Language Critique

# Version Information

Draft version. Alpha, 1.1

Hope to take it to a Beta soon.

Started: 7/29/23

Expected completion: 8/30/23

Contents

[Version Information 1](#_Toc143360194)

[Goal/Note 1](#_Toc143360195)

[Exploring Java through questions 1](#_Toc143360196)

[Brief answers 2](#_Toc143360197)

[Java Quiz 4](#_Toc143360198)

[Python 5](#_Toc143360199)

[Python Answers 5](#_Toc143360200)

# Goal/Note

1. This is expected to be a conversation about computer languages and not a test of one-up-ness. Through conversation one can understand each better, for much of joy in computers lie in languages.
2. A language refresher to come back to once in a while

# Exploring Java through questions

1. Briefly what are some key features of Java programming language?
2. What distinguished Java when it came in as a programming language?
3. What are major critiques of the Java programming language?
4. How come UI libraries written in Java have not really taken off?
5. What are more recent enhancements to the Java programming language?
6. What are some key enhancements to the Java language since Java 8?
7. When were streams introduced into Java?
8. What are still some criticisms of Java?
9. In what ways Java improved upon C++?
10. Tell me an example of Java streams?
11. What kind of a method is sum() above?
12. Can you have a custom terminal operation?
13. What are key classes in Java streams package?
14. Explain to me suppliers and consumers in streams?
15. What are some of the most critical packages in Java?
16. What is behind the statement that "new" is considered harmful in Java and other OO languages?
17. What other languages support annotations?
18. Give me an example of attributes in C#?

## Brief answers

1. **Key Features of Java:**
   * Platform independence
   * Object-oriented
   * Robust and Secure
   * Multithreading
   * Garbage collection
   * Rich API
2. **Distinguishing Features of Java:**
   * Platform independence (Write Once, Run Anywhere)
   * Object-oriented nature
   * Robust standard library
3. **Major Critiques of Java:**
   * Verbosity
   * Performance overhead
   * Lack of unsigned types
   * Complexity of the language
4. **Challenges for UI Libraries in Java:**
   * Swing's perceived complexity
   * JavaFX's late arrival
   * Competition from web-based technologies
5. **Recent Enhancements to Java:**
   * Records
   * Pattern matching for instanceof
   * Text blocks
   * Sealed classes
6. **Key Enhancements Since Java 8:**
   * Lambdas
   * Streams API
   * Date and Time API
   * CompletableFuture
   * Default methods in interfaces
7. **Introduction of Streams in Java:**
   * Streams were introduced in Java 8.
8. **Ongoing Criticisms of Java:**
   * Slower startup time
   * Boilerplate code
   * Verbosity
   * Absence of value types
9. **Ways Java Improved Upon C++:**
   * Simplicity
   * Platform independence
   * Garbage collection
   * Standard libraries
10. **Example of Java Streams:**
    * **IntStream.range(1, 10).filter(x -> x % 2 == 0).forEach(System.out::println);**
11. **Sum() Method in Streams:**
    * It is a terminal operation that calculates the sum of elements in a stream.
12. **Custom Terminal Operation:**
    * Yes, custom terminal operations can be created using the **collect()** method with a custom **Collector**.
13. **Key Classes in Java Streams Package:**
    * **Stream**
    * **Collectors**
    * **Optional**
    * **IntStream**
14. **Suppliers and Consumers in Streams:**
    * Suppliers produce elements for streams.
    * Consumers consume elements from streams.
15. **Most Critical Packages in Java:**
    * **java.lang**
    * **java.util**
    * **java.io**
    * **java.net**
    * **java.awt**
    * **java.util.concurrent**
16. **"new" Considered Harmful in Java:**
    * Can lead to tight coupling, reduced flexibility, and difficulty in testing and maintainability.
17. **Languages Supporting Annotations:**
    * Kotlin, C#, Python, C/C++, Swift, TypeScript, Scala, PHP, Ruby, and more.
18. **Example of Attributes in C#:**
    * **DeprecatedAttribute** marking a method as deprecated.

## Java Quiz

What is odd about this Java code:

var count = 10;

var message = "Hello, world!";

var names = new ArrayList<String>();

# Python

1. Functional programming overview
2. What is the current state of Clojure?
3. What is the level of support for functional programming in Python?
4. Is Julia considered a functional programming language?
5. What are some examples of higher-order functions in Python?
6. Can you give an example of immutability in Python?
7. Why is immutability important? Can you give an example?
8. Is Scala a functional programming language?
9. What is fold? Can you give an example?
10. Is fold the same as reduce?
11. What do you think of the "reasoning" debate in programming languages?
12. What are some key features of Python?
13. What are some key programming concepts in Python that are quite unique to Python?
14. What is duck typing again?
15. What other languages support duck typing?
16. Does Julia support duck typing?
17. Does C# support duck typing?

## Python Answers

1. **Functional programming overview**: Functional programming is a programming paradigm focused on treating computation as the evaluation of mathematical functions, emphasizing immutability, pure functions, and higher-order functions.
2. **Current state of Clojure**: As of my last update in September 2021, Clojure is a mature, dynamically-typed functional programming language that runs on the Java Virtual Machine (JVM) and is widely used for web development and data-intensive applications.
3. **Level of support for functional programming in Python**: Python supports functional programming concepts, including higher-order functions, lambda functions, map, filter, and reduce, making it suitable for functional-style programming.
4. **Is Julia considered a functional programming language?**: Julia is a multi-paradigm language that supports functional programming but is not purely functional. It is designed for high-performance scientific computing and features multiple dispatch.
5. **Examples of higher-order functions in Python**: Functions that take other functions as arguments or return functions are higher-order functions. Examples in Python include **map()**, **filter()**, and **reduce()** from the **functools** module.
6. **Example of immutability in Python**: Immutable objects cannot be modified after creation. Examples in Python include strings and tuples, where operations create new objects rather than modifying the original ones.
7. **Importance of immutability with an example**: Immutability helps ensure data integrity, simplifies code reasoning, and enables safer concurrent programming. Example: Immutable strings in Python prevent unexpected changes.
8. **Is Scala a functional programming language?**: Scala is a multi-paradigm language that strongly supports functional programming, including first-class functions, immutability, and higher-order functions.
9. **Explanation and example of fold**: Fold (or reduce) is a higher-order function that accumulates the elements of a collection into a single result by applying a binary operation. Example: Using **reduce()** to sum elements of a list in Python.
10. **Is fold the same as reduce?**: Yes, in functional programming, "fold" and "reduce" are often used interchangeably to describe the same higher-order function.
11. **Opinion on the "reasoning" debate in programming languages**: As an AI language model, I don't have opinions. The debate centers around trade-offs between features like expressiveness, performance, and readability in programming languages.
12. **Key features of Python**: Python features readable and expressive syntax, dynamic typing, object-oriented and functional programming support, extensive standard library, and cross-platform compatibility.
13. **Key programming concepts unique to Python**: Python's unique concepts include significant whitespace (indentation), duck typing, list comprehensions, generators, default function arguments, and extensive documentation.
14. **Explanation of duck typing**: Duck typing is a concept in dynamically typed languages where an object's type is determined by its behavior (methods and attributes) rather than its explicit type or inheritance.
15. **Languages supporting duck typing**: Duck typing is common in dynamically typed languages like Python, Ruby, JavaScript, Perl, Lua, Objective-C, Smalltalk, and Erlang.
16. **Does Julia support duck typing?**: Yes, Julia is a dynamically typed language that supports duck typing, allowing objects to be treated based on their behavior rather than explicit type.
17. **Does C# support duck typing?**: No, C# is a statically typed language and does not support duck typing. It requires explicit type declarations at compile-time.
18. **Does Java know the type information at runtime?**: Yes, Java knows the type information at runtime. It is a statically typed language but uses reflection to access and manipulate type information at runtime.