2Q−8

Smalltalkにおけるリソースプログラミング*

○石間宏之 疋田輝雄†
明治大学 理工学部‡

1 はじめに

Smalltalkにおけるリソースプログラミングについて論じる．リソースはMacintoshやX Windowのプログラミングにおいて一般的な，ソースコードとデータを分離するための概念である[1],[3]．どのような場合にデータがリソースであるかというのは議論のあるところだが，ここでは次のような意味とする．一般にデータがリソースであるとは，それがある程度の大きさをもち，コードから分離され，複数のコードに共用され，使用において適宜置換可能なものであることである．

ところで，オブジェクト指向言語システムは，オブジェクトの集合体であると言うことができる．特にSmalltalk[2]はその感が強い．従来のリソースの概念を当てはめると，オブジェクトはいわばリソースに他ならない．このリソースの考え方は，システムに対してグローバルなリソースである．ここでは，オブジェクト個々の内部に存在するリソース，つまりクラス内リソースという考え方について述べる（図1）．

2 リソースプログラミングの例

以下に，クラス内リソースの考え方を使った簡単なプログラミング例を示す．クラスExampleClassのインスタンス変数myImageに，イメージ（ビクチャ）を持たせるものとする．

```
ExampleClass methodsFor: initializing

initializeMyImage
    myImage := Image extent: 16@16 depth: 1
                    palette: MappedPalette whiteBlack
                    bits: #[.....] pad: 16
```

ここで，斜字体で示した部分をリソースとして分離させる．

```
ExampleClass methodsFor: resources-Image

myImage
    ^Image extent: 16@16 depth: 1
            palette: MappedPalette whiteBlack
            bits: #[.....] pad: 16
```

このように，イメージのインスタンスを生成するセンテンスをリソースとして分離しておけば，後で修正・変更が容易になる．このリソースメソッドを利用するために，最初に示したメソッドinitializeMyImageを次のように変更する．

```
ExampleClass methodsFor: initializing

initializeMyImage
    myImage := self myImage
```

3 クラス内リソース

3.1 オブジェクト指向言語におけるリソース

一般に，あるクラスのインスタンスであるオブジェクトは，クラスで定義されたいくつかのオブジェクトとメソッドを持っている．また，メソッドはオブジェクトに対するメッセージセンディングによって構成される．メッセージは引数としてオブジェクトを伴うことがしばしばである．このように，メソッドの内部つまりクラス内部には，様々なオブジェクトが存在していることになる．これらのオブジェクトの中には，リソースとしての性格をもつものが存在している．

リソースオブジェクトのありえる位置は，メソッド上の次の三つである．

- レシーバオブジェクトがリソースである場合
- メッセージ引数がリソースである場合
- センテンス全体がリソースである場合

これらの，メソッド中に埋もれたオブジェクトを，リソースとして分離しようというのがクラス内リソースの趣旨である．

一般に，オブジェクト指向言語において，リソースを分離する方法は二つ考えられる．一つはリソースのみを提供するクラスを作成することである（これをリソースクラスと呼ぶことにする）．リソースが必要になったときに，リソースクラスのインスタンスに対しメッセージを送ることにより，リソースを得ることができる．もう一つの方法は，通常作成するクラス内にリソースを保持させることである（これをクラス内リソースと呼ぶ）．前者の方法は，リソースクラスが保持しているリソースの使用者を特定することができない．それに対して後者は，リソースの使用者が明白である．リソースを必要としているクラスを特定できることが重要と考え，ここでは後者のクラス内リソースを採用する．

3.2 リソースの形態

クラスの内部には，上述のように，様々なオブジェクトが存在している．クラスあるいはインスタンスの初期設定段階で，内部変数に代入されるという形で存在しているオブジェクトもあれば，メソッド記述の中に現れるオブジェクトもある．

3.2.1 リソースオブジェクト

リソースオブジェクト(resource object)とは，クラス内でリソースとしての性格を持つオブジェクトである．内部変数によって参照されているオブジェクトは，多くの場合リソースオブジェクトであるといえるであろう．基本的に，そのオブジェクトがリソースであるかという判断は，仕様記述段階でなされるべきものである．

内部変数（アトリビュート）は一般に，インスタンスの生成段階で初期化されるが，その内部変数に代入されるオブジェクトを生成するセンテンスがメソッド中に記述されていることがしばしばである．しかしそのオブジェクトがリソースとしての性格を持つとしたら，そのオブジェクトがメソッド中に記述されてしまうことは，リソースのソースコードからの分離という観点から考えて好ましくない．このことを避けるために，リソースメソッドという概念を導入する．

3.2.2 リソースメソッド

リソースメソッド(resource method)とは，リソースオブジェクトを生成するセンテンスが記述され，かつそのリソースオブジェクトを返すメソッドである．リソースメソッドが記述されているクラスのインスタンスに対して，そのリソースメソッドを起動するメッセージを送ると，目的のリソースオブジェクトが返ってくる．つまりリソースメソッドとは，目的のリソースオブジェクトを生成し，かつメッセージセンダに対してそのリソースオブジェクトを返すためのメソッドである．

---

*Resource Programming on Smalltalk

†Hiroyuki Ishima, Teruo Hikita
‡School of Science and Technology, Meiji University

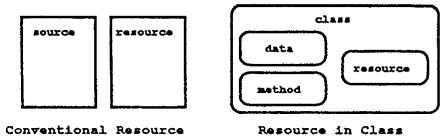

図 1: クラス内リソース

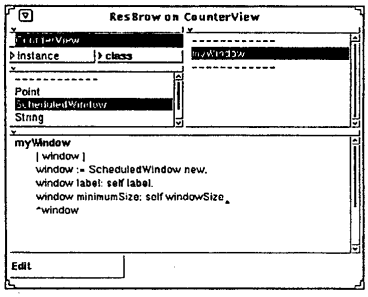

図 2: リソースブラウザ

ここで注意すべき点は，クラスはリソースをインスタンスの形（つまり何等かの変数から参照されている形）ではなく，メソッドとして保持することである．このことは，リソースを保持するためにそのリソースを参照する変数を持たなくてよいことを意味する．また，リソースメソッドはリソースオブジェクトを生成するためにのみ存在しているので，リソースの分離という目的を達成している．

3.3 クラスリソースとインスタンスリソース

3.3.1 クラスリソース

クラスリソース (class resource) はクラスから直接得ることのできるリソースである．つまりクラスリソースのリソースメソッドは，クラスメソッドとして定義される．クラスリソースが，クラス変数に代入されたオブジェクトと異なるのは，クラス変数はそのインスタンスから直接参照できるのに対して，クラスリソースはそれができないということである．インスタンスから参照する場合には，そのクラスを介して間接的に行うことになる．

3.3.2 インスタンスリソース

インスタンスリソース (instance resource) は，そのリソースメソッドがインスタンスメソッドとして定義されたものである．当然ながらインスタンスからは，いつでもインスタンスリソースを得ることができる．つまり，プログラマはインスタンスリソースをインスタンス変数と同様の感覚で扱うことができる．

3.4 リソースの種類

リソースの種類はクラスの種類と一致する．リソースはオブジェクトに他ならないから，リソースは必ず何等かのクラスのインスタンスである．

4 リソースブラウザ

ここまで述べてきたリソースオブジェクトを取り扱うためには，リソース専用のツールが必要になる．例えば，Macintosh には ResEdit と呼ばれる，リソースを作成・編集するためのツールが用意されている [1].

Smalltalk 上でリソースを取り扱うためのツールとして，リソースブラウザ (resource browser) を作成した（図 2）．このリソースブラウザによってクラス内のリソースを手軽に扱うことができ，システムブラウザと併用することによって，リソースの概念を取り入れたオブジェクト指向プログラミングが可能になる．リソースブラウザは，Smalltalk 上の既存のツールであるクラスブラウザに似ている．クラスブラウザは，ある特定の一つのクラスだけをブラウジングするためのツールであり，そのクラスのすべてのメソッドをブラウジングできる．これに対してリソースブラウザは，ある特定のクラスのリソースメソッドだけをブラウジングできるものである．

あるメソッドがリソースメソッドであるかどうかの判断は，プロトコル名（メッセージカテゴリ名）で行うことにする．リソースメソッドのプロトコル名の構造は次のようになっている．

resources-ClassName

リソースメソッドのプロトコル名は必ず resources- で始まると定義する．その後に続く ClassName とは，そのリソースメソッドが返すリソースオブジェクトのクラス名である．そのリソースメソッドが定義されているクラス名ではないことに注意されたい．例えば，'Foo' というクラスに定義されている 'resources-String' というプロトコル名のリソースメソッドは，String クラスのインスタンスであるリソースオブジェクトを返す．

プロトコル名で，そのメソッドがリソースメソッドであるかどうかを判断するこの方法は，従来の Smalltalk の枠組を逸脱することなく，かつリソースブラウザによってリソースを分離して扱うことができる．

リソースブラウザの左上のウィンドウにはリソースメソッドのプロトコル名が表示される．システムブラウザ等で見ると resources-ClassName のように表示されるが，リソースブラウザではクラス名のみが表示され，resources- は省略される．メソッド定義は通常のブラウザと全く同様に行なうことができる．

左下には Edit というボタンがある．これは他のブラウザーには無いリソースブラウザに特徴的なものである．このボタンは，作成したいリソースが属するクラスに応じた専用のエディタが用意されているときに，そのエディタを起動するためのものである．

5 まとめ

本稿では，オブジェクト指向言語でのリソースプログラミングを可能にするクラス内リソースの概念について提案した．さらに，Smalltalk を例に実際のプログラミングを試み，その有意性を論じた．

クラス内リソースの考え方は，Smalltalk に限らず，オブジェクト指向言語全般に適応出来る単純な概念であり，ソフトウェアの部品化にもつながるものである．しかし，リソースを容易に扱うことのできるツールを用意しなければ，この概念を持ち込む意味は半減してしまうであろう．ここで述べたリソースブラウザはそのツールの一例であるが，これだけでは不十分で，5 節で述べたように，クラスごとに専用のエディタを完備してはじめて真価を発揮できる．

また，ここではリソースをリソースメソッドという形で分離している．リソースメソッドは，実行時に初めてリソースオブジェクトを生成するので，実行時以外に，不必要なオブジェクトのためのメモリを消費しないという利点がある．

今後は，Smalltalk 上においてリソースブラウザ周辺の環境を整備し，リソースプログラミングがより快適に行なえるプログラミング環境を実現したいと考えている．

参考文献

[1] Alley,P. and Strange,C., 井川俊彦訳: ResEdit 完全版, トッパン,1992.

[2] Goldberg,A. and Robson,D.: Smalltalk-80 the Language and Its Implementation, Addison Wesley,1983.

[3] Jones,O., 西村享監修, 三浦明美, ドキュメントシステム訳: X Window ハンドブック, アスキー,1990.

[4] Meyer,B., 二木厚吉監訳, 酒匂寛, 酒匂順子共訳: オブジェクト指向入門, アスキー,1990.