Object Encodings

combine data and behavior

self-aware

mutation

* classes — sets of objects
* encapsulation

FbSR

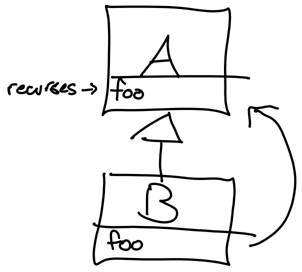
$$e ::= \cdots \mid \underbrace{\{ l = e, \ldots \} \mid e.l}_{FbR} \mid \underbrace{\text{Ref } e \mid !e \mid e := e}_{FbS}$$

- Track game history
  - wins
  - losses
  - determine total games

$$\frac{e_1 \Rightarrow v_1 \quad \cdots \quad e_n \Rightarrow v_n}{\{l_1 = e_1 ; \cdots ; l_n = e_n\} \Rightarrow \{l_1 = v_1 ; \cdots ; l_n = v_n\}}$$

```
Let obj = { wins = Ref 0;
            losses = Ref 0;
            getTotal =
               (Function this -> !this.wins + ! this.losses)
          }
In
  obj.getTotal obj
```

$$\underbrace{(\text{Function } x \to \text{Function } y \to x+y)(!\text{wins})(!\text{losses})}$$



$\boxed{\text{Object Extension}}$

```
Let obj ... In
Let obj2 = { ties = Ref 0;
             wins = obj.wins;
             losses = obj.losses;
             getTotal = Function this -> !this.ties + (obj.getTotal this)
           }
In
```

$$\begin{tikzcd} \text{Java, Python} \arrow[rr] & & \text{Dynamic Dispatch} \\ \text{C++} \arrow[urr, "\text{virtual}"] \arrow[drr, "\text{default}"'] & & \\ & & \text{Static Dispatch} \end{tikzcd}$$

this : (subobject - obj2)

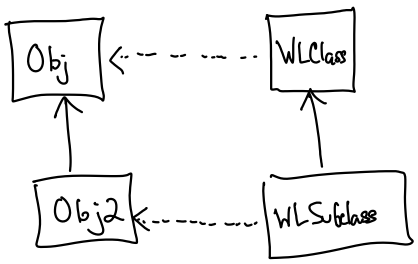
obj :

Classes

```
Let make = Function junk ->
                 { wins = Ref 0;
                   losses = Ref 0;
                   getTotal = ....
                 }
In
  Let objA = make {} In
  Let objB = make {} In
```

```
class A:
    x = 5
```

```
Let wLClass =                     wLClass (not to
                                     a WL obj)
  { new = Function this ->
                 { wins = Ref 0;
                   losses = Ref 0;
                   getTotal = ....
                 }
         ;
  }
In
Let  wL1 = wLClass.new wLClass In
```



"Let over Lambda" — Closure —

```
Let obj =
     Let private =
            { wins = Ref 0;
              losses = Ref 0;
            }
     In
       { getTotal = Function this -> !private.wins + !private.losses }
  In
  ●
```