Improvement of Fill-in-the-blank Questions for Object-Oriented Programming Education

Miyuki Murata (National Institute of Technology, Kumamoto College)
Naoko Kato (National Institute of Technology, Ariake College)
Tetsuro Kakeshita (Saga University)

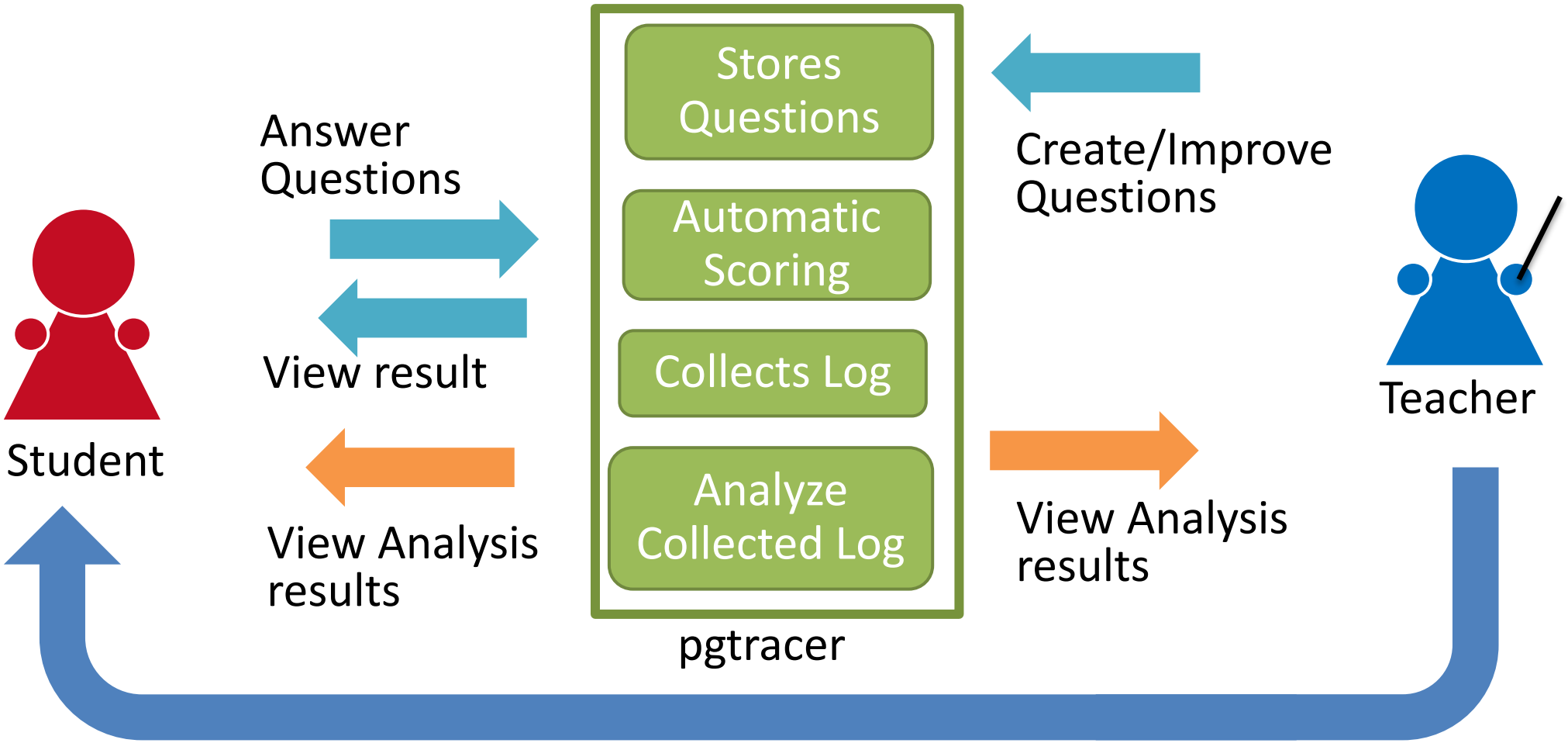
Background

Object-oriented technology is important to improve software quality from various perspectives

Developing a Programming Education Support Tool pgtracer for C program

- Provides fill-in-the-blank questions to students
- Automatically collects learning activity data of the students and provides data analysis functions.
- Obtain useful knowledge for C programming education

Programming Education Process utilizing pgtracer



Our Previous Observations

We provide programming homework using pgtracer **for C programming** to the students since 2016

- The understanding of the program becomes better for the students better understanding the concept of the trace table.
- The students having high programming skill take time to consider before filling the blanks.
- Student's programming skill affects their answering process.
- Usual students tend to fill the blanks in the order from the top.
- More difficult for the students to fill in the program than to fill in the trace table.
- The accuracy rate is low for the blanks related to iteration in the program.

Research Questions

Research Purpose

- Obtain useful knowledge in Java programming education
  - Extend pgtracer for Java program
  - Analyze the collected data by pgtracer using LA methods

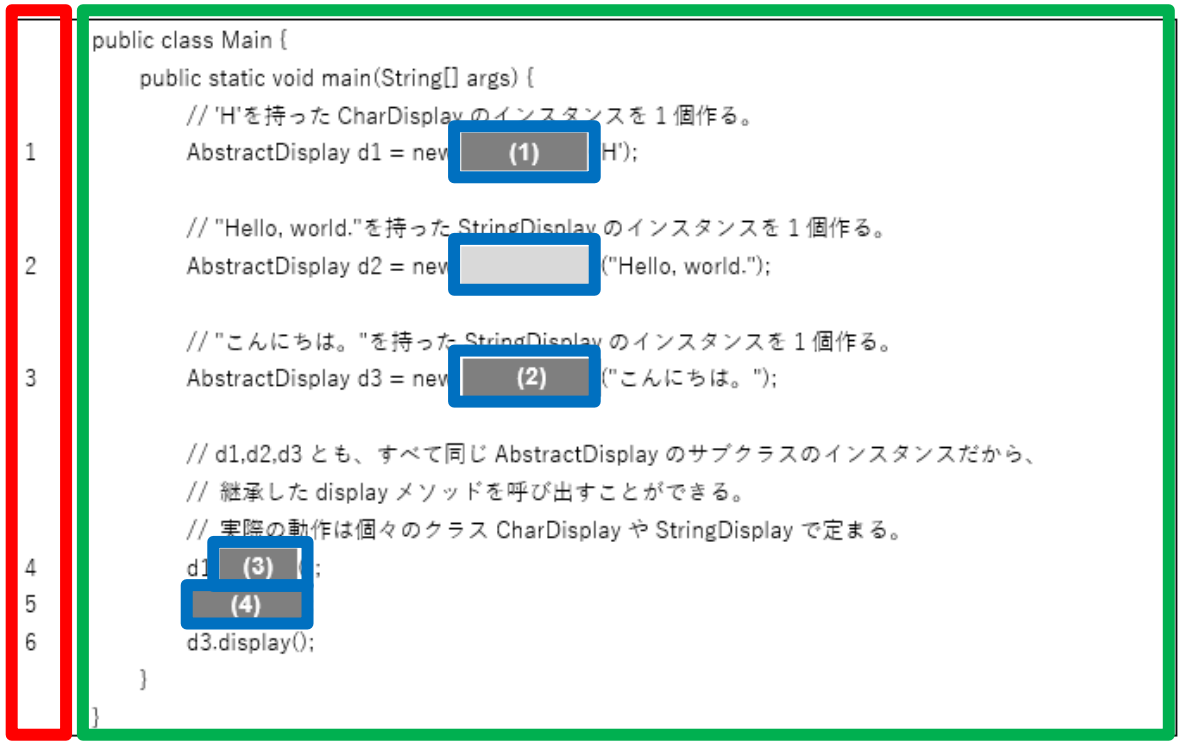
In This Paper, We

- Define the blanks for a Java source program and a trace table. The trace table represents the execution sequence and output.
- Adjust the difficulty level of the fill-in-the-blank questions by defining the blanks.
- The created fill-in-the-blank questions were used in an actual class and evaluated by the students.

Extension of Fill-in-the-Blank Question for Java Program

<table>
<tr><th></th><th>Program</th><th>Trace Table</th></tr>
<tr><td>Representation</td><td>• Contains multiple source files, each corresponds to a class</td><td>• Order of executing steps for each object
• Message passing between objects</td></tr>
<tr><td rowspan="2">Definition of blanks</td><td>• Token
• Sequence of tokens
• Expression
• Statement, etc.</td><td>• Variable value
• Step number
• Variable name
• Method name</td></tr>
<tr><td colspan="2">• Define the type of blank
• Introduce a blank which does not need to answer</td></tr>
</table>

Representation of Program



Source Code

Fill-in-the-blank question contains multiple source files.

Step number

- Assign to each statement such as assignment, method call, if, else, etc.
- Correspond to the step numbers in the trace table

Blanks

(n)

Blanks which requires answer

Blank which dose not require answer
(Called Ignorable blank)

- Hide the statements containing hints

Difficulty level of a blank can be controlled more flexibly

Representation of a Trace Table

- Define a trace table for each instance
- Represent each message passing in the trace table
- Expression of instance is “ClassName#X” (X:the oder of generation)
- If a statement contains multiple operations, the statement will be split into multiple rows.

Related to Message Passing

Representing the Values

Caller of the Method			Step of Called the Method			Argument of the Method	Instance Value	Local Variable of the Method	Output / Return Value of the Method
Caller of			Class	method	step	char	char	int	output
object	method	step				ch	ch	i	
Main	main	1	CharDisplay		1				
Main	main	1	CharDisplay	CharDisplay	1	H	H		
Main	main	(1)	AbstractDisplay	display	1		H		
			(2)	open	1		H		<<
			AbstractDisplay	display	2		H	0	
			AbstractDisplay	display	2.1		H	0	

Example of Trace Table(Instance ID = charDisplay#1)

Expression of Method Call in a Trace Table

Caller Method (Main)

Main.java

```
   public class Main {
      public static void main(String[] args) {
        // 'H'を持ったCharDisplayのインスタンスを1個作る。
1       AbstractDisplay d1 = new CharDisplay('H');
        // "Hello, world."を持ったStringDisplayのインスタンスを1個作る。
2       AbstractDisplay d2 = new StringDisplay("Hello, world.");
```

Called Method (charDisplay)

CharDisplay.java

```
   // CharDisplayは、AbstractDisplayのサブクラス。
   public class CharDisplay extends AbstractDisplay {
1     private char ch;                    // 表示すべき文字。
      public CharDisplay(char ch) {       // 渡された文字chを、
1       this.ch = ch;                     // フィールドに記憶しておく。
      }
```

Main

Step of the Method			Local Variable of the Method			External Object		
Object	method	step	AbstractDisplay d1	AbstractDisplay d2	AbstractDisplay d3	CharDisplay charDisplay#1	StringDisplay stringDisplay#1	StringDisplay stringDisplay#2
Main	main	1				CharDisplay		
Main	main	1	charDisplay#1					
Main	main	2	charDisplay#1				StringDisplay	
Main	main	2	charDisplay#1	stringDisplay#1				
Main	main	3	charDisplay#1	stringDisplay#1				StringDisplay
Main	main	3	charDisplay#1	stringDisplay#1	stringDisplay#2			
Main	main	4	charDisplay#1	stringDisplay#1	stringDisplay#2	display		
Main	main	5	charDisplay#1	stringDisplay#1	stringDisplay#2		display	

Message Sending to External Object

CharDisplay#1

Caller of the Method			Step of Called the Method			Argument of the Method	Instance V
Caller of object	method	step	Class	method	step	char ch	char ch
Main	main	1	CharDisplay		1		
Main	main	1	CharDisplay	CharDisplay	1	H	H
Main	main	(1)	AbstractDisplay	display	1		H
			(2)	open	1		H
			AbstractDisplay	display	2		H
			AbstractDisplay	display	2.1		H
			(3)	print	1		H

Caller object, method and step number

Class which defines the called method and step number

Argument of the Method

Blank within a Trace Table

- The value of each cell
  - variable values, output values, step numbers, object identifiers, class names, method names, etc.
- Sub-items of method arguments, instance variables, local variables, and external objects
  - a blank can be defined for their data types or class names, variable names, or object names
- Ignorable blanks can be defined

Caller of the Method			Step of Called the Method			Argument of the Method	Instance Value	Local Variable of the Method	Output / Return Value of the Method
Caller of			Class	method	step	char	char	int	output
object	method	step				ch	ch	i	
Main	main	1	CharDisplay		1				
Main	main	1	CharDisplay	CharDisplay	1	H			
Main	main	(1)	AbstractDisplay	display	1				
			(2)	open	1				<<
			AbstractDisplay	display	2			0	
			AbstractDisplay	display	2.1			0	
			(3)	print	1			(4)	H
			AbstractDisplay	display	2			1	
			AbstractDisplay	display	2.1			1	

Example of Trace Table(Instance ID = charDisplay#1)

Creation of Fill-in-the-Blank Questions

Prepare 12 Fill-in-the-Blank Questions based on the course content

Three question levels according to the course contents and the order of teaching

Students can learn typical techniques of Java programming through learning design patterns

Introduction to Design Patterns in Java

Level	Order	Design Pattern	The section in the Textbook
Beginner			
	1	TemplateMethod	3
	2	FactoryMethod	4
	3	Iterator	1
	4	Composite	11
Intermediate			
	5	Decorator	12
	6	Strategy	10
	7	AbstractFactory	8
	8	Observer	17
Advanced			
	9	Adapter	2
	10	Builder	7
	11	Command	22
	12	Visitor	13

Development Policy of Fill-in-the-Blank Questions

	Beginner	Intermediate	Advanced
Blanks within Program	• Individual Token	• Token • Expression	• Token • Expression • Statement
Blanks within Trace Table	• Variable Value	• Variable Value • Instance identifiers • Methods	• Variable Value • Instance identifiers • Methods • Step Number • Class Name
Common Consideration	• Define approximately ten blanks per question. Here, ignorable blanks are not counted. • Create two problems for one topic with different difficulty levels. • Clarify the educational objective of each question. • Clarify the intention for each blank.		

Example of a Fill-in-the-Blank question

Topic	Template Method
Educational Objective	Recognize usages of abstract classes and abstract methods.

Blank#	Right answer	Intention of the Blank
(1)	CharDisplay	Understand a constructor call.
(2)	StringDisplay	Understand a constructor call.
(3)	display	Can describe method calls.
(4)	d2.display()	Can derive an instance and a method from the output.

main.java

```
    public class Main {
        public static void main(String[] args) {
            // 'H'を持った CharDisplay のインスタンスを 1 個作る。
1           AbstractDisplay d1 = new  (1)  ('H');

            // "Hello, world."を持った StringDisplay のインスタンスを 1 個作る。
2           AbstractDisplay d2 = new          ("Hello, world.");

            // "こんにちは。"を持った StringDisplay のインスタンスを 1 個作る。
3           AbstractDisplay d3 = new  (2)  ("こんにちは。");

            // d1,d2,d3 とも、すべて同じ AbstractDisplay のサブクラスのインスタンスだ。
            // 継承した display メソッドを呼び出すことができる。
            // 実際の動作は個々のクラス CharDisplay や StringDisplay で定まる。
4           d1.  (3)  ();
5           (4)  ;
6           d3.display();
        }
    }
```

CharDisplay.java

```
    // CharDisplay は、AbstractDisplay のサブクラス。
    public class CharDisplay  (1)  AbstractDisplay {
1       private char ch;                        // 表示すべき文字。

        public CharDisplay(char ch) {           // コンストラクタで渡された文字 ch を、
1           this.ch = ch;                       // フィールドに記憶しておく。
        }
```

StringDisplay.java

```
    // StringDisplay も、AbstractDisplay のサブクラス。
    public class StringDisplay        (1)        {
1       private String string;                  // 表示するべき文字列。
2       private int width;                      // バイト単位で計算した文字列の「幅」。

        // コンストラクタで渡された文字列 string を、フィールドに記憶。
        // それからバイト単位の幅もフィールドに記憶しておいて、後で使う。
        public StringDisplay(String string) {
1           this.string = string;
2           this.width = string.getBytes().length;
        }
```

Evaluation Experiment

Target Class to provide fill-in-the-blank questions

Exercise in Programming III

- for the third academic year of the computing courses at Saga University.
- The students first learn object-oriented programming using Java

Programming Skill of the Target Students

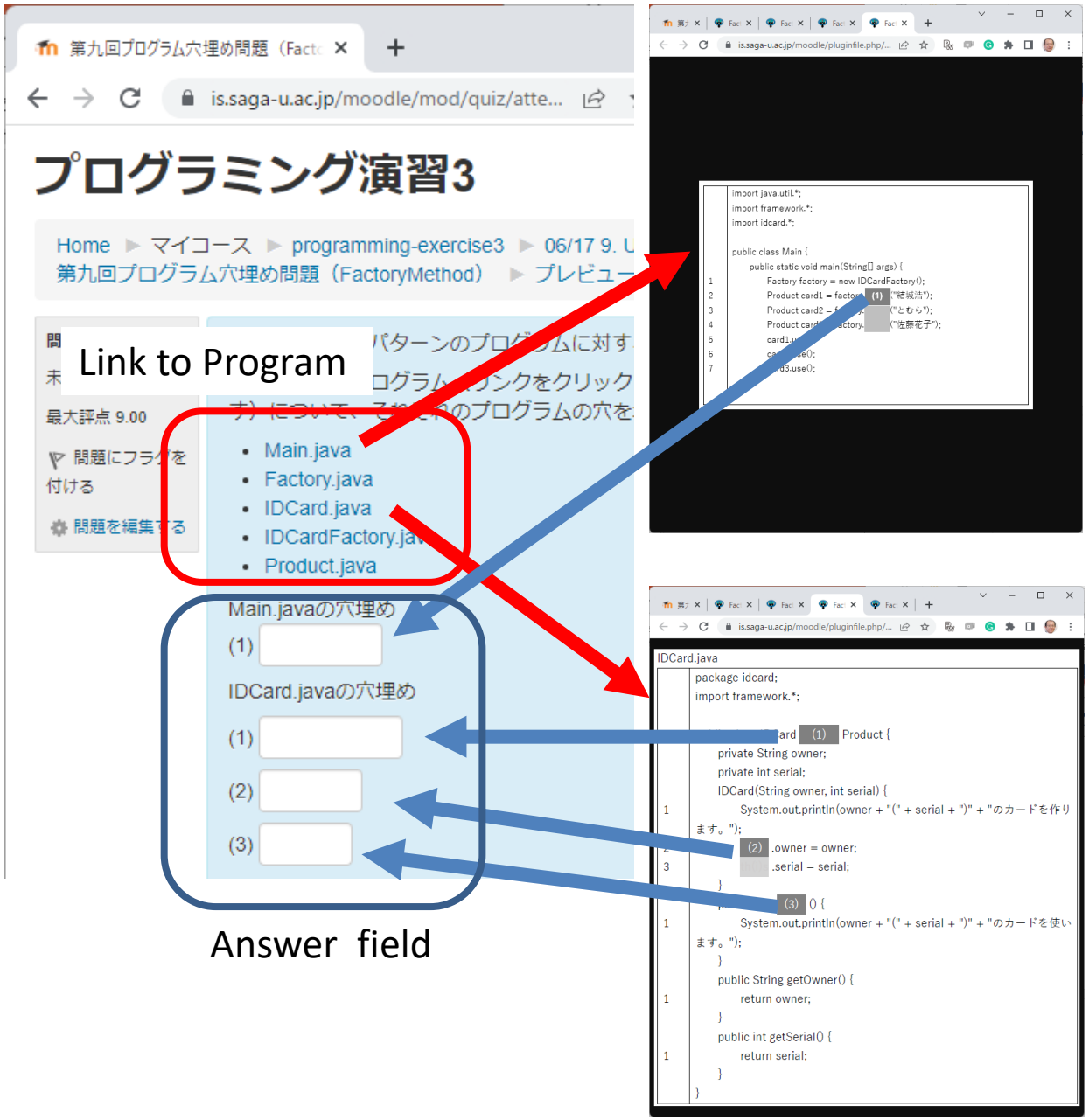
- Python in the 1st academic year
- Structured programming using C++ in the 2nd academic year

Purpose of Exercise in Programming III

- Intends to learn the purpose and usage of various tools used in the practical field of software development to maintain efficiency and high quality.
- In parallel with the lesson plan, the students are provided with fill-in-the-blank questions to help them master the Java language

Preliminary Trials

- Provide fill-in-the-questions using in Moodle, because the Java version of pgtracer in still under development
  - The programs and trace tables for the fill-in-the-blank questions were manually created and displayed as images.
  - programs and trace tables are displayed in a separate tab on the browser.
  - The answer fields were created using the embedded answer (cloze) feature of Moodle.



Questions and Number of Examinees

Week	Design Pattern	level	Problem Type	Problem ID	# of Blanks	# of Examinees
1	Exercise to understand fill-in-the-blank questions and trace tables		Program	Ex01_PR	6	67
			Trace table	Ex01_TR	4	
6	Template Method	Beginner	Program	Ex06_PR	12	71
		Beginner	Trace table	Ex06_TR	12	
7	Factory Method	Beginner	Program	Ex07_PR	9	58
8	Simple Example	Beginner	Trace table	Ex08_SP	2	67
	Iterator	Beginner	Trace table	Ex08_TR	10	
10	Factory Method	Beginner	Trace table	Ex10_TR	9	61
11	Composite	Beginner	Program	Ex11_PR	11	69
13	Strategy	Intermediate	Program	Ex13_PR	6	66
14	Observer	Intermediate	Program	Ex14_PR	10	65

Answer Result: “Program” Question Type

Level of Problem	# of Problems	Blanks containing a Token		Blanks containing Multiple Tokens	
		# of Blanks	Average Correct Answer Ratio (%)	# of Blanks	Average Correct Answer Ratio (%)
Beginner	3	22	76.5	10	57.7
Intermediate	2	11	67.2	9	41.9

Blanks containing multiple tokens are more difficult for the students.

At the intermediate level, we define blanks that require the students to understand the execution flow of the program to answer correctly.

The blanks reflected the difficulty level we intended.

Answer Result : “Trace Table” Question Type

Type of Blanks	# of Blanks	Average Correct Answer Ratio (%)
Call for constructor or method	16	72.5
Variable value (Basic type)	4	71.1
Return value (Instance)	5	64.0
Called Class	2	61.7
Step number of the calling method	3	46.0
Variable value (Instance)	1	38.8

The average correct answer ratio is low for the variable values that answer the step number and instance of the method caller.

It is difficult for the students to answer the questions related to object-oriented specific method invocations and instances.

Questionnaire Result

- 69 students answered.
- 63.8% of students answered that the questions were rather difficult.
  - This indicates that the level of difficulty of the problem is appropriate.
- 87.0% of students answered that the number of blanks within a question was appropriate.
  - This indicates that ignorable blanks could control the number of blanks.

15 Comments from Students

- Switching between program and trace table tabs is cumbersome (5 comments).
- Request explanation (4 comments).
- This was a complaint about wrong answers due to the difference between upper and lower case letters when typing(3 comments) .
- Increase in easy problems(1 comment)
- The ability to read programs a little better(1 comment)

Improvement of Fill-in-the-Blank Questions

Program

- Adjusting the location of the blanks so that the correct answer is unique.

(example) if(i < 5) Miss judged answer : 5 > i, i <=4

if(i< 5)

- Modified the program for the fill-in-the-blank questions because the textbook has been revised.
- Not to assign step numbers to the definition statement

Trace table

- Use the function which create automatically the trace table giving programs and input data.

Concluding Remarks

Conclusion

- Extended program and trace tables for Java programs
- Used the fill-in-the-blank questions in an actual class and evaluated by students
- Based on the evaluations, we improved the fill-in-the-blanks questions.

Future Work

- Complete pgtracer for Java program and conduct experiments using the problems we have created.
- By using the collected data, we will be able to extract the tendency of students' answers and the tendency of questions that are prone to mistakes using LA technology
- we believe that the obtained knowledge will contribute to Java programming education.