The Eighteenth International Symposium on Artificial Life and Robotics 2013 (AROB 18th '13),
Daejeon Convention Center, Daejeon, Korea, January 30-February 1, 2013

Programming Learning Support Systems Focused on Structures of Programming Language and Code

Masanori Ohshiro[1], Takashi Yamaguchi1, and Eiji Nunohiro[1]

[1]Tokyo University of Information Sciences, 4-1 Onaridai, Wakaba-ku, Chiba, Chiba, 265-8501, Japan
(Tel: +81-43-236-4667, Fax: +81-43-236-4667)
ohshiro@rsch.tuis.ac.jp

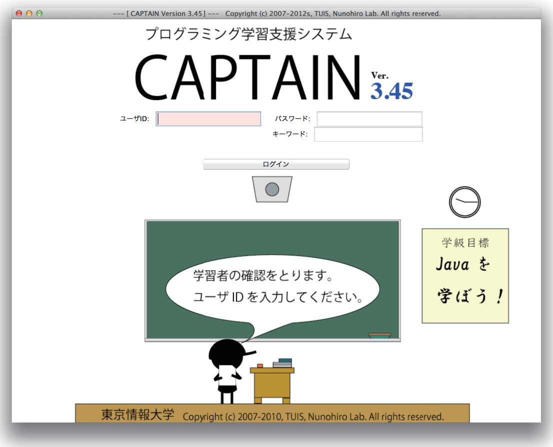
Abstract: The authors have developed a programming training system CAPTAIN (Computer Aided Programming Training And INstruction). In this training system, each complete program is fragmented randomly into a few lines by the system. Students sort the lines as an original program similarly to solving a puzzle game. In this paper, we propose an advanced feature for the system. In order to write correct programs, students must know important structures of language 's syntax. For example, block syntax is used for significant structures in Java. Therefore, in the new system, a program is divided into block syntax elements. First, contents of theses elements are empty except for their frame. Students are instructed to place them into correct position and to fill contents of the block syntax elements. It is expected that students will understand the structures of the programs in such process and their ability of writing programs will be improved.

Keywords: programming, learning, game, java

1 INTRODUCTION

The authors have developed a programming training system CAPTAIN (Computer Aided Programming Training And INstruction) and have applied the system in an actual programming course. Fig.1 is a login window of CAPTAIN[1, 2, 3, 4, 5, 6]. In this training system, learners create programs similarly to solving a puzzle game as follows. Each complete runnable program is fragmented randomly into a few lines by the system. Users must sort the lines as an original source program (fig.2). The system compiles the source program sorted by the user and checks the correctness of it.

In our previous version of the system, exercises are executed in style of a puzzle, sorting lines into correct source programs as mentioned above. It seems that such a puzzle method improves students ' ability of reading source programs and understanding algorithms in the codes. However, it will not improve learners ' ability for creating and writing programs. A typical and simple learning method for program writing is typing programs in a programming editor. But, such a typing method brings many unexpected troubles and needs sufficient time. On the other hand the puzzle method is advantageous for lessons with a time limit, for example class in school. Accordingly, we designed a new method for lesson of writing and creating programs in the puzzle-style method.


--- [CAPTAIN Version 3.45] --- Copyright (c) 2007-2012s, TUIS, Nunohiro Lab. All rights reserved.

プログラミング学習支援システム

CAPTAIN Ver. 3.45

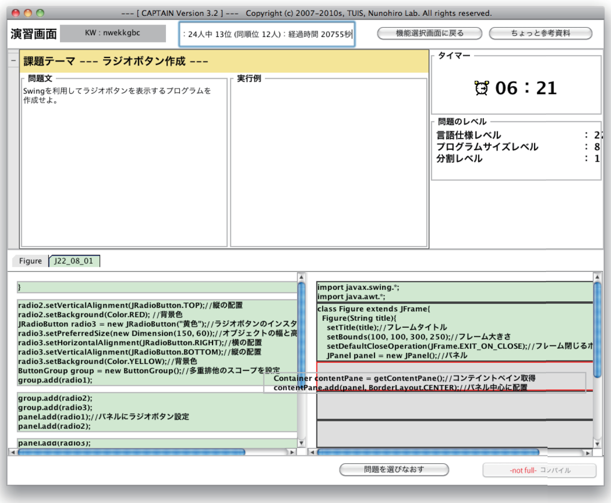
学習者の確認をとります。
ユーザ ID を入力してください。

東京情報大学 Copyright (c) 2007-2010, TUIS, Nunohiro Lab. All rights reserved.

Fig. 1. CAPTAIN

2 SYNTAX-ORIENTED FRAGMENTATION

In order to write correct source programs, students must know important structures of language's syntax. For some popular programming languages (Java, C, C++, etc.), block syntax is used as significant structures. Fig.3 shows the typical applications of block syntax in Java. These structures based on the block syntax are important part of programs. Such block syntax element is a span of tokens enclosed by a pair of parentheses, brackets, and braces. Almost part of a source program written in these languages consists of block structures. Therefore, in the new system, a program source is divided in block syntax elements. At the first, contents of theses elements are empty except their block frame. Students are instructed to place them into correct position. In succession, students must fill contents of the block syntax elements. It is expected that students will understand the structures of the programs in such process and their ability of writing programs will be improved.



: 24人中 13位 (同順位 12人) : 経過時間 20755秒

課題テーマ --- ラジオボタン作成 ---

問題文

Swingを利用してラジオボタンを表示するプログラムを作成せよ。

実行例

問題のレベル

言語仕様レベル : 2
プログラムサイズレベル : 8
分割レベル : 1

```
}
```

```
radio2.setVerticalAlignment(JRadioButton.TOP);//縦の配置
radio2.setBackground(Color.RED); //背景色
JRadioButton radio3 = new JRadioButton("黄色");//ラジオボタンのインスタ
radio3.setPreferredSize(new Dimension(150, 60));//オブジェクトの幅と高
radio3.setHorizontalAlignment(JRadioButton.RIGHT);//横の配置
radio3.setVerticalAlignment(JRadioButton.BOTTOM);//縦の配置
radio3.setBackground(Color.YELLOW);//背景色
ButtonGroup group = new ButtonGroup();//多重排他のスコープを設定
group.add(radio1);
```

```
group.add(radio2);
group.add(radio3);
panel.add(radio1);//パネルにラジオボタン設定
panel.add(radio2);
```

```
panel.add(radio3);
```

```
import javax.swing.*;
import java.awt.*;
```

```
class Figure extends JFrame{
  Figure(String title){
    setTitle(title);//フレームタイトル
    setBounds(100, 100, 300, 250);//フレーム大きさ
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//フレーム閉じるホ
    JPanel panel = new JPanel();//パネル
```

```
Container contentPane = getContentPane();//コンテイントペイン取得
contentPane.add(panel, BorderLayout.CENTER);//パネル中心に配置
```

Fig. 2. Each learner must sort fragmented and randomly listed parts of a program at the lower left panel into the correct order at right lower panel using drag and drop.

3 METHOD OF THE FRAGMENTATION

The method of syntax-oriented fragmentation besed on block syntax is displayed in fig.4. Lines of a source code are symbolized and analyzed using simple syntax parsing. Blank lines in the source program are regard as equivalent to each other. These blank lines are defined as follows:

```
blankline = B3 = B6 = B9;
```

These blanklines are ignored in symbolic expressions mentioned below. All lines that contains the same contents are regard as equivalent to each other and defined as follows:

```
C3 = D4 = C6 = C9 = C12 = C15 = A3;
```

A Class 'Person' is analyzed and described as follows:

```
"class Person"
  = A1, A2 { B1, B2, B4, B5,
             B7, B8, B10 }, A3;
```

The above symbolic expression describes that a class 'Person' is defined as sequence of A1, A2, A3. Moreover, the expression shows A2 is set of B1, B2, B4, B5, B7, B8, B10 (the blank lines are ignored). In brief, contents of a pair of braces may appear in random order in these symbolic expressions. If some part of contents of such a pair of braces must be in particular order for some context, we may use a pair of brackets as follows:

```
{ B1, [ B2, B3 ] }
```

In this case, two instances shown below are permitted.

```
B1, B2, B3
B2, B3, B1
```

In many cases, such necessity of particular order is caused by semantic context, therefore `[ ]` may be specified manually. The expression below describes method 'setAge' is defined as B5 or a sequence of C4, C5, C6.

```
class A {
  // members
}
```

```
int max ( int a, int b ) {
  // statements
}
```

```
if( x > 0 ) {
  // statements
}
```

```
for ( i = 0; i < n; i++ ) {
  // statements
}
```

```
else {
  // statements
}
```

```
try {
  // statements
}
```

```
while ( i < n ) {
  // statements
}
```

```
catch ( Exception e ) {
  // statements
}
```

Fig. 3. Typical applications of the block syntax in Java.

A1					class Person {	1
A2	B1				private int age;	2
	B2				private String name;	3
	B3					4
	B4	C1			public int getAge() {	5
		C2	D1		return age;	6
		C3			}	7
	B5	C4			public void setAge(int age) {	8
		C5	D2		if(age >= 0) {	9
			D3	E1	this.age = age;	10
			D4		}	11
		C6			}	12
	B6					13
	B7	C7			public String getName() {	14
		C8	D5		return name;	15
		C9			}	16
	B8	C10			public void setName(String name) {	17
		C11	D6		this.name = name;	18
		C12			}	19
	B9					20
	B10	C13			public Person(int age, String name) {	21
		C14	D7		setAge(age);	22
			D8		setName(name);	23
		C15			}	24
A3					}	25

```
blankline = B3 = B6 = B9;
C3 = D4 = C6 = C9 = C12 = C15 = A3;
"class Person"
  = A1, A2 { B1, B2, B4, B5, B7, B8, B10 }, A3;
"method setAge" = B5 = C4, C5{D2, D3{ E1 }, D4}, C6;
```

Fig. 4. An example of syntax-oriented fragmentation besed on block syntax.

```
  "method setAge" = B5
    = C4, C5 { D2, D3 { E1 }, D4 }, C6;
```

This expression can be written in two expressions as follows:

```
   D3 = { E1 };
  "method setAge" = B5
    = C4, C5 { D2, D3, D4 }, C6;
```

class Person {	A1	"class Person"
}	A3	
private int age;	B1	
public Person(int age, String name) {	C13	"ctor Person" (B10)
}	C15	
setAge(age);	D7	
this.age = age;	E1	
return name;	D5	
public void setName(String name) {	C10	"method setName" (B8)
}	C12	
	B9	
	B3	
setName(name);	D8	
public String getName() {	C7	"method getName" (B7)
}	C9	
private String name;	B2	
if(age >= 0) {	D2	C5
}	D4	
	B6	
this.name = name;	D6	
public int getAge() {	C1	"method getAge" (B4)
}	C3	
return age;	D1	
public void setAge(int age) {	C4	"method setAge" (B5)
}	C6	

Fig. 5. An example of fragments in random order.

The source program is analyzed in the manner mentioned above. At the next, the fragments are sorted in random order. It is important that lines of every block syntax are integrated into one fragment without its contents in this process (fig.5). By the way, white spaces for indent is omitted in the display because such indents are unexpected hints.

4 USER OPERATION

Fig.6 shows typical user operation. At the first, fragments are displayed. A user is instructed to place them into correct position. User can select and move the fragments to construction panel using drag and drop (fig.6-(1)). Especially, contents of every block syntax fragment can be inserted (fig.6-(2)). A user can fill and sort fragments in construction panel any number of times. At the last, a user check the correctness of the sorted fragments.

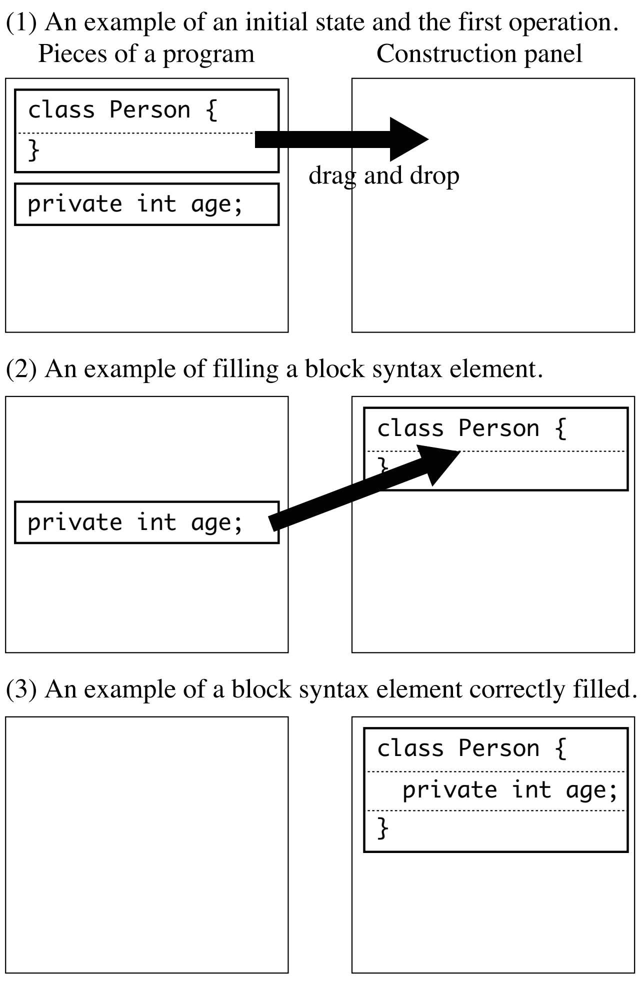

(1) An example of an initial state and the first operation.
Pieces of a program
Construction panel
class Person {
}
private int age;
drag and drop
(2) An example of filling a block syntax element.
private int age;
class Person {
}
(3) An example of a block syntax element correctly filled.
class Person {
private int age;
}

Fig. 6. An example of typical user operation.

5 CHECKING CORRECTNESS OF RESULTS

Fig.7 is a example of an answer. In order to check the correctness of the answer, the system matching the symbolic expressions extracted from original source code with the symbolic expressions extracted from user's answer according to the rules as follows:

(1) Blank lines are ignored.

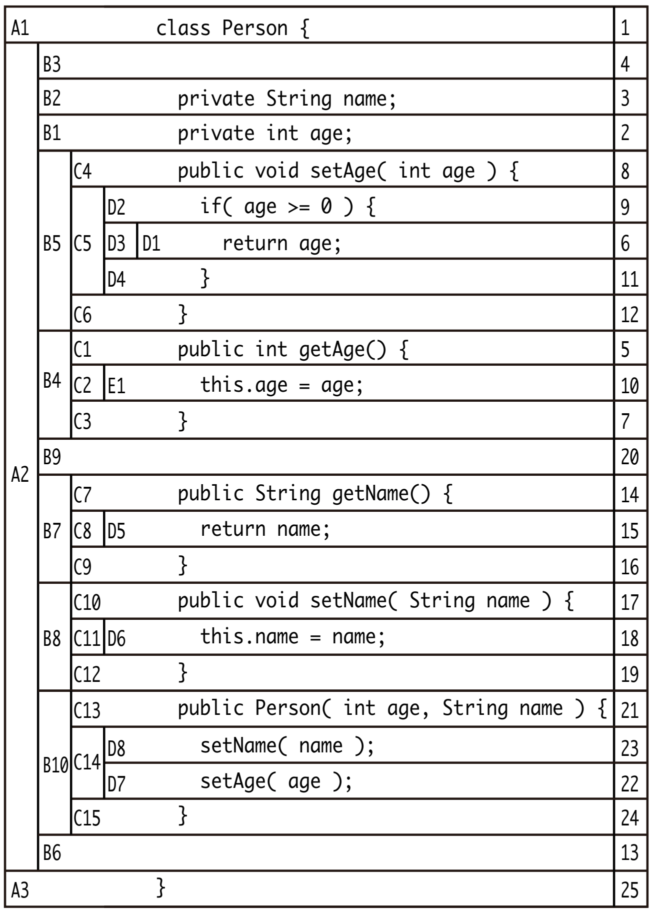
(2) Symbols in `{ }` may appear in random order.

(3) Inside `{ }`, symbols in `[ ]` must appear in specified order.

For example of fig.4 and fig.7, the first different point is detected as follows:

```
Original:
  "method setAge" = B5
    = C4, C5 { D2, D3 { E1 }, D4 }, C6;
Answer:
  "method setAge" = B5
    = C4, C5 { D2, D3 { D1 }, D4 }, C6;
```

The different part of these symbolic expressions is underlined. Therefore, the answer is incorrect.



```
A1              class Person {                          1
   B3                                                   4
   B2                   private String name;            3
   B1                   private int age;                2
      C4                public void setAge( int age ) { 8
         D2               if( age >= 0 ) {             9
   B5 C5 D3 D1              return age;                 6
         D4               }                             11
      C6                }                               12
      C1                public int getAge() {           5
   B4 C2 E1               this.age = age;               10
      C3                }                               7
   B9                                                   20
A2    C7                public String getName() {       14
   B7 C8 D5               return name;                  15
      C9                }                               16
      C10               public void setName( String name ) {  17
   B8 C11 D6              this.name = name;             18
      C12               }                               19
      C13               public Person( int age, String name ) {  21
          D8              setName( name );              23
   B10 C14
          D7              setAge( age );                22
      C15               }                               24
   B6                                                   13
A3              }                                       25
```

```
"class Person"
   = A1, A2 { B2, B1, B5, B4, B7, B8, B10 }, A3;
"method setAge" = B5 = C4, C5{D2, D3{ D1 }, D4}, C6;
```

Fig. 7. An example of an answer.

6 CONCLUSION

We proposed a new feature, syntax-oriented fragmentation for CAPTAIN. It is expected that students may pay much more attention to structures of source code and syntax of Java using this new feature. It seems that the feature is useful to know weak points of students' because it is easy to detect where the students made mistakes. We will implement the feature in CAPTAIN and examine the effect of the feature.

ACKNOWLEDGMENTS

This research was supported by The Ministry of Education, Culture, Sports, Science and Technology (MEXT), Grant-in-Aid for Scientific Research(C) (24501207).

REFERENCES

[1] Nunohiro E, Mackin K, Ohshiro M, Matsushita K, Yamasaki K (2007), Implementation of a GA driven programming training support system, The 12th International symposium artificial life and robotics, pp.517-522.

[2] Nunohiro E, Matsushita K, Mackin K, Ohshiro M, Yamasaki K (2008), Design and development of a puzzle based programming learning support system with genetic algorithm (in Japanese), Transactions of Japanese society for information and system in education, vol.25, No.2, pp.207-213.

[3] Yoneyama Y, Matsushita K, Mackin K, Ohshiro M, Yamasaki K, Nunohiro E (2008), Puzzle Based Programming Learning Support System with Learning History Management, The 16th International Conference on Computers in Education (ICCE2008), pp. 623-627

[4] Yamakawa Y, Ohshiro M, Matsushita K, Mackin K, Nunohiro E (2010), Programming Learning Support System 'CAPTAIN' with Motivational Study Model, The 18th International Conference on Computers in Education (ICCE2010), pp. 171-175

[5] Ohshiro M, Mackin K, Matsushita K, Nunohiro E, Yamakawa Y (2010), Programming Learning Support System with Learning Progress Monitoring Feature, Joint 5rd International Conference on Soft Computing and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems (SCISISIS 2010), pp.1465-1468

[6] Ohshiro M, Matsushita K, Mackin K, Yamaguchi T, Nunohiro E (2012), Programming Learning Support System with Competitive Gaming Using Monitoring and Nicknames, The 17th International symposium artificial life and robotics, pp.238-241.