COMS W4115 Programming Languages and Translators

Liva Project Proposal, summer 2016

Shanqi Lu	Jiafei Song	Zihan Jiao	Yanan Zhang
sl4017	js4984	zj2203	yz3054

1 Introduction

Liva is a general-purpose programming language and a lite version of Java. Having realized that the focus of this project should be applying compiler design theories in practice rather than innovation, we decided to develop a language that has the similar syntax and abstract data types in Java. Past projects related to general-purpose languages all have their own interesting features like Cpi (2013) implemented a data type called "struct" and Dice (2015) developed their own version of inheritance mechanism. Our preference is to develop the object-oriented model that resembles Java and implement data structures like Map and List. This language is compiled down to LLVM.

2 Description

Unlike domain-specific programming languages are designed for specific fields, our language is designed for general purpose, hence serving as a portable language that runs on many platforms as long as LLVM is runnable. Programs written in Liva will look like Java in many ways including variable declaration and class declaration. Common algorithms like GCD can be easily implemented using our language.

We will complete the HashMap class to make Liva support more algorithms which need map data structure. HashMap maintains key-value pairs and their mapping relationship. It is efficient for locating a value based on a key which is an operation appears very often in common algorithms.

3 Key Features

3.1 Arithmetic Calculations

Our language should support basic calculations by using arithmetic operators like "+", "-", "*", "/", "%", etc.

3.2 Control Flows and Basic Operators

Control Flow tokens and basic operators should include: if, else, for, ==, !=, etc.

3.3 Object-oriented programming

Our language should allow users to create their own classes (along with methods and attributes).

3.4 Strings

The String class represents character strings. All string literals in our programs, such as "abc", are implemented as instances of this class. The String class includes the following methods:

charAt(int index)	Returns the char value at the specified index.
length()	Returns the length of this string.
split(String regex)	Splits this string around matches of the given regular expression.
toCharArray()	Converts this string to a new character array.

3.5 Data structures

Our language supports several implementations of common data structures including array and HashMap as well as their key methods. The HashMap class should contain the following methods. In addition, we allow users to use instances of user-defined classes to build a HashMap.

clear()	Removes all of the mappings from this map.
containsKey(Object key)	Returns true if this map contains a mapping for the specified key.
put(K key, V value)	Associates the specified value with the specified key in this map.
get(Object key)	Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
keySet()	Returns a Set view of the keys contained in this map.

4. Example Code

4.1 Arithmetic Calculations Example

Example of adding integers:

```
public class cal {
        public void main(char[][] args) {
                print(15+15);
        }
}
```

4.2 Strings Example

Example of string class:

```
import Liva.stdlib;
public class testS{
        private class String x;
        public void main(char[][] args) {
        class String msg = new String("Thanks a lot!");
        this.x = msg;
```

```
        print(this.x.string());
        }
}
```

4.3 Boolean Example

Example of Boolean expression:

```
public class testB {
        public void main(char[][] args) {
                 print(100<200);
                 print(45>2);
                print(2==3);
                 print (2 !=100);
        }
}
```

4.4 Array Sort Example

Example of using Array in Utility, control flows and function:

```
import Liva.util;
Import Liva.Exception;
public class MainClass {
  public void main(String args[]) throws Exception {
        int array[] = { 2, 5, -2, 6, -3, 8, 0, -7, -9, 4 };
        Arrays.sort(array);
        printArray("Sorted array", array);

  }
  private void printArray(String message, int array[]) {

    for (int i = 0; i < array.length; i++) {
        if(i != 0){
        print(", ");
        }
        print(array[i]);
    }
        print();
  }
}
```

4.5 Object-Oriented Programming Example

The Liva language supports the Object-Oriented Programming (OOP).

4.5.1 Classes

Everything in Liva is defined in a class which defines a collection of data.

Example:

```
class Student {
        String name;
        String ssn;
        String emailAddress;
        int yearOfBirth;
```

```
}
```

4.5.2 Objects

Example of creating and using instances of the **Student** class:

```
import Liva.util;

public class StudentDemo {
        public void main(String[] args) {
        Student e1 = new Student ();
        e1.name = "yanan";
        e1.ssn = "888-12-345";
        e1.emailAddress = " shanqi@company.com";
        Student e2 = new Student();
        e2.name = "shanqi";
        e2.ssn = "456-78-901";
        e2.yearOfBirth = 1992;
        print("Name: " + e1.name);
        print("SSN: " + e1.ssn);
        print("Email Address: " + e1.emailAddress);
        print("Year Of Birth: " + e1.yearOfBirth);
        print("Name: " + e2.name);
        print("SSN: " + e2.ssn);
        print("Email Address: " + e2.emailAddress);
        print("Year Of Birth: " + e2.yearOfBirth);
        }
}
```

4.5 HashMap Example

Example of using HashMap in Liva:

```
import Liva.util
public class Program {
  public void main(String[] args) {

                HashMap<String, String> hash = new HashMap<>();
                hash.put("zihan", "EE");
                hash.put("jiafei", "CS");
                hash.put("shanqi", "EE");
                hash.put("yanan", "CS");
                String  major1= hash.get("zihan");
                String  major2= hash.get("jiafei");
                print(major1);
                print(major2);
                Set<String> keys = hash.keySet();
                for (String key : keys) {
                        print(key);
        }
}
```