1. Introduction

Espresso is a hybrid object-oriented language and functional language. The key goal of the project is to practice design of a simple language and use Ocaml to implement it. The basic operators, conditional statement, looping and data types will be supported, also with Object-oriented behaviour and some new lambda related features. Related former projects include Dice(2015), Liva(2016) and Scala--(2016). Espresso is compiled to LLVM.

2. Project Description - Syntax, semantics of the subset of Java that we support

Espresso is a general usage language, users can use it to practice object-oriented programming and will also have some basic functional aspects. The code of Espresso possesses some similarities in terms of structure with Java, and it will be compiled to LLVM. To simplify, Espresso will not support Access modifiers such as public, private, protected and default.

3. Language Features

   a. Comments: // and /* */
   b. Operators
      i. Arithmetic - +,-,*,/,%
      ii. Relational - =, ==, !=
      iii. Boolean && || ^
   c. Basic Data types
      i. Boolean - Standard boolean values
      ii. Int - Numeric Values (Integers)
      iii. String - Sequence of characters
   d. Conditional Statements
      i. if-then-else
   e. Looping Constructs
      i. for loop
      ii. for each loop
      iii. while loop
   f. Data Structures
      i. Array
      ii. List
      iii. HashMaps
   g. Built-in Functions - for data structures like Strings, Arrays, HashMaps. Ex: String.length().
h. OOPs Features
   i. Classes
i. Functional Features
   i. Lambdas - basic
      1. map, filter, foreach,

4. Examples
   a. Comments: // and /* */

// a comment example
/* another comment example */

b. Operators
   i. Arithmetic - +,-,*,/,%
   ii. Relational - =, ==, !=
   iii. Boolean && || ^

c. Basic Data types
   i. boolean
   ii. int
   iii. String

d. Conditional Statements
   i. If-then-else

```java
int a = 10, b = 5;
boolean hasGoogleOffer = true;
String congrad = “Congratulations!”;
if( a / b == a % b){
    print(“Oops, cute!”);
}
else if(a%b==0 && hasGoogleOffer){
    print(congrad);
}
else{
    print(“Mike, you should try harder”);
}
```

e. Looping Constructs
   i. for loop
   ii. for each loop
   iii. while loop

```java
int [] nums = new int[10];
```
for(int i =0; i < nums.length;i++){
    nums[i] =i;
}
List list = Array.toList(nums);
foreach (int item in list){
    Item *=item;
}
while(!list.empty()){  
    list.removeHead();
}

f. Data Structures  
   i. Array  
   ii. List  
   iii. HashMaps 

g. Built-in Functions - for data structures like Strings, Arrays, HashMaps. Ex:
    String.length().

// init and constructors
boolean [] flags = {true, false};
int [] nums = new int[k];
String [] fluits = {“Apple”,”Banana”,”Orange”};
String [] fluitscopy = Array(fluits);
List list = new List(int);
HashMap map = new HashMap(String,int);
// built in functions
fluits.length; 
fluits[0].length();
list.length();
list.empty();
list.add(10);
list.removeHead();
map.contains(“Apple”);
map.put(“Apple”,5);
map.remove(“Orange”);

h. OOPs Features  
   i. Classes

Class People {
    String name;
    int age;
}
Class Women extends People{
}

i. Functional Features
   i. Lambdas - basic
      1. map, filter, foreach,

int square(int x ){
    return x*x;
}

int [] quares = Array.map(square, nums);
int sum = List.foldleft(((a,b) ->(a+b)),0,list);
fluits.foreach(print);
int [] tenmore = Array.filter( item -> item >10);