CS131 Homework6 Summary

1 Introduction

In this summary, we briefly analyze various language aspects of Dart compared to other languages such as Ocaml, Java, Python. Then we determine that whether this language is suitable for writing the interface of our new garage sale application – 'GarageGarner' – which helps the user find best deals in a garage sale using machine learning model with TensorFlow Lite.

2 Evaluating Dart features

In this section, we will discuss different language aspects of Dart and compare it with other languages, such as Ocaml, Java and Python. Also, we will comment on its impact on the quality of our application.

2.1 Language Type

Dart is an Object-Oriented Language, and every variable in Dart is an object. Like Java and Python, this feature would be convenient for us to organize code and write complex applications.

2.2 Typing

Like Ocaml, Dart is statically typed and uses type inference. This is also an advantage since type checking is enforced at compile time, and developers does not need to explicitly write the type of a variable.

2.3 Garbage Collection

Like Java, Dart has a garbage collector and uses the generational approach that is very efficient. This provides the convenience of garbage collection without negatively impact the performance of our application.

2.4 Asynchronous Programming

Dart supports Asyncio, and to perform asynchronous operations in Dart, we can use the Future class and the async/await keywords. This is like what Python provides with its Asyncio Library. Support for asynchronous programming would greatly improve the performance of our application written in Dart, as our application would be constantly waiting for user inputs.

2.5 Multithreading

Like Python, Dart is based on single-threaded model. However, Dart has 'isolates', which is considered as threads in Dart, but they resemble processes in other languages. This is because isolates do not share memory and communicates through ports and messages. Isolates could run in parallel and utilize many CPU cores.

This approach toward multithreading is between what Java and Python provides. In Java, a program can spawn multiple threads that run in true parallelism, whereas in Python, a program is always single-threaded. This approach of using isolates has some benefits. Since each isolate has its own memory, memory allocation and garbage collection does not need to be locked, thus greatly improve the performance of these operations.

3 Language Aspects

In this section, we will discuss five important language aspects, namely ease of use, flexibility, generality, performance, reliability, in the case of Dart.

3.1 Ease of use

Dart has several features that makes the program easy to use. It is Object-Oriented, uses type inference, and has garbage collection. Hence this language would be relatively easy to learn for people with prior experience in Python, Java or C/C++. However, considering that it is a relatively new language, its documentation and community support might not be as resourceful as that of Python and Java. But as a popular language for mobile development, the community size is not small either.

3.2 Flexibility

Dart has many convenient features, but these features would sometimes limit the flexibility of the program. For example, since Dart has garbage collection, the programmer cannot control when to free a dynamically allocated object. Also, the absence of pointers means that programmers cannot modify any part of the memory arbitrarily.

3.3 Generality

As a programming language mainly used for creating UI, Dart is optimized for the needs of user interface creation. For example, it has ‘hot reload’, which helps programmer to see the changes in code on the resulting UI directly. Hence Dart would fit perfectly with the development of our application, but compared to languages like Java and Python, which are used in many different fields, Dart lacks general optimization for other uses.

3.4 Performance

Like Python, Dart is based on single-threaded model, which may put some constraint on the performance of our application. However, it is fully optimized for event-driven code with async/await, which improves the efficiency of our application. Also, the ability of creating multiple isolates would mean that our application could run in true parallelism.

3.5 Reliability

Dart is a statically typed language with type inference. This means that not only does Dart have the same convenience in declaring variables as Python, it also performs more thorough type checking at compile time, and this in turn makes our application more reliable. Also, Dart has the same syntax as Java to handle errors and exceptions, that is using the ‘Try…Catch…Finally…’ block, so that they can be handled without affecting the whole program.

4 Conclusion

As a language that specializes for creating user interface, Dart’s language features such as Object-Oriented programming, type inference and garbage collection would make developing our application easier. Dart also supports asynchronous programming, which is crucial for mobile app, since the application is constantly waiting for user input. The single-threaded model of Dart might become a bottleneck for our application, but we can take advantage of isolates to speed up our application. In summary, Dart’s language features fit with the development of our mobile application and would support the functionalities well.

5 References

A tour of the Dart language:

https://dart.dev/guides/language/language-tour

Asynchronous programming: futures, async, await:

https://dart.dev/codelabs/async-await

Why Flutter Uses Dart:

https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf

Flutter: Don’t Fear the Garbage Collector:

https://medium.com/flutter/flutter-dont-fear-the-garbage-collector-d69b3ff1ca30