Advanced Software Paradigms

Assignment 1_part 1 Ilyas Karimov

1. First of all, I would like to start with the discussion of the former. Using multiple-line comments is beneficial for long comments, since it will be **simple (readability)** and **expressive (writability)**; For long comments, if we can use only symbols twice (/* */), and if so, why use them in many lines (// in every line)?! The disadvantage of multi-line comments is that forgetting it at the end makes it unreliable since you either need to go line-by-line to check where it might end or erase the whole comment from start to solve the issue. The advantage of the one-line code would be you can turn the code on and off by erasing one // in huge comments without affection of other lines (for multi-line comments, you either need to cut that code and copy it after the comments). The disadvantage of the using one-line comments is we need to use the same symbols in every new line. It decreases **readability** because we need to check every line, which it is not simple. Additionally, we might forget one of the comment symbols just in one line, this will give an error if it is not a code, or a part of code (maybe variable declarations happened before and are in the comment).

2. The first and the main reason why Java does not allow pointers is **security**. By knowing memory location, you can traverse through memory locations and retrieve data, even the secured data (private variables) can be accessed this way. This is what the hackers might take advantage of. That is what makes C/C++ **unreliable**.

The second reason is the usage of pointers usually ends up with bugs and they can be confusing, specifically, for beginners. "Cost" indicated the training of the programmers and I believe, the usage of pointers increases the cost of programming language.

Another reason might the type *pointer_name. * has different meanings in C. Other than the declaration of pointers, it is used for multiplication operations. This decreases the Readability in Language valuation Criteria, specifically, simplicity.

3. **Scala** – It is a Java-like programming language, it is the combination of functional and object-oriented programming. It runs Java Virtual Machine and Scala was intended to work flawlessly with Java. It is rather created based on the criticisms of Java though. Its classes can call Java methods and create Java methods. I would say, Scala falls under Java (or Under Java and JavaScript) in the genealogy.

F# - it is a multi-cross, functional-first programming language which include imperative, functional and object-oriented programming languages. It was developed by Microsoft and it is a part of .Net family. It is thoroughly utilized for Machine Learning projects and had been impacted by Python, C#, Haskell, Scala and Earlang. In my opinion, F# falls under C# and Python.

Go – It is statically-typed, compiled programming language developed by Google. It has been impacted a lot by C++ and Assembly language. Its syntax is similar with C, but with support of garbage collector and structural typing. So I believe, it falls under C++ in the genealogy.

Dart – It is another programming language developed by Google. It is object-oriented and supports garbage collector as well. It's usually utilized for server and browsers. It's more for building User Interfaces for fast applications, therefore, called client optimized. Dart can either compile to native code or JavaScript. It has been influenced by C, Java, JavaScript, Kotlin, Ruby. Even its syntax is more like C. I believe, it falls under Algol category, specifically, after under the JavaScript and C++.

References:

1. https://way2java.com/java-general/why-java-does-not-support-pointers/
2. https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/
3. https://www.scala-lang.org
4. https://scala-lang.org/files/archive/spec/2.13/
5. https://fsharp.org
6. https://golang.org/