

Language Design Proposal: pOOP

Student Name(s): Kyle Dewey

Language Name: pOOP

Compiler Implementation Language and Reasoning: Java. I'm already familiar with it, and I'm not planning to get into optimizations. Learning a new language is an unnecessary risk.

Target Language: C

Language Description: (Pathetic) object-oriented programming. The goal is for me to better understand how object-oriented programming languages work. I want to implement a Java-like language with classes and subclasses. I'm intentionally picking C because it is pretty low-level, but it's not so low-level that it will require me to spend a lot of time understanding the target language.

Planned Restrictions: there is no way to reclaim allocated memory (either automatically or manually), and no optimizations.

Abstract Syntax:

var is a variable

classname is the name of a class

methodname is the name of a method

str is a string

i is an integer

type ::= Int | Boolean | Void | **Built-in types**

classname **class type; includes Object and String**

op ::= + | - | * | / **Arithmetic operations**

exp ::= var | str | i | **Variables, strings, and integers are expressions**

this | **Refers to my instance**

println(exp) | **Prints something to the terminal**

exp op exp | **Arithmetic operations**

exp.methodname(exp*) | **Calls a method**

new classname(exp*) | **Creates a new instance of a class**

(type)exp **Casts an expression as a type**

vardec ::= type var **Variable declaration**

stmt ::= vardec; | **Variable declaration**

var = exp; | **Assignment**

while (exp) stmt | **while loops**

break; | **break**

{ stmt* } | **block**

```

    if (exp) stmt else stmt | if/else
    return exp; | return an expression
    return; return Void
access ::= public | private | protected
methoddef ::= access type methodname(vardec*) stmt
           vardec's are comma-separated
instancedec ::= access vardec; instance variable declaration
classdef ::= class classname extends classname {
           instancedec*
           constructor(vardec*) stmt   vardec's are comma-sep
           methoddef*
           }
program ::= classdef* exp   exp is entry point

```

Computation Abstraction Non-Trivial Feature: Objects + methods with class-based inheritance.

Non-Trivial Feature #2: Subtyping

Non-Trivial Feature #3: Static access modifier checking

Work Planned for Custom Milestone: Access modifier checking. Until that point, everything is implicitly considered public.