The Coral Language Specification

Markéta Nikola Lisová

April 21, 2014

Contents

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

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

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