



Chris Murphy  
Ryan Overbeck  
Lauren Wilcox  
Joeng Kim

# Introduction



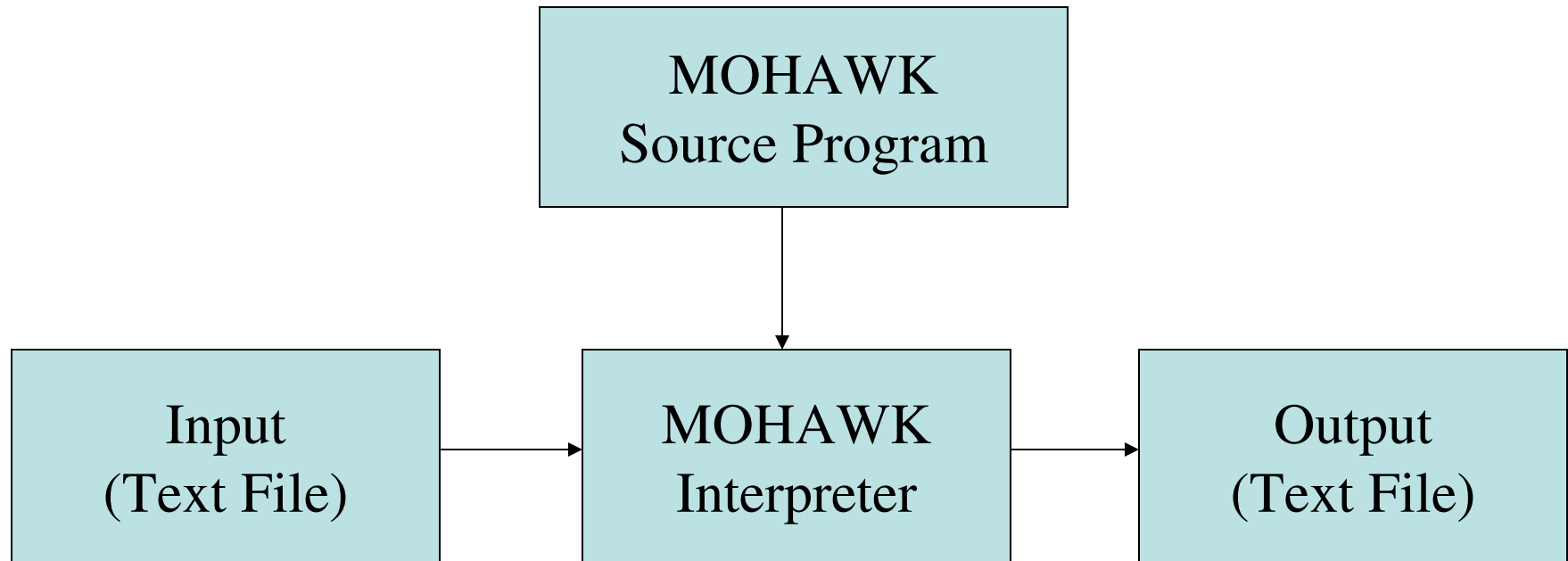
Oi! Oi! Oi!

AWK

# What is MOHAWK?

- A programming language for processing tabular data from text files
- Largely based on AWK

# Runtime Architecture



# Mohawk Program

- A series of pattern/action statements
- Patterns are:
  - “-:-|” (Forward MOHAWK)
  - “|:-” (Reverse MOHAWK)
  - A numeric, boolean, or string valued expression
  - Nothing.
- Actions are a series of:
  - Math, boolean, and string expressions
  - Conditional statements
  - Function calls
  - I/O Control

# Distinguishing Features

- Uses ‘!’ as a statement terminator
- No data types (like AWK)
- Java style comments
- Linking numbered fields with meaningful string identifiers
- New variable scoping rules
- Most keywords are IRC style smilies

# Tutorial Introduction/Example

```
:-: -|
{
max = 0!
:-O( "**BIWEEKLY WAGE REPORT**" )! //print line
}

{
totalhours = $2 + $3!
wage -> $4! //linking
total = totalhours * wage!

:-/($1 =~ "[Rr]yan$") //if statement and reg ex to match Ryan or ryan
{
:-|! //next //if it's ryan, skip this record
}

:-O($1 + ": " + $2 + " " + $3 + " " + wage + " " + total)!
:-/ (total > max)
{
name = $1!
max = total!
}
:-O()!
}

| :-: -
{
:-O( "max = " + name + " with " + max )!
:-O( "The end" )!
}
```

Annotations in the image:

- A blue arrow labeled "BEGIN" points to the line `:-O( "**BIWEEKLY WAGE REPORT**" )! //print line`.
- A blue arrow labeled "BODY" points to the line `:-|! //next //if it's ryan, skip this record`.
- A blue arrow labeled "END" points to the line `:-O( "The end" )!`.

Bottom status bar: `--:-- Total.oi (Fundamental)--L15--All--`

Input file

```
emacs@firefly.cs.columbia.edu
File Edit Options Buffers Tools Help
Chris 9 7 255.00
Lauren 3 13 245.3
Ryan 8 8 285.00
Joeng 7 9 245.1
:-- data.txt (Text CVS:1.4 Fill)--I
```

**MOHAWK**

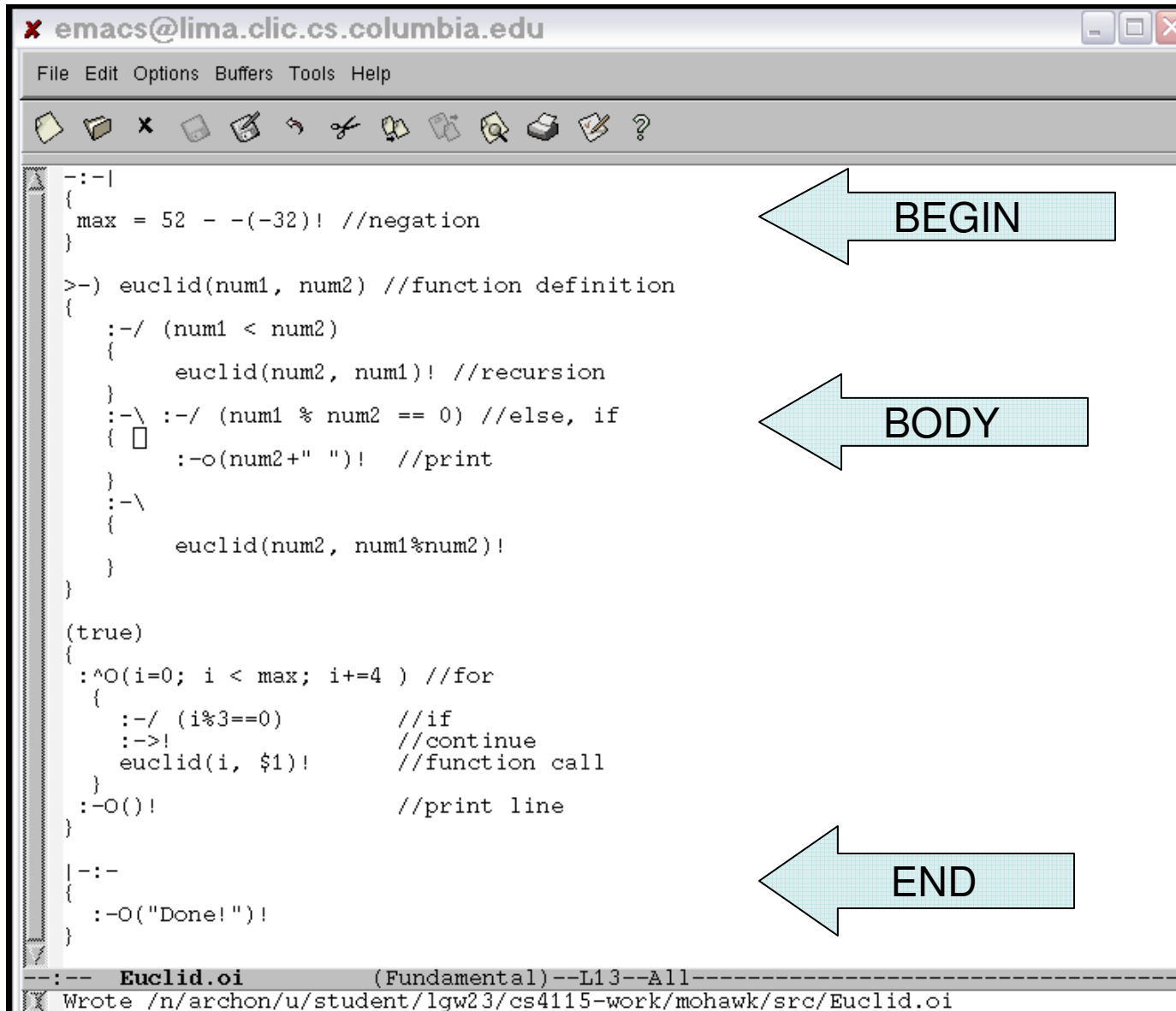
```
xterm
$ mohawk Total.o1 data.txt > data2.txt
```

Output file

```
emacs@firefly.cs.columbia.edu
File Edit Options Buffers Tools Help
**BIWEEKLY WAGE REPORT**
Chris: 9 7 255.00 4080.0
Lauren: 3 13 245.3 3924.8
Joeng: 7 9 245.1 3921.6
max = Chris with 4080.0
The end
```



# Tutorial Introduction / Example



```
emacs@lima.clic.cs.columbia.edu
File Edit Options Buffers Tools Help
[Icons]
-:-|
{
max = 52 - -(-32)! //negation
}
>-) euclid(num1, num2) //function definition
{
:-/ (num1 < num2)
{
euclid(num2, num1)! //recursion
}
:-\ :-/ (num1 % num2 == 0) //else, if
{
:-o(num2+" ")! //print
}
:-\
{
euclid(num2, num1%num2)!
}
}
(true)
{
:^O(i=0; i < max; i+=4 ) //for
{
:-/ (i%3==0) //if
:->! //continue
euclid(i, $1)! //function call
}
:-o()! //print line
}
|:-
{
:-o("Done!")!
}
}
--- Euclid.oi (Fundamental)--L13--All-----
Wrote /n/archon/u/student/lgw23/cs4115-work/mohawk/src/Euclid.oi
```

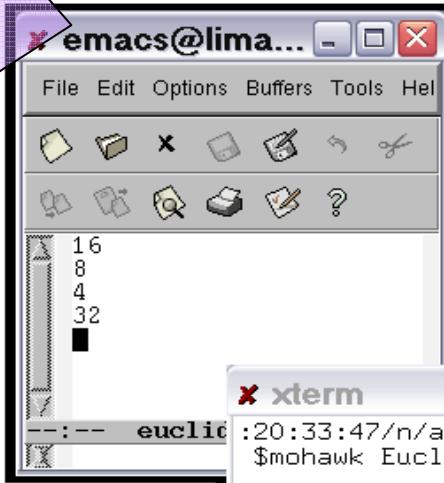
← BEGIN

← BODY

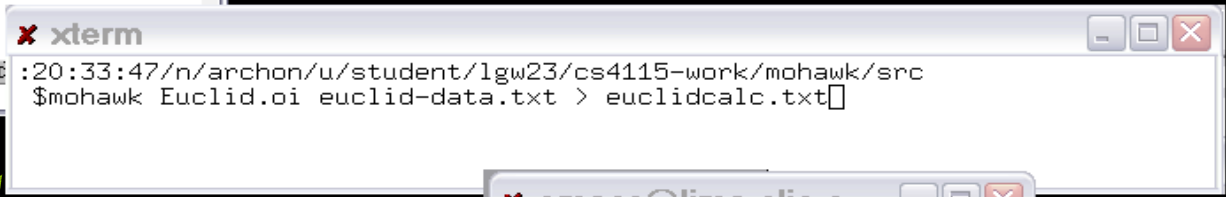
← END

# Tutorial Introduction / Example

Input file



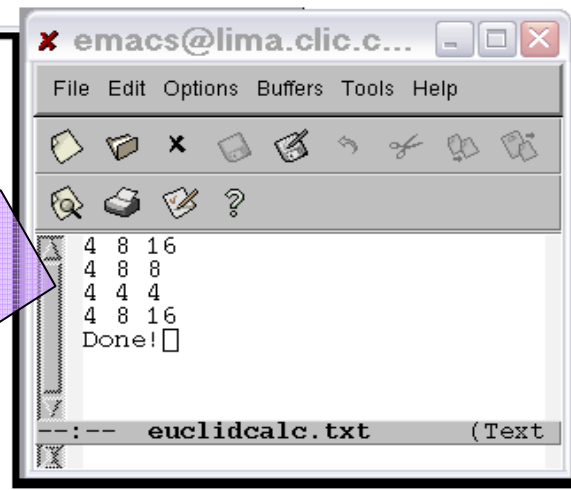
emacs@lima...  
File Edit Options Buffers Tools Hel  
16  
8  
4  
32



xterm  
euclid :20:33:47/n/archon/u/student/lgw23/cs4115-work/mohawk/src  
\$mohawk Euclid.o euclid-data.txt > euclidcalc.txt

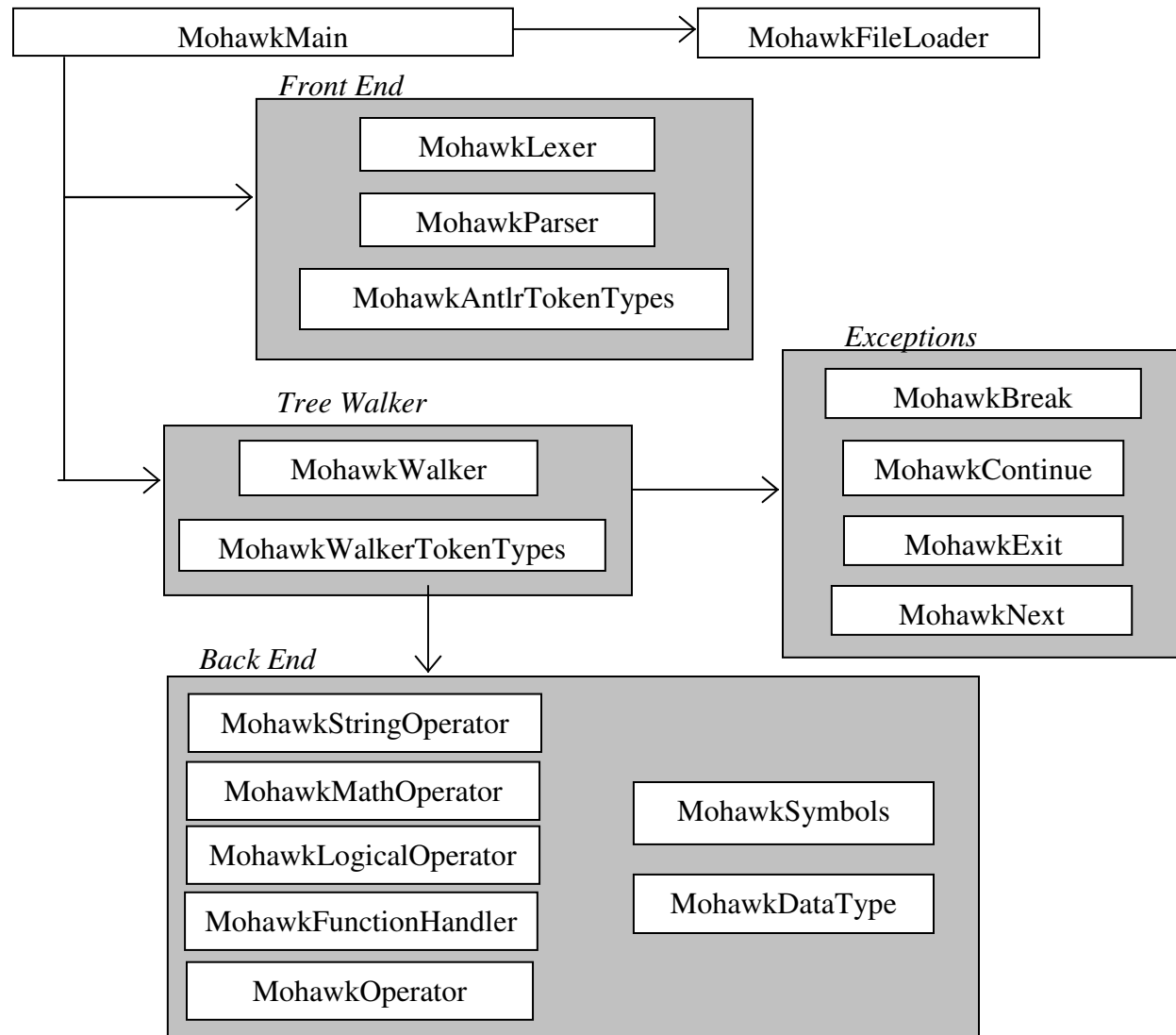
MOHAWK

Output file

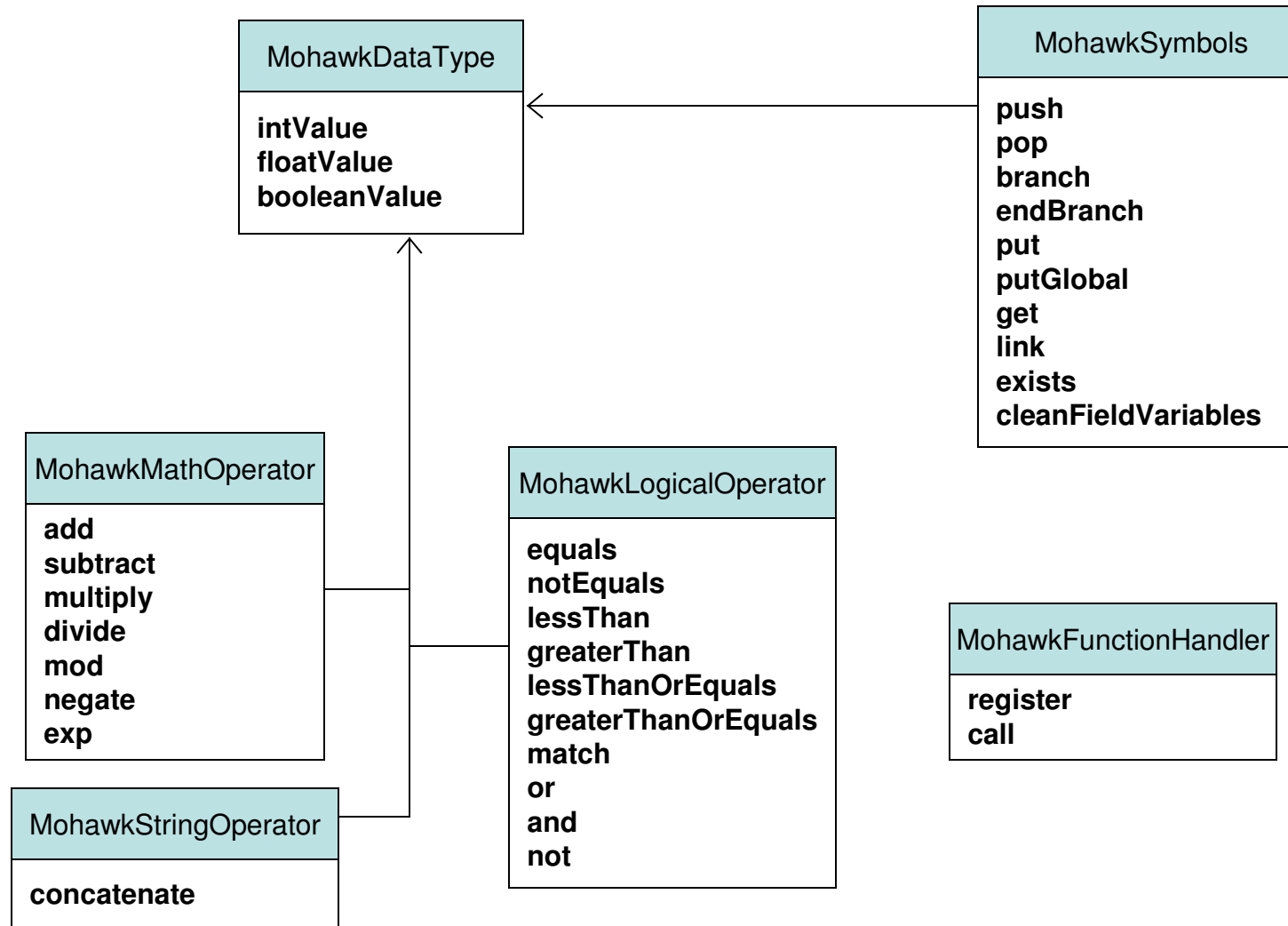


emacs@lima.clic.c...  
File Edit Options Buffers Tools Help  
4 8 16  
4 8 8  
4 4 4  
4 8 16  
Done!  
euclidcalc.txt (Text)

# MOHAWK Component Architecture

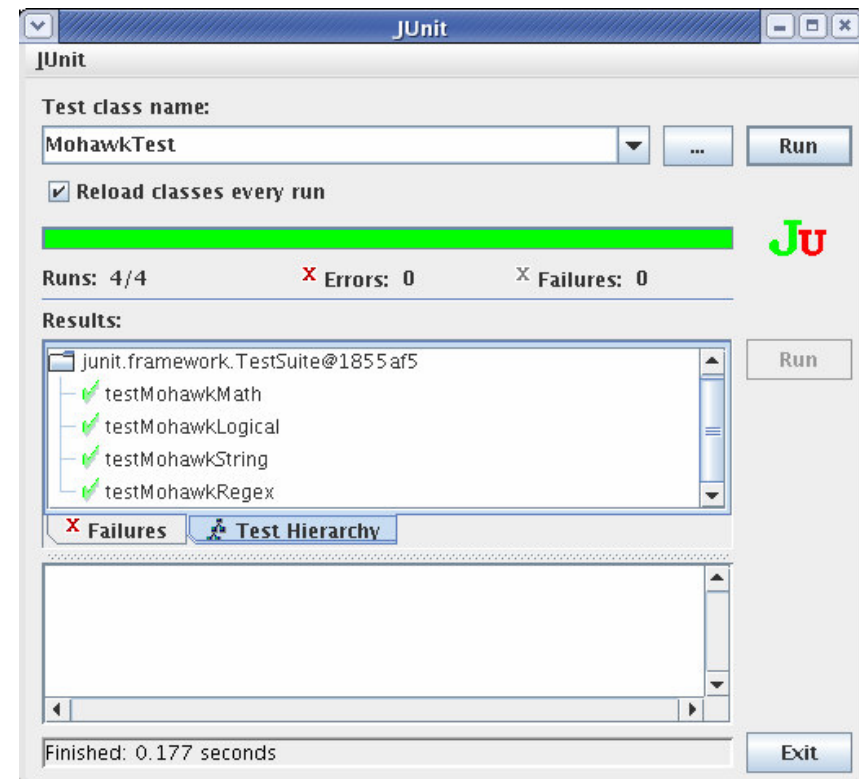


# Back End Components



# Testing

- Unit Testing
  - JUnit
  - Test operators
- Parser Testing
  - Expected Tokens
- Function Testing
  - Expected Output



# Lessons Learned

- Good language design requires balance
- Impossible to stick to original specifications
- Details behind creating a language
  - Operator Precedence
  - Variable Scoping
  - Data type Manipulation
- Importance of Testing Early and Testing Often