**Language Design Proposal: ODANG**

**Student Names**: Charles Dang, Giovanni Orozco, Marius Larnoey

**Language Name**: ODANG

**Compiler Implementation Language and Reasoning**: Javascript. We’re going to learn it in our Web Engineering classes and hopefully use it to make Java more efficient. We also believe using Javascript to compile our language into Java would not be as complex as other languages.

**Target Language**: Java

**Language Description**: Object-oriented programming. We’re going to create a Java-like language that is more optimized and useful.

**Planned Restrictions**: No garbage collection.

**Abstract Syntax**:

var is a variable

i is an integer

str is a string

classname is the name of the class

methodname is the name of the method

interfacename is the name of an interface

type ::= Int | String| Char | Boolean | Float | Double | Short | Byte

op ::= \* | / | + | - | % | > | < | && | ! | ||

expr ::= var | str | i |

this | // reference to current object

println(expr\*) | //prints text on console

expr op expr | //arithmetic operations

expr.methodname(expr\*) | //call a method with any number of expressions as parameters

new classname(expr\*) | /\*create a new class with any number of expressions as parameters\*/

type(expr) // cast the expression to a type

vardec ::= type var // declare a variable

stmt := vardec; | //simple variable declaration

var = expr; | assign an expression to a variable

while (expr) stmt | // while loop

for (expr) stmt | // for loop

break; | // break

{ stmt\* } | // block of statements

if (expr) stmt [else stmt] | // if statement

return expr; | // return expression

return; | // simple return

methoddef ::= access type methodname(vardec\*) stmt /\* a method definition that contains any possible number of variable declarations separated by commas as parameters\*/

interfacedef ::= interface interfacename {

vardec\*

methoddef\*

}

classdef ::= class classname [extend classname | implement interfacename] { /\* the “implement interfacename” part will be where typeclass will be implemented \*/

vardec\*

classname (vardec\*) stmt /\* a constructor that has any number of variables that are separated by commas\*/

[classname operator op (classname &var) stmt] // implement operator overloading

methoddef\*

}

program ::= classdef\* expr

**Computation Abstraction Non-Trivial Feature**: Type classes

**Non-Trivial Feature #2**: General Optimizations

**Non-Trivial Feature #3**: Operator Overloading

**Work Planned for Custom Component**: General Optimizations will be done throughout the semester.