A comparison of C++ and Java.

Dominic Rathbone

March 5, 2016

0.1 Introduction

The aim of this report is to compare and contrast the programming languages, C++ and Java. This comparison will be based around a number of aspects that define a programming language such as it's type system, how it is compiled and the paradigm's it utilises.

0.2 History

C++ was developed by a Computer Scientist named Bjarne Stroustrup, arising from an early project of his, "C with classes". The intention of this was to produce a superset of the C language that supported the object oreinted . His inspiration for this came from his experience with the programming language, Simula 67 which exposed him to the object-oriented programming paradigm. Eventually, in 1983, C With Classes evolved into C++ with a new feature set being added. [1]

On the other hand, Java was produced at Sun Microsystems (Now, Oracle) in secret, by team known as the "Green Team" with the intentions of creating a modern alternative to C++ which was the most popular language at the time. The idea was spawned out of the green team's frustations with the C/C++ APIs that Sun had been using. It aimed to retain most of the feature set that C++ had whilst removing the unnecessary, outdated parts. During the process of becoming Java, it went through two iterations of names, "C++ ++–" and then Oak.[2]

0.3 Paradigms

As mentioned above, both C++ and Java are languages that support the object oriented paradigm. This concept describes how a system can be modelled as structures known as "objects" that contain the data that describe them as well as the operations that manipulate and use this data. This is based on the imperative programming paradigm, where a language has the ability to control the flow and state of a system.

0.4 Typing

Java is statically typed the code is type-checked at compilation time and strongly typed (relative to other languages) as a new variable declared as one type cannot be initialised or changed to be another type. For example, if a new integer is declared and initialised as a string, the compiler will throw an error as the types are mismatched. This is in contrast to dynamic typing where type-checking happens at runtime. Whilst static typing restricts what a programmer can do with a variable, it has the obvious benefit of preventing potentially serious errors from occuring.

Although C++ is statically typed, it can be said to be weakly typed as the compiler does not prevent some errors from occuring and thus they can occur at runtime.

Both languages contain the notion of primitive types which are basic, predefined types such as integers or booleans which have the purpose of acting as building blocks within the language.

0.5 Memory Management

Java Garbage Collection C++ Pointers

0.6 Compilation

0.7 Concurrency

Preventing the corruption of shared data between concurrently executing threads/ tasks. Synchronisation of threads/ tasks to prevent race conditions. Can the user influence the priority of threads/ tasks

0.8 Error Handling

0.9 Reflection

Bibliography

[1] www.cplusplus.com/info/history/

[2] https://www.santarosa.edu/~dpearson/mirrored_pages/java.sun.com/Java_Technology_-_An_early_history.pdf

Appendix A

```
//this will cause a compilation error
// as "hello" isn't of type int.
int a = 1;
a = "hello";
```

```
//this will cause a compilation error
//as the variable isn't declared.
a = 1;
```