Graphical Tutoring of Message Passing
to Promote Understanding of Java Framework

Atsuko Ueno, Hiroshi Taguchi, and Hiromitsu Shimakawa
Computer Science Department
Ritsumeikan University
Kusatsu, Shiga, Japan

Abstract

Many students feel that Java programming is difficult. It attributes to the difficulty of programming with Java frameworks and the problem of the programming education environment. This paper proposes a method that supports learners to understand Java frameworks. The method provides self-study support tool to bridge a lecture and a programming exercise. This tool uses the UML sequence chart to provide a learner with knowledge to describe messages. The method enables to guide a learner, checking code dependency and reference of objects while the learner specifies his program. An experiment has proved that the method improves understanding of learners.

1. Introduction

Many universities teach Java as a prime object oriented programming language. Most students, however, feel difficulties in studying Java programming. Those students are not necessarily novice programmers. Some of them have already achieved programming abilities with procedural programming languages such as C language. It is crucial that many students struggle with high hurdles in Java programming in spite of their programming experiences with other programming languages. It may attribute to both complexity in learning Java programming and environments of programming education.

One of the reasons why learners feel complexity in learning Java programming is a great variety of constraints in Java programming. Java forces programmers to observe many constraints such as frameworks and the message passing. Java programming founds on reuse of existing software components. A Java framework is a combination of characteristics of components and ways to build a program of components. It is inevitable to understand the order of the creation and the construction of components. Message passing is an activity to send a message from one object to another. The execution of a method in Java programming corresponds to message passing from a sender object to a receiver object. Since any code in a Java program founds on message passing, programmers must always be conscious of a sender and a receiver. In addition to that, they must pay attention to arguments passed to a receiver. When a message is specified in a sender, the sender has to be able to refer all arguments in the message, using variables available in the current context.

The current environments of programming education also have a problem in that all students are not given guidances enough to understand Java programming. Most universities teach programming with a pair of lectures and exercise classes. A lecture to teach knowledge on a Java framework is followed by an exercise class to put the knowledge into practice. Since many learners cannot understand the knowledge only in a lecture, they often feel difficulties for Java programming in an exercise class. Though they need guidances from instructors, shortage of instructors and teaching assistants in an exercise class prevents them from receiving guidances enough to understand practical usages of a Java framework.

To solve these problems, the paper proposes a method to support learning of Java frameworks. Based on the method, we have developed a tutoring system which has the following features:

- display of the knowledge necessary to write codes on a UML sequence chart, and
- guidances to learners while they are programming, using animation.

The paper explains the tutoring system based on the method. The tutoring system supports for learners to confirm the knowledge taught in a lecture by themselves. In the self learning time after a lecture, learners study the knowledge of a Java framework to prepare a programming exercise. The tutoring system graphically presents a learner with
a sender/receiver of a message, variables which can be referred to specify a message to a receiver in the current context of the sender method. It also checks learner codes from the view points of description order of codes, and availability of variables for the reference. The graphical presentation makes learners conscious of information necessary to use a Java framework, while the check realizes guidances to learners during their programming. The timely guidance to learners in the self learning time also gives a solution for shortage of instructors and teaching assistance. With the tutoring system, learners can acquire the knowledge to use a Java framework, because it plays a roll of instructors and teaching assistants to guide them during the programming.

2. Education for Java Programming

2.1. Hurdles in Java Programming

In Java programming, a program is assumed to be constructed with predefined software components. Many constraints should be satisfied so that these components may work well together. A programmer has to understand the constraints to build up a program from the components. To understand Java programming, the constraints would be hurdles to be overcome. As the first hurdles, most Java novice programmers would face the concepts of ‘framework’ and ‘message passing’.

The framework[4] is a combination of characteristics of components and ways to build a program of them. The framework strictly specifies ways to create new components and to build up them, which are realized with message passing. In this process, messages must be specified, following a predefined order. A learner of Java programming cannot construct a program, even though he knows characteristics of each component. He is required to correctly understand a method to construct them. Since the method is implied by model codes, he has to spend much time to read carefully textbooks which explains model codes.

Java programs consist of various message passing between objects. Figure.1 show an example of message passing which occurs in a Java program using Swing components. The upper part of the figure explains the behavior of setLayout() method, while the lower part shows the message passing in the method execution.

One method execution corresponds to a message passing from a sender object to a receiver object. In the figure, main() method sends setLayout() message to receiver container, an instance of Container class. A Java programmer must be conscious of a sender and a receiver of a message he wants to use. In addition to that, when a programmer specifies an object as arguments in the message invocation, he must know whether the object can be referred in the sender.

2.2. Current Status of Education

In many universities in Japan, Java programming is taught to learners, using a lecture succeeded by an exercise in each week. Lectures give knowledge on the object oriented programming and the Java frameworks, and so on. Exercises provide learners with opportunities for them to write correct Java codes with the knowledge. Though learners grasp a rough image of the Java framework in the lecture, they cannot get deep knowledge on actual usage of it without exercises. However, if learners do not understand message passing yet, they face not only syntax errors but also errors in program execution in exercises. The lack of knowledge on the Java framework and the message passing prevents learners from achieving Java programming abilities. In addition to that, a few instructors teach dozens of learners both in lectures and exercises in many universities. It is difficult for instructors to teach enough knowledge to each of learners.

2.3. Related Works

We can find several existing works related to Java programming education. A GUI tool teaches interactions between objects in [5]. An Web system proposed in [2] enables feedbacks from instructors to learners for frequent communication. These works, however, do not address tutoring specific to Java framework.

Tutoring for learners are studied in [8, 6]. A system in [8] assigns learners with fill-in-the-gap exercises to confirm the validity of codes in the gap. The quick feedback to learners decreases the burden of instructors. Though the system can find errors in codes, it cannot judge the conformance of the codes to Java framework. Aiming to support learning of C, Pascal, and Smalltalk, [6] realizes a tutoring system, which
can point out mistakes during programming. Nonetheless, it pays no attention to understanding of Java framework, which is a high hurdle for novices for Java programming.

3. Graphical Tutoring of Java Programming

3.1. Learning Flow with Tutoring System

Aiming at the improvement of understanding of the Java framework and the message passing, this paper proposes a graphical tutoring method, and explains a tutoring system based on the method. Learners are expected to study by themselves with the tutoring system after each lecture to establish knowledge taught in the lecture. Learners can examine their understanding of the knowledge in an exercise class following the lecture.

The features of the proposed method are enumerated as

- display of the knowledge necessary to write codes on a UML sequence chart, and
- guidances to learners while they are programming, using animation.

In Java programming, a sender is required to send a message to a receiver with variables it can refer in its current context. In other words, the sender should prepare variables needed for the message in its context. In this paper, the variables the sender can refer in its context as to an available variable list (AVL). The proposed method represents a Java program with a UML sequence chart, which shows a sender, a receiver, the sequence of messages to be sent, and an AVL. Every time a learner writes codes to send a message, the tutoring system requires him to specify a sender and a receiver and select arguments from AVL. It promotes the learner’s understanding for message passing.

The Java framework prescribes the order of codes to build up software components. The tutoring system checks the order of codes when a learner specifies a message. Every time a learner specifies a code to send a message, the tutoring system checks whether variables used in the code belong to the AVL of the sender. The check mechanism enables the tutoring system to guide learners while they are writing Java codes. The timely guidance to learners during the programming enables learners to acquire the knowledge to use a Java framework. The introduction of the tutoring system improves programming education environments because the system takes the place of instructors and teaching assistants to guide learners in exercise classes.

Figure.2 shows the flow of the learning. An instructor prepares assignments used in the tutoring system in advance. Learners can study a typical usage of a Java framework, solving the assignments. The tutoring system displays a UML sequence chart. In the initial state, nothing is specified in methods in the UML sequence chart. A learner tries to fill up methods with selection of messages, each of which triggers the checks based on the order of codes and the availability of variables in the context. According to results of these checks, an animated tutor in the system gives a guidance to a learner. Instructors are expected to register model codes of Java. The tutoring system compares the model codes with those are specified by a learner. The tutoring system indicates the matching rate of these codes.

3.2. Code Dependency

A Java program consists of codes described to specify functions such as creation and combination of components. Java frameworks prescribe how codes are described. Especially, the order of codes is important. For example, in the Swing framework, no component can be placed on a panel if the panel has not created yet. In our method, the order of codes are represented by code dependency. We define that code $A$ depends on code $B$, if $B$ should be described before $A$. We represent it with the notation $A \rightarrow B$.

The code dependency is classified into the two categories:

- one coming from instance existence, and
- one coming from an instance state.

The code dependency coming from instance existence occurs when one code depends on another code which specifies the creation of an instance. Figure 3 shows an example in the Swing framework, where code 1 creates a button and it will be placed on the panel with code 3, respectively. An instance of JButton is set to variable button in code 1, while button is used in the add message as its argument in code 3. The example indicates code 1
must be described before code 3 because of the instance creation. Without code 1, code 3 would have nothing to be specified in the argument. On the other hand, the code dependency coming from an instance state implies that the state of an instance should be set to a specific one before a certain code is executed. In figure 3, code 2 specifies a component layout manner of the panel. code 2 should be specified before code 3, because the layout manner of the panel needs to be set appropriately before the placement of the button. This kind of the code dependency is usually explained to learners as “the layout manner must be specified in a container before a component is placed on the container”.

The explanation of code dependencies plays a vital role for learners to understand Java frameworks. When an instructors make an assignment, he is required to enumerate all code dependencies which may occur in the assignment. He is expected to state the explanation which promotes the understanding of learners.

3.3. Reference from Object

Variables in a Java program indicates references to instances and primitive values. The scope of a variable depends on where the variable is declared; variables declared inside an instance can be referred to anywhere in the instance, while variables declared in a method can be referred to only from the inside of the method. In the proposed method, an AVL (available variable list) is displayed, reflecting a scope of each variable. To send a message in a method, it is necessary to be able to use reference of instances specified as a receiver and arguments. The proposed method pays attention to the necessity as the object reference availability. When a learner specifies a code to send a message, it is checked whether target instances can be referred to through values of variables in the code.

3.4. Validation

In the proposed method, an instructor is expected to prepare an assignment, which should be solved with objects taught in a lecture. Learners study a Java framework with the tutoring system which checks the code dependency and the references from objects. In the tutoring system, learners study by themselves, specifying codes to send messages. Since the tutoring system knows objects appearing the assignment, it can show a list of constructors and methods of an object specified as a receiver. When a learner specifies a code to send a message to an object, he can selects a constructor or a method from the list. He can also select arguments of the message from the AVL.

The tutoring system checks the validity of the code from the view points of the code dependency and the object reference availability. The checking process is shown in figure 4. When a learner specifies a code in a method, the
tutoring system checks whether all codes it depends are included in the same method. It is also checked whether all instances specified as message arguments can be referred from the context of the sender. If the method does not meet any of them, the tutoring system informs the mistake of the learner, and gives a guidance to lead him a correct answer. Otherwise, the tutoring system compares the method of the learner with a model description of the method. The coincidence of the two methods triggers update of the matching ratio of codes, which is displayed on the tutoring system.

4. Tutoring System

4.1. System Components and Configuration

The tutoring system is implemented with Adobe Flash[1]. Instructors are expected to make an XML file corresponding to an assignment to learners. Each XML file specifies blank sender methods, which should be filled up by learners, and enumerates receiver methods required to fill them up. For each method, the XML file also contains description on code dependency.

The execution of the tutorial system starts from reading an XML file. At the initialization, blank sender methods are displayed on the screen of the system, with all receiver methods expanded inside the system.

4.2. Assignment Specified with XML

Instructors need to specify the following two kinds of XML files for an assignment:

- **myclass.xml** specifying blank sender methods, and
- **class.xml** enumerating receiver methods.

Instructors can make new contents of assignments, exchanging these XML files.

In myclass.xml, an instructor specifies a class and types of its methods to be filled up by learners. Based on the XML file, the tutoring system create a screen for learners to operate on. An example is shown in figure.5, which specifies Hello class and main method. The description for main method includes a list of model codes. In the model codes, an instructor indicates combinations of a receiver and its method. The example illustrates the model codes such as the creation of JFrame, the execution of getContentPane method of JFrame, and so on.

On the other hand, class.xml enumerates receiver objects and their methods, which are necessary to fill blank sender methods up. The description of each method states both arguments of the method and codes the method depends on. The description method of class.xml is illustrated with figure.6, which takes setLayout method as its example. More than one arguments should be described in the file, including correct ones and wrong ones. As for wrong arguments, the description includes a reason to explain why the selection of the arguments is wrong and a guidance for learners. As the figures shows, param tag inside setLayout specifies arguments. Since the second argument jframe is a wrong one, hint attribute of it indicates the reason and the guidance to lead learners to the correct selection. Codes setLayout message depends on are represented with prerequisite tags, with which receivers and their methods in the codes are encompassed. In the example, getContentPane method of JFrame is specified as a code setLayout message depends on. When a learner write a code of setLayout message, the tutoring system checks whether getContentPane method is described before it in the same sender method. If it is not described, the sentence in hint tag is presented to the learner.

1The XML file includes some Japanese sentences because the tutoring system aims to support self learning of Japanese students.
4.3. System Behavior

Figure 7 shows a screen shot when a learner is using the tutoring system. In the figure, a dark gray area with a tab and a light gray area correspond to a class and a method, respectively. The screen shot indicates a situation where a constructor of Greeting class is selected. When the system is started, all sender methods are vacant. Learners can specify the contents of the sender method with only mouse operations. The system provides learners with functions for

- creating an instance,
- sending a message to a receiver,
- assigning a value to a variable, and
- undoing an operation.

Operations to send a add message to an instance of container are illustrated with the figure. 7. As the first step, a learner selects a sender and a receiver of the message passing. The figure indicates Greeting constructor as the sender and an instance of Container as the receiver, respectively. The selection draws an arrow from the sender to the receiver to show the direction of the message passing. Since the receiver is identified, the tutoring system can show a list of messages of the receiver to the learner, using class.xml file. As the second step, the learner selects a message name and arguments. The tutoring system checks the selection from the view points of the code dependency and the object reference availability. If there is any mistake in the selection, a guidance and a hint are given from an
animated tutor, which is located in the right bottom side of
the screen. The coincidence of a learner code with a model
code increases the matching ratio on the screen.

5. Experiment for Java Novices

5.1. Outline of Experiment

We have conducted an experiment to confirm the pro-
posed method improves the understanding of learners for
a Java framework. The learners in the experiment are 10
undergraduate students. Though they have studied Java lan-
guage specification in lectures, none of them has much pro-
gramming experience with Java. We make the students
study according to the sequence of a lecture, an exercise,
and tests as follows:

1. a placement test of programming in 30 minute,
2. a lecture on a Java framework in 30 minutes,
3. self learning in 30 minute,
4. a written examination in 30 minute, and
5. programming exercise using two assignments in 50
minutes

In this experiment, the Swing components are adopted as
a Java framework. Using scores of the programming place-
ment test before the lecture, the students are divided into
two groups so that the average scores may be equal in both
group. In the self learning after the lecture, one group is
forced to use the tutoring system, while the other group is
required to study contents of the lecture without the tutor-
ing system. We have compared the understanding of the
two groups based on scores of the written examination and
the programming exercise after the self learning.

5.2. Experiment Result

Table.1 indicates the average scores of each group in the
placement test, the written examination, and the program-
ing exercise using two assignments. At the programming
test before the lecture, the two group have the same score.
After the self learning, however, the group learning with the	
tutoring system is superior to the group learning without it,
as for the score in the programming exercise. On the other
hand, the group learning without the tutoring system excels
the group learning with the tutoring system in the average
scores in the writing examination.

We have conducted a questionnaire for 5 students. The
result is summarized in table. 2. For each item, the students
answer in the following way. The learner is

<table>
<thead>
<tr>
<th>No</th>
<th>items in questionnaire</th>
<th>answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The system improves understanding of Java programming</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>The system improves understanding of Swing framework</td>
<td>3.6</td>
</tr>
<tr>
<td>3</td>
<td>Self learning with the system excels one without the system</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Graphical Representation of programs is preferable</td>
<td>3.8</td>
</tr>
<tr>
<td>5</td>
<td>Guidance during programming leads to error identification</td>
<td>3.8</td>
</tr>
</tbody>
</table>

1. strongly against the item.
2. against the item a little.
3. neutral.
4. for the item a little.
5. strongly for the item.

In most of items, the students are supportive for the tu-
toring system, rating it worthy more than 3. Especially, all
students have rated the system with 4 for the forth item,
which proves the tutoring system promotes understanding
of Java programming. However, the system is not evaluated
effective in term of the third item.

5.3. Discussion on Result

In the programming exercise after the self learning, the
students using the tutoring system make better scores than
the students without it. It attributes to a smooth flow of
learning. In the proposed method, an instructor specifies
his own assignment on the tutoring system. Since the in-
structor can use consistent contents in all of the lecture, the
self learning, and the exercise in the experiment. The con-
sistency results in a smooth flow of learning, which enable
for an instructor to transfer the contents to learners in an ex-
plcit manner. The tutoring system facilitates an instructor
to teach a typical way of thinking in Java programming.

Let us consider the writing examination. There is much
possibility of few guidances from the tutoring system in
poor score of the students using the tutoring system. In the current implementation, the tutoring system gives guidances only when it finds mistakes in learner selection. Since the learners use the tutoring system in the 30 minute self learning time, they have a few chances to be notified of their mistakes. On the contrary, they can be aware of their mistakes in the programming exercise because of compile errors.

The questionnaire reveals that self learning with the tutoring system is not supported by the learners so much. It might come from the large amount of operations in the tutoring system. Learners would not prefer tools with a lot of operations. The tutoring system should be improved in its usability.

6. Conclusion

In this paper, a new method is proposed to promote Java novice programmers to understand Java frameworks.

The tutoring system based on the proposed method plays a role as a self learning tool, which is used after a lecture on a Java framework, but before the programming exercise. The tutoring system represents a Java program with a UML sequence chart indicating senders and receivers of message passing. The graphical representation facilitates understanding what is necessary for coding. The method finds out mistakes of learner, checking the code dependency and the object reference availability. Because of the check on the spot, the tutoring system gives guidances to learners during their programming, which leads to improvement programming education environment. The tutoring system is implemented with Adobe Flash. It enables for instructors to change assignments to learners, using the specification of contents with XML files.

An experiment has been conducted to confirm the effectiveness of the method. It reveals that learners using the tutoring system in self learning get higher scores in a programming exercise than learners without it. The experiment, however, implies that the method leaves some issues to be improved, especially complex operations of the tutoring system. Our future works include refinement of the method through practical uses of the tutoring system.

References