Objects & Inheritance

Section 7
Implementing Objects in 401

Ways of implementing objects:

- Use closures as objects
- Use tables as objects
Closures as Objects

function myObject (field1, field2, field3 ...) {
  function (methodName, param1, param2) {
    if (methodName == "method1") {
      /* method1 code */
    } else if (methodName == "method2") {
      /* method1 code */
    } else {
      error("invalid action")
    }
  }
}

def x = myObject(0,1,2) // Create instance of data type called x
x("method1", 10, "asd") // Call method1 on object x
Tables as Objects

```
Datatype name       Datatype fields

def myObject = {field1 = x, field2 = y, field3 = z, ...}

myObject[“method1”] = function(self, param1, param2, ...) { /* method1 code */ }
myObject[“method2”] = function(self, param1, param2, ...) { /* method2 code */ }

// Create instance of data type called x and call method 1

def x = { field1 = 5, method1 = myObject[“method1”], method2 = myObject[“method2”] }

x[“method1”](x,a,b,c);

We can use de-sugaring to improve syntax. For example:

x:method1(a,b,c) becomes x[“method1”](x,a,b,c)
```
Exercise 1

a) Use closures to build a custom datatype Square. The data structure should have one attribute length and two methods getArea and setLength.

b) Use tables to define the same Square datatype.

For this exercise, assume no de-sugaring.
Prototypes

- Prototypes are just like regular objects
- Contain shared methods and attributes
- All objects of the a datatype extend the prototype of that datatype
  - Similar to lexical scoping, objects have a pointer to their prototype
  - When we cannot find a field or method in the object, we search the prototype
  - We keep the prototype pointer in the special \_index field.
Square example

/* Create the prototype object */
def Square = {length = 5}
Square["getArea"] = function(self) { length * length }
Square["setLength"] = function(self, newLength) { self.length = newLength }
Square["new"] = function(self) {
    def o = {}
    o.__index = self
    o
}

/* Create instance of datatype */
def x = Square["new"](Square)    or with desugaring it simply becomes:    def x = Square:new()
How `__index` is used.

⇒ def x = Square:new()

⇒ x.length = x.length + 10
Exercise 2

Using tables and prototypes, define the following datatypes and create one instance of each type. You may assume desugaring.

- Name: **Shapes**
  - No attributes
  - Methods: new

- Name: **Square**
  - Attributes: length
  - Methods: setLength, getArea

- Name: **Circle**
  - Attributes: radius
  - Methods: setRadius

- Name: **Rectangle**
  - Attributes: width
  - Methods: setWidth, getArea
Fin.