A-032

# オブジェクトの場所を区別せずにプログラムを書く事が出来る言語機能の提案
# Programming Environment without Taking Care of the Locations of Objects

原田 英明† 山本 浩樹†
Hideaki Harada Hiroki Yamamoto

## 概 要

メモリ上、ネットワーク上、ハードディスク上など色々な場所に存在しているオブジェクトを使う時には、その場所に応じたプログラムを書く必要がある。ここで提案するのは、オブジェクトの場所を気にすることなくプログラムを書くことが出来る言語機能である。今回はオブジェクトがメモリ上とハードディスク上にある場合に限定し、この２つを区別せずにプログラムが書ける機能を考えた。この機能は Java の学習用言語である Nigari[1] を拡張する形で実装されている。そして、この言語を用いて簡単なプログラムを作成し、データがメモリ上にあるのかハードディスクに保存されているのか区別せずにプログラムが書けることを確認した。

## 1. 背景

Java などの言語では、オブジェクトはいたるところに存在している。ネットワーク上の別のコンピュータであったり、ローカルのハードディスク等である。プログラマはそのオブジェクトの場所を意識してプログラムを書く必要がある。たとえばネットワーク上の別のコンピュータのオブジェクトを扱う時には RMI を用いるし、ハードディスクに保存したオブジェクトを扱おう時にはデシリアライズを用いる。そこで、プログラマがオブジェクトの場所を意識せずにプログラムを書くことが出来るようなシステムを考えることにした。

## 2. 参照の仕組み

Java などでは内部でポインタを用いてオブジェクトの参照を表しているが、これはオブジェクトが一時的なものであるから可能である。今回は永続的に保存したオブジェクトも扱うので、以前に保存したオブジェクトも参照することが出来るポインタのようなものが必要になる。そのため、時間的にもオブジェクトを一意に識別する ObjectID というものを用意した。この ObjectID を用いてオブジェクトの参照を表すようにする。

ObjectID というのはこれまで作られたオブジェクトの総数を利用しており、オブジェクトの通し番号である。VM が終了する際に、ObjectID を外部に記憶しておいて、再起動したときにその値を読み取ってくる。このため、将来にわたって同じ ID を持つオブジェクトが重複することもない。

今回の提案するのは、オブジェクトを管理するシステム (以後 ObjectList と呼ぶ) を用意し、そのシステムが ObjectID を使い間接的にオブジェクトを参照する仕組みである。オブジェクトの参照があった場合、参照元のオブジェクトは参照先オブジェクトの ObjectID を ObjectList に登録する。参照先オブジェクトが ObjectList を参照、ObjectList はその参照先のオブジェクトを参照することにより、オブジェクト間の参照が成立する。

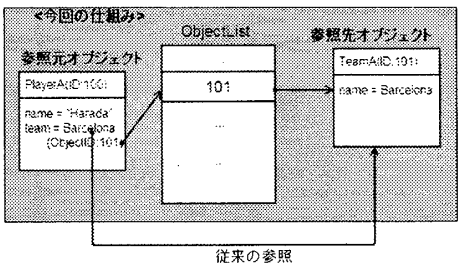


図 1: オブジェクトの参照の仕組み

## 3. 保存の仕組み

オブジェクトの参照があった場合、参照先のオブジェクトの ObjectID を保存することにより、オブジェクトが保存できる。また VM 終了時にメモリ上にあるオブジェクトを全て保存するが、１オブジェクトを１ファイルとして保存する。

### 3.1 保存すべきもの

１つのファイルに保存すべきものは次の３つである。これらをテキスト形式で保存する。

- ObjectID
- オブジェクトの元となるクラスの名前
- 「変数名・型・その値」のペア

メソッドは、クラス名からクラスファイルを探し出し、クラスファイルから復元可能なので保存する必要はない。

## 4. 復元の仕組み

オブジェクトを復元する際に、参照しているオブジェクトを全て復元してしまうのではなく、使われるオブジェクトだけを随時自動的に復元する。ここで２章で提案したオブジェクトの参照の仕組みを利用することが出来る。オブジェクトが参照状態にある場合、参照先オブジェクトの ObjectID を ObjectList に登録する。しかし、参照先のオブジェクトは復元せず、ObjectList とのリンクしない。

参照先オブジェクトの ObjectID で、ObjectList と参照先オブジェクトがリンクされていればそのオブジェクトはメモリ上にあり、リンクされていない場合は、メモリ上にないということである。

メモリ上にないオブジェクトが使われるときには、自動的に復元し ObjectList と参照先オブジェクトをリンクする。こうすることにより、使われるオブジェクトだ

†早稲田大学大学院 理工学研究科 筧研究室
{hide,hiroki}@kake.info.waseda.ac.jp

けを効果的に復元することが出来る。また、オブジェクトがメモリ上に展開されているのか、まだ展開されていないのかという情報を ObjectList が管理するため、プログラマがその事を意識することはない。

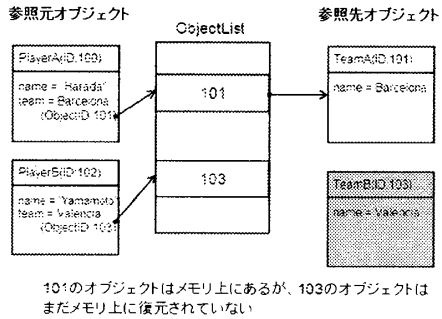


図 2: 復元されたオブジェクトの参照方法

### 4.1 復元のタイミング

参照先のオブジェクトが使われる時には、自動的に復元しなくてはならない。「オブジェクトが使われる時」を次のように定義した。

- オブジェクトからメソッドが呼ばれたとき
- オブジェクトの変数が呼ばれたとき

## 5. 実装

今回のシステムを実現するにあたり、所属研究室で開発されている言語 Nigari を用いた。所属研究室で開発されているため VM や言語仕様の変更などを含めて変更が容易であるからである。

### 5.1 保存例

図 3 の関係にあるオブジェクトを保存すると、次の図 4、図 5 のようになる。

TString、TObject、TInt は Nigari で用いられている型である。

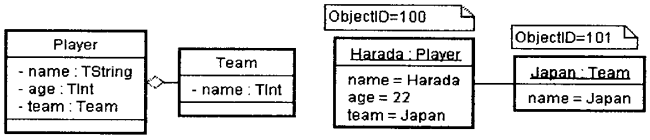


図 3: オブジェクトの参照関係

```
@100    Player
name    TString   Harada
age     TInt      22
team    TObject   10
```

図 4:ObjectID が 100 のオブジェクトを保存したファイル

```
@101    Team
name    TString   Japan
```

図 5:ObjectID が 101 のオブジェクトを保存したファイル

### 5.2 復元方法

#### 5.2.1 復元用メソッド

オブジェクトが自動的に復元されるとはいえ、ルートとなるオブジェクトは明示的に復元する必要がある。このため、Nigari に readObject という組み込みメソッドを用意した。引数として ObjectID の値をとる。ObjectID が 100 のオブジェクトを復元するには

```
object = readObject(100);
```

と書く。

#### 5.2.2 サンプルコード

次のプログラムは図 4、図 5 で保存されたオブジェクトを復元するプログラムである。3 行目と 4 行目では同じコードを書いているが、3 行目ではオブジェクトを復元し、4 行目ではメモリから参照している。

```
//ID:100 のオブジェクトを復元する
1:harada = readObject(100);

//この段階では team オブジェクトを復元しない
2:japan = harada.team;

//初めてオブジェクトの参照先を読んだので、オブジェクトを復元
3:print(japan.name);

//上と同じだが、オブジェクトは復元済みでメモリ上のデータを扱っている
4:print(japan.name);
```

## 6. 課題

### 6.1 ObjectID の問題

ObjectID には作られたオブジェクトの個数を用いているため、オブジェクトが大量に作られると、その値が非常に大きくなってしまう。

### 6.2 ネットワークを介したオブジェクト

今回はローカルの HD と、メモリ上のオブジェクトしか扱わなかったが、ネットワークを介した別のコンピューター上にあるオブジェクトも扱えるようにしたい。

## 参考文献

[1] 長 慎也他：Nigari - Java 言語へも移行しやすい初学者向けプログラミング言語
(2003 年度　情報処理学会　研究報告 「コンピュータと教育」 No.071)

[2] 原田 英明：外部リソースと、メモリ上のデータを区別しないプログラム手法の提案
http://athena.kake.info.waseda.ac.jp/~hide/files/thesis.pdf
(2003 年度　早稲田大学理工学部情報学科卒業論文)