Indonesian Journal of Electrical Engineering and Computer Science
Vol. 17, No. 2, February 2020, pp. 845~849
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v17.i2.pp845-849

JAVA and DART programming languages: Conceptual comparison

Afaf Mirghani Hassan
Computer Department, Tabuk University (Branch Duba), Saudi Arabia

Article Info

Article history:

Received Jun 3, 2019
Revised Aug 5, 2019
Accepted Aug 19, 2019

Keywords:

Control statement
Dart
Data type
Java
Semantic
Subprogram

ABSTRACT

This paper elaborates on the concepts of a new programming language "Dart", which has been developed by Google and considered for future use. Here, we compare it to the most famous, real time, and updated language "Java". This is to define similarities and differences between the two important languages, explain programs' behavior, with a focus on investigating alternative implementation strategies and problem definitions. We used programming languages' concepts and terminologies to compare between the main characteristics of the two languages, Dart & Java.

Corresponding Author:

Afaf Mirghani Hassan,
Computer Department,
Tabuk university,
(Branch Duba), Saudi Arabia,
Email: drafafmirg@gmail.com

1. INTRODUCTION

Google has released a new language aimed at developing complex, Google-scale web applications In October 2011. The aim was to develop a language that is better language for the web than JavaScript. That was because of the frustration with the slow progress in evolving JavaScript, partly caused by the so many interested parties. The main goal was to sustain the dynamic nature of JavaScript, but have a better performance and is extendable to tooling for large projects. It would also be able to cross-compile to JavaScript. This language was given the name Dart [1].

Dart is a general purpose programming language. It is a new language in the C tradition, designed with ease of use, familiarity to the vast majority of programmers, and scalability in mind. It is purely object-oriented, class-based, programming language. Dart is intended to provide a platform that is specifically crafted to support future needs and emerging software/hardware platforms. As such it hides low level details of the underlying platform, while enabling programmers to use the powerful facilities new platforms have to offer [2].

It is an open source, structured language to create complex, browser-based web applications. Applications usually run in Dart either by the browser directly, which supports Dart code, or by compiling code to JavaScript. Dart has a familiar syntax, and it's class-based, It has a concurrency model called isolates that allows parallel execution. In addition to running code in web browsers and converting it to JavaScript, it can also run code on the command line [3-10]. For client side web app development, Dart has many advantages over JavaScript. These include but are not limited to improved speed, enforcement of programmatic structure, and improved facilities for software reuse. Best of all, Dart is automatically converted to JavaScript so that it works with all web browsers, Dart is a fresh start, without the baggage of

the last two decades of the webDart language Designer has pragmatic choice to make smooth experience coding [3-10].

Java is an efficient programming language likable by developers and so is Dart. Both languages have powerful concepts such as object creation, concurrency, serialization, reflection, and many more, all in real time [3, 4, 11-17]. Java has evolved over time; newer versions of Java increase the need for specific best-practices advice for multiple paradigms, functional interfaces, lambda expressions, method references, and streams, Default and static methods in interfaces, resources' statement, New library features such as the Optional interface, java.time, and factory methods for collections. All of that, so developers can convert to dart with relative ease [11-17]. One example of a Dart implementation is Flutter, a mobile app SDK from Google, which has Java integrity. The app Create a simple **Dart** class, Use optional parameters (overloading), Create a factory, Implement an interface, Use **Dart** for functional programming [3-12]. The usability and familiarity of the language makes it a good candidate to implement complex engineered systems such as those in [18-21].

We would like to find a geometric recognition language, a graphics interpreter, a rule-based control interpreter, and an object-oriented language interpreter to work together all at once [22]. One good practice is to structure a complex program as a collection of languages, each of which provides a different viewpoint, different way for different program elements [22]. It might be this is the reason why our programs are becoming increasingly complex thinking more explicitly about languages might be the best way to deal with this complexity. The basic idea is that the interpreter itself is just a program that is written in some language, whose interpreter is another program, which is written in some other language etc.

One main objective or strategy concept of a programming language is to distinguish itself from other languages based on the characteristics and usages or utilities of the language. In this study, we try to investigate the programming language Dart, by comparing to the important programming language (JAVA), concentrating on the similarities and differences of the two. Section 2 detals this comparison and present it in an easy readable table format. The paper is concluded in Section 3.

2. DART AND JAVA: COMPARISON OF CONCEPTS

Google is a real time interactive system application dealing with search, electronic mail, translation, play, images, drive and many other applications. Hence, the company is in a constant search to develop programming languages that connect all of these, and also future applications. Dart programming language has come to meet this need [23].

Java is considered a general-purpose programming language while Dart is a client-optimized programming language. There are many similarities and differences between the two programming languages. They are similar in criteria such as readability, reliability, cost, portability, and generality [4]. Both languages are writable and well-defined or precise languages. The two languages are roughly OOP languages (Object Oriented Programming), they are classes' structure, and both based on C structure i.e. similar software syntax in C. Also, the two of them are web software languages, and both are lovely languages for developers [24].

Some of the differences between the two are: while Java is general purpose language, Dart is considered Google specific language. Dart is a class structure same as Java, however in Dart class code cannot be written. This is in contrast to Java, a class-based object-oriented programming language not a pure object- oriented one. That is, Java has a second scoping mechanism (Package Scope) that can be used in place in all classes in a package, in case there are no access control modifiers that are visible throughout the package [6].

Table 1 shows a Comparison between Dart and Java in terms of behavior, syntax, sematic, value, environment, expression, procedure and conditional clauses. With Dart we can create applications on the web, smart phones and servers [22-26].

Table 1. Comparing Dart and Java

	Comparison points	Java	Dart
1	Authority	Sun company, now Oracle	Google company
2	Generation	Updated interactive language	Future language
3	Script language	Can be Web programming	Interactive web language
4	Syntax	-many similar structure as in Dart - contain keywords	• with Construct class as in java • not contain keywords • depends on has letters (_) that is enters to special library words or commands.
5	Sematic	Example is Java's static semantics rule: else matches with the nearest if.	Example is the Flutter semantics package.

	Comparison points	Java	Dart
6	Portability	(Java Virtual Machine) JVM concept, JIT(Just In Time) compilers	Working on different platforms
7	C language architecture	C, java are imperative language (same categories)	Based On
8	Java Script language	similar syntax	supports a multi-tasking feature like JavaScript
9	false result	more than one false result (null, false, 0)	one false result (False),
10	Cascade Notation	N/A	(..)
11	Comment	//	//
12	Run	UNICODE, ASCII	UTF-32 code points of a string
13	Asynchrony support	code run line by line	libraries are full of functions that return Future or Stream objects
14	Exception Handling and Event Handling	-All exceptions are objects of classes Throw able (Error- exception). -Java Swing GUI Components. -Java Event Model	Exceptions: exception, error, throw, Catch
15	Data Types	Defined all: -Primitive Data Types -Character String (String class) -Array (index integer types) -Array Initialization string object- support jagged arrays -not support unions -allows replace pointers	Defined as: Built-in types (numbers, strings, Booleans, lists (also known as arrays), sets, maps, runes (for Unicode characters in a string), symbols). Initializing list
16	Expressions and Assignment Statements	-assignment statement produces a result and can be used as operands -Mixe Mode (widening assignment)	Specific defined operators You can override many of these operators, as described in Overridable
17	Control Structures	-all control expression must be Boolean. -Java's static semantics rule: else matches with the nearest if. -Multiple-Way Selection (Switch). Unconditional labeled e (break). -labeled versions of continue. -do not support goto statement.	• Control flow statements (if and else, • For loops, while and do-while loops • break and continue, switch and case assert)
18	Libraries	Use import to specify how a namespace from one library	Use import to specify how a namespace from one library
19	Lexical scope	:: inherent class	follow the curly braces outward
20	Subprograms	-All parameters are passed by value. -Object parameters are passed by reference. Require Type Checking Parameters. --Array inherits a named constant length. -allow programmers to write multiple versions of subprograms with the same name. -predefined overloaded subprograms -allow Generic Subprograms	-Function. Type (class objects, Anonymous, scope, Lexical closures, testing equality. -Methods provide behavior for an object. -Callable classes. -Annotations for public APIs, (function works if you omit types). -allow Generic Subprograms

21	An Example structure	class StackClass {
		private:
		private int [] *stackRef;
		private int [] maxLen, topIndex;
		public StackClass() { // a constructor
		stackRef = new int [100];
		maxLen = 99;
		topPtr = -1;
		};
		public void push (int num) {…};
		public void pop () {…};
		public int top () {…};
		public boolean empty () {…};
		}
			// Define a function.
			printInteger(int aNumber) {
			print('The number is $aNumber.'); // Print to console.
			}
			
			//This is where the app starts executing.
			main() {
			var number = 42; // Declare and initialize a variable.
			printInteger(number); // Call a function.
			}

3. CONCLUSION

Dart is a powerfull, interactive language that is expected to get widely adopted by developers the same way Java is adopted today. Dart code can be reused for either smat phones (clients) or servers; howver, it still lacks the general-purpose property of Java. In domain-specific (scientific, business, artificial intelligence, web and system) applications, The programming language's domain is extended to the special objectives of that domain; In this capacity Dart can be considered as a domain-specific programming language. Dart as a domain-specific programming language will be one of the Web Software with three branches of web or Eclectic collection of languages software, which are markup (HTML), script (PHP) and general purpose(JAVA).

REFERENCES

[1] Chris Buckett, *Dart in action*, Manning Publications, 2013.
[2] Gilad Bracha, *The dart programming language*, Addison Wiseley Educational Publishers, 2015,
[3] Gilad Bracha, *The dart programming language*, Publishing Addision Wesely 2016.
[4] David Kopec, *Dart for absolute beginners*, 1st ed. Edition, Kindle Edition, 2014.
[5] Chris Buckett, *Dart in action*, Second Edition, Publishing manning Shelter Island, 2013.
[6] D. Mitchell, S. Akopkokhyants, and Ivo Balbaert, *Dart scalable application development*, Packt publishing 2017.
[7] Ivo Balbaert and Dzenan Ridjanovic, *Learning dart*, Packt publishing, 2015.
[8] Davy Mitchell, *Dart by example*, Packt publishing, 2015.
[9] Martin Sikora, *Dart essential*, Packt publishing, 2015.
[10] Sergey Akopkokhyants, *Dart mastering*, Packt publishing, 2014.
[11] Andria Redko, *Advanced java: Preparing you for java mastery*, Exelixis Media P.C., 2015
[12] Oracle D61796FR10, Edition 1.0, D61796FR10_EP, *Fundemantals of the java programming language SE6*, Oracle and java copyright 2007, 2009 sunmicrosystem.
[13] Tony Stubblebine, *Regular Expression Pocket Reference Regular Expressions for Perl, Ruby, PHP, Python, C, Java and.NET*, 2nd edition, O'Reilly Media, 2007).
[14] Barry Burd, *Beginning programming with java for dumm*ies, 4th Edition, John Wiley & Sons, Jul 2014.
[15] Joshua Bloch, *Effective java*, 3rd Edition, Addison-Wesley Professional, 2017.
[16] Herbert Schildt, *Java the complete reference*, seven Edition, McGraw-Hill Education, 2010.
[17] Chris Mayfield and Allen Downey, *Think java: How to think like a computer scientist*, Green Tea Press, 2016
[18] Zainab Mahmod, Aisha Hassan Abdalla Hashim, Othman O. Khalifa, Farahat Anwar, and Shihab A. Hameed, " The Effect of Network's Size on the Performance of the Gateway Discovery and Selection Scheme," *Indonesian Journal of Electrical Engineering and Informatic*, Dec 2017.
[19] Md. Reza Ranjbar, and Aisha Hassan Abdalla Hashim, "Development of an autonomous remote access water quality monitoring system," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 8, no. 2, pp. 467-474 2017.
[20] Senan Shayma and Aisha Hassan Abdalla Hashim, "Performance Analysis of HRO-B+ scheme for the nested mobile networks using OPNet", *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 8, no. 2, pp. 522-532, 2017.
[21] T.S. Gunawan, I.R.H. Yaldi, M. Kartiwi, H. Mansor, "Performance evaluation of smart home system using Internet of Things," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 1, pp. 400-411, 2018.
[22] Daniel P. Friedman and Mitchell Wand, *Essentials of Programming Languages*, Third Edition, MIT Press, 2008
[23] Codelabs, Intro to Dart for Java Developers: Create a simple Dart class, Codelabs Developers Google, [Online]. Available: https://codelabs.developers.google.com/codelabs/from-java-to-dart/#1, [Accessed July, 14, 2019].
[24] Sebesta W. Robert, *Concept Of programming Language*, Eight Edition, Wesley Longman Publishing, 2007.
[25] Codelabs, Dart cheatsheet, [Online]. Available: https://dart.dev/codelabs/dart-cheatsheet, [Accessed July,14, 2019].
[26] J. C. Mitchell, *Concepts in programming languages*, Cambridge University Press, 2003.

APPENDIX

Below example code that can be compiled with DART, HTML, CSS on the same time; with HTML output and CONSOLE [26].

```
1  import 'dart:html';
2  import 'dart:math' show Random;
3  // We changed 5 lines of code to make this sample nicer on
4  // the web (so that the execution waits for animation frame,
5  // the number gets updated in the DOM, and the program ends
6  // after 500 iterations).
7  main() async {
8   print('Compute π using the Monte Carlo method.');
9   var output = querySelector("#output");
10  await for (var estimate in computePi().take(500)) {
11  print('π ≅ $estimate');
12  output.text = estimate.toStringAsFixed(5);
13  await window.animationFrame;
14  }
15 }
16 /// Generates a stream of increasingly accurate estimates of π.
17 Stream<double> computePi({int batch: 100000}) async* {
18  var total = 0;
19  var count = 0;
20  while (true) {
21  var points = generateRandom().take(batch);
22  var inside = points.where((p) => p.isInsideUnitCircle);
23  total += batch;
24  count += inside.length;
```

```
  var ratio = count / total;
  // Area of a circle is A = π·r², therefore π = A/r².
  // So, when given random points with x ∈ <0,1>,
  // y ∈ <0,1>, the ratio of those inside a unit circle
  // should approach π / 4. Therefore, the value of π
  // should be:
  yield ratio * 4;
  }
 }
 Iterable<Point> generateRandom([int seed]) sync* {
  final random = Random(seed);
  while (true) {
  yield Point(random.nextDouble(), random.nextDouble());
  }
 }
 class Point {
  final double x, y;
  const Point(this.x, this.y);
  bool get isInsideUnitCircle => x * x + y * y <= 1;
 } import 'dart:html';
 import 'dart:math' show Random;
 // We changed 5 lines of code to make this sample nicer on
 // the web (so that the execution waits for animation frame,
 // the number gets updated in the DOM, and the program ends
 // after 500 iterations).
 main() async {
  print('Compute π using the Monte Carlo method.');
  var output = querySelector("#output");
  await for (var estimate in computePi().take(500)) {
  print('π ≅ $estimate');
  output.text = estimate.toStringAsFixed(5);
  await window.animationFrame;
  }
 }
 /// Generates a stream of increasingly accurate estimates of π.
 Stream<double> computePi({int batch: 100000}) async* {
  var total = 0;
  var count = 0;
  while (true) {
  var points = generateRandom().take(batch);
  var inside = points.where((p) => p.isInsideUnitCircle);
  total += batch;
  count += inside.length;
  var ratio = count / total;
  // Area of a circle is A = π·r², therefore π = A/r².
  // So, when given random points with x ∈ <0,1>,
  // y ∈ <0,1>, the ratio of those inside a unit circle
  // should approach π / 4. Therefore, the value of π
  // should be:
  yield ratio * 4;
  }
 }
 Iterable<Point> generateRandom([int seed]) sync* {
  final random = Random(seed);
  while (true) {
  yield Point(random.nextDouble(), random.nextDouble());
  }
 }
 class Point {
  final double x, y;
  const Point(this.x, this.y);
  bool get isInsideUnitCircle => x * x + y * y <= 1;
 }
```