Names, Scopes, Bindings

Textbook Chapter 3

Binding

- Association of name with value
  - language design time
  - language implementation time
  - program writing time
  - compile time
  - link time
  - load time
  - run time
- Static vs. dynamic

Object Lifetimes

- Creating of object
- Creation of binding
- Use of binding
- Deactivation/activation of binding
- Destruction of binding
- Destruction of object
Storage Allocation

- **Static**
  - allocated at link time
  - static fields
- **Stack**
  - allocated in function call
  - local vars
- **Heap**
  - allocated dynamically
  - using new

Static vs. Dynamic Scope

```
(define x 1)
(define (bar)
  x)
(define (foo f)
  (define x 2)
  (f))
(foo bar)
```

- Static scoping: $x = 1$
- Dynamic scoping: $x = 2$

Implementation of Static Scoping

- Use Closures as function values
  - Pointer to code
  - Pointer to environment
- Remember environment in which function is defined
Construction of Closures

```
(define x 1)
(define (bar) x)
```

Global scope:

```
x 1
bar closure → (lambda () x)
```

Closure contains environment in which function is defined

Function Call

- Take environment out of closure
- Create function scope
- Define parameters in function scope
- Evaluate function body in fun scope

Construction of Function Scopes
Functions as Return Values

(define (add x)
    (lambda (y)
        (+ x y)))

(define add1 (add 1))
(define add5 (add 5))
(define i (add1 7))
(define j (add5 7))

Environments for Add Example