Javaアプレットのプログラミング初学者のための トレース能力の修得を目的とした学習支援システムの設計

山本 雅也, 下川 亮, 高橋 陽介, 三好 健一, 福原 幸, 高橋 和也, 石崎 由也, 中村 駿介, 荒井 正之

帝京大学理工学部情報科学科

1. はじめに

Javaアプレットを用いて，プログラミングの導入教育を行っている。Javaアプレットは、GUIであるため学習者の興味、関心を引く、実行結果を視覚的に確かめられる等の利点がある。しかし、Java アプレットはイベントドリブン型のため、学習者がイベントの発生によって実行されるメソッドのトレースができない、また、メソッド内のソースコードのトレースができない、などの問題があることがわかってきた。

本研究の目的は、これらの問題点を解決するための学習支援システムを開発することである。

2. システムの概要

この章では、図1の学習者用インターフェースを例にシステムの概要について述べる。

2.1 学習者用インターフェース

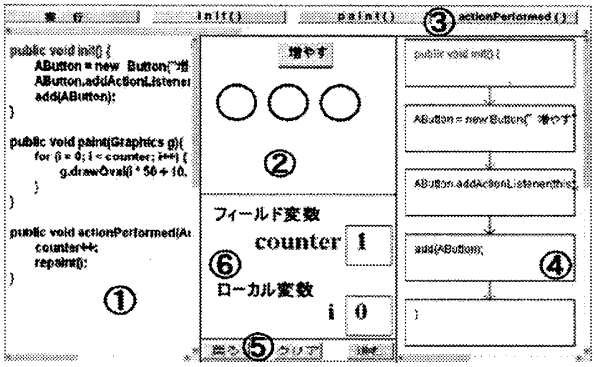

図1. 学習者用インターフェース

図1の画面構成の番号は以下のものを表している。①ソースプログラム表示部：学習者がトレースを行うソースプログラムが表示される。②アプレット表示部：①で示したソースプログラムの実行結果であるアプレットが表示される。③実行メソッド表示部：②のアプレットで「ボタンを押す」等のイベントが発生すると、そのイベントに対応したメソッドがボタンとして順番に表示される。④フローチャート表示部：③にある、メソッドを表したボタンを押すことで、そのメソッドの処理内容と手順がフローチャートで表示される。⑤トレース用ボタン表示部：進む、戻る、クリアの3つのボタンが設けられており、進む・戻るボタンを押すと①④の、該当部分が赤くなり、プログラムを1行単位でトレースすることができる。⑥変数値表示部：選択中のメソッド内のローカル変数とフィールド変数を表しており、⑤のボタンを押すことで、①④と共に変化していく。

2.2 プログラムの可視化に必要なデータ構造と画面の連動に用いるデータ構造

図 1 の①から⑥までの画面を連動させるために，本システムでは「スケジュールデータ」[1]と「実行メソッドデータ」を用いる。

図2の(1)にスケジュールデータの生成方法を示す。図1の①に示したソースプログラムの1行1行に対して、システムはスケジュールデータを得るための命令を自動的に挿入する。スケジュールデータはクラスとして実装されており、図2の(2)の"[0] m=1,1,i=0,counter=1,img=null"は、"m=1"が実行されたメソッドの順序、次の"1"がメソッド内のソースコードの行番号、i=0がメソッドで使用されるローカル変数名とその値、"counter"がフィールド変数名とその値、"img"が実行中のアプレットの画像を表している。

図 2 の(3)は、実行されたメソッドを表すボタンの情報であり、「実行メソッドデータ」と呼ぶ。スケジュールデータ内のメソッド順序と実行メソッドデータ内のボタン配列はリンクしており、例えば、実行メソッドデータの配列番号[1]の「init() スケジュールデータ内からメソッドの順序が 1 であるデータを呼び出す」は，スケジュールデータのメソッド順序が"m=1"のデータを参照する。図 1 の②に表示されるアプレットのボタンなどを押して、イベントが発生すると生成される。

3. イベント発生時の実行メソッドの可視化

本章では、2章で述べた可視化についての内部処理を、上記のスケジュールデータと実行メソッドデータを基にして述べる。

3.1 メソッド発生順序の可視化

図1の③に示したメソッドの発生する順序の可視化を行うために、実行メソッドデータを用いる。例えば、実行メソッドデータの内容が図2(3)の場合、実行メソッドデータに格納された順

「Design of a learning support system to obtain the capability of tracing for Java applet's novice programmers」

†Masaya YAMAMOTO , Ryo SHIMOKAWA , Yosuke TAKAHASHI , Ken-ichi MIYOSHI , Miyuki FUKUWARA , Kazuya TAKAHASHI , Yuya ISHIZAKI , Syunsuke NAKAMURA , Masayuki ARAI Department of Information Sciences, School of Science and Engineering, Teikyo University

に init(),paint(),actionPerformed() ボタンが図 1 の③に表示される。表示された各ボタンを押すことによって、図 1 の①②④⑥の表示が変わる。例えば、init()ボタンを押した場合は、①のソースプログラムは init()メソッドの部分を赤く表示し、④には init()メソッドのフローチャートを表示する。

(1)スケジュールデータの生成方法

```
public void init(){
        incrementButton = new Button("増やす");
        incrementButton.addActionListener(this);
        add(incrementButton);
}
```

学習者に表示するソースプログラム

↓

```
public void init() {
    data.add(new SData(m, 1, 0, counter, img))
    incrementButton = new Button("増やす");
    data.add(new SData(m, 2, 0, counter, img));
    incrementButton.addActionListener(this);
    data.add(new SData(m, 3, 0, counter, img));
    add(incrementButton);
    data.add(new SData(m, 4, 0, counter, img));
    data.add(new SData(m, 5, 0, counter, img));
}
```

スケジュールデータを生成するための命令をシステムが挿入

(2) スケジュールデータ生成の例

配列	メソッド順序,	行,	ローカル変数,	フィールド変数,	画像識別子
[0]	m=1,	1,	i=0,	counter=1,	img=null
[1]	m=1,	2,	i=0,	counter=1,	img=null
[2]	m=1,	3,	i=0,	counter=1,	img=null
[3]	m=1,	4,	i=0,	counter=1,	img=null
[4]	m=1,	5,	i=0,	counter=1,	img=null
[5]	m=2,	101,	i=0,	counter=1,	img=null
⋮					

(3)実行メソッドデータ

ボタン配列	実行メソッド名	ボタンを押した時の処理
[1]	init()	スケジュールデータ内からメソッド順序が1であるデータを呼び出す
[2]	paint()	スケジュールデータ内からメソッド順序が2であるデータを呼び出す
[3]	actionPerformed()	スケジュールデータ内からメソッド順序が3であるデータを呼び出す
⋮		

図 2. スケジュールデータと実行メソッドデータ

```
Image  img = null ;
paint (Graphics g){
    g = getGrubedGraphics();
    data.add(new SData(m, 101, 0, counter, img ));
    g.drawOval(……);
        DispCapture();
        img  = getImage();
        data.add(new SData(m, 102, 0, counter, img ));
    g.drawString(……);
        DispCapture();
        img  = getImage();
        data.add(new SData(m, 2, 103, counter, img ));
}
```

gの内容を内部処理するために画像変換用の命令を挿入

画像取得用の命令を挿入

画像取得用の命令を挿入

図 3. アプレットの画像化の処理の概要

3.2 実行中のアプレットの画像化

プログラムの実行によりアプレットも変化することがある。本節では，プログラムの実行により変化するアプレットを可視化することにより、トレースする方法について述べる。図 2(1)に示したように、システムはスケジュールデータを取得するための命令を自動的に 1 行ごとに挿入する。アプレットが変化する命令の直後には、図 3 に示すようにさらに画像を得るための命令を挿入する。図 3 は、drawOval と drawString によって描かれたアプレットの画像を取得して，img という変数に代入した後に、これらをスケジュールデータに格納する例を表している。このように、アプレットの画面が変化するたびに画像データが格納される。これらを用いることにより、アプレットもトレースできるようにした。

4. メソッド内の変数の値と処理手順の可視化

本章では図 1 の①④⑥に示したプログラムの実行行、制御構造、変数の値の可視化について述べる。

4.1 変数の値の表示

スケジュールデータを用いて、図 1 の⑥に示した変数の値を表示する。例えば、図 2 の(2)の“m=1,1,i=0, couter=1,img=null” というデータを用いた場合、ローカル変数 i の値は 0、フィールド変数を表す counter は 1 と表示される。

4.2 フローチャートの表示

図 1 の④に示したフローチャートの生成方法について説明する。システムは，図 1 の①に示したソースプログラムを構文解析し、フローチャート図記号および各命令の実行順番に関する情報を得る。これらの情報をもとにフローチャートを描く。例えば、if 文の場合は条件判断を示す記号、for 文や while 文の場合は、繰り返しを示す記号を表示する。

4.3 実行中のソースコードの表示

スケジュールデータを用いて、図 1 の①⑥に示した実行中の命令と命令に対応するフローチャート記号を赤く表示する。例えば、図 2(2)に示した”m=1,1,i=0,counter=1, img=null”というスケジュールデータの場合，2 番目のデータ”1” が実行行を表しており、ソースコードの 1 行目と、それに対応したフローチャート記号を赤く表示する。

5. おわりに

本研究では、Java アプレットプログラミングの初学者を支援するためのシステムを提案した。本システムは、実行されるメソッドの可視化機能、プログラムの実行により表示されるアプレットの可視化機能、メソッド内の制御構造の可視化機能などを持つ。

今後の課題としては、システムの実現、実授業におけるシステムの評価などが挙げられる。

参考文献

[1]山崎倫巳、荒井正之：プログラミング初学者のトレース能力の修得を目的とした学習支援システムの開発、情報処理学会第 68 回全国大会 4v-11(2006)。