1L-2

表入力による JAVA プログラムの設計支援ツール

湯浅 正徳† 梅澤 大志† 今野 俊直† 吉田 信一朗† 渡辺 康之† 紫合 治†
東京電機大学 情報環境学部†

1. はじめに

既存の UML ツール[1][2][3]は，クラス図を作成する際，クラス図の描画と設計情報の入力を同時に行なうものが多い．だが，そうした入力形式では，マウス操作とキー操作が混同し，入力に余計な時間がかかってしまう．そこで本研究では表入力を用いてキー操作に統一することで短時間での入力が可能となるプログラムの設計支援ツールを開発した．図 1 に実行画面の全体を示す．表で設計情報の入力を先にまとめて行い，その後クラス図を作成することで，入力だけに集中する事ができるため，クラス図の作成までの時間を短縮することができる．

本ツールは，設計情報の登録を表入力の形式で行い，入力が全て完了した後にクラス図の表示を行う．また，それとは別に表の右側に設計情報の更新を随時反映させた簡易的なクラス図

図 1　ツール実行画面

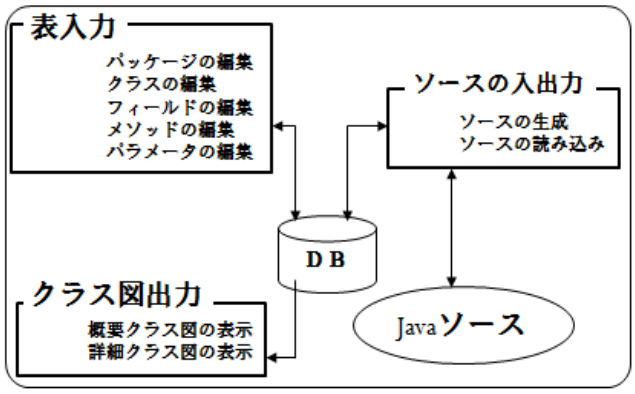

図2　全体のシステム構成

Java Program Design Support Tool using Table Input
† Masanori Yuasa, Masashi Umezawa, Toshinao Konno, Shinichiro Yoshida, Yasuyuki Watanabe and Osamu Shigo
† School of Information Environment , Tokyo Denki Universitv

を表示ができる．そして，表に入力された設計情報をソースとして書き出すことも可能になっている．ツールの構成を図 2 に示す．

2. ツールの概要

2.1. 設計情報の登録

設計情報の登録は図3，図4，図5，図6で示す表で行う．入力可能な設計情報はパッケージ，クラス，フィールド，メソッド，引数に関する名前や値，型などのプログラムの骨組みにあたる部分である．

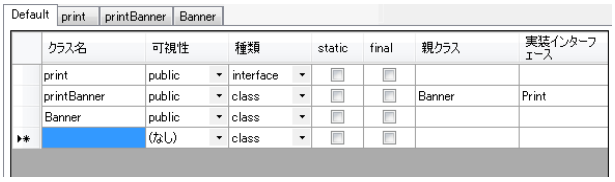

図3 表入力部分(パッケージ)

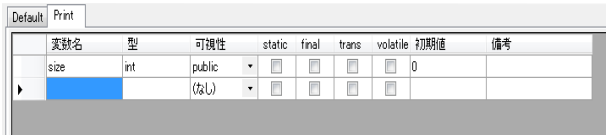

図4 表入力部分(フィールド)

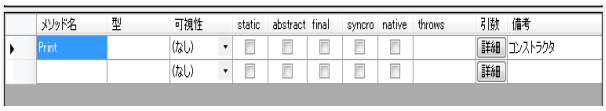

図5 表入力部分(メソッド)

図6 表入力部分(引数)

表のセルの形式は，テキスト型，リストボックス型，チェックボックス型，ボタン型の4種類があり，入力し易いようになっている．また，パッケージとクラスごとにタブを作成することができ，タブを切り替えることでそれぞれの設計情報を入力していく．パッケージのタブでは図3，クラスのタブでは図4，図5が表示され，図6は図5内の引数のボタンを押下することで表示される．

2.2. 簡易クラス図

簡易なクラス図とはクラスの名前だけを表示したクラス図であり，クラス間の関係を把握するためのものである．表の右側に常に表示されている．

また，設計情報を更新する度に簡易クラス図も更新されるようになっている．図7に表示例を示す．

図7 簡易クラス図表示の例

2.3. 詳細なクラス図

簡易クラス図とは異なり，詳細なクラス図は，メソッドやフィールドも表示した完全な形のクラス図を指す．こちらは別ウィンドウで表示される．メソッドの引数やフィールドの初期値はマウスオーバーによって表示される．表からクラス図を出力することによりクラス間の関係が視覚化出来ることや，メソッドやフィールドの表示によってプログラムの作成を支援することを目的とした機能である．図8に詳細なクラス図の表示例を示す．

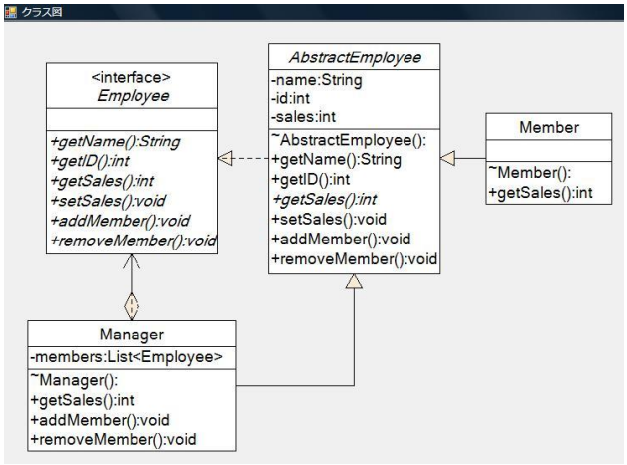

図8 詳細なクラス図表示の例

2.4. ソースの読み込み及び，書き出し

本ツールにはソースの入出力機能を備えている．既に作成されたJavaのソースを読み込み本ツールの設計情報DBに自動変換することができる．このときソース中のコメントの一部は備考欄の情報としてDBに取り込まれる．また，本ツールで作成した設計情報を基に，スケルトンソースを出力できる．

本ツールには，メソッドの中身などをプログラミングするエディタの機能が備わっていない．そのため，メソッドの中身をプログラミングする場合ソースを一旦出力し，他のエディタなどを使用してコーディングする必要がある．メソッド内部の入力支援を行えるエディタの開発に関しては，今後の課題として取り組む必要がある．

3. 実験，評価，考察

今回評価実験を行うにあたり，クラス図作成では比較を行う既存ツールとして，我々が使いやすいと感じた「astah」を用いた．内容は，こちらで用意したソースを基にクラス図を作成してもらい，完成までにかかった時間と入力やクラス図の誤りの数を計った．ソースの構成は，クラスが5つ，インターフェースが4つで，それぞれのメソッド数を平均4つとした．実験は7名で行い，その結果を表1に示す．

表1 実験結果

被験者	時間(分)		誤り	
	Astah	本ツール	Astah	本ツール
A	43	34	3	1
B	33	24	2	1
C	65	60	7	6
D	55	30	4	0
E	38	37	1	0
F	30	20	3	1
G	45	35	3	0
平均	44.1	34.3	3.3	1.3

結果として，既存ツールに比べ入力時間は平均28.5%早く完了させることができた．また，誤りの数は既存ツールに比べ，平均2.5倍少なくなった．その理由として，既存ツールの主な誤り原因となった，クラス間の依存や汎化等の関係線を引き間違えることがなかったことが挙げられる．

4. おわりに

表入力によるJAVAプログラムの設計支援ツールについて述べた．今回の実験により本ツールが既存ツールより入力に優れていることが確認された．

今後の課題として，まずテキスト入力支援機能の充実を図る必要がある．入力予測やスペルチェックによって，入力ミスによるエラーを減らし，早く入力することができる．また，インポートへの対応がある．現在はインポートが自動で追加されないため，ソース出力後ユーザが入力しなければならない．これらを改善しユーザ負担を減らしていきたい．その他の課題として，メソッド内部の入力支援を行うためのエディタの開発や，JAVA以外の言語(C++,C#など)にも対応し，必要に応じて切り替えを行えるようにすることも考えている．

・参考文献

[1] astah http://astah.changevision.com/ja/
[2] 1stModeller
http://hp.vector.co.jp/authors/VA017111/
[3] Jude http://jude.change-vision.com/jude-web/index.html