Ceylon (programming language)

Ceylon is an object-oriented, strongly statically typed programming language with an emphasis on immutability, created by Red Hat. Ceylon programs run on the Java virtual machine (JVM), and could be compiled to JavaScript.[6][7] The language design focuses on source code readability, predictability, toolability, modularity, and metaprogrammability.[8]

Important features of Ceylon include:[9]

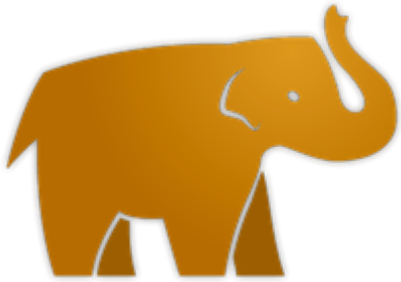
- A type system enforcing null safety and list element existence at compile time
- Regular syntax and semantics, avoiding special cases and primitively-defined constructs in favor of syntactic sugar
- Support for generic programming and metaprogramming, with reified generics
- Modularity built into the language, based on JBoss modules, interoperable with OSGi[10] and Maven[11]
- powerful tools, including an Eclipse-based IDE[12]

The name "Ceylon" is an oblique reference to Java, in that Java and Sri Lanka, formerly known as Ceylon, are islands known for growth and export of coffee and tea.

In August 2017, Ceylon was donated to the Eclipse Foundation.

Contents

Ceylon



Paradigm	Object-oriented
Designed by	Gavin King
Developer	Eclipse Foundation
First appeared	2011
Stable release	1.3.3 / August 21, 2017
Typing discipline	Static, strong, safe
Platform	Java virtual machine, JavaScript
License	Apache License 2.0
Filename extensions	.ceylon[1]
Website	ceylon-lang.org (http://ceylon-lang.org)
Influenced by	
Java,[2] Scala, Smalltalk, ML,[3] Lisp,[4] Whiley[5]	

Language features

Ceylon is heavily influenced by Java's syntax, but adds many new features.

Type system

One of the most novel aspects of Ceylon compared to Java is its type system. Ceylon foregoes Java's primitive types[13] and boxing in favor of a type system composed entirely of first-class objects. While this may cause boxing overhead in some situations, it makes the type system more uniform.

Ceylon allows for union and intersection types, in a similar fashion to TypeScript, Whiley and Flow.

Union types, written `A|B`, allow a variable to have more than one type. The following example shows a Ceylon function which may take either an integer or a string:

```
shared void integerOrString(Integer|String input) {
    if (is Integer input) {
        print("Got the integer ``input``");
    } else {
        print("Got the string '``input``'");
    }
}
```

Intersection types, written `A&B`, are the theoretical foundation of flow-sensitive typing:

```
shared void integerOrString(Integer|String input) {
    Integer added = input + 6; // illegal; the + operator is not defined on Integer|String

    if (is Integer input) {
        Integer added = input + 6; // legal; input is now known to be an Integer
        print("Got the integer ``input``");
    } else {
        print("Got the string '``input``'");
    }
}
```

The condition `is Integer input` narrows the type of `input` to `<Integer|String> & Integer`, which distributes to `Integer&Integer | String&Integer`, which, as `String` and `Integer` are disjoint types, is equivalent to `Integer&Integer | Nothing` (`Nothing` is the empty bottom type), which simplifies to just `Integer`.

Null safety

Union and intersection types are used to provide null safety. The top type of the Ceylon type hierarchy is the class `Anything`, which has two subclasses: `Object`, the superclass of all normal classes and all interfaces, and `Null`, with the only instance `null`. Since `Object` and `Null` are disjoint types, most regular types like `Integer` or `List<String>` are not nullable; a nullable type is the union `Integer|Null`, abbreviated `Integer?`.[14]

Intersection types can be used to get a non-optional type out of a possibly-optional type, such as a type parameter. For example, the signature of a function that removes `null` elements from a stream of values could be:

```
Iterable<Element&Object> removeNulls<Element>(Iterable<Element> stream);
```

When `removeNulls` is called with a stream of `Integer|Null` elements, the result will be a stream of `<Integer|Null> & Object` elements, which simplifies to `Integer`.

Functions

Similarly to many modern languages, Ceylon supports first class functions and higher order functions, including function types and anonymous functions[15]

```
// A top-level higher-order function using block syntax (not associated with any user-created
classes)
String process(String text, String transformString(String toChange)) {
    return transformString(text);
}

// A top-level function calling String.reverse in expression form.
String reverse(String text) => text.reversed;

// A function reference to String.reversed but mostly equivalent to the function above.
String(String) reverseFunctionReference = String.reversed;

// An example where the top-level function above is provided as an argument to the higher-
order function above
String reversed1 = process("one", reverse);

// An example where an anonymous function - (text) => text+text - is provided to the higher-
order function above.
String reversed2 = process("one", (text) => text+text);
```

Enumerated types

Similar to Java and many other languages, and with a similar mechanism as algebraic types, Ceylon supports enumerated types, otherwise known as enums. This is implemented in Ceylon with a pattern of limiting the instances of an abstract class at declaration to a limited set of objects (in this case, singleton instances). Another way to implement this pattern is with the new constructor feature in Ceylon 1.2 where the objects are implemented as different named constructor declarations.[16]

```
// Traditional syntax for enumerated type, in this case, limiting the instances to three
objects(for this purpose: Singletons)
abstract class Vehicle(shared String name) of plane | train | automobile {}

object plane extends Vehicle("plane") {}
object train extends Vehicle("train") {}
object automobile extends Vehicle("automobile") {}
// Compile error: type is not a subtype of any case of enumerated supertype: 'boat' inherits
'Vehicle'
//object boat extends Vehicle("boat") {}

// New (as of Ceylon 1.2.0) constructor-based syntax
class Vehicle of plane | train | automobile {
    String name;

    abstract new named(String pName) {
        name = pName;
    }

    shared new plane extends named("plane") {}
    shared new train extends named("train") {}
    shared new automobile extends named("automobile") {}
    // Compile error: value constructor does not occur in of clause of non-abstract
```

```
enumerated class: 'boat' is not listed in the of clause of 'Vehicle'
    //shared new boat extends named("boat") {}
}
```

Type inference

Ceylon is strongly and statically typed, but also has support for type inference. The `value` keyword is used to infer the type of a variable, and the `function` keyword is used to infer the type of a function. The following two definition pairs are each equivalent:

```
Integer i = 3;
value i = 3;

Integer add(Integer i1, Integer i2) {
    return i1 + i2;
}
function add(Integer i1, Integer i2) {
    return i1 + i2;
}
```

However, to make single-pass type inference possible, type inference is only allowed for non-toplevel and unshared declarations.[17]

Entry point with names

By default the starter (`ceylon run`) runs the shared run() function of a module:

```
/* The classic Hello World program */
shared void run() {
    print("Hello, World!");
}
```

but any other shared function without parameters can be used as main calling the program with the --run parameter, like this:

```
ceylon run --compile=force --run hello default
```

Versions

Versions of Ceylon released:[18]

- M1 0.1 "Newton" (Dec 20 2011)
- M2 0.2 "Minitel" (Mar 2 2012)
- M3 0.3 "V2000" (Jun 21 2012)
- M3.1 0.3.1 "V2000" (Jul 6 2012)
- M4 0.4 "Analytical Engine" (Oct 29 2012)
- M5 0.5 "Nesa Pong" (Mar 13 2013)
- M6 0.6 "Virtual Boy" (Sep 23 2013)
- 1.0 beta "Virtual Boy" (Sep 24 2013)
- 1.0.0 "No More Mr Nice Guy" (Nov 13 2013)
- 1.1.0 "Ultimate Ship The Second" (Oct 09 2014)

- 1.2.0 "A Series of Unlikely Explanations" (Oct 28 2015)
- 1.2.1 "Irregular Apocalypse" (Feb 11 2016)
- 1.2.2 "Charming But Irrational" (Mar 11 2016)
- 1.3.0 "Total Internal Reflection" (Sep 19 2016)
- 1.3.1 "Now We Try It My Way" (Nov 22 2016)
- 1.3.2 "Smile Tolerantly" (Mar 02 2017)
- 1.3.3 "Contents May Differ" (Aug 21 2017)

License

All parts of Ceylon are available as free software, mostly the Apache License.[19] Part of the source code is licensed under LGPL.

See also

- Dart (programming language), has its own VM, compiles to JS, type system not very strict, supports mixins
- Fantom (programming language), compiles to JVM, type system not very strict, supports mixins

References

1. King, Gavin. "The Ceylon Language: §4.1 Compilation unit structure" (http://ceylon-lang.org/documentation/1.2/spec/html_single/#compilationunitstructure). Retrieved 2015-12-04. "A *compilation unit* is a text file, with the filename extension `.ceylon`."
2. "Frequently Asked Questions: What is Ceylon?" (http://ceylon-lang.org/documentation/1.2/faq/#what_is_ceylon). Retrieved 2015-12-04. "Ceylon is a new programming language that's deeply influenced by Java"
3. "ceylon/user - Gitter" (https://gitter.im/ceylon/user?at=5660a7242cbea1d7054de9d9). Retrieved 2015-12-04.
4. "ceylon/user - Gitter" (https://gitter.im/ceylon/user?at=5660a90e5057376520db6f8b). Retrieved 2015-12-04.
5. "Top 10 Ceylon language features Java wishes it had" (https://jaxenter.com/top-10-ceylon-language-features-java-wishes-it-had-108003.html). Retrieved 2019-11-29.
6. "Ceylon 1.0 beta" (http://ceylon-lang.org/blog/2013/09/22/ceylon-1/). Retrieved 2013-09-26.
7. "Project Ceylon – Red Hat builds Java replacement" (https://www.theregister.co.uk/2011/04/13/red_hat_unveils_project_ceylon). The Register. 2011-04-13. Retrieved 2011-11-27.
8. King, Gavin (2012-01-10). "Principles that guide this project" (http://ceylon-lang.org/blog/2012/01/10/goals/). Retrieved 2015-12-04.
9. "FAQ about language design: Goals" (http://ceylon-lang.org/documentation/1.2/faq/language-design/#goals). Retrieved 2015-12-04.
10. Festal, David (2014-10-10). "Write in Ceylon, deploy as OSGI, use in Java EE" (http://ceylon-lang.org/blog/2014/10/10/ceylon-osgi-jee/). Retrieved 2015-12-04.
11. "Maven repositories" (http://ceylon-lang.org/documentation/1.2/reference/repository/maven/#maven_repositories). Retrieved 2015-12-04.
12. "Features of Ceylon IDE" (http://ceylon-lang.org/documentation/1.2/ide/features/). Retrieved 2015-12-04.

13. King, Gavin. "Ceylon: Language Design FAQ" (http://ceylon-lang.org/documentation/faq/language-design/).
14. King, Gavin. "The Ceylon Language: §1.4.3 Compile-time safety for null values and flow-sensitive typing" (http://ceylon-lang.org/documentation/1.2/spec/html_single/#compiletimesafety). Retrieved 2015-12-04.
15. King, Gavin. "The Ceylon Language: 4.7 Functions" (http://ceylon-lang.org/documentation/1.2/spec/html/declarations.html#functions). Retrieved 5 December 2015.
16. King, Gavin. "The Ceylon Language: 4.5.8 Enumerated classes" (http://ceylon-lang.org/documentation/1.2/spec/html/declarations.html#classeswithcases). Retrieved 6 December 2015.
17. King, Gavin. "The Ceylon Language: §3.2.9 Type inference" (http://ceylon-lang.org/documentation/1.2/spec/html_single/#typeinference). Retrieved 2015-12-04.
18. https://ceylon-lang.org/download-archive/ Ceylon: Download Previous Ceylon versions
19. "Ceylon: Licenses" (http://ceylon-lang.org/code/licenses). Retrieved 2015-12-04.

External links

- Official website (https://ceylon-lang.org/)