
Comprehension of OOP Concepts using Dolittle in elementary school

JongHye Kim, *jonghye.kim@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

SookKyoung Choi, *sookkyoung.choi@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

HanSung Kim, *hansung.kim@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

JeongA Jang, *jeonga.jang@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

WonGyu Lee, *lee@comedu.korea.ac.kr*

Professor, Dept of Computer Science Education, Korea University, South Korea

Abstract

This article describes the research on Object-Oriented Programming (OOP) learning of the elementary students. The elementary school students were exposed to OOP through Dolittle, which is an educational programming language. Teaching concepts of OOP to elementary school students need high cognitive states because most students have concrete operational period. Our teaching approach focused on OOP concepts and it also featured programming-language oriented thinking method. Evaluation methods were both qualitative and quantitative. At the end of tutoring, most elementary students understood the basic principles of OOP and they naturally developed procedural thinking ability. The two main contributions of this research are: (1) Dolittle is appropriate educational programming language for teaching OOP concepts to elementary school students, and (2) students who are in the concrete operational period can achieve abstract thinking ability through learning OOP.

Keywords

OOP (Object-Oriented Programming), Dolittle, Educational object-oriented programming language, Procedural thinking ability, GALT (Group Assessment of Logical Thinking), Elementary school student

1. Introduction

In December 2005, the "Guidelines for Information and Communications Technology Training in Elementary and Secondary Schools" was published in Korea (Ministry of Education and Human Resources Development, 2005). The biggest change among revised ICT education curriculum was that it focused on learning computer science principles. There exists a virtual consensus that teaching fundamentals of computer science should be focused on scientific principles, problem-solving and project development skills, rather than on specific artefacts such as programming languages and operating systems (Gal-Ezer, Beer, Harel and Yehudai, 1995). One of the education curriculum goals from Grade 3 to Grade 4 is improving problem-solving methods. Students in Grade 5 start to learn programming using programming languages. Abstraction dimension, such as mapping from the problem domain to the programming domain, is important for beginners (Detienne, 2001). However, teaching concepts of OOP (Object-Oriented Programming) to beginners need high cognitive states (Hadjerrouit, 1998). Understanding abstraction dimensions for elementary school students is not easy because they are in the period of concrete operations which is one of the stages of cognitive development formulated by Piaget (Wadsworth, 1989). When elementary school students learn programming with high-level programming language such as Java, they spend a lot of time learning syntaxes instead of focusing on improving problem-solving skills (Dan Aharoni, 2000). Therefore, this study focuses on teaching the concept of OOP to elementary school students and examines procedural thinking ability by using Dolittle, which is an educational OOP language with syntax minimization (Susumu Kanemune et al., 2004). In addition, it examines whether elementary school students can understand concepts of OOP concepts and apply these concepts in real-life problems.

2. Related work

Piaget stated that the concrete operations stage of cognitive development can be found in children ranging from 7 to 11 years old. Although this age range is not fixed, children around these ages start developing logical operations (Wadsworth, 1989). However, logical operation in the period of concrete operations is not sufficiently developed such that children who are in this period can deal with only concrete objects. In other words, children can not understand abstract concepts and they can not solve language problems. This study focused on elementary school students and it examined stages of cognitive development by evaluating students' logical minds before teaching programming. Instructors agree that there is a need for a different pedagogical approach for teaching OOP (Bishop, 1997). In introducing objects, the following concepts have to be treated: the object state, changing the object state, and the way objects relate to each other (Woodman, M. & Holland, S, 1996). It is better to hold off the study of control flow, complex statements and advanced OOP topics such as inheritance (Stein, 1997). Some authors gave a detailed list of students' misconceptions regarding OOP concepts. For example, students only view programming as a computational tool but not as a changing attributes; confusion between an object and an attribute (Holland, S. Griffiths, R & Woodman, M, 1997). To account for these misconceptions, educational programming software called Dolittle will be used in this study to evaluate how appropriate the software is for elementary school students when they learn OOP concepts.

Dolittle – Educational Object-Oriented Programming Language

In order to select an appropriate programming language based on elementary school students' cognitive development level, Dolittle was tested. Dolittle is an educational OOP language and a prototype-based language which copies objects without the concept of class. It is a text-oriented

language and it consists of simple structure as well as easy error treatment. Figure 1 shows a sample Dolittle program that is written in both Korean and English.

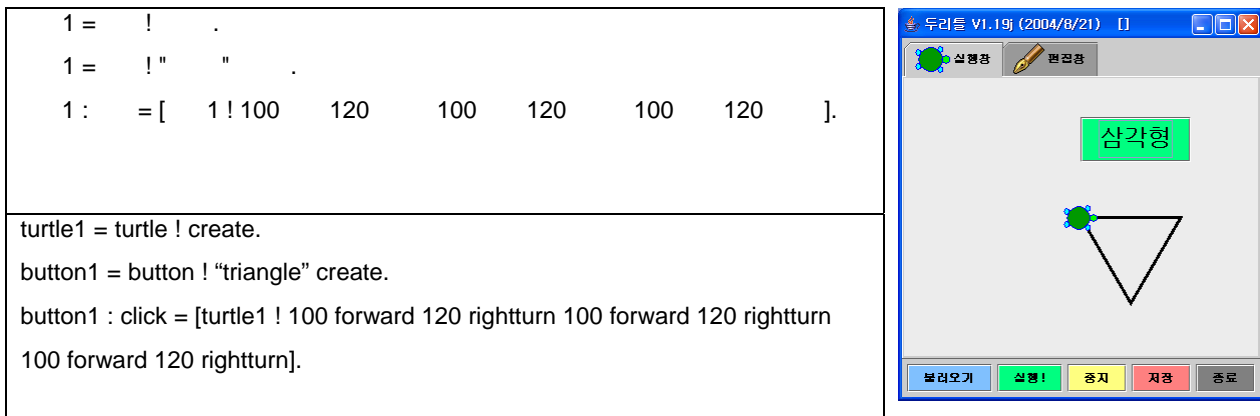


Figure 1 A sample Dolittle program and its execution

Dolittle is known for its syntax minimization. As an example, ‘turtle1=turtle ! create.’ can be seen from Figure 1. Dolittle programs call on objects using “!”. This simple statement sends a “create” message to the prototype object “turtle” to let it clone. The period “.” indicates the end of the statement.

In programming, creating an object using Dolittle is different from other programming languages such as Java. In Java, users create objects using classes while users in Dolittle simply copy an object instead of creating them. For example, ‘button1 = button ! “triangle” create. button1 : click = [turtle1 ! 100 forward 120 rightturn 100 forward 120 rightturn 100 forward 120 rightturn].’ can be seen in Figure 1. This statement creates a button object called “button1”. Next statement assigns a block to the “click” attribute of the “button1” object. The “click” is the method executed when the button is clicked. When you click the button, the execution message is sent to the “turtle1”, and the “turtle1” executes the block. As a result, the screen shows the triangle.

Dolittle is easy to elementary and middle school students because it is in their first(mother) language (Susumu Kanemune et al., 2004). Five under-achieved students who were in Grade 7 got interested in programming after learning programming with Dolittle (HeeKang, 2005). 5 under-achieved students who were in Grade 9 improved their programming skills after learning programming with Dolittle (HeyMinGil, 2004). In other study, 64 Grade 11 students were divided to 2 classes and were taught about logical circuit. One class was taught with an aid of Dolittle, while the other did not. A class that learned logical circuit with Dolittle showed high interaction and improved their problem-solving skills and long-term memories (SuKyungYoo, 2005). These results proved that Dolittle can reach its full effectiveness in secondary education in short period time.

3. Method

Phenomenographic research methodology describes the variation of understanding of a particular phenomenon found in a group of people. The unit of phenomenographic research is a way of experiencing something, and the purpose of the research is the variation in ways of experiencing phenomena (Anna Eckerdal, Anders Berglund, 2005).

Lessons

This research used a phenomenographic research methodology (Anna Eckerdal, Anders Berglund, 2005). Table 1 shows the lesson plan for teaching OOP concepts.

Lessons	Subject	Contents	OOP Concepts
1,2	Understanding Objects	<ul style="list-style-type: none"> ·Pre-interview ·Introduction of Dolittle ·Object creation and Modification 	<p>Object</p> <p>Method</p>
3,4	Understanding Method and Instance	<ul style="list-style-type: none"> ·Understanding Method ·Understanding Instance 	<p>Method</p> <p>Instance</p>
5	GUI programming	<ul style="list-style-type: none"> ·Figures creation and modification ·Button, Label, Field creation and modification 	Encapsulation
6,7	Game programming	<ul style="list-style-type: none"> ·Timer creation and modification ·Collision method modification 	Modularity
8	Final project	<ul style="list-style-type: none"> ·Final programming ·Evaluation(OOP concepts and Procedural thinking ability) 	

Table 1 Lesson Plan for teaching OOP concepts from Grade 3 to Grade 6

At the beginning of the 1st class, a pre-survey which asks about programming using experiences was given. After the survey, participating students used Dolittle and moved objects inside the program. They learned how to operate and correct errors in the program. During the 3rd and 4th class, the students learned methods and instances by operating the object. In 5th class, the students made figures and used methods for moving those figures. Additionally, the students learned event programming which relates objects inside programming with GUIs such as buttons. The next 2 classes taught the students how to program games using timer and collision method. For the final class, the students were asked to build programs based on what they have learned from previous classes.

A total of 8 lessons were held with 12 male students ranging from Grade 3 to Grade 6. Dolittle was used for 1 hour in 8 different lessons. Final project consisted of game programming in order to develop an interest for programming. During the lessons, the teacher told students the name of "Object" but did not tell students concepts of OOP. The outcomes of each process were then uploaded to the website and these results were shared among students. During the final lesson, students were asked to create their own programs based on the training they had received. This survey used the rating scales and contained evaluation process after each lesson. Formative assessment was made through quizzes at the end of each lesson.

Evaluation

The tests included qualitative and quantitative evaluation. Evaluation consisted of pre-tests, process tests, and post-tests. Pre-tests used Group Assessment of Logical Thinking (GALT) test and surveys (Shayer and Adey, 1981). GALT test evaluates student's cognitive development level. Survey evaluates whether elementary students learned programming and/or OOP concepts. Also, it evaluates whether they know the meaning of flow-chart. During Dolittle lessons, phenomenographic research was used to evaluate student's satisfaction. Post-tests evaluate OOP concepts and procedural thinking ability. Table 2 shows the evaluation criteria, which are

OOP concepts that are likely hard for novices to understand (Noa Ragonis & Mordechai Ben-Ari, 2005).

Problems relating to OOP concepts should be used in real-life problems, not just relating to coding or computer system. Problems relating to procedural thinking ability consist of “Elevator’s Operation Sequence” and “Vending Machine Operation”. The evaluation used in “Elevator’s Operation Sequence” was objective, while “Vending Machine Operation” was subjective. In addition, “Vending Machine Operation” used flow-chart as an additional method.

Domain	Evaluation Criteria
OOP Concepts	Object Creation
	Identification of Objects
	General understanding of Method
	General understanding of Instance
	Understanding Encapsulation
	Understanding Modularity
Procedural thinking ability	Elevator’s operation sequence, Vending machine operation

Table 2. Evaluation Criteria about OOP concepts and procedural thinking ability

4. Results and Discussion

Pre-tests

The result of pre-tests shows that most students belong to concrete operational period and there is no one who understands the OOP concepts. The results of tests for checking students' initiative learning ability can be divided into GALT and questionnaires. GALT measures 6 kinds of logical thoughts such as conservation logic, combinational logic, proportional logic, probabilistic logic, correlation logic, and controlling variables logic. Table 3 shows the result of cognitive development type of the students.

	concrete operational period	transitional period	formal thinking period
Number of students	8	3	1
	Grade 3 - 5	Grade 5,6	Grade 6

Table 3 Type of cognitive development after GALT test

As shown in Table 3, most of the students belong to concrete operational period in which their abstract conception is not formed completely yet. Only one student was in the period of formal thinking.

As a result of questionnaires, students never heard of OOP concepts such as objects, encapsulation, and modularity. It was also found that students' never heard of the term 'flow-chart'.

Process tests

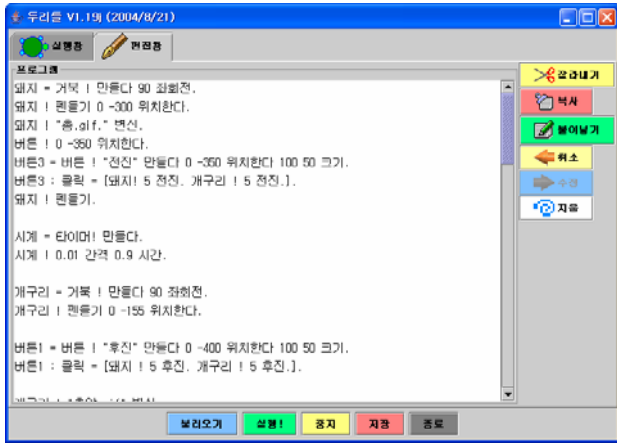
After analyzing process tests, it indicated that the procedural thinking ability and abstract thinking ability of some concrete operational period students were improving.

According to many researches, Dolittle generally raises students interest in programming (HeyMinGil, 2004, SuKyungYoo, 2005). However, many students experienced difficulties from Lesson 6, which goal was to devise various problem-solving methods while they learn programming with Dolittle.

For example, Grade 3 students lost interest when they encountered "timer" object. Confusion arose from the fact that the "timer" was not a visible object. The results showed that Grade 3 students were no longer interested in OOP when they were required to 'think outside the box'.

In case of Grade 4 students, not all of them lost interest when taking process tests. Dolittle was an appropriate educational programming language tool for Grade 4 students without cognitive load. Program which was made by Grade 4 students showed that they improved procedural thinking ability as well as abstract thinking ability.

Figure 2 is about an example of game programming made by Grade 6 students.



```

Pig = turtle ! create 90 leftturn.
Pig ! penup 0 -300 position.
Pig ! "gun.gif" looks.

Frog=turtle ! create 90 leftturn.
Frog ! penup 0 -155 position.

Button ! 0 -350 position.
Button3=button ! "forward" create 0 -350 position 100 50 size.
Button3 : click =[pig ! 5 forward. Frog ! 5 forward.].
Pig ! penup.

Time = timer ! create.
Time ! 0.01 period 0.9 time.

Button1=Button ! "backward" create 0 -400 position 100 50 size.
Button1 : click = [Pig ! 5 backward. Frog ! 5 backward.].
        
```

Figure 2 A sample Dolittle program written by Grade 6 student

The transitional period level student made a monster game as shown in Figure 2. The goal of the game is to shoot the flying monsters. The program used 6 "turtle" objects and 8 "button" objects. In the program, the monster can change its movement when the monster was shot. Moreover, the objects were copied in order to make more monsters and each was assigned a different time value in the timer. Also, the students understood the procedural process of the game programming and gained more confidence when they completed their own programming.

Post-tests

The post-tests results indicate that most students understand OOP concepts as well as procedural thinking ability using Dolittle. A test about the OOP concepts comprehension was conducted. Figure 3 shows the quantitative evaluation result of students' understanding of OOP concepts. After evaluating the results, most students were able to comprehend OOP concepts.

Apart from 2 students, all students were able to solve the problem that was connected to real life objects, methods, instances, encapsulation, and modularity. The 2 students only understood the concept of objects, methods, and instances. These results show the limitation that is connected to real life and abstract thinking ability of computing. Figure 3 shows the students' average score of OOP concepts problems. Most students could distinguish instances and methods and learn the major concepts of OOP.

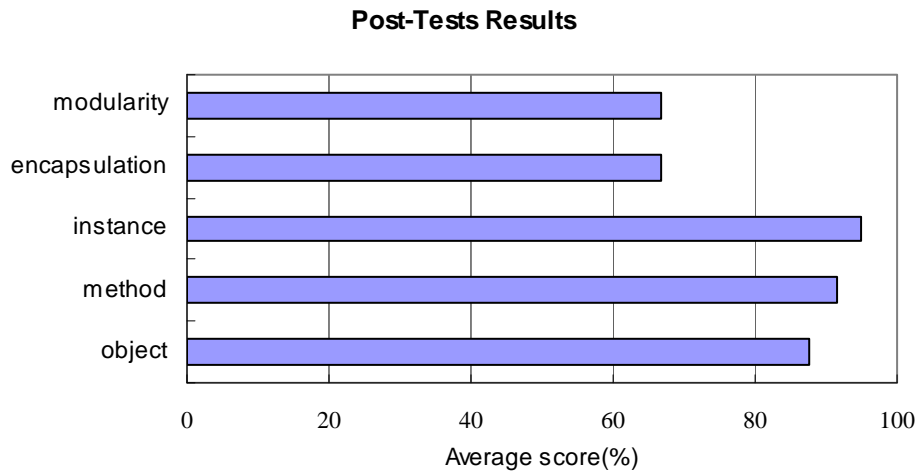


Figure 3 The result of post-tests is that most students understand OOP concepts.

After conducting OOP concepts test, procedural thinking ability test was given to the students. The evaluation of procedural thinking ability test shows that most students understood the method of problem solving. During the lessons, they did not learn programming education using flow-chart. Although students did not cognize the concept of flow-chart, it is worthy of notice that most students could solve using flow-chart. The students naturally understood the processes of the conditional and repetition though they could not understand the diagram.

Conclusions

This paper researched that learning computer programming with Dolittle improves the procedural thinking ability of students and helps them understand the concepts of OOP. Additionally, it showed that the elementary students gained confidence upon completion of their own computer program using Dolittle. Despite the fact that Grade 4 students are in concrete operational period, it was found that Dolittle could help elementary students understand OOP concepts and develop procedural thinking ability as early as Grade 4. The post-tests results provided further evidence towards the fact that abstract thinking ability and procedural thinking ability can be developed at early stage of education. However, the experiment in this paper was only based on 12 students. In order to further support aforementioned results, experiments with more participants are recommended for future study.

References

- Anna Eckerdal, Anders Berglund(2005), *What Does It Take to Learn 'Programming Thinking'?*. ICER'05.
- Bishop, J.M.(1997) *A Philosophy of teaching Java as a first teaching language*. ACM SIGCSE Bulletin, 29(1),pp140-142.
- Dan Aharoni(2000) *Cogito, Ergo Sum! Cognitive Processes of students dealing with data structures*. SIGCSE.
- Detienne, F(2001) *Software design-Cognitive Aspects*. London:Springer.
- Hadjerrouit, S.(1998) *Java as first programming language : A critical evaluation*. ACM SIGCSE Bulletin, 30(2), 43-47.
- HeeKang(2005), *Learning Motivation induction of under-achievement student using Dolittle*. Korea University.
- HeyMinGil(2004) *Application and Evaluation of Object-Oriented Educational Programming Language 'Dolittle'*. Korea University.
- Holland, S. Griffiths, R & Woodman, M(1997), *Avoiding object misconceptions*. ACM SIGCSE Bulletin, 29(10), pp131-134.
- Leron, U(1987) *Abstraction barriers in mathematics and computer science*. In J.Hilel(Ed.), Proceedings of the third International Conference for Logo and mathematics Education. Montreal.
- Ministry of Education and Human Resources Development(2005) *Guidelines for Information and Communications Technology Training in Elementary and Secondary Schools*. Ministry of Education and Human Resources Development.
- Noa Ragonis & Mordechai Ben-Ari(2005) *A Long-Term investigation of the Comprehension of OOP concepts by Novices*. Computer Science Education, pp203-221.
- Shayer, M. and P, Adey(1981) *Toward a science of science teaching : cognitive development and curriculum demand*. London : Heiemann educational Books.
- Susumu Kanemune, Shingo Fukui, Yasushi Kuno, Takako Nakatani, Rie Mitarai((2004) *Dolittle-Experiences in Teaching Programming at K12 Schools*.
- Stein, L.A.(1997), *Beyond objects*. Educations Symposiun, Conference on OOP Systems, Languages, and Applications, Atlanta, Georgia. Retrived October 30 2004 from <http://222.ai.mit.edu/projects/cs101/beond-objects.ps>
- SuKyung Yoo(2005), *Analysis on Learning Achivemnet, Instructional Design and development on 'Digital Logic Circuit' using Dolittle*. Korea University.
- Tony Jenkins(2002) *On the difficulty of Learning to Program*. 3rd Annual LTSN-ICS Conference, pp.53-58.
- Wadsworth, B. J(1989) *Piaget's Theory of Cognitive and Affective Development(4th ed.)*. N. U: Longman Inc.
- Woodman, M. & Holland, S(1996) *Form software user to software author:An initial pedagogy for introductory object-oriented computing*. ACM SIGCSE Bullen, 28(SI), pp60-62.