# Proj2 Lab 8 Report

这个Lab的内容与上个lab相差不多

## 实现方式：

1. 根据获得的 `slot-name` 生成 `generic-function` ：
    - 首先，获得的 `slot-name` 的数据格式是list的形式，
      即 `all-slots` (附加了输出)：

      ```
      TOOL==> (define-class <cat> <object> name size)
      namesize(defined class: <cat>)
      TOOL==> (define-class <fat-cat> <cat> fat)
      fatnamesize(defined class: <fat-cat>)
      ```

    - 之后，对每个 `slot` 建立一个 `generic-function` ，有可能出现的情况是，对应 `slot` 的 `generic-function` 本身就存在，因此需要首先验证对应的gf是否存在，如果存在则直接跳到下一步
      验证操作可以采用上一个lab中的寻找变量的谓词

      ```
      (define (is-generic-function-in-env? var env)
       (let ((b (binding-in-env var env)))
        (if (found-binding? b)
            (let ((gf (tool-eval var env)))
                      (generic-function? gf))
            #f)))
      ```

    - 不存在时，通过 `make-initialized-environment` 相似的操作建立一个新的 `generic-function` ，这里建立的gf基于class的env
2. 建立基于gf的 `method` ：
    1. `generic-function` 的名称继承自对应的 `slot-name`
    2. 需要用到的 `param-list` 由 `class-name` 和一个局部变量 `x` 构成

3. `procedure` 可以用 `get-slot` 构造，参数是局部变量 `x` 和 `slot-name`

```
(tool-eval
  '(define-method ,entry ((x ,name)) (get-slot x ',entry)
)
env)
```

3. 完整代码：

```
;; Lab 8
(define (eval-define-class exp env)
(let ((superclass (tool-eval (class-definition-superclass ex
p)
                             env)))
 (if (not (class? superclass))
     (error "Unrecognized superclass -- MAKE-CLASS >> "
            (class-definition-superclass exp))
     (let ((name (class-definition-name exp))
           (all-slots (collect-slots
                       (class-definition-slot-names exp)
                       superclass)))
       (let ((new-class
              (make-class name superclass all-slots)))
         (define-variable! name new-class env)
         (list 'defined 'class: name)
         (for-each
          (lambda (entry)
            (begin
              (if (is-generic-function-in-env? entry env)
                  (list 'generic 'function entry 'exist)
                  (begin
                    (tool-eval
                     '(define-generic-function ,entry)
                     env)
                    ))
              (tool-eval
               '(define-method ,entry ((x ,name)) (get-slot x
 ',entry))
               env)
              ))
          all-slots)
         )))))
```

4. 测试用例：

```
TOOL==> (define-class <cat> <object> size name)
#<void>
TOOL==> (define x (make <cat> (size 1) (name 2)))
*undefined*
TOOL==> (size x)
1
TOOL==> (name x)
2
TOOL==> (define-class <fat-cat> <cat> fat)
#<void>
TOOL==> (define y (make <fat-cat> (size 100) (name 'fat-boy) (fat 'ton)))
*undefined*
TOOL==> (size y)
100
TOOL==> (fat y)
ton
TOOL==> (name y)
fat-boy
```