

The Coral Language Specification

Markéta Nikola Lisová

April 21, 2014

Contents

1	Lexical Syntax	3
1.1	Identifiers	4
1.2	Keywords	4
1.3	Newline Characters	4
1.4	Operators	4
1.5	Literals	4
1.5.1	Integer Literals	4
1.5.2	Floating Point Literals	4
1.5.3	Imaginary Number Literals	4
1.5.4	Units of Measure	4
1.5.5	Boolean Literals	4
1.5.6	String Literals	4
1.5.7	Symbol Literals	4
1.5.8	Type Parameters	4
1.5.9	Regular Expression Literals	4
1.5.10	Collection Literals	4
1.6	Whitespace & Comments	4
1.7	Preprocessor Macros	4
2	Identifiers, Names & Scopes	5
3	Types	7
3.1	Paths	8
3.2	Value Types	8
3.2.1	Value Type	8

3.2.2	Type Projection	8
3.2.3	Type Designators	8
3.2.4	Parametrized Types	8
3.2.5	Tuple Types	8
3.2.6	Annotated Types	8
3.2.7	Compound Types	8
3.2.8	Function Types	8
3.2.9	Existential Types	8
3.3	Non-Value Types	8
3.3.1	Method Types	8
3.3.2	Polymorphic Method Types	8
3.3.3	Type Constructors	8
3.4	Relations Between Types	8
3.4.1	Type Equivalence	8
3.4.2	Conformance	8
4	Basic Declarations & Definitions	9
4.1	Variable Declarations & Definitions	10
4.2	Property Declarations & Definitions	10
4.3	Instance Variable Definitions	10
4.4	Type Declarations & Aliases	10
4.5	Type Parameters	10
4.6	Variance of Type Parameters	10
4.7	Function Declarations & Definitions	10
4.7.1	Positional Parameters	10
4.7.2	Optional Parameters	10
4.7.3	Repeated Parameters	10
4.7.4	Named Parameters	10
4.7.5	Procedures	10
4.7.6	Method Return Type Inference	10
4.8	Use Clauses	10

5	Classes & Objects	11
5.1	Class Definitions	12
5.1.1	Class Linearization	12
5.1.2	Constructor & Destructor Definitions	12
5.1.3	Class Block	12
5.1.4	Class Members	12
5.1.5	Overriding	12
5.1.6	Inheritance Closure	12
5.1.7	Modifiers	12
5.2	Mixins	12
5.3	Unions	12
5.4	Enums	12
5.5	Compound Types	12
5.6	Range Types	12
5.7	Units of Measure	12
5.8	Record Types	12
5.9	Struct Types	12
5.10	Object Definitions	12
6	Expressions	13
6.1	Expression Typing	14
6.2	Literals	14
6.3	The Nil Value	14
6.4	Designators	14
6.5	Self, This & Super	14
6.6	Function Applications	14
6.6.1	Named and Optional Arguments	14
6.6.2	Input & Output Arguments	14
6.6.3	Function Compositions & Pipelines	14

6.7	Method Values	14
6.8	Type Applications	14
6.9	Tuples	14
6.10	Instance Creation Expressions	14
6.11	Blocks	14
6.12	Prefix & Infix Operations	14
6.12.1	Prefix Operations	14
6.12.2	Infix Operations	14
6.12.3	Assignment Operators	14
6.13	Typed Expressions	14
6.14	Annotated Expressions	14
6.15	Assignments	14
6.16	Conditional Expressions	14
6.17	Loop Expressions	14
6.17.1	Classic For Expressions	14
6.17.2	Iterable For Expressions	14
6.17.3	Basic Loop Expressions	14
6.17.4	While & Until Loop Expressions	14
6.17.5	Conditions in Loop Expressions	14
6.18	Collection Comprehensions	14
6.19	Return Expressions	14
6.19.1	Implicit Return Expressions	14
6.19.2	Explicit Return Expressions	14
6.19.3	Structured Return Expressions	14
6.20	Raise Expressions	14
6.21	Rescue & Ensure Expressions	14
6.22	Throw & Catch Expressions	14
6.23	Anonymous Functions	14
6.24	Conversions	14
6.24.1	Type Casting	14

7	Implicit Parameters & Views	15
8	Pattern Matching	17
8.1	Patterns	17
8.1.1	Variable Patterns	17
8.1.2	Typed Patterns	17
8.1.3	Literal Patterns	17
8.1.4	Constructor Patterns	17
8.1.5	Tuple Patterns	17
8.1.6	Extractor Patterns	17
8.1.7	Pattern Alternatives	17
8.1.8	Regular Expression Patterns	17
8.2	Type Patterns	17
8.3	Pattern Matching Expressions	17
8.4	Pattern Matching Anonymous Functions	17
9	Top-Level Definitions	19
9.1	Compilation Units	19
9.2	Modules	19
9.3	Module Objects	19
9.4	Module References	19
9.5	Top-Level Classes	19
9.6	Programs	19
10	Annotations	21
11	Naming Guidelines	23
12	The Coral Standard Library	25
12.1	Root Classes	25
12.1.1	The Object Class	25

12.1.2 The Nothing Class	25
12.2 Value Classes	25
12.3 Standard Reference Classes	25
A Coral Syntax Summary	27

Preface

Coral is a Ruby-like programming language which enhances advanced object-oriented programming with elements of functional programming. Every value is an object, in this sense it is a pure object-oriented language. Object blueprints are described by classes. Classes can be composed in multiple ways – classic inheritance, mixin composition, union and compound types.

Coral is also a functional language in the sense that every function is also an object. Therefore, function definitions can be nested and higher-order functions are supported out-of-the-box. Coral also has a limited support for pattern matching, which can emulate the algebraic types used in other functional languages.

Coral has been developed from 2012 in a home environment out of pure enthusiasm for programming and out of a desire for a truly versatile language. This document is a work in progress and will stay that way forever. It acts as a reference for the language definition and some core library classes.

Chapter 1

Lexical Syntax

Coral programs are written using the Unicode character set; Unicode supplementary characters are supported as well. Coral programs are preferably encoded with the UTF-8 character encoding. While every Unicode character is supported, usage of Unicode escapes is encouraged, since fonts that IDEs might use may not support the full Unicode character set.

1.1 Identifiers

1.2 Keywords

1.3 Newline Characters

1.4 Operators

1.5 Literals

1.5.1 Integer Literals

1.5.2 Floating Point Literals

1.5.3 Imaginary Number Literals

1.5.4 Units of Measure

1.5.5 Boolean Literals

1.5.6 String Literals

1.5.7 Symbol Literals

1.5.8 Type Parameters

1.5.9 Regular Expression Literals

1.5.10 Collection Literals

1.6 Whitespace & Comments

1.7 Preprocessor Macros

Chapter 2

Identifiers, Names & Scopes

Chapter 3

Types

3.1 Paths

3.2 Value Types

3.2.1 Value Type

3.2.2 Type Projection

3.2.3 Type Designators

3.2.4 Parametrized Types

3.2.5 Tuple Types

3.2.6 Annotated Types

3.2.7 Compound Types

3.2.8 Function Types

3.2.9 Existential Types

3.3 Non-Value Types

3.3.1 Method Types

3.3.2 Polymorphic Method Types

3.3.3 Type Constructors

3.4 Relations Between Types

Chapter 4

Basic Declarations & Definitions

4.1 Variable Declarations & Definitions

4.2 Property Declarations & Definitions

4.3 Instance Variable Definitions

4.4 Type Declarations & Aliases

4.5 Type Parameters

4.6 Variance of Type Parameters

4.7 Function Declarations & Definitions

4.7.1 Positional Parameters

4.7.2 Optional Parameters

4.7.3 Repeated Parameters

4.7.4 Named Parameters

4.7.5 Procedures

4.7.6 Method Return Type Inference

4.8 Use Clauses

Chapter 5

Classes & Objects

5.1 Class Definitions

5.1.1 Class Linearization

5.1.2 Constructor & Destructor Definitions

5.1.3 Class Block

5.1.4 Class Members

5.1.5 Overriding

5.1.6 Inheritance Closure

5.1.7 Modifiers

5.2 Mixins

5.3 Unions

5.4 Enums

5.5 Compound Types

5.6 Range Types

5.7 Units of Measure

5.8 Record Types

Chapter 6

Expressions

6.1 Expression Typing

6.2 Literals

6.3 The Nil Value

6.4 Designators

6.5 Self, This & Super

6.6 Function Applications

6.6.1 Named and Optional Arguments

6.6.2 Input & Output Arguments

6.6.3 Function Compositions & Pipelines

6.7 Method Values

6.8 Type Applications

6.9 Tuples

6.10 Instance Creation Expressions

6.11 Blocks

Chapter 7

Implicit Parameters & Views

Chapter 8

Pattern Matching

8.1 Patterns

8.1.1 Variable Patterns

8.1.2 Typed Patterns

8.1.3 Literal Patterns

8.1.4 Constructor Patterns

8.1.5 Tuple Patterns

8.1.6 Extractor Patterns

8.1.7 Pattern Alternatives

8.1.8 Regular Expression Patterns

8.2 Type Patterns

8.3 Pattern Matching Expressions

8.4 Pattern Matching Anonymous Functions

Chapter 9

Top-Level Definitions

9.1 Compilation Units

9.2 Modules

9.3 Module Objects

9.4 Module References

9.5 Top-Level Classes

9.6 Programs

Chapter 10

Annotations

Chapter 11

Naming Guidelines

Chapter 12

The Coral Standard Library

12.1 Root Classes

12.1.1 The Object Class

12.1.2 The Nothing Class

12.2 Value Classes

12.3 Standard Reference Classes

Chapter A

Coral Syntax Summary