# Principles of Programming

Jacques Carette

McMaster University

Week #3 - Fall 2013

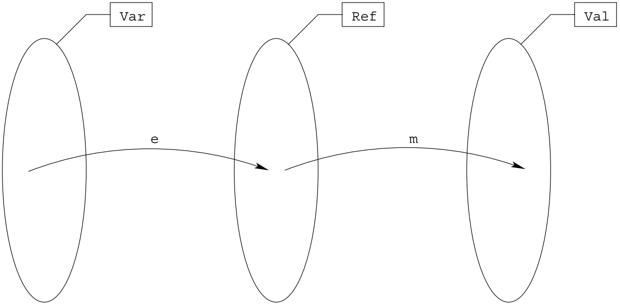
# Java Pseudo Knowledge

Things you need to use now even you do not understand now

```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello, World");

    }

}
```

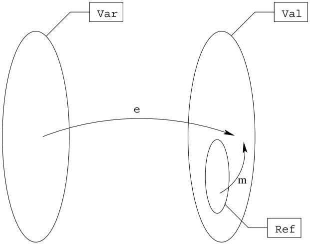
# Semantics of Core - Decomposition of State

$s : State = FinV \rightarrow Val$
$e : Environment = FinV \rightarrow Ref$
$m : MemoryState = Ref \rightarrow Val$
$s = e; m$



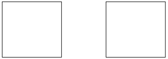
# Decomposition of State (Cont.)

- (Update function on states) $\oplus = +_e ; +_m$ where
  - (Update function on *environment*): $e +_e (x = r)$
  - (Update function on *memory state*): $m +_m (r = v)$
- Remember *Constants*! The Solutions:
  - Dependent Type: (overly complicated)
    $EnV : \{mutable, \ constant\} \times Var \to (Ref \vee Val)$:
    $EnV : (t : \{\{mutable, constant\}) \times Var \to$
    if $t = mutable \Rightarrow Ref$ else $Val$
  - $Ref \subset Val$

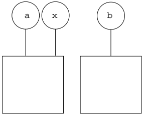


# Visual Representation of a State

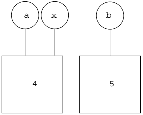
- Reference: by box



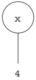
- Environment: by adding one or more labels to certain references




- Memory state: by filling each square with a value




- When a variable is associated directly with a value in the environment:



# The Value of Expressions

- $\Theta : Expr \times Env \times Mem \rightarrow Val$
  example: $\Theta(x+3, [x = r_1, y = r_2], [r_1 = 5, r_2 = 6]) = 8$
- $kind : Var \rightarrow \{mitable, constant\}$
- For Java:
  - $\Theta(x, e, m) = m(e(x))$ if $kind(x) = mutable$
  - $\Theta(x, e, m) = e(x)$ if $kind(x) = constant$
  - $\Theta(c, e, m) = c$ if $c$ is a constant
  - $\Theta(t+u, e, m) = \Theta(t, e, m) + \Theta(u, e, m)$
    ($+$ in the left side is defined on Epr and $+$ in the right side is the usual one)
    ($+$ can be replaced by any elements $\in \{+, -, *, /, \%\}$)
  - $\Theta((b)?t : u, e, m) =$
    if $\Theta(b, e, m) = true$ then $\Theta(t, e, m)$,
    if $\Theta(b, e, m) = false$ then $\Theta(u, e, m)$,

# The Value of Expressions (Cont.)

- For C: the same as Java
- For Caml: the same except
  $\Theta(x, e, m) = e(x)$ where the variable x is either mutable or constant.
  Caml also has a construct ! such that
  $\Theta(!t, e, m) = m(\Theta(t, e, m))$

# Execution of Statements

Now, the declaration of "$\Sigma$" is "$Stat \times Env \times Mem \rightarrow Mem$"
The definition of $\Sigma(p, e, m)$ depends on $p$

- a mutable variable declaration of the form "$\{T\ x = v; s\}$"
  $\Sigma(\{T\ x = v; s\}, e, m) = \Sigma(s, e \oplus (x = r), m \oplus (r = \Theta(v, e, m)))$
  where $r$ is "fresh"
- a constant variable declaration of the form "$\{final\ \ T\ x = v; s\}$"
  $\Sigma(\{final\ \ T\ x = v; s\}, e, m) = \Sigma(s, e \oplus (x = \Theta(v, e, m)), m)$
- an assignment of the form "$x = v;$"
  $\Sigma(x = v; , e, m) = m \oplus (e(x) = \Theta(v, e, m))$

# Execution of Statements - Cont.

- a sequence of the form "$\{s1\ s2\}$"
  $\Sigma(\{s1\ s2\}, e, m) = \Sigma(s2, e, \Sigma(s1, e, m))$
- a test of the form "*if* $(b)$ $s1$ *else* $s2$"
  1. *if* $\Theta(b, e, m) = true$ *then* $\Sigma(s1, e, m)$
  2. *if* $\Theta(b, e, m) = false$ *then* $\Sigma(s2, e, m)$

## Execution of Statements - Cont.

- a loop of the form "*while* $(b)$ *s*"

  a. Imaginary Statements

     1. skip; with $\Sigma(skip;, e, m) = m$
     2. give-up; with $\Sigma(give-up;, e, m) = \bot$

  b. Approximation *while* $(b)$ *s*

     p0 = if (b) giveup; else skip;
     p1 = if (b) { s p0 } else skip;
     p2 = if (b) { s p1 } else skip;
     .
     pn+1 = if (b) { s pn } else skip;

  If loop terminates then $\Sigma(pn, e, m)$ is eventually constant
  $\Sigma(while\ (b)\ s, e, m) = lim_{n \to \infty}\ \Sigma(pn, e, m)$