1V-03　Java上でのオブジェクトの動的変更の実現

瀬川 淳一　外山 春彦　海邊 裕
(株) 東芝 研究開発センター コンピュータ・ネットワークラボラトリ

1　はじめに

プログラム実行時にオブジェクトのメソッドやフィールドの追加、変更、削除といったオブジェクトの動的な変更が可能であれば、実行時にプログラムの機能の変更や拡張が容易になる [1]。また、プログラムの小規模の修正とテストの繰り返しが行いやすいため、プロトタイピング的な開発が行いやすい。

そこで今回我々は、動的に変更可能なオブジェクトを Java 上で扱うための枠組みの実装を行った。

2　Java 上に実装する際の制約

Java 上に動的変更可能なオブジェクトを扱うための枠組みを実装するには次の三つの方法が考えられる。

1. Java の Virtual Machine に Java のオブジェクトを直接書き換える機能を拡張する
2. 動的変更可能なオブジェクトを扱うインタプリタ言語を Java 上に開発する
3. 複数の Java のオブジェクトから構成される一つの複合オブジェクトを動的変更可能なオブジェクトとして扱う

1 の方法を用いると、利用者は拡張れた Virtual Machine 上でしか動的変更可能なオブジェクトを動かすことができないため、Java の利点であるプラットフォーム非依存性が利用できない。

2 の方法は、インタプリタ言語独自のオブジェクトを動的に変更可能なものとして扱うため、Java の標準 API として提供されているクラスから動的に変更可能なオブジェクトを生成するのが難しい。

3 の方法は、複数の Java のオブジェクトと Java のリフレクション API を用いて実現できるため、Java の実行環境には変更を加えない。また、Java の標準 API として提供されているクラスから生成したオブジェクトを動的変更可能なオブジェクトとして扱うことができる。

Dynamic change of Objects Behaiviors on Java
Jun'ichi SEGAWA, Haruhiko TOYAMA, Hiroshi KAIBE
Computer and Network Systems Laboratory,
Corporate Research and Development Center,
TOSHIBA Corporation

上記理由により我々は、複数の Java のオブジェクトを一つの動的変更可能なオブジェクトとして扱うことのできる複合オブジェクトを実装した。

3　動的変更可能なオブジェクトの実装

我々の実現した複合オブジェクトは図 1 のように、インタフェイスオブジェクトと、複数の Java のオブジェクトから構成される。

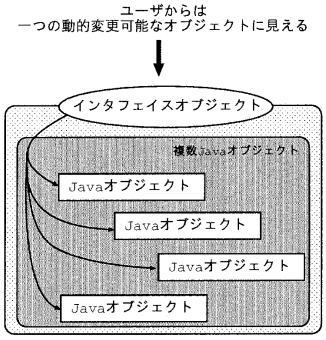

図 1: 複合オブジェクトの構成

インタフェイスオブジェクトは、動的変更可能なオブジェクトとしてのインタフェイスの提供と、内部にある複数の Java オブジェクトの管理を行う。ユーザは、インタフェイスオブジェクトの提供するインタフェイスを通じて、複数の Java のオブジェクトを操作することにより、複合オブジェクトを一つの動的変更可能なオブジェクトとして扱う。

このインタフェイスは、大きく分けて二種類ある。一つ目が、現在複合オブジェクトを構成する複数の Java オブジェクトが保持している全てのメソッドとフィールドの一覧の出力、あるメソッドの呼び出し、あるフィールドの書き込みあるいは読み出しといった、複合オブジェクトへの動的な変更を伴わない操作である。二つ目が、複合オブジェクトへの新たなオブジェクトの追加や、既に管理されているオブジェクトの削除や、既に管理されているオブジェクト同士の入れ替えといった、複合オブジェクトの動的変更を行う操作である。

インタフェイスオブジェクトは、複数の Java のオブジェクトを図 2 のように層状に重ね合わせて管理する。各層には優先度を設定しており、上の層ほど

優先度が高い。この優先度はメソッドとフィールドの検索に用いる。

3.1 メソッドの操作

インタフェイスオブジェクトが、ユーザからの複合オブジェクトに対するメソッド呼び出しを受け取ると、要求されたメソッドを持つ Java のオブジェクトを Java のリフレクション機構を用いつつ優先度に従い探し出す。次に、見つかったオブジェクトのメソッド呼び出しを行い、さらにフィールドの統合を行う。フィールドの統合については後述する。

ここで、インタフェイスオブジェクトを介した複合オブジェクトに対するメソッドの呼び出しにおいて、同一メソッドシグネチャで宣言されたメソッドが複数の Java オブジェクトに存在する場合、上の層のメソッドが優先度に従い呼び出されるため下の層のメソッドは上の層のメソッドにオーバーライドされる。

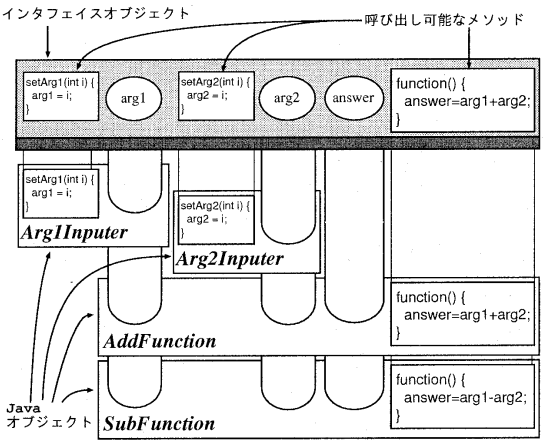

図 2: オブジェクトの層構造

同様に、このような層構造に対し、新しい Java のオブジェクトの追加、指定した Java のオブジェクトの削除、特定の二つの層の入れ替えを行うことにより、動的変更可能 なオブジェクトとしてのメソッドの追加、削除、変更が可能となる。

例えば、図 2 において、`function()` の呼び出しを要求すると、*AddFunction* が持っているメソッド`answer = arg1 + arg2` が実行されるが、*AddFunction* と *SubFunction* を入れ替えると *SubFunction* が層の上側になるので *SubFunction* が持っている`answer = arg1 - arg2` が実行される。

3.2 フィールドの統合

ここで、ある Java のオブジェクトのメソッド`func()` の呼び出しにより、そのオブジェクトの持つフィールド`val` の値が変更されるとする。

このとき、もし上の層で同じ名前のフィールド`val` が存在すると、層構造により下の層のフィールドがオーバーライドされるためメソッド`func()` により、変更されたフィールド`val` の値をユーザは得ることができない。

また、他の層のオブジェクトがフィールド`val` を持っている場合にも、メソッド`func()` によるフィールド`val` の値の変更はそのメソッドを持っているオブジェクトに局所化される。他の層のオブジェクトのフィールド`val` の値は変更されない。このため一つの動的変更可能なオブジェクトとして考えた場合、フィールド`val` の一貫性が保てない。

そこで、我々は新しくフィールドの統合機能を実装した。これは、あるオブジェクトを指定すると、そのオブジェクトの持つすべてのフィールドの値を複合オブジェクトとしてのフィールドに反映させる。インタフェイスオブジェクトは、あるオブジェクトの指定でもってフィールドの統合が要求されると、そのオブジェクトの持つすべてのフィールドの値を層構造中のすべての Java のオブジェクトの同一の名前のフィールドの値に代入する。これにより、メソッド実行によるフィールドの値の変更を層構造中の Java オブジェクト全体に波及することができ、複合オブジェクトのフィールドの値の一貫性が保てる。

4 おわりに

本稿では、Java 上で動的変更可能なオブジェクトとして扱える複合オブジェクトの実装について説明をおこなった。オブジェクトの動的な変更は、実行時に、Java のオブジェクトの層構造を操作することにより実現している。また、メソッドの呼び出しによるフィールドの変更を層構造中の Java のオブジェクト全体に波及させるために、フィールドの統合機能を提供した。

これにより、Java 上にプログラム実行時にフィールドやメソッドの変更可能なオブジェクトが実現できた。この枠組みを利用することにより、プログラムが動作中でも機能の改良や追加が可能となる。

また、層構造に取り込むオブジェクトが Java のオブジェクトであるため、Java の 標準 API として提供されている豊富なクラスから生成されるオブジェクトを動的に変更可能なオブジェクトとして利用することが可能である。

参考文献

[1] Gunther Blaschek.
Object-Oriented Programming with Prototypes
Springer-Verlag