Project 3. CLOS Interpreter.
COSC 252
Jeremy Bolton

You will implement CLOS (Common Lisp Object System) interpreter. You will simply add this Object System to your mini-scheme, thus you do not need to create this interpreter from scratch.

You will need to update the design of the language, the token set, the grammar, and the semantics accordingly.

When in doubt of desired behavior feel free to default to our course server: cs-class.uis.georgetown.edu; the command for the interpreter is clisp. Type *man clisp* to learn more. If you do not have a course server acct, **let me know ASAP!**

Instructions:

I. Language Additions. Add to the simple calculator the following:
   a. "Built-in" functions
      - defclass, :initarg, :initform
      - make-instance
      - defparameter
      - setf
      - slot-value
      - defgeneric
      - defmethod
   b. Capabilities
      - inheritance
      - dynamic dispatch (polymorphism)
      - dynamic classes
      - multiple inheritance
      - generics (explicit polymorphism)
   c. Example Usage:
      - (defclass bank-account ()
        ```
        ((customer-name
        :initarg :customer-name)
        (balance
        :initarg :balance
        :initform 0))) ([x 'a])
        (let ([f (lambda (y) (list x y))])
            (f 'b)))
        ```
      - (slot-value *account* 'customer-name)
      - (defmethod initialize-instance :after ((account bank-account) &key)
        ```
        (let ((balance (slot-value account 'balance)))
        ```

```
(setf (slot-value account 'account-type)
        (cond
                ((>= balance 100000) :gold)
                ((>= balance 50000) :silver)
               (t :bronze)))))
```

II. Design and build CLOS

a. Design tokens and build tokenizer
    - Updates to tokenizer are likely not needed

b. Design grammar and build parser.
    - Updates to Grammar are likely not needed.
    - NOTE:
        - Debugging help:
            a. Test your parser before adding semantics!!!
            b. Hint: Have your parser compose a string that illustrates the path of the function call chain during the parse, (this mimics the parse tree). Use the parse tree rendering using tool (http://mshang.ca/syntree/):

c. Implement Semantic Analysis. Incorporate semantic evaluation directly into the parser.
    - See discussion in Section I.
    - Include appropriate error messages and recovery.
    - When in doubt of desired behavior feel free to default to [Se], THE course server, and/or ask me.

III. RUBRIC

Requirements		**Points Allocated**
clos	Token / Parsing Updates	20%
	OOP def, acc, setf, mk-inst	70%
	Multi-Inheritance*	10%
	Dynamic Classes*	10%
	Generics*	10%
	Error Messages	10%
TOTAL		**100% + 30% extra credit**