Development of Tools to Support the Creation of Programming Test Questions

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 "pgf-umlcd" 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

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.

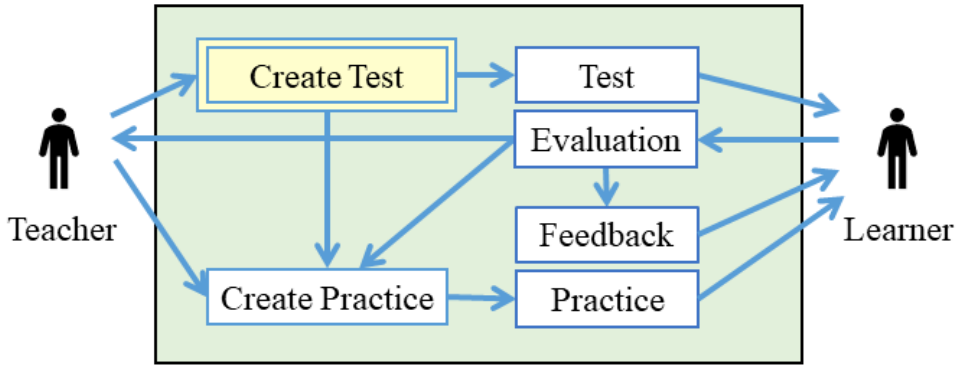

Figure 1: Overview of programming practice support system.

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.

```java
// OOAJOO Taro Dendai
/**
 * StrawberryMain
 *
 * @question Strawberry (File name to submit: Strawberry.java)
 * This question is a program that handles "strawberry".
 * Create the program using steps (1) to (3).
 *
 * @img s001.jpg
 *
 * @subquestion Create a skeleton source code from the class diagram "Strawberry".
 *
 * @make.inputClass Strawberry1
 * align center
 * class Strawberry
 *
 * @subquestion Create the operation confirmation program "StrawberryMain.java".
 *
 * @source StrawberryMain.java
 *
 * @execution StrawberryMain
 *
 * @subquestion Implement "Strawberry.java" to meet the API specification "Strawberry".
 *
 * @api Strawberry
 *
 * @clearpage
 *
 * @author kohama
 *
 */
public class StrawberryMain{
    /**
     * main
     *
     * @param args
     */
    public static void main(String[] args){
        Strawberry strawberry1 = new Strawberry("Amaou", 50);
        Strawberry strawberry2 = new Strawberry("Tochiotome", 40);

        String name1 = strawberry1.getName();
        String name2 = strawberry2.getName();
        int weight1 = strawberry1.getWeight();
        int weight2 = strawberry2.getWeight();
        System.out.println(name1 + " " + weight1);
        System.out.println(name2 + " " + weight2);
    }
}
[EOF]
```

```java
/**
 * Strawberry
 *
 * @author kohama
 *
 */
public class Strawberry {
    private String name;
    private int weight;
    /**
     * Constructs a new Strawberry object.
     * The argument "name" is the strawberry cultivar.
     * The argument "weight" is the strawberry weight.
     *
     * @param name
     * @param weight
     */
    public Strawberry(String name, int weight) {
        this.name = name;
        this.weight = weight;
    }
    /**
     * Returns the strawberry name.
     *
     * @return
     */
    public String getName() {
        return this.name;
    }
    /**
     * Returns the strawberry weight.
     *
     * @return
     */
    public int getWeight() {
        return this.weight;
    }
} [EOF]
```

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

Question 2: Strawberry (File name to submit: Strawberry.java)

This question is a program that handles "strawberry". Create the program using steps (1) to (3).

(1) Create a skeleton source code from the class diagram "Strawberry".

Strawberry
-name:String -weight:int
+Strawberry(name:String,weight:int) +getName():String +getWeight():int

(2) Create the operation confirmation program "StrawberryMain.java".

```java
// 00AJ00 Taro Dendai
public class StrawberryMain{
    public static void main(String[] args){
        Strawberry strawberry1 = new Strawberry("Amaou", 50);
        Strawberry strawberry2 = new Strawberry("Tochiotome", 40);

        String name1 = strawberry1.getName();
        String name2 = strawberry2.getName();
        int weight1 = strawberry1.getWeight();
        int weight2 = strawberry2.getWeight();
        System.out.println(name1 + " " + weight1);
        System.out.println(name2 + " " + weight2);
    }
}
```


Execution image

```
>java StrawberryMain
Amaou 50
Tochiotome 40
```

(3) Implement "Strawberry.java" to meet the API specification "Strawberry".

API Strawberry	
Strawberry	Constructs a new Strawberry object. The argument "name" is the strawberry cultivar. The argument "weight" is the strawberry weight.
getName	Returns the strawberry name.
getWeight	Returns the strawberry weight.

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.

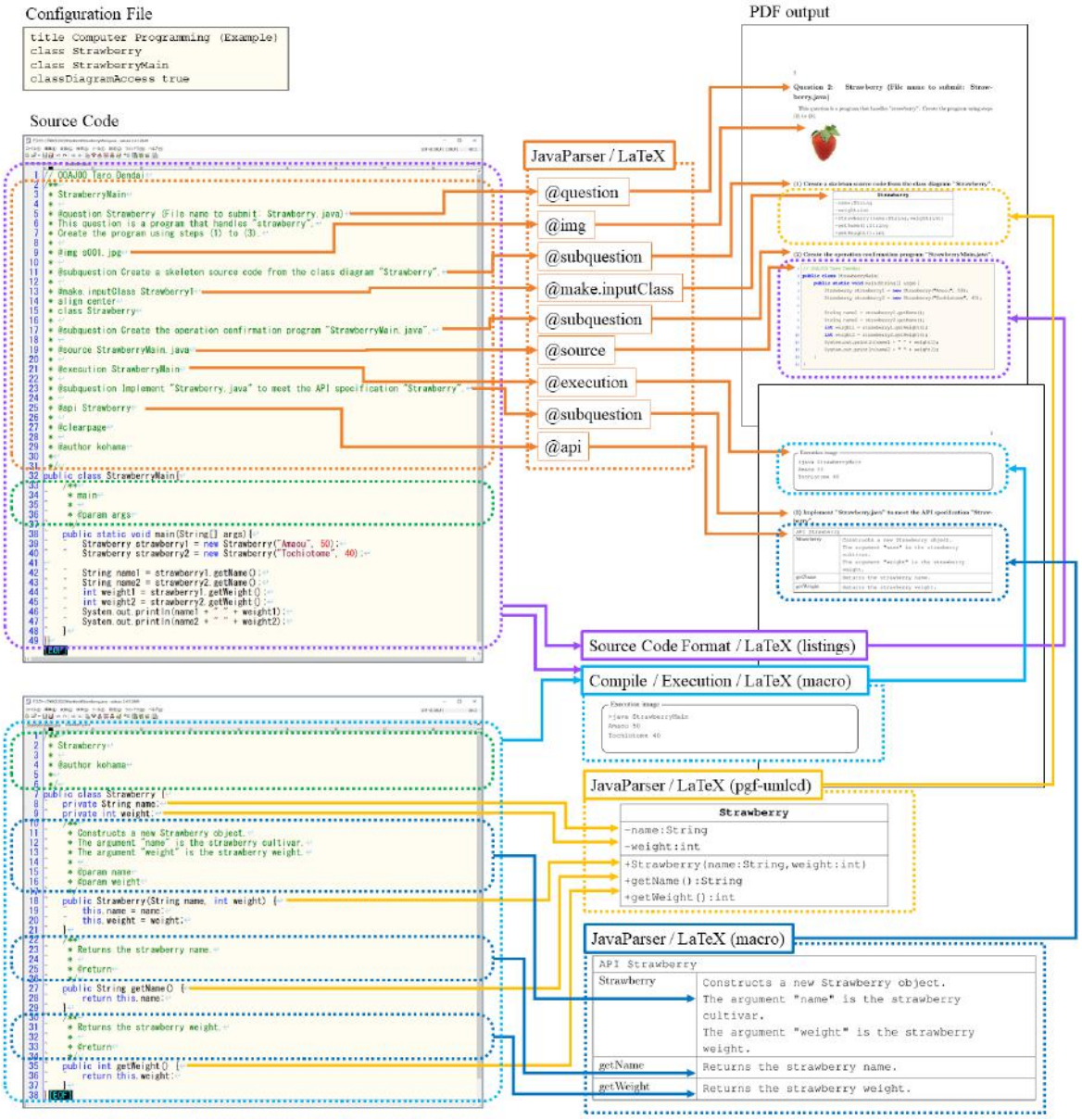

Figure 4: An example of data conversion by the tools.

```
¥ifdefined¥StrawberryClassWidth%
¥else
 ¥newlength{¥StrawberryClassWidth}
¥fi
¥settowidth{¥StrawberryClassWidth}{¥tt +Strawberry(name:String,weight:int) }
¥addtolength{¥StrawberryClassWidth}{+3mm}
¥ifdefined¥StrawberryClass%
¥else
 ¥newcommand{¥StrawberryClass}[3][text width=¥StrawberryClassWidth]{%
  ¥begin{class}[#1]{Strawberry}{#2}{#3}
   ¥attribute{-name:String}
   ¥attribute{-weight:int}
   ¥operation{+Strawberry(name:String,weight:int)}
   ¥operation{+getName():String}
   ¥operation{+getWeight():int}
  ¥end{class}
 }
¥fi
¥ifdefined¥StrawberryClassWidthName%
¥else
 ¥newlength{¥StrawberryClassWidthName}
¥fi
¥settowidth{¥StrawberryClassWidthName}{¥tt Strawberry}
¥ifdefined¥StrawberryClassName%
¥else
 ¥newcommand{¥StrawberryClassName}[3][text width=¥StrawberryClassWidthName]{%
  ¥begin{class}[#1]{Strawberry}{#2}{#3}
  ¥end{class}
 }
¥fi
¥def¥StrawberryAssociationString#1#2{
 ¥unidirectionalAssociation{Strawberry}{}{}{String}
}
¥def¥StrawberryDashedLine#1#2{
 ¥draw[umlcd style dashed line,->] (Strawberry) -- (#1);
}
[EOF]
```

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.

Table 1: Details of the tags.

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.

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.

3

問題 1 クラス図からソースプログラムの機械的導出
(提出物 StrawberryFrame.java)

クラス図 StrawberryFrame からソースコードを機械的に導出しなさい.

StrawberryFrame
name:String weight:int
StrawberryFrame(name:String,weight:int) getName():String getWeight():int

4

問題 2 いちご (提出物 Strawberry.java)

この問題は,「いちご」を取り扱うプログラムです.(1) ～ (3) の手順にしたがって,プログラムを完成しなさい.

(1) クラス図 Strawberry からソースコードを機械的に導出しなさい.

Strawberry
name:String weight:int
Strawberry(name:String,weight:int) getName():String getWeight():int

(2) 動作確認用プログラム StrawberryMain.java を作成しなさい.

```
// 00A100 電大太郎
public class StrawberryMain{
    public static void main(String[] args){
        Strawberry strawberry1 = new Strawberry("あまおう", 50);
        Strawberry strawberry2 = new Strawberry("とちおとめ", 40);

        String name1 = strawberry1.getName();
        String name2 = strawberry2.getName();
        int weight1 = strawberry1.getWeight();
        int weight2 = strawberry2.getWeight();
        System.out.println(name1 + " " + weight1);
        System.out.println(name2 + " " + weight2);
    }
}
```

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

5

実行イメージ

```
>java StrawberryMain
あまおう 50
とちおとめ 40
```

(3) API仕様 Strawberry を満足するように Strawberry.java を実装しなさい.

API仕様 Strawberry	
Strawberry	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,重さです. 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.
getWeight	「いちご」の重さを返却します.

6

問題3 プラスチックトレイ (提出物 PlasticTray.java)

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.「プラスチックトレイ」には,複数の「いちご」が入ります.(1) ～ (3) の手順にしたがって,プログラムを完成しなさい.

(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.

PlasticTray
arrayList:ArrayList<Strawberry> = new ArrayList<Strawberry>()
PlasticTray() put(strawberry:Strawberry):void getStrawberry(index:int):Strawberry getSize():int getTotalWeight():int getAverageWeight():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).

7

(2) 動作確認用プログラム PlasticTrayMain.java を作成しなさい.

```
// 00AJ00 電大太郎
public class PlasticTrayMain {
    public static void main(String[] args) {
        PlasticTray plasticTray = new PlasticTray();
        plasticTray.put( new Strawberry("あまおう", 50) );
        plasticTray.put( new Strawberry("とちおとめ", 40) );

        int size = plasticTray.getSize();
        System.out.println("いちごの個数 " + size);
        for(int i = 0; i < size; i++) {
            Strawberry strawberry = plasticTray.getStrawberry(i);
            String name = strawberry.getName();
            int weight = strawberry.getWeight();
            System.out.println(name + " " + weight);
        }
        int totalWeight = plasticTray.getTotalWeight();
        System.out.println("合計 " + totalWeight);
        int averageWeight = plasticTray.getAverageWeight();
        System.out.println("平均 " + averageWeight);
    }
}
```

実行イメージ

```
>java PlasticTrayMain
いちごの個数 2
あまおう 50
とちおとめ 40
合計 90
平均 45
```

8

(3) API仕様 PlasticTray を満足するように PlasticTray.java を実装しなさい.

API仕様 PlasticTray	
PlasticTray	コンストラクタです.
put	「プラスチックトレイ」に「いちご」を追加します. 引数 strawberry は,「いちご」です. 「いちご」をインスタンス変数 arrayList に追加します.
getStrawberry	「プラスチックトレイ」から指定された「いちご」を返却します. インスタンス変数 arrayList から,引数 (index) で指定された位置にある「いちご」を取り出し (get),返却します.
getSize	「プラスチックトレイ」に入っている「いちご」の個数を返却します.
getTotalWeight	「プラスチックトレイ」に入っている「いちご」の重さの合計を計算し,返却します.
getAverageWeight	「プラスチックトレイ」に入っている「いちご」の重さの平均を計算し,返却します.

(補足) ArrayList<Strawberry> クラスの抜粋

ArrayList<Strawberry>()	コンストラクタです.
add(strawberry:Strawberry):boolean	リストの最後に,指定された要素を追加します.
get(index:int):Strawberry	リスト内の指定された位置 (index) にある要素を返します.ただし,順番は 0 から始まります.
size():int	リスト内にある要素の数を返します.

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