全体ミーティング(2010/6/16)

M1　池尻　拓朗

今日の話題

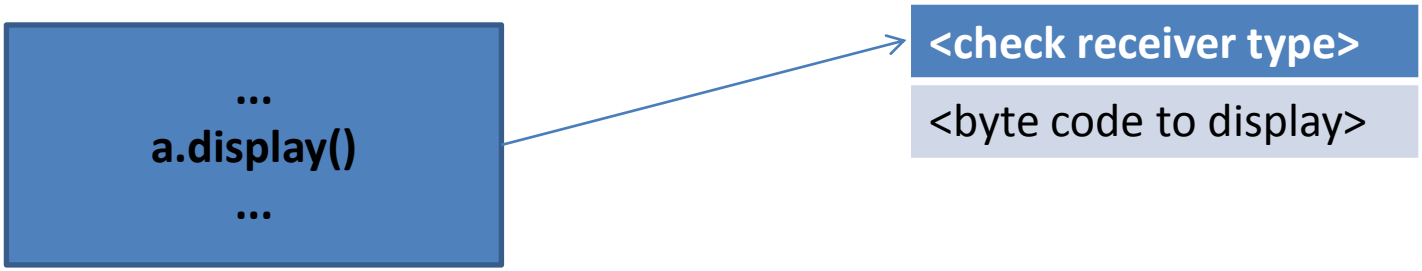
- Selfの処理系についての雑多な情報
  - 発表というより進捗報告

前回の復習:Lookup Cache

- HWのキャッシュと同じように、最新のメソッド検索結果を保存しておく
  - (オブジェクトの型、メソッドの名前)とメソッド本体のアドレスを対応させるテーブル
- キャッシュに対応するメソッドが載っていなければ、普通のルックアップルーチンを呼び出す

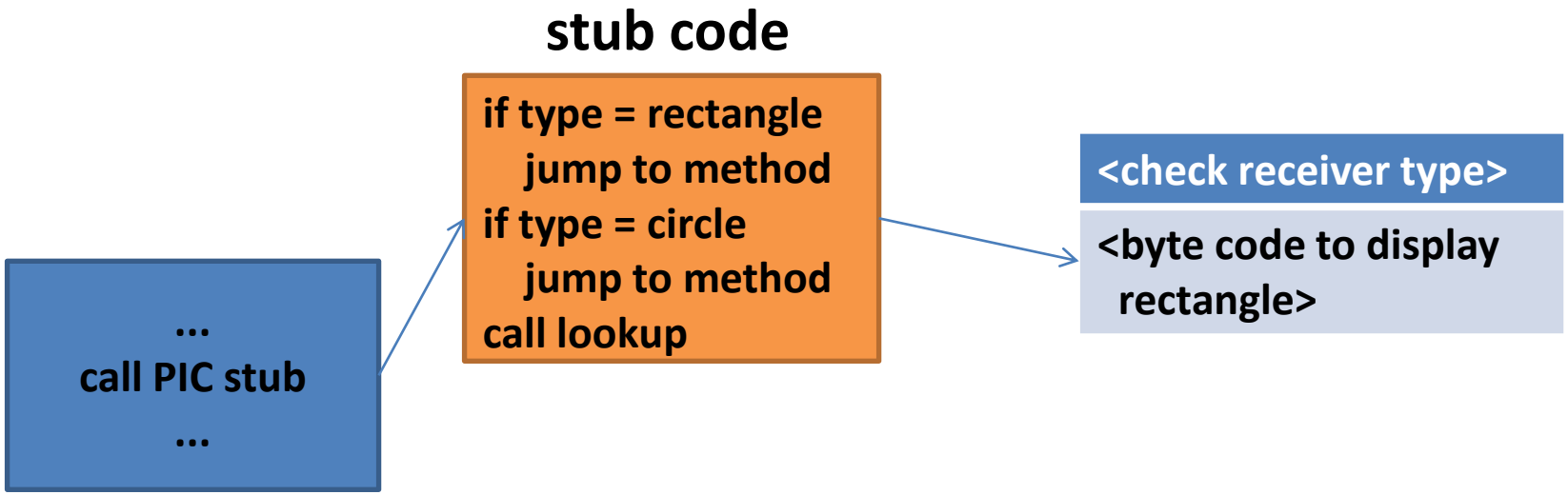
前回の復習:Inline Cache

- メソッド呼び出し側でメソッドのアドレスをキャッシュしておく
  - コードのある個所のメソッド呼び出しにおいては対象のオブジェクトの型はほとんど変わらないという仮定



前回の復習:Polymorphic Inline Cache

- ある多相的なメソッド呼び出しの全ての検索結果をスタブルーチンにキャッシュする
  - 最新の検索結果だけではない



Selfの処理系

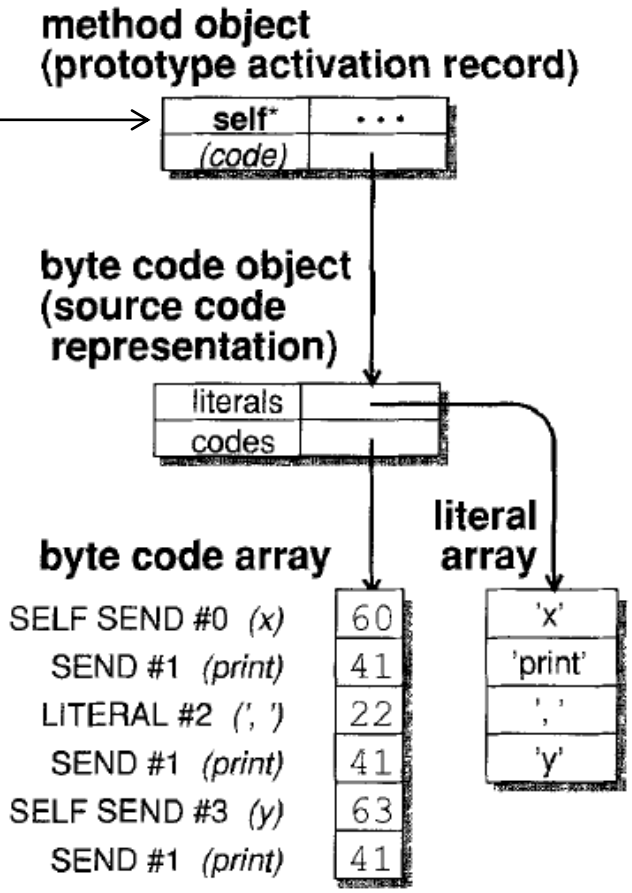
- 基本的にインタプリタ
  - C++で実装
- メソッドオブジェクトはSelfのバイトコードにコンパイルされ、SelfVM上で実行される
  - Javaと同じくスタックマシン

例

....	....
distance	
...	

Point A

parent*	
x	3
y	4
x:	←
y:	←



SimpleLookup.cpp

- 継承のルールに従った単純なルックアップ
- 継承にサイクルができていないかをチェックする工夫
  - 一度訪れたオブジェクトにマークを付けておく

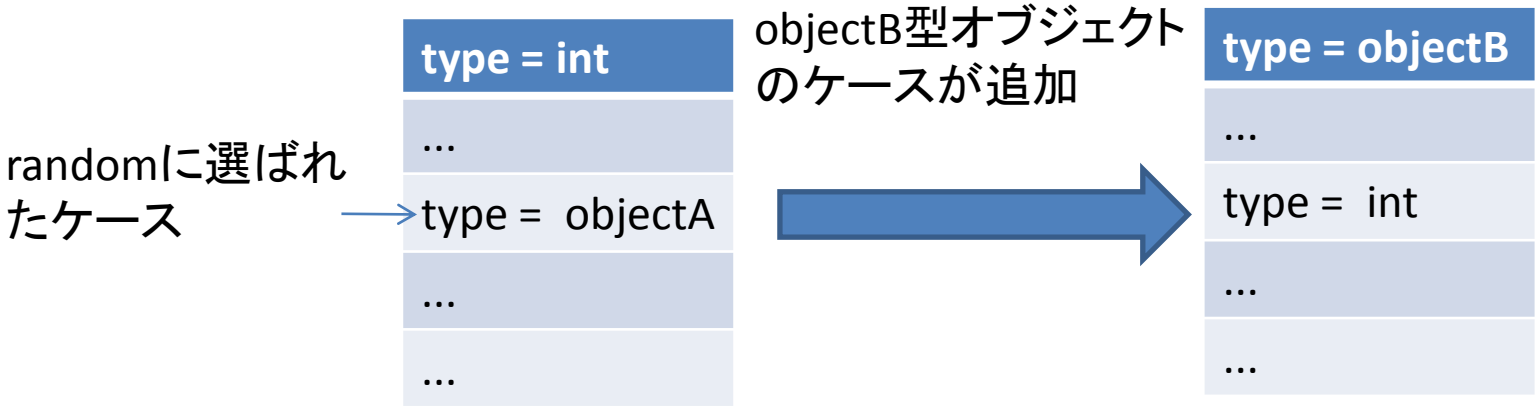
ComplexLookup.cpp

- インラインキャッシュ内のルックアップなど複雑なルックアップ

- compilingLookup
  - Lookupで見つかったメソッドをコンパイルしている模様
  - 最初のlookupまでメソッドはバイトコードになっていないかもしれない

CacheStub.cpp

- キャッシュミス時の処理
  - ランダムに最初以外のケースを選び、そこに最初のケースを移動する
  - その後、最初のケースに先程ミスしたケースを追加する



キャッシュでの型の判定

- lookup cache, inline cache, PICは全て対応するメソッドを探すのにオブジェクトの型の判定を必要とする
- 先程のソース内には該当する部分が無いと思われる
- どのソースにも現れるMapというものに注目

Map(1)

- Selfではほとんどのオブジェクトはcloneによって作られるので、プロトタイプと共通するフォーマットを持つ

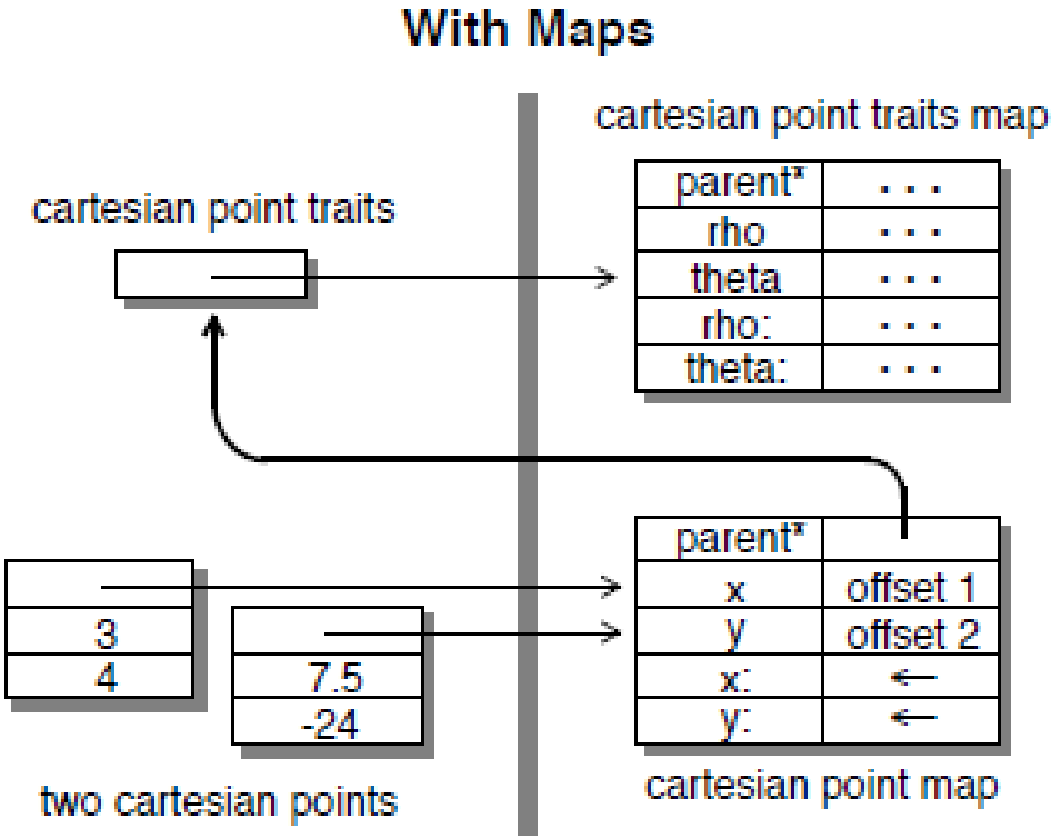
- clone family
  - あるプロトタイプオブジェクトに対して、assinableなスロットの値が異なるだけで、他はすべて同じのクローンオブジェクトの集合

Map(2)

- 一つのclone familyとそれに属するオブジェクトの対応関係

- プログラム中でオブジェクトのフォーマットが変わったり、コンスタントな値が変更されると昔のMapはそのオブジェクトに適用できない
  - 変更されたオブジェクトに対して新たなMapが作られる

例


With Maps
cartesian point traits map
cartesian point traits
parent*
rho
theta
rho:
theta:
. . .
parent*
x
y
x:
y:
offset 1
offset 2
3
4
7.5
-24
two cartesian points
cartesian point map

Mapと型の関係

- 同じMapに属するオブジェクトは同じ型として扱って良いはず

- これでオブジェクトの型チェックを行っているのではないか？

reference

- An Efficient Implementation of SELF, a Dynamically-Typed Object-Oriented Language Based on Prototypes CRAIG CHAMBERS et.al (1991)

- ソースコード: http://github.com/russellallen/self