7A-5

オブジェクトの構造をテストする表明文の導入

榊原　正天[†]　　櫻井　孝平[‡]，山崎　雄大[‡]，斉藤　瞳[†]，古宮　誠一[‡]

[†]芝浦工業大学工学部情報工学科　　[‡]芝浦工業大学大学院電気電子情報専攻

1　はじめに

オブジェクト指向プログラムにおける単体テストには，JUnit[1]というフレームワークを用いた手法がよく知られている．これはテストしたいクラスのメソッドごとにテスト用メソッドを実装し，表明文を記述する．つまり期待値との比較を行い誤りを検出しようとするものである．しかし表明によって比較できるものは，ごく一部に限られる．結果として、テストの工程に置いて大量のテストコードを記述する必要がある.またテストを進めるためにはオブジェクトの具体的な状態および抽象的な状態を知る必要がある[2].

本研究ではクラスのオブジェクトを一つずつではなく，あるまとまり，つまりオブジェクトの構造をまとめてテストする新しい表明文を導入する.

2　オブジェクトの構造の表明文

オブジェクトの構造をテストする例題を 2 つ挙げ，解決となる新しい表明文を提案する.

2.1　階層構造のテストケース

例えば，Componentインターフェースを実装したFile，Directoryクラスによって階層構造を構築できるプログラムがあり，以下のように利用できるとする.

```
Directory root = new Directory();
Directory child1 = new Directory();
Directory child2 = new Directory();

File file = new File();

root.add(child1);
root.add(child2);
child2.add(file);
```

FileとDirectoryクラスはそれぞれファイルとディレクトリを表現している．Directoryのaddメソッドは渡されたComponentオブジェクトを子として保持し，getChild(int)メソッドによって子の要素を得ることが出来るとする.
上のコードにより図 1のようなオブジェクトのグラフが構築される

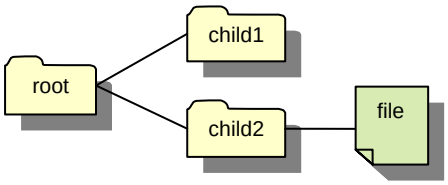

図 1：ディレクトリとファイルによる木構造の例

JUnit を利用したテストでは次のような表明文によって構築したオブジェクトのテストをする.

```
assertSame(child1, root.getChild(0) );
assertSame(child2, root.getChild(1) );
assertSame(file, child2.getChild(0) );
```

assertSame は第一引数に期待されるオブジェクトの参照を取り，第二引数に実際の値を取ることで参照の比較が行われる．上記のような表明文は各 Component に記述する必要があり，またテスト設計者はオブジェクトの状態を知るために getChild メソッドなどの定義の詳細を知らなければならない．（例えば，追加した順番に参照が得られるかどうか，など．）より簡潔で宣言的なテストコードを書くために次のような表明文を導入する.

```
from(root).toSeq(child1,
                 from(child2).to(file))
.assert();
```

この表明文では from(root)により，root オブジェクトから参照を得て到達可能なオブジェクトを格納するリストを生成する．toSeq で順番を指定して到達可能であると期待される child1,child2 オブジェクトを渡し，child2 からさらに file オブジェクトに到達可能であることを from(child2).to(file)によって定義している．最終的に assert の呼び出しによりこの表明文の検査が実行され，失敗すれば例外が投げられる.

2.2　状態遷移のテストケース

提案した表明文の別の適用例として状態遷移のテストケースを考える．以下のプログラムは表を表現する table オブジェクトを Html 形式の String 型文字列に出力する toHtmlStr メソッドのテストコードである.

The Introduction of Assertion for Testing Object Structure to Java Language
Masanori Sakakibara†,Kohei Sakurai‡,Yudai Yamazaki‡, Hitomi Saito,Seiichi Komiya‡
†Shibaura Institute of Technology
‡Graduate School of Engineerimg, Shibaura Institute of Technology

```
...
String str = table.toHtmlStr();
int i=0;
for(String line : str.split("\\n")) {
  if(i==0
     && line.indexOf("<table>")!=-1 )
    i = 1;
  else if(i==1
     && line.indexOf("<tr>")!=-1)
    i = 2;
  ...
  else if(i==10
     &&line.indexOf("</table>")!=-1)
    i = 11;
}
assertEquals(11, i);
```

出力 str の各行が<table>, <tr>,... </table> の順で現れることを str.split(“\\n”)の戻り値である String 型配列の中身をチェックすることでテストしている．

このように状態遷移に基づくテストコードは if 文と状態を表す変数を用いてそれぞれの状態を順に確認する必要がある．状態は数値コード化され，変更に弱く手続き的で誤りを引き起こしやすい．本研究では前節で提案した表明文を拡張し次のように簡潔な記述を可能にする．

```
from( str.split(“\\n”) ).toSeq(
   call(“indexOf”,THIS,“<table>”)
      .eval(“!=”,-1),
   call(“indexOf”,THIS,“<tr>”)
      .eval(“!=”,-1),
   ...
   call(“indexOf”,THIS,“</table>”)
      .eval(“!=”,-1)).assert();
```

from(str.split(“\\n”))の str.split(“\\n”) オブジェクトを起点としてつまり，split(“\\n”)によって改行で分割された String 配列の内容が指定された順序の indexOf メソッド呼び出し式の列に適合する（各 indexOf メソッドの比較にマッチする String が順番に出現する）ことを assert メソッドで検証している．

3　表明文ライブラリ

提案する表明文ライブラリの API を説明する．AssertStruct クラスを使う．

```
class AssertStruct{

  static AssertStruct from(Object src)
  ...
}
```

from(Object src)は，srcを起点とするAssertStructオブジェクトを構築し返す．AssertStructオブジェクトには，次のようなメソッドが用意されている．

AssertStruct to(Object obj)はオブジェクトの構造に対して期待される参照の到達を指定できる．引数に参照を得て到達可能であると期待されるオブジェクトを渡す．

AssertStruct toSeq(Object... seq)は順序つき集合の構造に対する参照の到達を指定できる．ある順序で到達することが期待されるオブジェクトを同じ順序で引数に渡す．

AssertStruct toSet(Object... set)は順序がない集合の構造に対する参照の到達を指定できる．toSeqと異なり，どの順序で引数にオブジェクトを渡してもよい．

static CallExpr call(String methodname, Object... args)は到達先にオブジェクトを指定する代わりに述語を指定する．argsを引数(インスタンスメソッドの対象はargs[0])としてメソッド名がmethodnameであるメソッドを呼び出し，実行した結果で到達性を判断する．

CallExprオブジェクトは，メソッド呼び出し式を表現していてeval(String op, Object obj)が用意されている．

eval(String op, Object obj)は，演算子によって式を連結する．callで呼び出されたメソッドとobjを文字列化した演算子opで連結する．

assertは，表明文の先頭からDJライブラリ[3]の探索戦略を用いて検査をする．

4　おわりに

既存の JUnit によるテストでは，オブジェクトの構造をテストする際に一つずつ表明文を記述する必要があることを例題によって示した．また，テスト設計者は参照をどのようにして得るかを知る必要があり，コードがどうなっているかを知らなければならない．さらに提案した表明文の適用例として状態遷移のテストケースを考え，テストコードの記述量を減らし簡略することができた．今後さらに一般的に適用可能な例を模索することと提案した表記の有効性の検証が課題となる．

参考文献

[1]http://www.junit.org/index.htm
[2]R.V.Binder.”Testing Object-Oriented Systems:A Status Report.”*American Programmer,* vol7, no.4,April 1994.
[3] Joshua Marshall, Doug Orleans, and Karl Lieberherr. *DJ: Dynamic Structure-Shy Traversal in Pure Java.* Technical report, Northeastern University, May 1999. http://www.ccs.neu.edu/research/demeter/DJ/.