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

The Barcelona Conference on Education 2023 Official Conference Proceedings

#### Abstract

The purpose of this research is to support the creation of programming test questions. Source code, class diagrams, specification tables and execution results are often used in programming test questions. In this case, these are related. Therefore, when these are created, the contents must match. However, when correcting a part, it is easy to make mistakes such as forgetting to correct. Therefore, we try to solve it by describing the information of the test questions in the answer source code. This paragraph describes programming test question creation support tools. This research targets Java language programs. First decide on the subject of the test question. Next, create the source code for the answer and write the test questions and specifications in Javadoc. Finally, run the tools. The tools automatically create class diagrams, specification tables, and execution results from the source code for the answer. In addition, these contents are combined into one and output as a PDF. The software used in this tool is Java, JavaParser, and LaTeX. The LaTeX macros "listings" and "pgfumlcd" were also used. JavaParser analyzes the source code. The analysis result is converted to a class diagram in pgf-umlcd format. Javadoc method comments are converted to specification tables. Javadoc class comments are converted to LaTeX-style test question text. The developed tools made it possible to output programming test questions in PDF format. The test questions consist only of the source code with Javadoc.

Keywords: Programming Education, Source Code, Programming Practice Support System

# iafor

The International Academic Forum www.iafor.org

#### Introduction

In recent years, the demand for programming education has been increasing. At primary educational institutions, education is provided to develop "programming thinking skills." In middle and higher education institutions, education on problem solving through programming is provided. From 2022, programming education will be compulsory in high schools. In universities, programming education has begun as part of general education.

In information science departments, there are many subjects for programming education. Information science students aim to become programmers or software engineers. Programming experience and programming skills are also required. Programming courses often include not only lectures but also practical training. In programming practice, it is effective for the instructor to provide appropriate guidance depending on the learner's level of proficiency and the progress of the task. However, when there are many learners, desk-based instruction alone is insufficient. Therefore, various programming practice support systems have been proposed (Azuma, H., et al., 2020; Zaffalon, F., et al., 2022). Also, regarding the test questions used in the practice, it is considered effective to tailor the questions to the learners. Systems based on item response theory have been proposed for some time. Such a system requires test questions of various difficulty levels.

In this research, we support the creation of test questions in a programming exercise support system. The target test questions are descriptive.

When creating programming test questions, source code, class diagrams, specification tables, and execution results are often used. In this case these are related. Therefore, when creating these, it is necessary to match the contents. However, when making partial corrections, it is easy to make mistakes such as forgetting to make corrections. Therefore, we will try to solve this problem by writing the test question information in the answer source code.

This paper describes a method for supporting test question creation and its results.

## **Overview of Programming Practice Support System**

The programming practice support system is shown in Figure 1. This system consists of creating test questions, testing, evaluation, feedback, creating practice questions, and practice. The test confirms the learner's level of proficiency. In evaluation, learners' answers are scored. The evaluation results are feedback to the learner. When creating practice assignments, explanations and assignments are created according to the learning level of the learner. Learners solve practice tasks and submit them for evaluation. By repeating these exercises, learners' understanding of programming will be improved.

Previously, the authors attempted to automate grading and feedback to support learners' practice (Takano, T., et al., 2023).

This paper describes the creation of test questions.



Figure 1: Overview of programming practice support system.

# **Creating Programming Test Questions**

The target programming test question is to write a program from a specification. In the proposed test question creation method, test question information is written in the source code. The reason is to centrally manage test questions and source code. Then, it generates a class diagram, specification table, and execution results from the source code.

Previously, when including class diagrams in test questions, the source code and class diagrams were created separately. Therefore, when modifying a test question, it was necessary to modify both the source code and the class diagram.

The proposed method maintains the consistency of the source code, class diagram, specification table, and execution results. The text of the test question is written in a comment in the source code. At this time, attributes are added to the comment by using "tags." This attribute makes it possible to specify test question headings, class diagrams, etc.

The advantage of the proposed method is that test questions can be created using only text. Also, since information about test questions is written in comments, the source code can be compiled and executed.

## How to Use the Tools

This section describes how to use the tools. First, decide on the subject of the test question. Next, create the source code for the answer and write the test questions and specifications in comments. An example of the source code is shown in Figure 2. Finally, run the tool.

The tool automatically creates class diagrams, specification tables, and execution results from the source code. Then, the contents are combined into one and a PDF file is output. Figure 3 shows an example of the generated PDF.

□ TX91+178ECE2023Resister(StewbertyMangove - sakas 241/2949 ついため 勝ちの 余物の 余気の つして おつの かつびかめ へんつい	
19月・日間とります。 おもや金金谷 (山田県村田)	B17500803_148021 43
StranburgManjana Strawberryana	· · · · · · · · · · · · · · · · · · ·
1 // OOAJOO Taro Dendai	
2 /**~	
3 * StrawberryMain⊖	
4 * ~	
5 * @question Strawberry (Fi	e name to submit: Strawberry.java)-
6 * This question is a progr	m that handles "strawberry". 🗠
7 * Create the program using	steps (1) to (3)
8 * -	
9 * @img s001. jpg-	
10 * ↔	later source and from the class discours "Chamberry" of
11 * Usubquestion Greate a ski	recon source code from the class diagram Strawberry
12 * -	ev1/1
14 * align center	ryr-
15 * class Strawberry-	
17 * @subquestion Create the	peration confirmation program "StrawberryMain java" 😔
18 * 4	porderion contributeron program certanoori judin. jara .
19 * @source StrawberryMain. i	va
20 * -	
21 * @execution StrawberryMain	
22 * -	
23 * @subquestion Implement "	trawberry java" to meet the API specification "Strawberry".4
24 * -	
25 * @api Strawberry←	
27 * @clearpage	
20 * Mouthor kohomo	
31 */~	
32 public class StrawberryMain	4.
33 /**-	
34 ° ∗ main –	
35 * ↔	
36 <sup>*</sup> ∉param args←	
37 */↩	
38 public static void main	String[] args) [~
39 Strawberry strawber	y1 = new Strawberry("Amaou", 50); e
40 Strawberry strawber	yZ = new Strawberry("lochiotome", 40); e
41	have a set News () is
42 String name1 = stra	berry: getName(),
AA int weight1 = straw	erryl getWeight ():-
45 ^ int weight? = straw	erry2 getWeight():
46 System out println(	ame1 + "" + weight1):
47 System out printing	ame2 + " " + weight2); e
48 }	and a Block
49 1	
(EOF)	
4	



Figure 2: An example of the source code for the answer.



Figure 3: An example of the generated PDF.

## **Implementation of Tools**

This section describes the implementation. The language of the programming test questions is Java. As an existing technology, we use the idea of Javadoc. The software used is Java, JavaParser, and LaTeX. Also, use the LaTeX macros "listings" and "pgf-umlcd."

• Java

Java (version 8) is used for tool development. Basic file operations, reading and writing files are done using standard libraries. Other software is called from processes.

• Javadoc

Javadoc is a documentation system. Author adds comments to Java source code according to Javadoc rules. Javadoc generates HTML-format API documentation from Java source code. The tools use extended Javadoc tags.

• JavaParser

JavaParser is a library that creates abstract syntax trees from Java source code. Application software uses JavaParser to parse Java source code and process syntax elements.

• LaTeX

LaTeX is a document processing system that is an extension of the typesetting system TeX. LaTeX creates reports, books, etc. from text written in markup languages. It is possible to import figures, tables, etc. using macro. The tool is used to output test questions to PDF. The LaTeX macro "listings" is used to display source code. Additionally, "pgf-umlcd" is used to create class diagrams.

## **Test Question Generation Details**

Details of data conversion by the tools are described below. An example of data conversion is shown in Figure 4.



Figure 4: An example of data conversion by the tools.

<pre>photo Bail photo Ph</pre>	□ デスクトップ¥BCE2023#tex/ksample#StrawberryClass.tex - takura 2.4.1.2849	- u x
<pre>3 2</pre>	ファイル() 編集() 実後() 後年() フール() 野芝() クインドウ() ヘルブ()	nuk-n(tark) (tark) ne:5
<pre>Numericans</pre>	19 11 11 11 11 11 11 11 11 11 11 11 11 1	
<pre>1 ¥ifdefined¥StrawberryClassWidth}~ 2 ¥else~ 4 ¥fi~ 5 ¥settowidth[¥StrawberryClassWidth] {¥tt +Strawberry(name:String, weight:int) }~ 4 ¥didolength[¥StrawberryClassWidth] {¥tt +Strawberry(name:String, weight:int) }~ 7 ¥ifdefined¥StrawberryClass[3][text width=¥StrawberryClassWidth][%~ 9 ¥newcommand[¥StrawberryClass][3][text width=¥StrawberryClassWidth][%~ 10 ¥begin[class][#1][Strawberry][#2][#3]~ 11 ¥attribute[-name:String]~ 12 ¥attribute[-name:String]~ 13 ¥operation[+Strawberry(name:String, weight:int)]~ 14 ¥operation[+Strawberry(name:String]~ 15 ¥operation[+strawberry(name:String]~ 16 ¥ond[class]~ 17 }~ 18 ¥fi~ 19 ¥ifdefined¥StrawberryClassWidthName%~ 20 ¥else~ 21 Ynewlength[¥StrawberryClassWidthName]~ 22 ¥fi~ 23 ¥settowidth[¥StrawberryClassWidthName] {¥tt Strawberry]~ 24 ¥ifdefined¥StrawberryClassWidthName] {¥tt StrawberryClassWidthName] {%~ 25 ¥else~ 26 ¥newcommand[¥StrawberryClassName][3][text width=¥StrawberryClassWidthName] {%~ 27 ¥begin[class] {#1][Strawberry] [#2] {#3]~ 28 ¥end[class] {#1][Strawberry] [#2] {#3]~ 29 }~ 20 ¥else~ 20 Ydef¥StrawberryAssociationString#1#2[~ 21 YnedficeTionalAssociation[Strawberry] [] {[String]~ 22 ¥ifi~ 23 ¥end[class] {#1][Strawberry] [#2] {#3]~ 24 ¥idef¥StrawberryDashedLine#1#2{~ 25 ¥else~ 26 ¥ifi~ 27 ¥begin[class] {#1][Strawberry] [#2] [#3]~ 28 ¥end[class] {#1][Strawberry] [#2] [#3]~ 29 }~ 20 ¥fi~ 20 ¥fi~ 20 ¥fi~ 20 ¥fi~ 21 ¥ifi~ 22 ¥ifi~ 23 ¥end[class] {#1][Strawberry] [#2] [#3]~ 24 ¥ifi~ 25 ¥else~ 25 ¥else~ 26 ¥ifi~ 27 ¥ifi~ 27 ¥ifi~ 28 ¥else~ 29 }~ 29 }~ 20 }~ 20 ¥ifi~ 20 ¥ifi~</pre>	StravberryClass.tex	•1 • • • • • • • • • • • • • • • • • • •
<pre>2 Yelse~ 3 Ynewlength[¥StrawberryClassWidth] ~ 4 Yfi~ 5 Ysettowidth[¥StrawberryClassWidth] [Ytt +Strawberry(name:String, weight:int) ] ~ 6 Yaddtolength[¥StrawberryClassWidth] [+3mm] ~ 7 Yifdefined¥StrawberryClass[3][text width=¥StrawberryClassWidth] [%~ 9 Ynewcommand[¥Strawberry[#2] [#3] ~ 1 Yattribute[-name:String] ~ 2 Yattribute[-name:String] ~ 2 Yattribute[-weight:int] ~ 3 Yoperation[+getMeme():String] ~ 4 Yoperation[+getMeme():String] ~ 5 Yoperation[+getWeight():int] ~ 18 Yfi~ 19 Yifdefined¥StrawberryClassWidthName%~ 20 Yelse~ 21 Ynewlength[¥StrawberryClassWidthName] ~ 23 Ysettowidth[¥StrawberryClassWidthName] {Ytt Strawberry} ~ 24 Yifdefined¥StrawberryClassNidthName] {Ytt Strawberry} ~ 25 Yelse~ 6 Ynewcommand[¥StrawberryClassName][3][text width=¥StrawberryClassWidthName] [%~ 7 Ybegin[class] [#1] [Strawberry] [#2] [#3] ~ 8 Yend[class] ~ 30 Yfi~ 30 Yfi~ 31 Ydef¥StrawberryAssociationString#1#2[~ 4 YunidirectionalAssociation[Strawberry] [] [String] ~ 32 YattawberryDashedLine#1#2[~ 4 Ydef¥StrawberryDashedLine#1#2[~ 4 Ydef¥StrawberryDashedLine#1#2[~ 4</pre>	1 ¥ifdefined¥StrawberryClassWidth%~	
<pre>3 Ynewlength [¥StrawberryClassWidth] e 4 ¥fie 5 Ysettowidth [¥StrawberryClassWidth] [¥tt +Strawberry (name:String, weight:int) ] e 6 ¥addtolength [¥StrawberryClassWidth] [+3mm] e 7 ¥ifdefined¥StrawberryClass%= 9 Ynewcommand [¥StrawberryClass] [3] [text width=¥StrawberryClassWidth] [%e 10 ¥begin [class] [#1] [Strawberry] [#2] [#3] e 11 ¥attribute [-mame:String] e 12 ¥attribute [-mame:String] e 13 ¥operation [+Strawberry (name:String, weight:int)] e 14 ¥operation [+getName() :String] e 15 ¥operation [+getName() :String] e 16 ¥end[class] e 17 ] e 18 ¥fie 19 ¥ifdefined¥StrawberryClassWidthName] e 21 ¥newlength [¥StrawberryClassWidthName] e 22 ¥fie 23 Ysettowidth [¥StrawberryClassWidthName] {¥tt Strawberry] e 24 ¥ifdefined¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%e 24 ¥ifdefined¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%e 25 ¥else: 26 ¥newcommand [¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%e 26 ¥newcommand [¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%e 27 ¥begin [class] e 28 ¥end[class] e 29 } e 20 ¥fie 24 ¥ifie 24 ¥ifie 34 ¥def¥StrawberryAssociationString#1#2[e 35 ¥def¥StrawberryDashedLine#1#2[e 36 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1); e 36 ¥def¥StrawberryDashedLine#1#2[e 37 ¥def¥StrawberryDashedLine#1#2[e 38 ¥def¥StrawberryDashedLine#1#2[e 39 ¥def¥StrawberryDashedLine#1#2[e 39 ¥def¥StrawberryDashedLine#1#2[e 30 ¥def¥StrawberryDashedLine#1#2[e 31 ¥def¥StrawberryDashedLine#1#2[e 32 ¥def¥StrawberryDashedLine#1#2[e 34 ¥def¥StrawberryDashedLine#1#2[e 34 ¥def¥StrawberryDashedLine#1#2[e 34 ¥def¥StrawberryDashedLine#1#2[e] \$ 34 ¥def¥StrawberryDashedLine#1#2[e 34 ¥def¥StrawberryDashedLine#1#2[e] \$ 35 ¥draw[umlcd style dashed line, -&gt;] (Strawberry] (#1); e 35 ¥def¥StrawberryDashedLine#1#2[e] \$ 36 ¥draw[umlcd style] \$ 37 ¥def¥StrawberryDashedLine#1#2[e] \$ 37 ¥def¥StrawberryDashedLine#1#2[e] \$ 38 ¥def¥StrawberryDashedLine#1#2[e] \$ 49 ¥def¥StrawberryDashe</pre>	2 ¥else	
<pre>4 ¥fi 5 ¥settowidth [¥StrawberryClassWidth] [¥tt +Strawberry(name:String, weight:int) ] ~ 6 ¥addtolength [¥StrawberryClassWidth] [+3mm] ~ 7 ¥ifdefined¥StrawberryClass] [3] [text width=¥StrawberryClassWidth] [%~ 9 ¥newcommand [¥StrawberryClass] [3] [text width=¥StrawberryClassWidth] [%~ 10 ¥begin [class] [#1] [Strawberry] [#2] [#3] ~ 11 ¥attribute[-name:String] ~ 12 ¥attribute[-name:String] ~ 13 ¥operation [+StrawberryClassString, weight:int)] ~ 14 ¥operation [+getName():String] ~ 15 ¥operation [+getWeight():int] ~ 16 ¥end[class] ~ 17 ] ~ 18 ¥fi~ 19 ¥ifdefined¥StrawberryClassWidthName%~ 20 ¥else~ 21 ¥newlength [¥StrawberryClassWidthName] ~ 22 ¥fi~ 23 ¥settowidth [¥StrawberryClassWidthName] {¥tt Strawberry] ~ 24 ¥ifdefined¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%~ 25 ¥else~ 26 ¥newcommand [¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%~ 27 ¥begin [class] [#1] [Strawberry] [#2] [#3] ~ 28 ¥end[class] ~ 29 } ~ 29 } [~ 30 ¥fi~ 31 ¥def¥StrawberryAssociationString#1#2[~ 32 ¥undirectionalAssociation[Strawberry] [] [] [String] ~ 33 } ~ 44 ¥def¥StrawberryDashedLine#1#2[~ 44 ¥def¥StrawberryDashedLine#1#2[~ 45 ¥define] [%~ 45 ¥define] [%~ 45 ¥define] [%~ 45 ¥define] [%~ 46 ¥fine] [%~ 46 ¥fine] [%~ 47 ¥undirectionalAssociation[Strawberry] [] [] [String] ~ 47 ¥def¥StrawberryDashedLine#1#2[~ 48 ¥def¥StrawberryDashedLine#1#2[~ 49 ¥define] [%~ 40 ¥define] [%~ 40 ¥define] [%~ 41 ¥def¥StrawberryDashedLine#1#2[~ 42 ¥define] [%~ 43 ¥def¥StrawberryDashedLine#1#2[~ 44 ¥def¥StrawberryDashedLine#1#2[~ 45 ¥define] [%~ 45 ¥define] [%~ 46 ¥define] [%~ 46 ¥define] [%~ 47 ¥define] [%~ 48 ¥define] [%~ 48 ¥define] [%~ 49 ¥define] [%~ 40 ¥define] [%~ 40</pre>	3 ¥newlength {¥StrawberrvClassWidth}	
<pre>5 Ysettowidth{YStrawberryClassWidth}{Ytt +Strawberry(name:String, weight:int)}~ 6 Yaddtolength{YStrawberryClassWidth}{+3mm}~ 7 YifdefinedYStrawberryClassWidth}{+3mm}~ 9 Ynewcommand{YStrawberryClass][3][text width=YStrawberryClassWidth][%~ 9 Ynewcommand[YStrawberryClass][3][text width=YStrawberryClassWidth][%~ 10 Ybegin[class][#1][Strawberry[#2][#3]~ 11 Yattribute[-mame:String]~ 12 Yattribute[-mame:String]~ 13 Yoperation[+Strawberry(name:String, weight:int)]~ 14 Yoperation[+getName():String]~ 15 Yoperation[+getWeight():int]~ 16 Yend[class]~ 17 }~ 19 ¥ifdefinedYStrawberryClassWidthName}~ 20 Yelse~ 21 Ynewlength{YStrawberryClassWidthName}~ 23 Ysettowidth{YStrawberryClassWidthName}~ 24 ¥ifdefinedYStrawberryClassWidthName}{~ 25 Yelse~ 26 Ynewcommand[YStrawberryClassNameM~ 27 Ybegin[class][#1][Strawberry][#2][#3]~ 28 Yend[class]~ 29 }~ 29 }~ 30 ¥fi~ 31 YdefYStrawberryDashedLine#1#2[~ 31 YdefYStrawberryDashedLine#1#2[~ 32 YetwerryDashedLine#1#2[~ 33 YdefYStrawberryDashedLine#1#2[~ 34 YdefYStrawberryDashedLine#1#2[~ 35 Ydiaw[umlcd style dashed line,-&gt;] (Strawberry) (#1);~ 36  - 37 Yopen</pre>	4 ¥fi⇔	
<pre>6 ¥addtolength [¥StrawberryClassWidth] [+3mm] ↔ 7 ¥ifdefined¥StrawberryClass [3] [text width=¥StrawberryClassWidth] [%↔ 8 ¥else↔ 9 ¥newcommand [¥StrawberryClass] [3] [text width=¥StrawberryClassWidth] [%↔ 10 ¥begin[class] [#1] [Strawberry[#2] [#3] ↔ 11 ¥attribute [-name:String] ↔ 12 ¥attribute [-meight:int] ↔ 13 ¥operation [+getWeight():int] ↔ 14 ¥operation [+getWeight():int] ↔ 15 ¥operation [+getWeight():int] ↔ 17 ] ↔ 18 ¥fi ↔ 19 ¥ifdefined¥StrawberryClassWidthName%↔ 19 ¥ifdefined¥StrawberryClassWidthName] ↔ 19 ¥ifdefined¥StrawberryClassWidthName] ↔ 19 ¥ifdefined¥StrawberryClassWidthName] ↔ 20 ¥else↔ 21 ¥newlength [¥StrawberryClassWidthName] {¥tt Strawberry] ↔ 23 ¥settowidth [¥StrawberryClassWidthName] {¥tt StrawberryClassWidthName] {¥text width=¥StrawberryClassWidthName] {% ↔ 24 ¥fic ↔ 25 ¥else↔ 25 ¥else↔ 26 ¥newcommand [¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [% ↔ 27 ¥begin[class] [#1] [Strawberry] [#2] [#3] ↔ 28 ¥end[class] {#1] [Strawberry] [#2] [#3] ↔ 29 } ↔ 20 ¥else↔ 20 ¥def¥StrawberryAssociationString#1#2[↔ 20 ¥unidirectionalAssociation[Strawberry] [] [String] ↔ 26 ¥def¥StrawberryDashedLine#1#2[↔ 27 ¥def¥StrawberryDashedLine#1#2[↔ 28 ¥ele↔ 29 } ↓ ↔ 20 ¥else↔ 29 } ↓ ↔ 20 ¥else↔ 20 ¥else↔ 20 ¥else↔ 20 ¥else↔ 21 ¥newlend(class] {#1] [Strawberry] [] [String] ↔ 22 ¥fi ↔ 23 ¥settowidth [¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [% ↔ 24 ¥ifdefined¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [% ↔ 27 ¥begin[class] {#1] [Strawberry] [] [] [String] ↔ 28 ¥end[class] {#1] [Strawberry] [] [] [String] ↔ 29 } ↓ ↔ 20 ¥else↔ 20 ¥else↔ 20 ¥else↔ 20 ¥ifi ↔ 20 ¥else↔ 20 ¥else↔ 20 ¥else↔ 20 ¥else↔ 20 ¥else↔ 21 ¥ifdefined¥strawberryClassName] {#2] [#3] ↔ 21 ¥ifdefined¥strawberryAssociationString#1#2[↔ 21 ¥unidirectionalAssociation[Strawberry] [] [] [String] ↔ 23 } ↔ 24 ¥else× 24 ¥unidirectionalAssociation[Strawberry] [] [] [String] ↔ 25 ¥else↔ 26 ¥else↔ 27 ¥unidirectionalAssociation[Strawberry] [] [] [String] ↔ 28 ¥else</pre>	5 ¥settowidth [¥StrawberrvClassWidth] [¥tt	+Strawberry (name:String, weight:int) }
<pre>7 ¥ifdefined¥StrawberryClass%~ 8 ¥else~ 9 Ynewcommand[¥StrawberryClass][3][text width=¥StrawberryClassWidth][%~ 10 ¥begin[class][#1][Strawberry][#2][#3]~ 11 ¥attribute[-name:String]~ 12 ¥attribute[-name:String]~ 13 ¥operation[+getName():String]~ 14 ¥operation[+getName():String]~ 15 ¥operation[+getWeight():int]~ 16 ¥end[class]~ 17 ]~ 18 ¥fi~ 19 ¥ifdefined¥StrawberryClassWidthName%~ 20 ¥else~ 21 ¥newlength[¥StrawberryClassWidthName]~ 22 ¥fi~ 23 ¥settowidth[¥StrawberryClassWidthName] {¥tt Strawberry}~ 24 ¥ifdefined¥StrawberryClassNidthName] {¥tt Strawberry}~ 25 ¥else~ 26 ¥newcommand[¥StrawberryClassName][3][text width=¥StrawberryClassWidthName][%~ 27 ¥begin[class][#1][Strawberry][#2][#3]~ 28 ¥end[class]~ 29 ]~ 29 ]~ 30 ¥fi~ 31 ¥def¥StrawberryAssociationString#1#2[~ 32 ¥unidirectionalAssociation[Strawberry] [] [][string]~ 33 ]~ 44 ¥def¥StrawberryDashedLine#1#2[~ 44 ¥def¥StrawberryDashedLine#1#2[~ 45 ¥def¥StrawberryDashedLine#1#2[~ 46 ¥def¥StrawberryDashedLine#1#2[~ 47 ¥def¥StrawberryDashedLine#1#2[~ 46 ¥def¥StrawberryDashedLine#1#2[~ 47 ¥def¥StrawberryDashedLine#1#2[~ 48 ¥def¥StrawberryDashedLine#1#2[~ 49 ¥def¥StrawberryDashedLine#1#2[~ 40 ¥def¥StrawberryDashedLine#1#2[~ 41 ¥def¥StrawberryDashedLine#1#2[~ 42 ¥def¥StrawberryDashedLine#1#2[~ 43 ¥def¥StrawberryDashedLine#1#2[~ 44 ¥def¥StrawberryDashedLine#1#2[~ 45 ¥def¥Strawbe</pre>	6 ¥addtolength{¥StrawberrvClassWidth} {+3	mm) 🕂
<pre>8 ¥elsed 9 Ynewcommand[¥StrawberryClass][3][text width=¥StrawberryClassWidth][%d 10 Ybegin[class][#1][Strawberry][#2][#3]d 11 Yattribute[-name:String]d 12 Yattribute[-name:String]d 13 Yoperation[+gtName]():string]d 14 Yoperation[+getWeight():int]d 15 Yoperation[+getWeight():int]d 16 Yend[class]d 17 ]d 18 Yfid 19 Yifdefined¥StrawberryClassWidthName%d 20 Yelsed 11 Ynewlength[¥StrawberryClassWidthName]d 21 Ynewlength[¥StrawberryClassWidthName]d 23 Ysettowidth[¥StrawberryClassWidthName]d 24 Yifdefined¥StrawberryClassWidthName]{Ytt Strawberry]d 25 Yelsed 26 Ynewcommand[¥StrawberryClassName][3][text width=¥StrawberryClassWidthName][%d 27 Ybegin[class]d]]Strawberry][#2][#3]d 28 Yend[class]d 29 }d 29 }d 29 }d 20 Yfid 31 Ydef¥StrawberryDashedLine#1#2[d 44 Vidf¥StrawberryDashedLine#1#2[d 44 Vidf¥StrawberryDashedLine#1#2[d 44 Vidf¥StrawberryDashedLine#1#2[d 44 Vidf¥StrawberryDashedLine#1#2[d 44 Vidf¥StrawberryDashedLine#1#2[d 45 Yifw]umidid style dashed line,-&gt;] (Strawberry) (#1);d 20 Id 20 Id 20 Id 20 Yend[dlass]d 20 Yend[dlashed line,-&gt;] (Strawberry) (#1);d 20 Id 20 Yend[dlashed line,-&gt;] (Strawberry) (#1);d 20 Yend[dlashed line,-&gt;] (Strawberry) (#1);d 20 Yend[dlashed line]d 20 Yend[dlashed line,-&gt;] (Strawberry) (#1);d 20 Yend[dlashed line]d 20 Yend[dla</pre>	7 ¥ifdefined¥StrawberrvClass%~	
<pre>9 Ynewcommand[YStrawberryClass][3][text width=YStrawberryClassWidth][%~ Yoegin[class][#1][Strawberry][#2][#3]~ Yattribute[-name:String]~ Yattribute[-weight:int]~ Yoperation[+Strawberry(name:String,weight:int)]~ Yoperation[+getWeight():int]~ Yoperation[+getWeight():int]~ Yet Yoperation[+getWeight():int]~ Yet Yet Yet Yet Yet Yet Yet Yet</pre>	8 ¥else	
<pre>10 Ybegin[class] [#1] [Strawberry] [#2] [#3] ~ 11 Yattribute[-name:String] ~ 12 Yattribute[-weight:int] ~ 13 Yoperation[+Strawberry(name:String, weight:int)] ~ 14 Yoperation[+strawberry(lame:String] ~ 15 Yoperation[+getWeight():int] ~ 16 Yend[class] ~ 17 ]~ 18 Yfi~ 19 YifdefinedYStrawberryClassWidthName%~ 20 Yelse~ 21 Ynewlength[¥StrawberryClassWidthName] ~ 22 Yfi~ 23 Ysettowidth[¥StrawberryClassWidthName] {Ytt Strawberry] ~ 24 YifdefinedYStrawberryClassWidthName] {Ytt Strawberry] ~ 25 Yelse~ 26 Ynewcommand[¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] {%~ 27 Ybegin[class] [#1] [Strawberry] [#2] [#3] ~ 28 Yend[class] [#1] [Strawberry] [#2] [#3] ~ 29 ]~ 20 Yfi~ 20 Yfi~ 21 Ydef¥StrawberryDashedLine#1#2[~ 22 Ydraw[umld style dashed line, -&gt;] (Strawberry) ~ (#1);~ 23 [~ 24 Yidaw[umld style dashed line, -&gt;] (Strawberry) ~ (#1);~ 24 Yidaw[umld style dashed line, -&gt;] (Strawberry) ~ (#1);~ 29 ]~ 20 Ydraw[umld style dashed line, -&gt;] (Strawberry) ~ (#1);~ 20 Ydraw[umld style dashed line, -&gt;] (Strawberry) ~ (#1);~ 20 Ydraw[umld style dashed line, -&gt;] (Strawberry) ~ 27 Ybegin[class] ~ 28 Ydraw[umld style dashed line, -&gt;] (Strawberry) ~ 29 ]~ 20 Ydraw[umld style dashed line, -&gt;] (Strawberry) ~ 20 Ydraw[umld style] ~ 20 Ydraw[umld style] ~ 20 Ydraw[umld] ~ 20 Ydr</pre>	9 ¥newcommand {¥StrawberrvClass} [3] [text	width=¥StrawberrvClassWidth][%
<pre>Yattribute[-name:String] { Yattribute[-name:String] { Yattribute[-weight:int] { Yoperation {+Strawberry(name:String, weight:int)] { Yoperation {+getName() {String] { Yoperation {+getNeight() : int] { YetNeight() : int] { Yoperation {+getNeight() : int] { Yoperation {+getNe</pre>	10 ¥begin[class] [#1] [Strawberry] [#2] [#3	]←
<pre>Vaturibute[-weight:int]+ Yaturibute[-weight:int]+ Yoperation[+getName():String]+ Yoperation[+getWeight():int]+ Yoperation[+getWeight():int]+ Yet Yet Yet Yet Yifdefined¥StrawberryClassWidthName}+ Yet Yet Yet Yet Yet Yet Yet Yet</pre>	11 ¥attribute (-name: String)	
<pre> 4 ¥operation {+Strawberry (name: String, weight: int) } ← 4 ¥operation {+getName(): String } ← 4 ¥operation {+getWeight(): int } ← 5 ¥operation {+getWeight(): int } ← 18 ¥fi ← 19 ¥ifdefined¥StrawberryClassWidthName% ← 20 ¥else ← 21 ¥newlength {¥StrawberryClassWidthName} ← 22 ¥fi ← 23 Ysettowidth {¥StrawberryClassWidthName} {¥tt Strawberry} ← 24 ¥ifdefined¥StrawberryClassWidthName] {¥tt Strawberry} ← 25 ¥else ← 26 ¥newcommand {¥StrawberryClassName% ← 27 ¥begin {class} [#1] {Strawberry} {#2} {#3} ← 29 } ← 29 } ← 20 ¥fi ← 21 ¥newlength {×StrawberryClassName} [3] [text width=¥StrawberryClassWidthName] {% ← 25 ¥else ← 26 ¥newcommand {¥StrawberryClassName} [3] [text width=¥StrawberryClassWidthName] {% ← 27 ¥begin {class} [#1] {Strawberry} {#2} {#3} ← 29 } ← 29 } ← 20 ¥fi ← 21 ¥def¥StrawberryAssociationString#1#2[← 22 ¥unidirectionalAssociation[Strawberry] [] {String} ← 23 } ← 24 ¥idef¥StrawberryDashedLine#1#2[← 24 ¥unidirectionalAssociation[Strawberry] [] {String} ← 25 ¥def¥StrawberryDashedLine#1#2[← 26 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1); ← 29 } ← 20 ¥fi ← 20 ¥fi ← 21 ¥def¥StrawberryDashedLine#1#2[← 22 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1); ← 23 } ← 24 ¥def¥StrawberryDashedLine#1#2[← 24 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1); ← 24 ¥def¥StrawberryDashedLine#1#2[ ← 24 ¥def¥StrawberryDashedLine#1#2[ ← 24 ¥def¥StrawberryDashedLine#1#2[ ← 24 ¥def¥StrawberryDashedLine#1#2[ ← 25 ¥def¥StrawberryDashedLine#1#2[ ← 26 ¥draw[umlcd style dashed line, -&gt;] (Strawberry] (#1); ← 26 } ↓ ← 27 ↓ ← 2</pre>	12 ¥attribute (-weight; int) +	
<pre>Voperation {+getName():String} ~ Voperation {+getName():String} ~ Voperation {+getWeight():int} ~ Voperation {+getWeight():int} ~ Voperation {+getWeight():int} ~ Voperation {+getWeight():int} ~ Voperation {+getName():String} ~ Voperation {+getName():int} ~ Voperation {+g</pre>	13 ¥operation + Strawberry (name: String	weight: int)}
<pre>Voperation {*getWeight():int} ~ Yoperation {*getWeight():</pre>	14 ¥operation (+getName(); String) e	no (Bite, Inc))
<pre>1</pre>	15 Yoperation (+getWeight(); int) 4	
<pre>10 Fore fords of the ford of the ford</pre>	16 ¥end (class) e	
<pre>18 ¥fie 19 ¥ifdefined¥StrawberryClassWidthName%e 20 ¥elsee 22 ¥fie 23 ¥settowidth{¥StrawberryClassWidthName} { 24 ¥ifdefined¥StrawberryClassName} { 25 ¥elsee 26 ¥newcommand{¥StrawberryClassName} [3][text width=¥StrawberryClassWidthName] {%e 27 ¥begin[class][#1][Strawberry] [#2] [#3] e 28 ¥end{class}] e 29 }e 29 }e 29 }e 20 ¥fie 31 ¥def¥StrawberryAssociationString#1#2[e 24 ¥unidirectionalAssociation[Strawberry] {] [] [String] e 33 }e 34 ¥def¥StrawberryDashedLine#1#2[e 44 ¥def¥StrawberryDashedLine#1#2[e 45 ¥def¥StrawberryDashedLine#1#2[e 46  ]e 46  ]e 47 ¥def¥StrawberryDashedLine#1#2[e 48 ¥enf¥StrawberryDashedLine#1#2[e] 49 }e 40 ¥def¥StrawberryDashedLine#1#2[e] 40 ¥def¥StrawberryDashedLine#1#2[e] 41 ¥def¥StrawberryDashedLine#1#2[e] 42 ¥def¥StrawberryDashedLine#1#2[e] 43 }e 44 ¥def¥StrawberryDashedLine#1#2[e] 45 ¥def¥StrawberryDashedLine#1#2[e] 46  ]e 46  ]e 47 ¥def¥StrawberryDashedLine#1#2[e] 48 ¥def¥StrawberryDashedLine#1#2[e] 49 }e 40 }e</pre>	17 10	
<pre>19 ¥ifdefined¥StrawberryClassWidthName%~ 20 ¥else~ 21 ¥newlength{¥StrawberryClassWidthName}~ 23 ¥settowidth{¥StrawberryClassWidthName}{ 24 ¥ifdefined4StrawberryClassName%~ 25 ¥else~ 26 ¥newcommand{¥StrawberryClassName}[3][text width=¥StrawberryClassWidthName]{%~ 27 ¥begin[class][#1][Strawberry][#2][#3]~ 28 ¥end[class]~ 29 }~ 30 ¥fi~ 31 ¥def¥StrawberryAssociationString#1#2[~ 24 ¥unidirectionalAssociation[Strawberry][][][String]~ 33 }~ 34 ¥def¥StrawberryDashedLine#1#2[~ 35 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1);~ 460]</pre>	18 ¥fi⇔	
<pre>20 Yelse 21 Ynewlength {¥StrawberryClassWidthName} ↔ 22 Yfi 23 Ysettowidth {¥StrawberryClassWidthName} {¥tt Strawberry} ↔ 24 Yifdefined¥StrawberryClassName%↔ 25 Yelse↔ 26 Ynewcommand {¥StrawberryClassName} [3] [text width=¥StrawberryClassWidthName] {%↔ 27 Ybegin [class] [#1] [Strawberry] [#2] {#3} ↔ 28 Yend [class] ↔ 29 ] ↓ 20 ¥fi↔ 20 ¥fi↔ 21 Ydef¥StrawberryAssociationString#1#2[↔ 21 Ydef¥StrawberryDashedLine#1#2[↔ 32 Ydraw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 31 [€] 32 Yend 33 ] ↔ 34 Ydef¥StrawberryDashedLine#1#2[↔ 34 Ydef¥StrawberryDashedLine#1#2[↔ 35 Ydraw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 35 Ydraw[umlcd style dashed line, -&gt;] (Strawberry]</pre>	19 ¥ifdefined¥StrawberrvClassWidthName%	
<pre>21 Ynewlength {¥StrawberryClassWidthName} ↔ 22 ¥fi↔ 23 Ysettowidth {¥StrawberryClassWidthName} {¥tt Strawberry} ↔ 24 Yifdef ined¥StrawberryClassName%↔ 25 Yelse↔ 26 Ynewcommand {¥StrawberryClassName} [3] [text width=¥StrawberryClassWidthName] [%↔ 27 Ybegin {class} [#1] {Strawberry] #2] [#3] ↔ 28 Yend {class} [#1] {Strawberry] #2] [#3] ↔ 29 } ↔ 29 } ↔ 29 } ↔ 20 ¥fi↔ 21 Ydef¥StrawberryAssociationString#1#2[↔ 22 YunidirectionalAssociation[Strawberry] {] {] {String} ↔ 23 } ↔ 24 ¥idef¥StrawberryDashedLine#1#2[↔ 24 ¥idef¥StrawberryDashedLine#1#2[↔ 25 ¥def¥StrawberryDashedLine#1#2[↔ 26 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1); ↔ 26 } ↓ 27 ↓ 28 ↓ 29 } ↔ 29 } ↔ 29 } ↔ 29 } ↔ 29 } ↔ 29 } ↔ 29 } ↔ 29 } ↔ 29 } ↔ 20 } ↔</pre>	20 Yelsed	
<pre>22 ¥fi 23 ¥settowidth {¥StrawberryClassWidthName} {¥tt Strawberry} 24 ¥ifdefined¥StrawberryClassName%↔ 25 ¥else↔ 26 ¥newcommand {¥StrawberryClassName} [3] [text width=¥StrawberryClassWidthName] {%↔ 27 ¥begin[class] [#1] {Strawberry] [#2] {#3} ↔ 28 ¥end[class] ↔ 29 }↔ 29 }↔ 20 ¥fi↔ 21 ¥def¥StrawberryAssociationString#1#2[↔ 22 ¥unidirectionalAssociation[Strawberry] [] [] {String} ↔ 23 ] ↔ 24 ¥def¥StrawberryDashedLine#1#2[↔ 25 ¥dfaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 26 [] ↔ 27 ¥begin[class] ↔ 29 ] ↔ 29 ] ↔ 29 ] ↔ 20 ¥fi↔ 20 ¥fi↔ 20 ¥fi↔ 21 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 21 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 25 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 27 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 28 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 29 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 20 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 27 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 28 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 29 {Varaw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔</pre>	21 ¥newlength [¥StrawberryClassWidthName]	4
<pre>23 ¥settowidth{¥StrawberryClassWidthName}{¥tt Strawberry}↔ 24 ¥ifdefined¥StrawberryClassName%↔ 25 ¥else↔ 26 ¥newcommand{¥StrawberryClassName}[3][text width=¥StrawberryClassWidthName]{%↔ 27 ¥begin{class}[#1]{Strawberry][#2]{#3}↔ 28 ¥end{class}↔ 29 }↔ 30 ¥fi↔ 31 ¥def¥StrawberryAssociationString#1#2[↔ 31 ¥def¥StrawberryDashedLine#1#2[↔ 33 }↔ 44 ¥def¥StrawberryDashedLine#1#2[↔ 35 ¥draw[umlcd style dashed line,→] (Strawberry) (#1);↔ 46 []↔ 45 []↔ 4</pre>	22 ¥fi⇔	
24       ¥ifdefined#StrawberryClassName%e         25       ¥elsee         26       ¥newcommand[¥StrawberryClassName][3][text width=¥StrawberryClassWidthName][%e         27       ¥begin[class][#1][Strawberry] [#2] [#3] e         28       ¥end{class}         29       }e         30       ¥fie         31       ¥def¥StrawberryAssociationString#1#2[e         32       ¥unidirectionalAssociation[Strawberry] [] [] [String] e         33       }e         34       ¥def¥StrawberryDashedLine#1#2[e         35       ¥draw[umlcd style dashed line, ->] (Strawberry) (#1); e         36        e	23 ¥settowidth [¥StrawberryClassWidthName]	{\tt Strawherry}
<pre>24 Filesed 25 Felsed 26 Ynewcommand [¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%@ 27 Yeegin [class] [#1] [Strawberry] [#2] [#3] @ 28 Yend [class] @ 29 ] @ 29 ] @ 30 ¥fi@ 31 Ydef¥StrawberryAssociationString#1#2[@ 24 YunidirectionalAssociation[Strawberry] [] [] [String] @ 33 ] @ 34 Ydef¥StrawberryDashedLine#1#2[@ 35 Ydraw[umlcd style dashed line, -&gt;] (Strawberry) (#1);@ 36 ] @ 37 [EOF] 38 [] [] [] [] [] [] [] [] [] [] [] [] []</pre>	24 ¥ifdefined¥StrawberrvClassName%	[rec del and rig]
<pre>25 Forewcommand [¥StrawberryClassName] [3] [text width=¥StrawberryClassWidthName] [%↔ 26 ¥newcommand [¥Strawberry] [#2] [#3] ↔ 28 ¥end [class] ↔ 29 } ↔ 30 ¥fi ↔ 31 ¥def¥StrawberryAssociationString#1#2 [↔ 32 ¥unidirectionalAssociation[Strawberry] [] [] [String] ↔ 33 } ↔ 34 ¥def¥StrawberryDashedLine#1#2 [↔ 35 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ 36 ] ↔ 37 [€06]</pre>	25 ¥elsed	
<pre>20 Therefund (for about you as draine) [0][text wheth-for a wheth you as what have [] [st 20 Yei] 20 Yei] 30 Yei] 31 Yei Yei Yassociation (Strawberry) [] [] [String] ↔ 33 Jei 44 Yei YStrawberryDashedLine#1#2[↔ 35 Yei raw[umlcd style dashed line, -&gt;] (Strawberry) (#1); ↔ 36 ]ei 40 Yei Yei Yei Yei Yei Yei Yei Yei Yei Yei</pre>	26 Ynewcommand [¥StrawherryClassName] [3]]	text width=¥StrawherryClassWidthName]
<pre>21 Hots file(lass) { # [[[[]]] [[]] [[]] [[]] [[]] [[]] [[</pre>	27 ¥begin (class) [#1] (Strawberry) (#2) [#3	
<pre>20 Fond [orads] 30 ¥fi ← 31 ¥def¥StrawberryAssociationString#1#2[↔ 32 ¥unidirectionalAssociation[Strawberry] [] [] [String] ↔ 33 }↔ 34 ¥def¥StrawberryDashedLine#1#2[↔ 35 ¥draw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ [EOF]</pre>	28 ¥end(class)	
25 ¥fi∉ 31 ¥def¥StrawberryAssociationString#1#2[↔ 32 ¥unidirectionalAssociation[Strawberry] [] [] [String]↔ 33 ]↔ 4 ¥def¥StrawberryDashedLine#1#2[↔ 35 ¥draw[umlcd style dashed line,->] (Strawberry) (#1);↔ 160F]	20 14	
<pre>St fifeFyStrawberryAssociationString#1#2[↔ St FyStrawberryDashedLine#1#2[↔ St FyStrawberryDashedLine#1#2[↔ St FyStraw[umlcd style dashed line, -&gt;] (Strawberry) (#1);↔ [EOF]</pre>	30 ¥fie	
2 ¥unidirectionalAssociation[Strammery] [] [] [String] ↔ 33 ] ↔ 34 ¥def¥StrawberryDashedLine#1#2[↔ 35 ¥draw[umlcd style dashed line, ->] (Strawberry) (#1);↔ 36 ] ↔	31 ¥def¥StrawberrvAssociationString#1#21	
34 ¥def¥StrawberryDashedLine#1#2{↔ 35 ¥draw[umlcd style dashed line,->] (Strawberry) (#1);↔ 36 ]→ [EOF]	32 Yunidirectional Association (Strawberry	1 [] [] [String]
4 ¥def¥StrawberryDashedLine#1#2[↔ 35 ¥draw[umlcd style dashed line,->] (Strawberry) (#1);↔ 36 ] EDE	33 ]-	100000
35 ¥draw[umlod style dashed line, ->] (Strawberry) (#1);↔ 36 ]}-	34 ¥def¥StrawherryDashed ine#1#2{~	
36  - [EOF]	35 Ydraw[umled style dashed line ->] (St	rawberry) (#1):4
	36 1	

Figure 5: An example of a TeX file.

# • Generate Class Diagram

The source code is parsed by JavaParser. After that, the instance variables and method information are extracted, and a TeX file is generated in the LaTeX macro "pgf-umlcd" format. An example of a TeX file is shown in Figure 5. TeX files are used to generate text for test questions.

• Generate API Specification Table

The source code is parsed by JavaParser. Then, the information in the Javadoc method comments is extracted and a TeX file is generated in LaTeX tabular format. TeX files are used to generate text for test questions.

• Generating Execution Result

The source code is compiled. If the class file contains a main method, it will be executed and the standard output will be output to the file. A TeX file is generated in the LaTeX "execution result" format. The "execution result" format is defined separately using a LaTeX macro. TeX files are used to generate text for test questions.

- Generating Source Code Diagrams The Javadoc part of the source code will be deleted. A TeX file in the LaTeX macro "listings" format is generated. TeX files are used to generate text for test questions.
- Generation Text for Test Questions The source code is parsed by JavaParser. Then, the information of the Javadoc class comments is extracted. The tag in the comment (keyword written with "@" at the beginning) is analyzed. Details of the tags are shown in Table 1. The text of the test question is constructed according to the tag information, and a TeX file is generated. The

TeX file is compiled with LaTeX. Class diagrams, specification tables, execution results, and source code are integrated. A PDF file of the test questions will be generated.

Tag	Detail
@question	Write the question heading and question text. The question number is automatically counted and auto-incremented.
@subquestion	Write the sub-question heading and question text. The sub-question number is automatically counted and auto-incremented.
@make.inputClass	Describe this when arranging a class diagram. The position and size of the figure can be adjusted using parameters. For example, "align center" causes centering. "scale 1.5" makes the figure 1.5 times larger. Also, multiple class diagrams can be placed.
@source	Write this when placing the source code. Specify the file name with the parameter. The view of the source code can be specified using the LaTeX macro "listings".
@execution	Describe this when placing the execution results. Specify the class name as a parameter. The view of the execution results can be specified using a LaTeX macro.
@api	Describe this when placing the API specification table. Specify the class name as a parameter. Table views can be specified using LaTeX macros.
@img	Describe this when placing an image. Specify the png or jpg image file name with the parameter.

Table 1: Details of the tage

## Experiment

It was actually used in the programming subjects shown below.

- Computer Programming III Assignment Exercises (2022/10/31)
- Computer Programming III Achievement Test (2022/12/22)
- Computer Programming III Supplementary Examination (2023/1/16)
- Computer Programming II Comprehensive Exercise 3 (2023/5/25)
- Computer Programming I Comprehensive Review 1 (2023/7/4)

Figure 6-8 shows an example of test questions actually used in computer programming (Japanese) (PDF).

Access modifiers (visibility notation) in class diagrams can be omitted. The class diagram in Figure 3 includes access modifiers. The class diagrams in Figures 6 and 7 omit the access modifiers in the class diagram.

In the PDF of the test questions, there were no mistakes between the source code, class diagram, specification table, and execution results. The results used in the exercise had a typo in the text of the test question, but there were no other flaws.

問題 1 クラス図からソースプログラムの機械的導出	問題 2 いちご (提出物 Strawberry.java)
提出物 StrawberryFrame.java) クラス団 StrawberryFrame からソースコードを機械的に専出しなさい.	この問題は、「いちご」を取り扱うプログラムです。(1)~〈3〉の手順にしたがっ プログラムを完成しなさい、
StrawberryFrame	
name:String	
StrawberryFrame(name:String,weight:int) getMame():String getMeight():int	
	<ol> <li>(1) クラス図 Strawberry からソースコートを機械的に導出しなさい。</li> <li>Strawberry</li> </ol>
	name:String
	weight:int
	Strawberry(name:String,weight:int) getName():String getWide():tot
	<pre>(2) 部行相当用 プログラム StrawberryMain_Java を作成しなさい. // OnLife StrawberryMain_Java を作成しなさい. // OnLife StrawberryMain public class StrawberryMain[] = new Strawberry(*あまおう*, 50); Strawberry strawberry2 = new Strawberry(*さまおう*, 50); Stramp mand = strawberry2.optName(); String mand = strawberry2.optName(); Int weight1 = strawberry2.optName(); Int weight1 = strawberry2.optName(); System.cot.printle(manel + * + weight1); System.cot.printle(mane2 + * * + weight2); ]</pre>

Figure 6: An example of test questions actually used (Japanese) (PDF) (page 1,2).

- 宋行イメージ			問題 3 プラスチックトレイ (提出物 PlasticTray.java)
>java Stra あまおう 50 とちおとめ 4	wberryMain D		この問題は、「プラスチックトレイ」を取り扱うプログラムです、「プラスキックトレイ には、複数の「いちご」が入ります。(1) ~ (3) の手順にしたがって、プログラムを5 成しなさい。
(3) API仕様 5	Strawberry を満足するように Strawberry.java を実装しなさい		1
API triffe Str. Strawberry	awberry コンストラクタです	-	
	引数 name は、「いちご」の名前です。 引数 weight は、重きです。 それぞれ、インスタンス変数 name, weight に代入します。		
getName	「いちご」の名前を返却します。		(1) クラス図 PlasticTray からソースコードを機械的に導出しなさい。
getWeight	「いちご」の重さを返却します。		PlacticTray
			<pre>riastictay() put(itankberry()troid getStawberry(index:int):Strakberry getSize():int getAverageWeight():int getAverageWeight():int</pre>
			<pre>#Jastactay() put(atawberry)Strawberry)ivoid getStrawberry(Indexint)iStrawberry getSirawberry(indexint)iStrawberry getSirawberg()int getAverageWeight()iInt getAverageWeight()iInt # ArrayListを利用するためには、以下の記述が必要です。 import java.util.ArrayList;</pre>
			Plasticitay() put (at axwherry) Strawberry) ivoid get.Strawberry (indexint) iStrawberry get.Strawberry (indexint) iStrawberry get.Strawbergeweight () int get.AverageWeight () int # ArrayList 参利用するためには、以下の記述が必要です. import java.util.ArrayList;
			<pre>#Jasicitay() put(atawberry)Strawberry):void getEtrawberry(indexint)Strawberry getEirawberry(indexint)Strawberry getEirawberry getEirawb</pre>
			Plasticitay() put (at axwberry) Strawberry) :void get Strawberry (indexint) : Strawberry get Strawberry (indexint) : Strawberry get Strawberght () : int # ArrayList を利用するためには、以下の記述が必要です. import java.util.ArrayList;
			Plasticitay() put (at axwberry) Strawberry) :void get Strawberry (indexint) : Strawberry get Site () : int get Total Weight () : int @ ArrayList を利用するためには、以下の記述が必要です. import java.util.ArrayList;

Figure 7: An example of test questions actually used (Japanese) (PDF) (page 3,4).



Figure 8: An example of test questions actually used (Japanese) (PDF) (page 5,6).

## Conclusion

The purpose of this research is to support the creation of programming test questions. In the proposed test question creation method, test question information is written in the answer source code. The test questions are only source code with Javadoc. Then, it generates class diagrams, schedules, and execution results from the source code.

The developed tools have made it possible to output programming test questions in PDF format. The created test questions were actually used in programming exercises. As a result, there were some typos, but there were no defects.

A future challenge is to create practice questions tailored to the learners for the programming practice support system.

#### Acknowledgements

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

## References

Azuma, H., Takenouchi, H., Takano, T., Miyakawa, O., & Kohama, T. (2020). Study on Computer-Adaptive Testing: Proposal of a Scaffolding Tool. The Asian Conference on Education 2020 Official Conference Proceedings, 289-298.

Javadoc. https://docs.oracle.com/javase/8/docs/technotes/guides/javadoc/index.html

JavaParser. https://JavaParser.org/

LaTeX. https://www.latex-project.org/

Listings. https://ctan.org/pkg/listings

Pgf-umlcd. https://ctan.org/pkg/pgf-umlcd

- Takano, T., Miyakawa, O., & Kohama, T. (2023). Development of a Tool to Analyze Source Code Submitted by Novice Programmers and Provide Learning Support Feedback With Comments. The Asian Conference on Education & International Development 2023 Official Conference Proceedings, 777-789.
- Zaffalon, F., Prisco, A., Souza, D. R., Teixeira, D., Paes, W., Evald, P., Tonin, N., Devincenzi, S., & Botelho, S. (2022). A Recommender System of Computer Programming Exercises based on Student's Multiple Abilities and Skills Model. 2022 IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/FIE56618.2022.9962646