that can reflect the teacher's intention in the visualization of an object-oriented language. The OOC Model can express the structural relationship between classes and instances, which is one of the essential concepts to learn object-oriented languages. We constructed a prototype learning support system for Java that realized the OOC Model visualization. Herein, we report the experimental findings concerning the usability and usefulness of the prototype system we created.

2. Visualization for an Object-Oriented Language

To investigate problems of visualization for an object-oriented language, we need to select a target language. Although there are many object-oriented languages, there are some differences in the language specifications of each language. In this research, we focus on Java as an example of an object-oriented language. This research considers only the object-oriented language specific issues of the Java language.

In Java, basic concepts such as variables, arrays, and repeating structures are common to other programming languages. There is no need to newly consider the visualization of common concepts in this study, because their visualization is already possible using, for example, Jeliot 3 (Moreno et al., 2014), Python Tutor (Guo, 2013), TEDViT, among others. We investigated the concepts that require Java-specific visualization in Java programming classes offered in the first year of the computer science department. As a result of excluding the above-mentioned common parts, we only focus the relationship class and instance. We do not consider the exception handling and IO streaming topics in this study.

When learning the class and instance topics in object-oriented languages, it is necessary to understand the three concepts, (C1) the class definition and instance generation concepts (the relationship between a defined class and a generated instance), (C2) the inheritance concept (the relationship between a superclass and a subclass), and (C3) the polymorphism concept (invoking method of each class by the same-name method). The C3 cannot be visualized on the existing system. The requirement is to be able to specify the relationship between "class definition" and "instance" and to be able to specify which class's method actually started. To visualize these relationships, we need a mechanism that can synchronously display the "program", "class diagram" and "instance diagram".

We propose the Object-Oriented Conceptual Model (OOC Model) as a visualization model that can realize the three concepts. To understand the program behavior of object-oriented languages, it is necessary to visualize diagrams showing the relationship between classes, methods, and instances. To show relationships between classes and instances, we can use the concepts of "class diagrams" and "object diagrams" used in UML diagrams. However, it is necessary to develop a framework that emphasizes class and instance (including fields and methods) in synchronization with class-instance relationships and program step execution.

3. System Implementation

Figure 1 shows an overview of the developed Prototype system. The left side shows the program, and the right side shows the OOC Model. The learner can proceed with single-step program execution by pressing the `[>]` button. In the Figure 1, the `public void printMemberInfo()` on the program side is surrounded by red lines. This indicates that the program is currently running up to this line. This example shows the situation after executing `ml[1].printMemberInfo()` has been executed. The `ml[1]` is an instance of `SpecialMember`, and the `printMemberInfo()` method defined in `SpecialMember` class should be activated. By highlighting both the program display on the left (the `printMemberInfo ()` method in class `SpecialMember` has been invoked) and the OOC Model on the right (`ml[1].printMemberInfo()` and the `printMemberInfo()` method of the `SpecialMember` class) at the same time, the learner can gain a deep understanding of polymorphism.

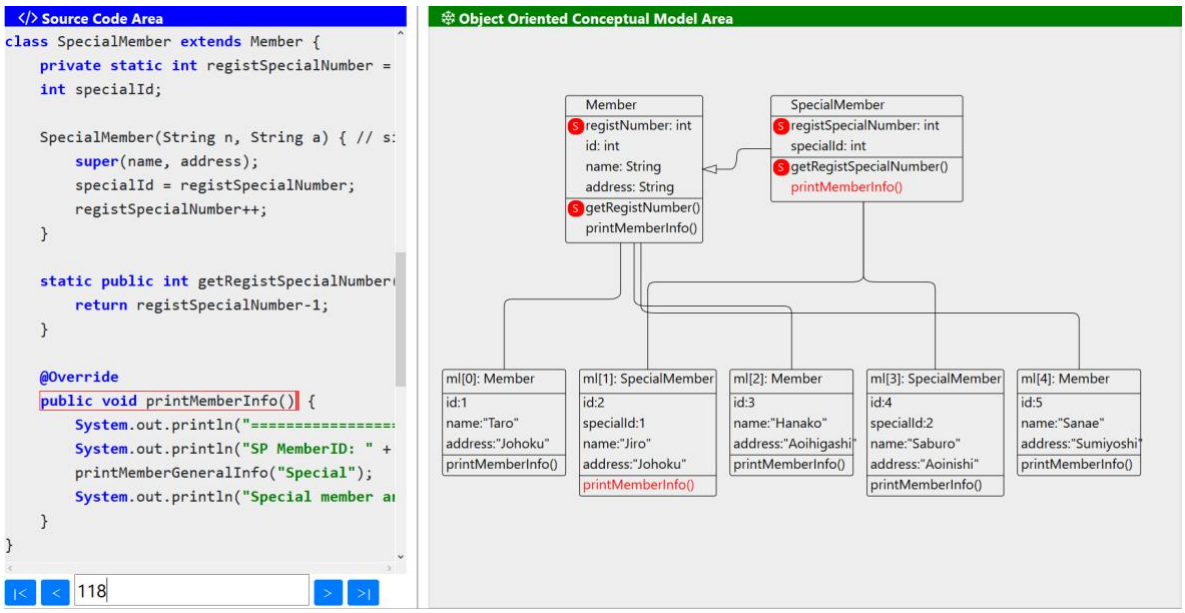

Figure 1. A Screen Shot of Prototype PV System.

4. Experimental Evaluation

The evaluation experiments investigate whether the following three hypotheses hold. ***H1*** is "the use of Java TEDViT raises the level of understanding of Java-specific concepts," ***H2*** is "the OOC Model is an aid to learners of programming," and ***H3*** is "by using Java TEDViT, it is possible to understand the flow of a program's operation, predict of its output, and understand the reasons for its behavior." The subjects for the experiments were 17 university students in the information system area. We gave the subjects the following procedure: (1) View a system tutorial, (2) Answer a questionnaire before the experiment, (3) Use the system, and (4) Answer a questionnaire after the experiment.

First, the rate of increase in self-evaluation for class and instance (***C1***)) was high at 82.3%. Next, the rate of increase in self-evaluation for inheritance (***C2***)) was 76.5%. We noted the opinion that "step flow of the constructor was understood well by executing step-by-step in the system". Finally, the rate of increase in self-evaluation for polymorphism (***C3***)) was 64.7%. It is considered that this did not improve much because the support system did not actively explain polymorphism. This result weakly supports that ***H1*** holds.

Moreover, we obtained subjective questionnaire on the usefulness of the system after the experiment. The results show that although the system does not adversely affect learning, it does not contribute to a clear improvement in understanding. Therefore, it is judged that this questionnaire result rejects ***H2*** and ***H3***. On the other hand, there were many positive opinions on the mechanism of proposed prototype PV system itself, and we obtained the result that the expectation for practical use can become higher by further improvement.

5. Conclusion

In this research, we examined the visualization of program behavior of an object-oriented language and proposed an Object-Oriented Conceptual Model (OOC Model). We also constructed a prototype of a PV system that allows teachers to control the visualization of program behavior based on the OOC Model. Evaluation of the constructed system was carried out by 17 university students. The results suggested that the use of a prototype system improves the understanding of object-oriented learning concepts. Although the evaluation of the usefulness of the system was not good, we obtained the finding that it would be useful if the system were improved.

In the future, therefore, we plan a number of enhancements, to include improving the user interface of the system, simplifying the rule setting method for the teacher, and using the system in an actual classroom setting.

Acknowledgements

This study was supported by JSPS KAKENHI Grant Numbers JP16K01084, JP18K11567, 19K12259 and, JP19K12265.

References

Guo, P. J. (2013). Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. *ACM Technical Symposium on Computer Science Education (SIGCSE)*.

Moreno, A., Sutinen, E., & Joy, M. (2014). Defining and evaluating conflictive animations for programming education: the case of Jeliot ConAn. *Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14)* (pp. 629-634). New York, NY, USA.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, *39*(4), 204-223.

Yamashita, K., Tezuka, D., Kogure, S., Noguchi, Y., Konishi, T., & Itoh, Y. (2018). A learning support system for visualizing behaviors of students' programs based on teachers' intents of instruction. Proceedings of the 26th International Conference on Computers in Education (pp.761-766).