

# Luke Cormier - Capstone Document

## Section 1: Program

Please include a zip file of the final repository in this directory.

## Section 2: Teamwork

For section 2, my partner Cameron Watson created tests for the project which would test the functionality of the Catscript program and document the results of the tests as well as exchange each other's guides for section 4.

The tests are as follows: Test 1:

```
@Test
void functionDeclarationPlusIteration(){
assertEquals("2\n3\n4\n5\n", executeProgram("function foo(x) { print(x+1) if(x = 3){
print(x) }\n"+
    "foo(1)\n"+
    "foo(2)\n"+
    "foo(3)\n"+
    "foo(4)\n"}));
}
```

This test would check the results of the function declaration in conjunction with the iteration of the Catscript program. Passing this test meant that the catscript program was able to declare its function while also being able to iterate. Test 2:

```
@Test
void VarStatementWithIf(){
assertEquals("2\n",executeProgram("var x = 1\n"+
    "var z = x + 1\n"+
    "if(z = 2){ print(z) }\n"+
    "else { print(1) }\n"));
}
```

Test 2 would test the functionality of the variable statement with an if statement inside it. Since the test returns a pass then the catscript language would have been able to handle a variable inside of an if statement. Test 3:

```
@Test
void returnStatementWithIteration(){
assertEquals("4\n",executeProgram("function foo(x : int) : int {\n"+
    "var z = x\n"+
    "z = z + 2\n"+
    "return z + 1"+
    "}\n" +
    "print(foo(1))"
));
}
```

Test 3 checks the results after testing the return statement with iteration. As the test passes, the catscript program is shown to be able to execute a return statement after iteration has been run before it.

For my own work on the project, I created three tests that would check the functionality of my partner's Catscript program checking if results match the expected output of each test.

Test 1:

```
@Test
void returnStatementWorksTest1() {
    assertEquals("11\n", executeProgram(
        "function foo(x : int) : int {\n" +
        "    return x + 2\n" +
        "}\n" +
        "print(foo(9))\n"
    ));
}
```

This test checks to ensure that the return statement works properly running it through a function then executing a return statement. Test 2:

```
@Test
void localAndVarStatementWorksProperlyTest() {
    assertEquals("16\n", executeProgram("var x = 16\n" +
        "var y = x\n" +
        "print(y)\n"));
    assertEquals("null\n", executeProgram("var x = null\n" +
        "print(x)\n"));
    assertEquals("5\n18\n93\n", executeProgram("for( x in [5, 18, 93] ) {\n" +
        "    var y = x\n" +
        "    print(y)\n" +
        "}\n"));
}
```

Test 2 checks the functionality of the local and variable statement works checking if the results return the correct variable value after printing the result.

Test 3:

```
@Test
void varInsideFunctionWorksProperlyTest() {
    assertEquals("82\n", executeProgram("function foo() : int {\n" +
        "    var x = 82\n" +
        "    return x\n" +
        "}\n" +
        "print( foo() )\n"));
    assertEquals("18\n21\n35\n", executeProgram("for(x in [18, 21, 35]) { print(x) }"));
}
```

Test 3 checks the functionality of the variable statement while inside a function after running the function to get the expected printed result at the end.

## Section 3: Design pattern

One design pattern utilized in the project was the memoize design pattern. The design pattern looks into a map determining if the map contains any value, otherwise it would create a new list type which would be inserted into the map before it would then return that map type. After this first call, the map will now come up as that new map type.

## Section 4: Technical writing. Include the technical document that accompanied your capstone project.

### Catscript Guide

Below is a guide on the functionalities of catscript.

#### Introduction

Catscript is a simple scripting language that resembles other programming languages like Java. In the guide below, you will learn the basic functionalities of Catscript.

```
var x = "Hello World"
print(x)
```

#### Features

##### Types

Catscript converts from an inputted string to an integer, String or boolean values.

```
var x = "String" //string
var y = 3 //int
var z = true //boolean
```

##### Comments

To type a comment in catscript, use "//" here is an example:

```
//comment
```

##### Operations

Catscript supports basic operations like multiplation, division, subbtraction and addition.

##### Factor Expressions

Catscript supports multiplication and divison, here is an example:

```
var x = 4
var y = 2
print(x/y) //2
print(x*y). //8
```

### Additive Expressions

It is possible to do simple subtraction and addition in catscript, here is an example:

```
var x = 2
var y = 2
print(x+y) //4
print(x-y) //0
```

### Boolean and null Expressions

Catscript supports boolean expressions and null expressions to manipulate logic with:

```
var x = true
var y = false
int x = null
```

### Lists

Catscript supports Lists, here is an example of a list in Catscript:

```
var x = [1, 2, 3]
print(x) // [1,2,3]
```

### Comparison Expressions

Catscript supports comparison expressions. The operators supported are greater than, greater than or equal to, less than, less than or equal to. Below is an example on how to compare values. Keep in mind, you have to initialize values first.

```
var x = 3
var y = 4
print(x > y) //returns false
print(x < y) //returns true
print(x >= y) //returns false
print(x <= y) //returns true
```

There is another comparison called == and !=, here is an example:

```
var x = 3
var y = 3
print(x==y) //returns true
print(x!=y) //returns false
```

### Equality Expressions

Catscript supports equality expressions. It is possible in catscript to compare two values. The operators supported are not equal to and equal to. Below is an example on how equality expressions work in Catscript. `x != y` `x == y`

## For Loop

Catscript supports for loops. Here is an example of a for loop: The for loop will iterate through the size of the given list.

```
for ( x in [1,2,3])
{
  print(x)
}
```

You can perform arithmetic inside for loops. Here is an example:

```
for ( z in [1,2,3])
{
  var y = z + 1
  print(y)
}
```

## If Statements

If statements are an important part of logic and catscript supports them. You can also create expressions inside of them. Catscript also supports else statements.

```
var x = 4
if ( x > 4)
{
  print("greater than 3")
}
else{
  print("less than 3")
}
```

## Unary Expressions

Catscript supports the implementation of unary expressions like "not" and "-"

```
not true // false
-1 //1
```

## Return Statements

Catscript supports return statements. This allows you to return a value from a function in Catscript.

```
function foo()
{
  var x = 1
  return x
}
```

It is also possible to perform arithmetic in the return statement. Here is an example:

```
function foo()
{
```

```
var x = 1
return x + 1
}
```

## Functions

Two important factors of functions are function declarations and function calls. Catscript supports both of these type of functionalities. Inside of the function you can add expressions. You can manipulate any type of data insdie the function body. Below is an example of how a function is implemented in catscript:

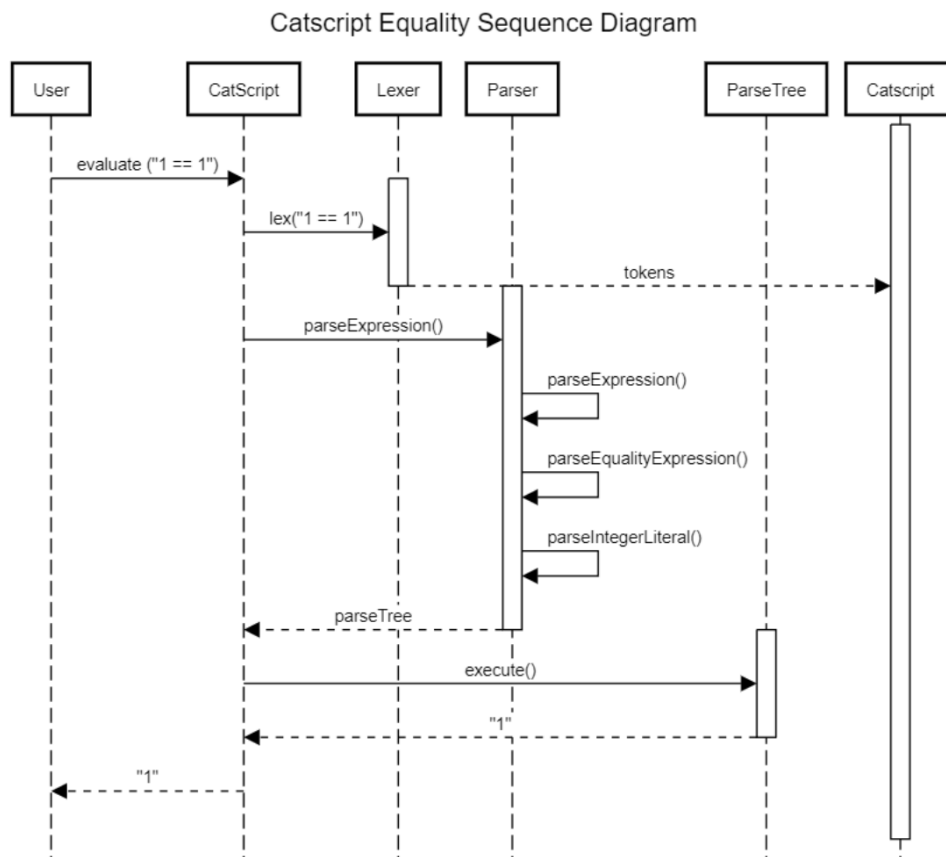
```
function example(a : object, b : int, c : bool, d : String)
{
    print(d)
}
```

Catscript is a versatile scripting language. It is simple to return a value passed to a function. Below is an example:

```
function example(a: int)
{
    return a
}
```

That covers the expressions, statements, and values available in Catscript.

## Section 5: UML.



This UML diagram depicts the equality sequence in the Catscript program and how it is processed in the language.

## Section 6: Design trade-offs

Trade-off for the design would be doing a by-hand parser instead of utilizing an auto generated parser for this project. This created a more intuitive recursive style algorithm guiding us into more how grammars worked rather than using lex to auto generate the parser for us.

## Section 7: Software development life cycle model

Test driven development life cycle model was utilized during this project in which several tests were presented to us for us to solve as we coded the project. Each test would notify us of the goal we sought to achieve in the code and the language being used by the test through the expressions. Once the tests were passing we'd move onto the next section of the project and complete those tests as well.