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

*Liva*

*A Lite Version of Java*

Table of Contents

[CHAPTER 1 INTRODUCTION 2](#_Toc456774624)

[CHAPTER 2 LEXICAL CONVENTIONS 2](#_Toc456774625)

[2.1 White Space 2](#_Toc456774626)

[2.2 Comments 2](#_Toc456774627)

[2.3 Identifiers 3](#_Toc456774628)

[2.4 Keyword 3](#_Toc456774629)

[2.5 Literals 3](#_Toc456774630)

[2.5.1 Boolean Literals 3](#_Toc456774631)

[2.5.2 Integer Literals 3](#_Toc456774632)

[2.5.3 Floating Point Literals 3](#_Toc456774633)

[2.5.4 Character Literals 4](#_Toc456774634)

[2.5.5 String Literals 4](#_Toc456774635)

[2.5.6 Escape Sequences for Character and String Literals 4](#_Toc456774636)

[2.5.7 The Null Literal 4](#_Toc456774637)

[2.6 Separators 4](#_Toc456774638)

[2.7 Operators 4](#_Toc456774639)

[CHAPTER 3 Types, Values, and Variables 5](#_Toc456774640)

[3.1 Primitive Types 5](#_Toc456774641)

[3.1.1 Integer Types 5](#_Toc456774642)

[3.1.2 Floating-Point Types 5](#_Toc456774643)

[3.1.3 The Boolean Type 6](#_Toc456774644)

[3.1.4 Class Types and The Class Object 6](#_Toc456774645)

[3.1.5 The Class String 6](#_Toc456774646)

# CHAPTER 1 INTRODUCTION

Liva is a general purpose programming language and a lite version of Java. It is designed to let programmers who are familiar with class-based languages feel comfortable with developing common algorithms like GCD. It is lite in the sense that it maintains some but not all features in Java. It has the similar syntax and abstract data types in Java and supports object-oriented paradigm and inheritance. However, generics and nested classes are beyond the scope of this project, hence they are not to be implemented.

The Liva programming language is strongly typed. The compiler checks whether arguments passed to a function match expected types and return an error if not. It is a portable language and compiled down to LLVM.

This language reference manual is organized as follows:

Chapter 2….

Chapter 3…

Chapter 4

Chapter 5

Chapter 6

# CHAPTER 2 LEXICAL CONVENTIONS

This chapter specifies the lexical conventions of Liva programming language. A compiler takes a program which consists of a sequence of characters and reduce it to a sequence of elements, which are tokens, white space and comments. The tokens are identifiers, keywords, literals, separators, and operators.

*Element: White Space| Comment| Token*

*Token: Identifier| Keyword| Literal| Separator| Operator*

## 2.1 White Space

White space in Liva is defined as space character, tab character, form feed character(page-breaking) and line terminator character. White space characters are ignored by a compiler except as they serve to separate tokens.

## 2.2 Comments

There is one kind of comments:

* /\* text \*/

All characters from “/\*” to “\*/” are ignored.

## 2.3 Identifiers

An identifier is a sequence of letters, digits and underscore ‘\_’. It can only begin with a letter. Identifiers are the names of variables, methods and classes. They are case-sensitive.

## 2.4 Keyword

Keywords are reserved and cannot be used as identifiers.

* *Keyword:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| for | *new* | *if* | *boolean* | *this* | *break* |
| *double* | *implements* | *else* | *import* | *return* | *extends* |
| *int* | *char* | *interface* | *void* | *class* | *float* |
| *while* |  |  |  |  |  |

## 2.5 Literals

Literals are syntactic representations of numeric, character, boolean or string data. They are used for representing values in programs. Literals are

### 2.5.1 Boolean Literals

There are two boolean literals:

* **true** represents a true Boolean value
* **false** represents a false Boolean value

### 2.5.2 Integer Literals

Integer numbers in Liva are in decimal format. Negative decimal numbers such as -10 are actually expressions consisting of the operator ‘-’ and integer literal. The primitive type of integer literal is **int**.

### 2.5.3 Floating Point Literals

Floating point numbers are expressed as decimal fractions and consist of:

1. an optional ‘+’ or ‘-’ sign; if omitted, the value is positive,
2. one of the following formats

|  |  |  |  |
| --- | --- | --- | --- |
| Format | | | Example |
| integer digits |  |  | 9 |
| integer digits | **.** |  | 7. |
| integer digits | **.** | integer digits | 17.31 |
|  | **.** | integer digits | .56 |

### 2.5.4 Character Literals

Character literals are expressed as a single quote: **'a'**, **'#'**, **'π'**

### 2.5.5 String Literals

String literals begin with a double quote character **"**, followed by zero or more characters and a terminating double quote **"**

Within string literals, there can be escape sequences but not unescaped newline.

|  |  |
| --- | --- |
| Name | Character |
| TAB | \t |
| newline | \n |
| double quote | \” |
| single quote | \’ |
| backslash | \\ |

### 2.5.6 Escape Sequences for Character and String Literals

An escape sequence is used to represent a special character. It begins with a backslash character (\), which indicates that the following characters should be treated specially. Escape sequences are listed in the above table.

### 2.5.7 The Null Literal

The null literal is formed as:

**null**

## 2.6 Separators

Separators are tokens used for separating. Follows are these separators:

{ } ( ) ; , .

## 2.7 Operators

The expression section of this manual will explain behaviors of operators. Here is a list of operators.

= > < ! == >=  
<= != & | + -  
\* \ %

# CHAPTER 3 Types, Values, and Variables

Primitive types and reference types

The Liva programming languages supports two kinds of types: primitive types and non-primitive types. Accordingly, there are two kinds of data values: primitive values and reference values.

There is also a special null type.

Primitive types are Boolean types and numeric types. Non-primitive types are class types, string types, array types and hashmap types.

The Liva programming language is statically-typed, which means that all variables must first be declared before they can be used, and the examples in this section assume that the relevant identifiers were previously declared.

## 3.1 Primitive Types

### 3.1.1 Integer Types

The integer data type is a 32-bit sequence of digits, which has a minimum value of -2^31 and a

maximum value of 2^31-1. An integer literal is a sequence of digits preceded by an optional negative sign. A single zero cannot be preceded by a negative sign.

x = 10;

y = -50;

z = 0;

### 3.1.2 Floating-Point Types

The double data type is a signed double-precision 64-bit IEEE 754 floating point. A double literal consists of an optionally signed integer part, a decimal point and a fractional part. Neither the integer part nor the fractional part can be missing.

x = 1.5;

y = -5.1;

z = 2.0;

### 3.1.3 The Boolean Type

The boolean data type is a 64-bit type with only two possible values: true and false. A boolean is its own type and cannot be compared to a non-boolean variable. Therefore, expression “true == 1” would lead to an error.

x = true;

y = false;

## 3.2 Reference Types and Values

### 3.2.1 Class Types and The Class Object

### 3.2.2 The Class String

# CHAPTER 4 Statements

# CHAPTER 5 Expressions