EECS 345: Programming Language Concepts

Interpreter Project, Part 5

Due Monday, April 27

In this homework, you will expand on the interpreter of part 4 by adding objects (instances of classes)

An example program is as follows:

```
class A {
  var x = 6;
  var y = 7;

  function prod() {
    return this.x * this.y;
  }

  function set2(a, b) {
    x = a;
    y = b;
  }
}

class B extends A {
  function set1(a) {
    set2(a, a);
  }

  static function main () {
    var b = new B();
    b.set1(10);
    return b.prod();
  }
}
```

As in part 4, your `interpret` function should take two parameters, a *file* and a *classname*. For example, `(interpret "MyProgram.j" "B")`, where *file* is the name of the file to be interpreted, and *classname* is the name of the class whose main method you are to run. The function should call `parser` on the file *file*, and then lookup `(string->symbol` *`classname`*`)` in the environment to get the desired class, and then lookup the `main` method of this class. The final value returned by your interpreter should be whatever is returned by `main`.

Details

1. Note that we now allow the object type in our language. So, objects can be assigned to variables, passed as parameters, and returned from functions.
2. All mathematical and comparison operators should only be implemented for integers, and logical operators should only be implemented for booleans.
3. You are *not* required to implement the `==` operator for objects, but you can if you wish.
4. The only operator that is required to work on objects is the `dot` operator.
5. The `new` operator will return an object of the desired class.
6. The `new` operator can only be used in expressions, not as the start of a statement.

7. Variables and methods can now be static (class) or non-static (instance).
8. The `main` method should be static.
9. The language supports use of `this` and `super` object references.
10. The top level of the program is only class definitions.
11. Each class definition consists of assignment statements and method definitions.
12. Nested uses of the `dot` operator are allowed.

Parser Constructs

```
class A {                   =>   (class A () body)
  body

class B extends A {         =>   (class B (extends A)  body)
  body

static var x = 5;           =>   (static-var x 5)
var y = true;               =>   (var y true)

static function main() {    =>   (static-function main () body)
  body

function f() {              =>   (function f () body)
  body


new A()                     =>   (new A)

a.x                         =>   (dot a x)

new A().f(3,5)              =>   (funcall (dot (new A) f) 3 5)
```

As there are no types, only one catch statement per try block is allowed.

Basic Tasks

Step 1: Write an interpreter that correctly handles objects and classes. You should be able to create objects (using a generic constructor), set values, call methods, and use values `this` and `super`. You do *not* have to support user defined constructors.

For Some Additional Challenge:

Add user-defined constructors. In the language, the constructor will look like a method that has the same name as the class name, but is not preceded with`function`, and in the parse tree it will be identified by `constructor`.

```
class A {
  A(x) {                =>  (constructor (x) body)
    body
  }
}
```

Constructors can be overloading, and constructors/new needs to have the following behavior:

1. Create the instance including space for all instance fields.

2. Lookup the appropriate constructor (if one exists). If no constructor exists, allow for a default constructor.
3. Call the constructor specified by the `super` or `this` constructor call that should be the first line of the constructor body (or automatically call the parent class constructor with no parameters if no `super()` is present).
4. Evaluate the initial value expressions for the fields of this class, in the order they are in the code.
5. Evaluate the rest of the constructor body.

As a hint, make the constructor list be a separate environment of the class from the method environment. That way constructors will not be inherited.