**Language Design Proposal**

**Student Name(s):** Miguel Cruz, Jacob Poersch, Nicholas Araklisianos, Carlos F. Sandoval  
  
**Language Name:** CoopJa  
  
**Compiler Implementation Language and Reasoning:** Java; Most group members are familiar with the language.  
  
**Target Language:** C  
  
**Language Description:** “C’s Cooperative Object Oriented Programming from Java” -- We plan to use Java’s Object Oriented nature and bring this to the C language. With this, we will be including class-based inheritance in our language. We were thinking of a few different target languages, but ultimately settled with C since we felt its differences with Java were significant enough that we could make some meaningful additions.  
  
**Planned Restrictions:** We will not be featuring any memory deallocation in our language, nor any garbage collection. We also will not be featuring Java’s Generics in our language.  
  
**Syntax (Subject to Change):**

*var* is a variable

*objectname* is the name of a class

*methodname* is the name of a method

*str* is a string

*i* is an integer

type ::= int | char | boolean | string | auto | [Built in types of variables]

objectname [Objects are also types]

op ::= + | - | \* | / | [Arithmetic operations]

> | < | >= | <= | == | != | ==| | [Comparison Operations]

| | & | ^ | >> | << | ~ [Bitwise Operators]

vardec ::= type var [Variable declarations]

exp ::= var | str | i | [Basic expressions]

exp op exp| [Arithmetic expression]

this [Refers to this instance]

objectname.Method(Var\*) [Call Method]

new objectName(exp\*) [Declare a new instance of an object]

access ::= Public | Private | Protected [access type for a method or var]  
stmt ::= vardec; | [Variable Declarations]

var = exp; | [assignment to variable]

If (exp) Block\_stmt else Block\_stmt | [standard if/else statement]

while (exp) Block\_stmt | [loop statement with restriction]

for (vardec; exp; exp;) Block\_stmt | [for loop statement]

break; | [escape loop statement]

return exp;| [return an expression]

return; | [Empty return]

println(str)| [Prints to the terminal, string only]

printf(str, exp\*) [C-Style printf statement]

Block\_stmt ::= {stmt\*} [block statement]

instancedec ::= [Access] vardec;

result\_type ::= type | void [Return types]

methodef::= [Access] result\_type methodname (vardec\*) Block\_stmt [Method declarations]

objectdefheader ::= access class objectname | access class objectname extends objectname

objectdef::= objectdefheader {

(vardec|methoddef)\* [declarations]

}

entrypoint ::= [access] result\_type main (vardec\*)Block\_stmt

objectdefmain :: = objectdefheader {

(vardec|methoddef)\* [declarations]

Entrypoint [main entry

(vardec|methoddef)\* [declarations]

}

program ::= objectdefmain\* | objectdef\* [Does not require entrypoint to compile]

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

**Non-Trivial Feature #2:** Access Modifiers (public and private types) referring to both Classes and variables.

**Non-Trivial Feature #3:** Type Inference, allowing for an “auto” type. The compiler will determine what the “auto” type actually is.

**Work Planned for Custom Milestone:** Access Modifiers. Until it is implemented, everything is treated as public.