Jedi 1.0

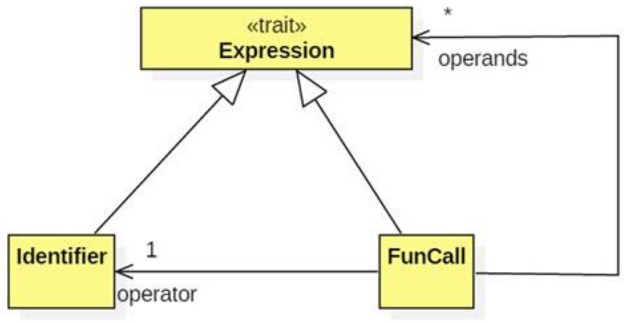
Jedi 1.0 is basically a calculator language. It is able to execute arithmetic and logical expressions. Programmers can store the values of expressions in identifiers which can be used in subsequent expressions.

1. Complete and test the implementations of the FunCall class and the alu singleton.

Here are some examples of FunCalls:

```
-> add(2, 3, 4)
9
-> mul(add(3, 4), sub(9, 3))
42
-> less(4, 3)
false
-> add(2, 3, 4.1)
9.1
-> 2 + 3 + 4
9
-> (3 + 4) * 2 + 1
15
-> 3 < 4.1
true
```

Here's a class diagram for FunCall:



Notes:

- A function call consists of an operator (which is an identifier) followed by operands-- a comma-separated list of zero or more expressions.
- Here's the EBNF rule

```
funcall ::= identifier ~ operands
operands ::= "(" ~ (expression ~ ("," ~ expression)*)* ~ ")"
```

- The operands may themselves be function calls or any other type of expression.
- The parser converts an infix expressions like 3 + 4 into the FunCall add(3, 4).
- FunCall.execute begins by eagerly executing all of its operands. This produces a list of values called arguments.
- Jedi 1.0 does not have user-defined functions (i.e., lambdas). All functions are implemented in the alu.
- FunCall.execute calls alu.execute(operator, arguments).
- alu.execute(operator, arguments) uses a match expression to call a lower level function such as alu.add(arguments)

Here's a test app for FunCalls:

FunCallTest.scala

2. Complete the implementations of Conditional, Conjunction, and Disjunction:

Here's a sample Jedi 1.0 session:

```
-> 3 < 4 && 2 == 1 && x
false
-> 2 == 1 || 3 < 4 || x
true
-> if (3 < 4) 3 + 4 else x
7
-> x
Undefined identifier: x
```

Notes:

- Conjunction (&&) and disjunction (||) use a form of lazy execution called short-circuit execution—execute operands from left-to-right until the answer is known, then stop executing operands. In both examples the expression x was never executed. We know this because executing x produces an undefined identifier error.
- Similarly conditionals (if/else) use a form of lazy execution called conditional execution—execute the condition. If it's true, execute the consequent and ignore the alternative. If it's false, execute the alternative (if it's not null, if it is, return Notification.UNSPECIFIED) and ignore the consequent.
- Jedi conditionals produce values and therefore are similar to Scala conditionals and the conditional expression in C:

```
(3 < 4)? 3 + 4: x
```

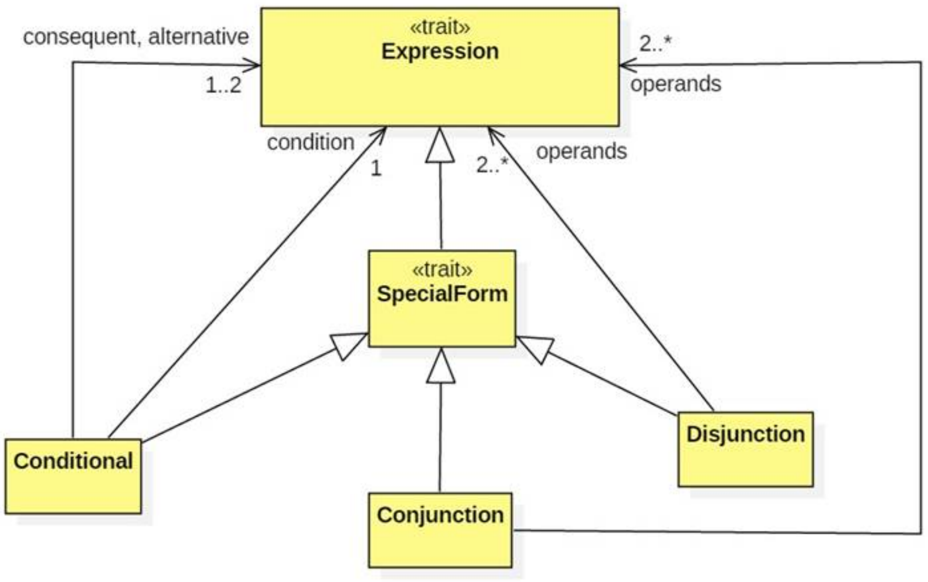
- Without some form of laziness it would be impossible to write code such as:

```
if (x != null && x.isInstanceOf[Double] && 0 <= x) y = math.sqrt(x)
```

- Here are the (simplified) EBNF rules for these expressions:

```
conditional ::= "if" ~ "(" ~ expression ~ ")" ~ expression ~ ("else" ~ expression)?
conjunction ::= expression ~ ("&&" ~ expression)*
disjunction ::= expression ~ ("||" ~ expression)*
```

- Here's a class diagram for these special forms:



3. Complete the implementation of the Declaration class

Jedi 1.0 allows programmers to bind identifiers to the values of expressions:

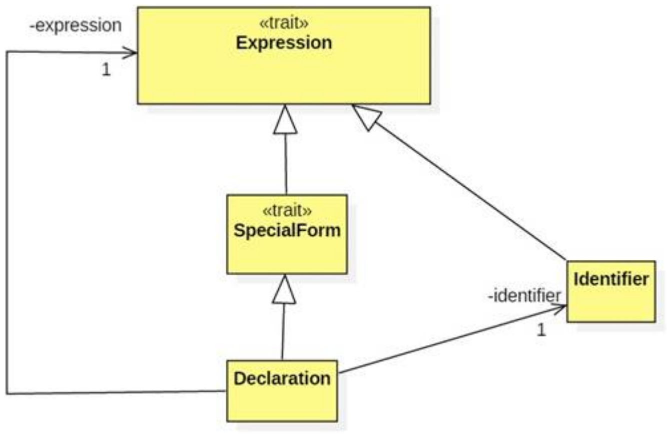
```
-> def pi = 3.14
ok
-> def e = 2.7
ok
-> def pie = pi * e
ok
-> pie
8.478000000000002
```

Notes:

- Here's the EBNF rule for declarations:

```
declaration ::= "def" ~ identifier ~ "=" ~ expression
```

- Here's the class diagram:



- Declaration.execute(env) adds a new row to env, then returns OK.

4. Implement the Jedi 1.0 context

Much of the Jedi context is implemented here:

`context/JediException.scala`

`context/Environment.scala`

`context/console.scala`

`context/alu.scala`

Notes:

- Much of Jedi's dynamic type checking happens in the ALU.

The Jedi1 Parsers are discussed here: Jedi 1.0 Parsers.

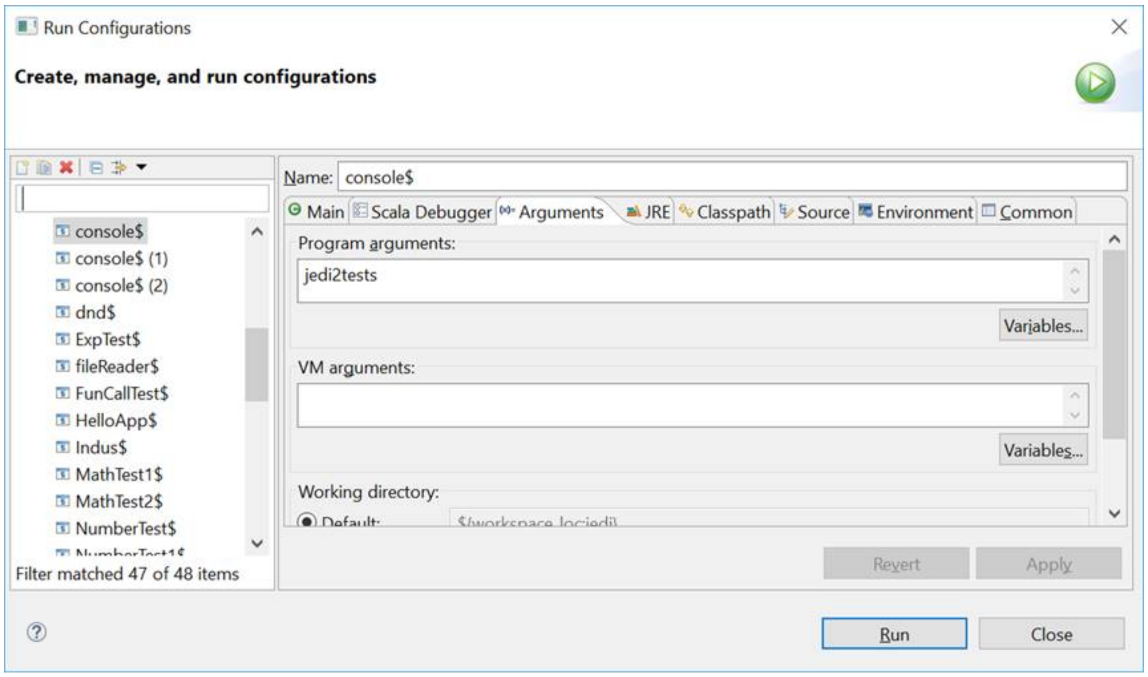
Jedi 1 Flow of Control

The flow of control through the Jedi interpreter.

Testing Jedi 1.0

In Eclipse add a file called jedi1tests.

In the Run Configurations dialog, set the program arguments to this file:



Run Configurations

Create, manage, and run configurations

Name: console$

Program arguments:

jedi2tests

VM arguments:

Working directory:

● Default: ${workspace_loc:jedi}

Filter matched 47 of 48 items

When the console runs, it should execute the expressions in this file instead of calling repl.

Here are my outputs.