Jedi 3.0

Jedi 3.0 adds variables, stores, and iteration to Jedi 2.0.

Variables

A variable is a value that contains another value.

```
-> def count = var(0)
ok
-> count
[0]
-> [count]
0
-> count = [count] + 1
done
-> count
[1]
```

We must dereference a variable to use its contents:

```
-> count + 1
Inputs to + must be numbers or texts
```

Variables may even contain other variables:

```
-> def pcount = var(count)
ok
-> pcount
[[0]]
-> [[pcount]]
0
```

Stores

A store is a variable on steroids. It can contain any number of values:

```
-> def scores = store(85, 23, 99, 56)
ok
-> scores
{ 85 23 99 56}
-> addLast(70, scores)
done
-> scores
{ 85 23 99 56 70}
-> contains(50, scores)
false
-> get(1, scores)
23
-> size(scores)
5
-> put(60, 2, scores)
done
-> scores
{ 85 23 60 99 56 70}
-> def scores2 = map(lambda(x) x + 1, scores)
ok
-> scores2
{ 86 24 61 100 57 71}
-> scores
{ 85 23 60 99 56 70}
-> filter(lambda(x) 70 < x, scores2)
{ 86 100 71}
```

Iteration

Jedi 3.0 provides a while loop for iterative execution:

```
-> while ([count] < 10) { write("calling incCount");count = [count] + 1 }
calling incCount
calling incCount
calling incCount
calling incCount
calling incCount
calling incCount
calling incCount
calling incCount
calling incCount
done
-> count
[10]
```

We can use this to define traditional iterative functions:

```
-> def tri = lambda (n) { def result = var(0); def count = var(0); while([count] < n +
1) {  result = [result] + [count];  count = [count] + 1 }; [result] }
ok
-> tri(5)
15
-> tri(6)
21
```

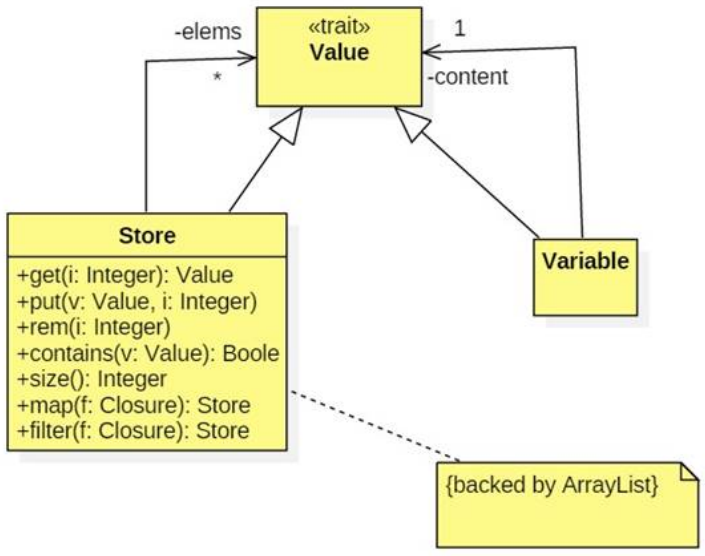
Here's a definition of tri with nice formatting:

```
def tri = lambda (n) {
   def result = var(0);
   def count = var(0);
   while([count] < n + 1) {
      result = [result] + [count];
      count = [count] + 1
   };
   [result]
}
```

Implementation

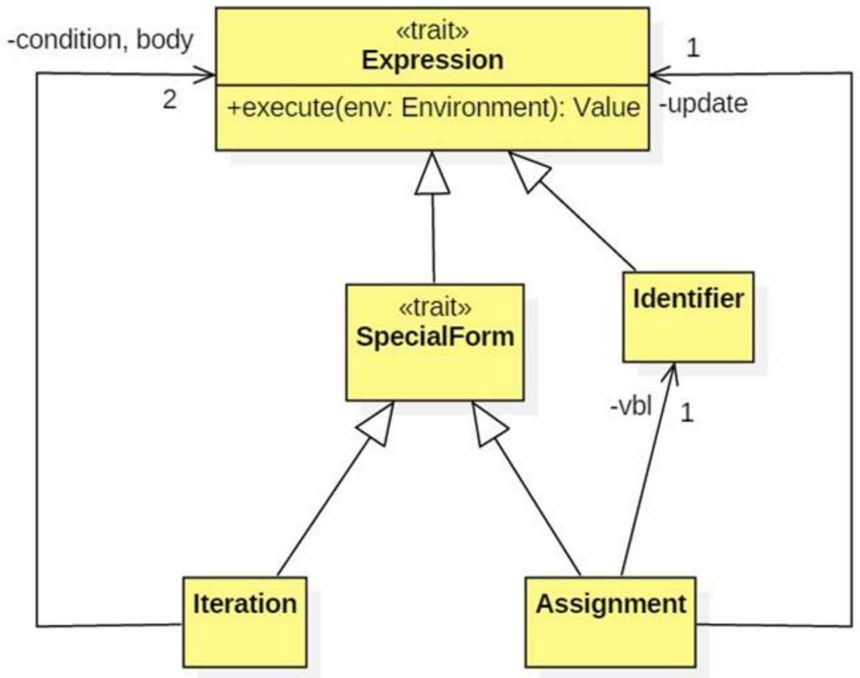
Variables and Stores

Variables and stores are new values:



Iteration and Assignment

Iteration and Assignment are new special forms:



Jedi 3.0 Grammar

Iterations are new expressions (they get parsed at the top); assignments and dereferences are new terms.

```
assignment ::= identifier ~ "=" ~ expression
iteration ::= "while" ~ "(" ~ expression ~ ")" ~ expression
dereference ::= "[" ~ expression ~ "]"
```

ALU Store and Variable Operations

Fortunately, many Jedi 3.0 features are just function calls that can get executed by the alu. We will need to add dereferencing and variable construction methods to the alu. We will also need to add all of the Store methods to the alu.

Some Files

Here's a start on the Store:

- Store.scala

Here's a start on the parsers:

- Jedi3Parsers.scala

Here's a start on the alu modifications:

`alu.scala`

This version of the console skips over blank lines in batch mode. This means you can separate expressions in test files with blank lines. Unfortunately, I can't figure out how to parse multi-line expressions. (Any suggestions?)

- console.scala

Here is a test file and the expected output:

- Jedi3Tests
- Jedi3TestResults