Dispense-o

Language Report

CSCI 310

Juan Tenezaca

May 9, 2016

**Introduction**

Dispense-o is a programming language that is designed for coffee bean dispensers. Dispense-o is designed to make sure that you get the appropriate flavor of coffee that you need in order to get through the day.

* 1. **Why call it Dispense-o?**

This programming language can be used with coffee bean dispensers and have coffee beans crushed at any time you want so you can blend your perfect cup of coffee.

* 1. **Why Dispense-o?**

A sane human cannot function without his cup of coffee. The American workforce with crumble under its own weight without the help of coffee. Dispense-o recognizes this and makes sure that we are fully caffeinated. In this world we have choices and when it comes to coffee we have hundreds of them. This programming language helps enable the user the choice of coffee bean. He also has the ability to setup the exact time to execute coffee bean dispensing.

* 1. **Language Design**
* Similar to C++ but we make assignment of variables much easier.
* Semicolons are needed in order to show the end of statement
* Case sensitive
* Left Associative
* **Input:**
	+ Clock running on 24 hours. Which can be used to setup whatever conditions that the user wants. Can be programmed to fit the users schedule
	+ Multiple containers to hold an assortment of coffee beans. All containers will have some sort of ID number which the program will identify.
	+ Scale inside these containers in order to notify the user when the containers are empty.
	+ These containers will all be supported by a contraption and all containers will have blades in order to crush the beans.

**Language Tutorial: Lexical Breakdown**

**2.1 Common Functions**

These are some of the common functions that this program will be using in order for the user to get the appropriate beans at the appropriate grind.

* **dispBeans(); -** Will dispense the bean for the user. Depending on time
* **setBeanChoice(); -** The user can choose what bean he wants
* **checkEmpty(); -** Checks if the container is empty
* **checkFull(); -** Checks if the container is full
* **setGrindTime(); -** Sets times for beans to be grinded
* **setAmount(); -** Sets amount of beans to be dispensed
* **setBean(); -** Sets the type of bean that is going to dispensed
* **Main(); -** Where all the magic happens.

**2.2 Reserved Words/Variable types/Operators**

These are key words reserved by the language and will cause errors when trying to name as variables

**Reserved Words:**

* Empty
* True
* False
* BeanChoice
* Container
* Amount
* Time

**Variable Types:**

* int
* double
* bool
* time
* container
* bean
* name

**Operators:**

* & and
* || or
* ++ increment by 1
* -- decrement by 1
* == is equals
* = equals
* : Is used to describe the time variable
* >= greater than or equal too
* <= less than or equal to
* ~~ comment

**Language Tutorial: Syntax**

**3.1** This section will show the BNF of our language

<main> ::= <variable assignment><functions>|<conditionals><comments>

<variables assignment>::= <variable types><variable name>

<variable type> ::= int | double | bool | time | container | bean

<variable name> ::= <A-Z> | <a-z> <0-9>

<name>::= <BeanChoice><Container><Amount>

<BeanChoice>::= <bean> type of bean that you want

<Container> ::= <container> which container the bean will be coming out

<Amount>::= This will be an amount in cup servings. Depending on the mug

**3.2** This section will show built in functions BNF

<functions>::= <built in functions> | <user created functions>

<built in functions>::= <dispBeans>| <setBeanChoice> |<checkEmpty> |<checkFull> | <setGrindTime>|<setAmount> | <setBean>

<dispBeans> ::= <BeanChoice>

<setBeanChoice>::= type of bean

<checkEmpty> ::= returns a bool ( true or false)

<checkFull> ::= returns a bool (true or false)

<setGrindTime> ::= time

<setAmount> ::= name.amount | Amount would be in cups

<setBean> ::= name.bean | type of bean that user would want

**3.3** This function will show the BNF of user created functions

<user created functions> ::= Function names, parameters, function contents, return

<conditionals> ::= <if> | <while> | <for> | <switch>

<if> ::= condition, do this thing | <else>

<else> ::= do that thing

<while> ::= condition, thing to do

<for> ::= amount to do for, stuff to do

<switch> ::= case 1, case 2, case 3…

<comments> ::= ~ comment content

<A-Z> ::= Uppercase <a-z>

<a-z> ::= Lowercase <a-z>

<0-9> ::= Numbers <0-9>

**Sample Programs**

**4.1** Sample #1

Main(){

 time morning = 06:00;

 time night = 18:00;

 bean dark = 1;

 container darkroast = 1;

name Juan = [bean: dark, container: darkroast, Amount: 1.00]

setGrindTime(morning);

 setBeanChoice(dark.bean);

 setAmount(juan.Amount);

 dispBean(juan.bean);

 }