ISSN: 2321-7782 (Online)
Volume 3, Issue 3, March 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Comparison between Object Oriented Programming Languages: Java and C++

Piyush Sudhakarrao Gondchawar
Computer Science & Engineering
Prof. Ram Meghe Institute of Technology & Research, Badnera
Amravati, India

Abstract: This paper construct a comparison between two main software's used in programming applications that are Java and C++, the comparison operation includes the time needed to perform some algorithm i.e. speed of operation, flexibility to adjusting some code, and efficiency. The same code is used to compare between the two software to determine which one is better. It is found that C++ needs less time to execute the same code comparing with Java. Java needs about 10% excess time to execute the same code segment comparing to C++.

Keywords: Java; C++; Algorithm Speed; Code Flexibility.

I. INTRODUCTION

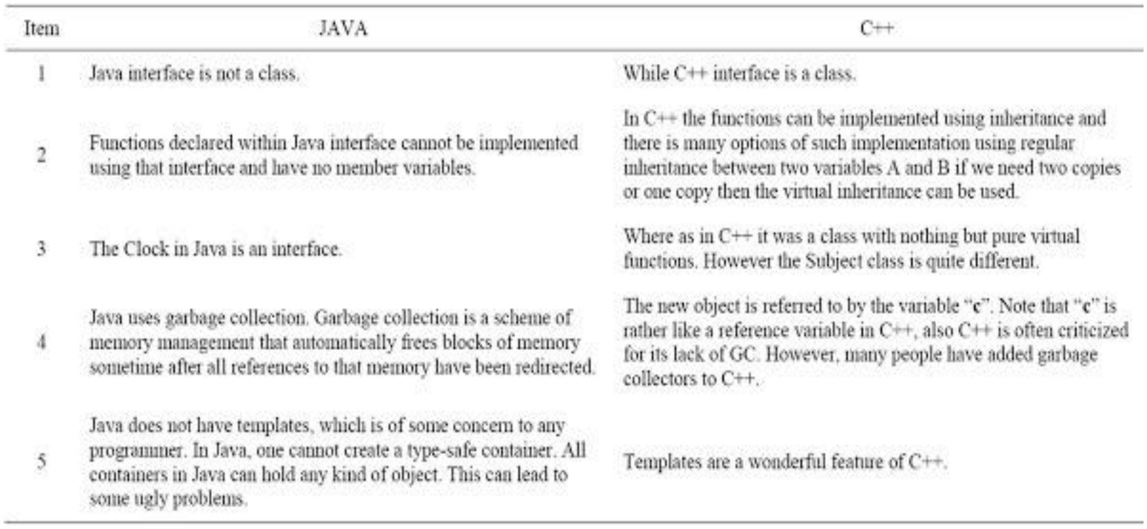
Java and C++ are the most used languages in program-ming for most of programmers and system designers. Java has a structure called an "Interface". Java interface is almost identical to a C++ class that has nothing but pure virtual functions. Inherent in java is not efficient from more than one base class; even if the base classes have nothing but abstract methods or pure virtual func-tions. The differences between Java and C++ can be summarized as in **Table 1**.

The time needed to execute some code, algorithm, program or complete system program is considered criti-cal in any programming language, in this paper the time needed to execute some same code in both Java and C++ is used to compare between such two languages. The minimum time needed to execution is an advantage be-cause it reflects how much is the language is powerful and efficient. The minimum time of execution means more speed of execution which is the main goal of any designer or programmer.

Many researches and studies discussed this issue, Lutz Prechelt, (1999), discussed the relative efficiency of Java programs, in particular in comparison to well established implementation languages such as C or C++. Java is of-ten considered very slow and memory-intensive. Most benchmarks, however compare only a single implemen-tation of a program in, say, C++ to one implementation in Java, neglecting the possibility that alternative imple-mentations might compare differently. In contrast, the current article presents a comparison of 40 different im-plementations of the same program, written by 40 dif-ferent programmers. The inter-personal program differ-ences are larger than those between the languages and the performance gap between Java and other languages is still shrinking rapidly, [1].

Peter Sestoft, 2010, they compare the numeric per-formance of C, C# and Java on three small cases. Man-aged languages such as C# and Java are easier and safer to use than traditional languages such as C or C++ when manipulating dynamic data structures, graphical user interfaces, and so on, [2]. Dirk E. *et al.* (2011), discussed the RC++ package simplifies integrating C++ code with R. It provides a consistent C++ class hierarchy that maps various types of R objects (vectors, matrices, functions, environments, ...) to dedicated C++ classes. Object in-terchange between R and C++ is managed by simple, flexible and extensible concepts which include broad support for C++ Standard Template Library idioms. C++ code can be compiled, linked and loaded on the y, or added via packages. Flexible error and exception code handling is provided. RC++ substantially lowers the bar-rier for programmers wanting to combine C++ code with R. [3,6].

Table 1. Differences between Java and C++



Item	JAVA	C++
1	Java interface is not a class.	While C++ interface is a class.
2	Functions declared within Java interface cannot be implemented using that interface and have no member variables.	In C++ the functions can be implemented using inheritance and there is many options of such implementation using regular inheritance between two variables A and B if we need two copies or one copy then the virtual inheritance can be used.
3	The Clock in Java is an interface.	Where as in C++ it was a class with nothing but pure virtual functions. However the Subject class is quite different.
4	Java uses garbage collection. Garbage collection is a scheme of memory management that automatically frees blocks of memory sometime after all references to that memory have been redirected.	The new object is referred to by the variable "c". Note that "c" is rather like a reference variable in C++, also C++ is often criticized for its lack of GC. However, many people have added garbage collectors to C++.
5	Java does not have templates, which is of some concern to any programmer. In Java, one cannot create a type-safe container. All containers in Java can hold any kind of object. This can lead to some ugly problems.	Templates are a wonderful feature of C++.

Michi H. *et al.* 2012, compared between Windows Communication Foundation and Java: Remote Method Invocation which are currently seen as major contenders in the middleware space, performance is often taken as the sole evaluation criterion, despite the fact that per-formance is only one of many factors that influence the choice of middleware. They provided a performance and scalability comparison of the three middleware platforms, and discussed when performance and scalability matter and when they do not, including their likely impact alongside other factors on the overall cost of a project. Finally, for those applications that indeed require high performance and scalability, the article points out a few techniques you can use to get the biggest bang for your buck, [7].

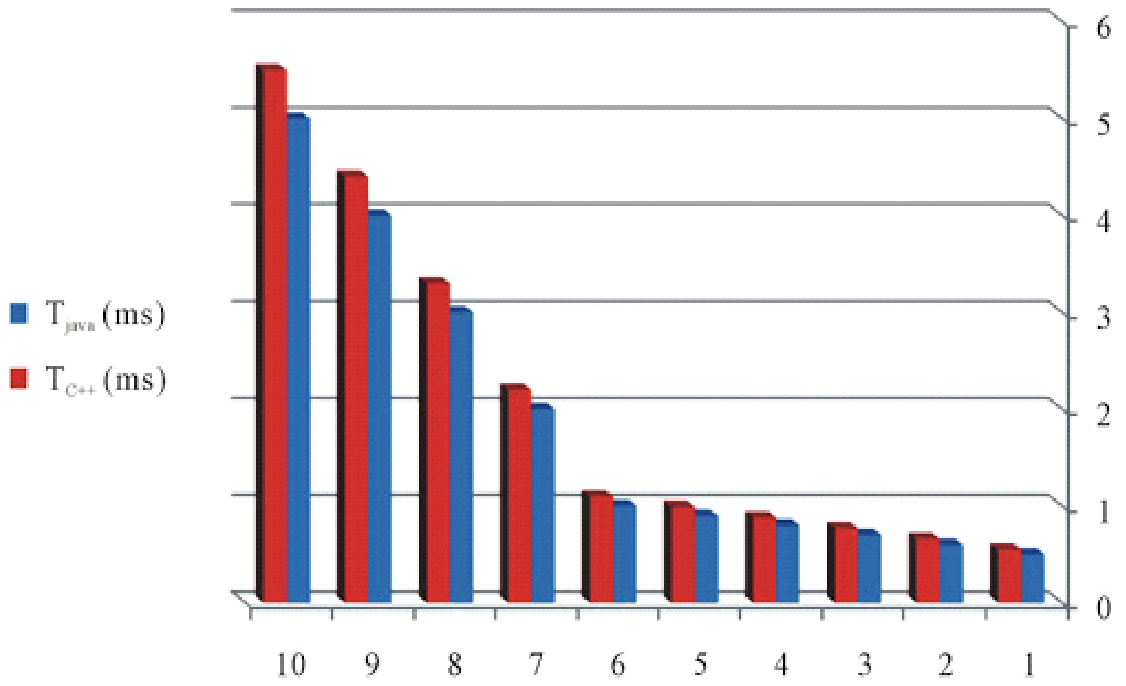


Fig. Comparing the execution time of JAVA and C++

Time Comparing between Java and C++ Software

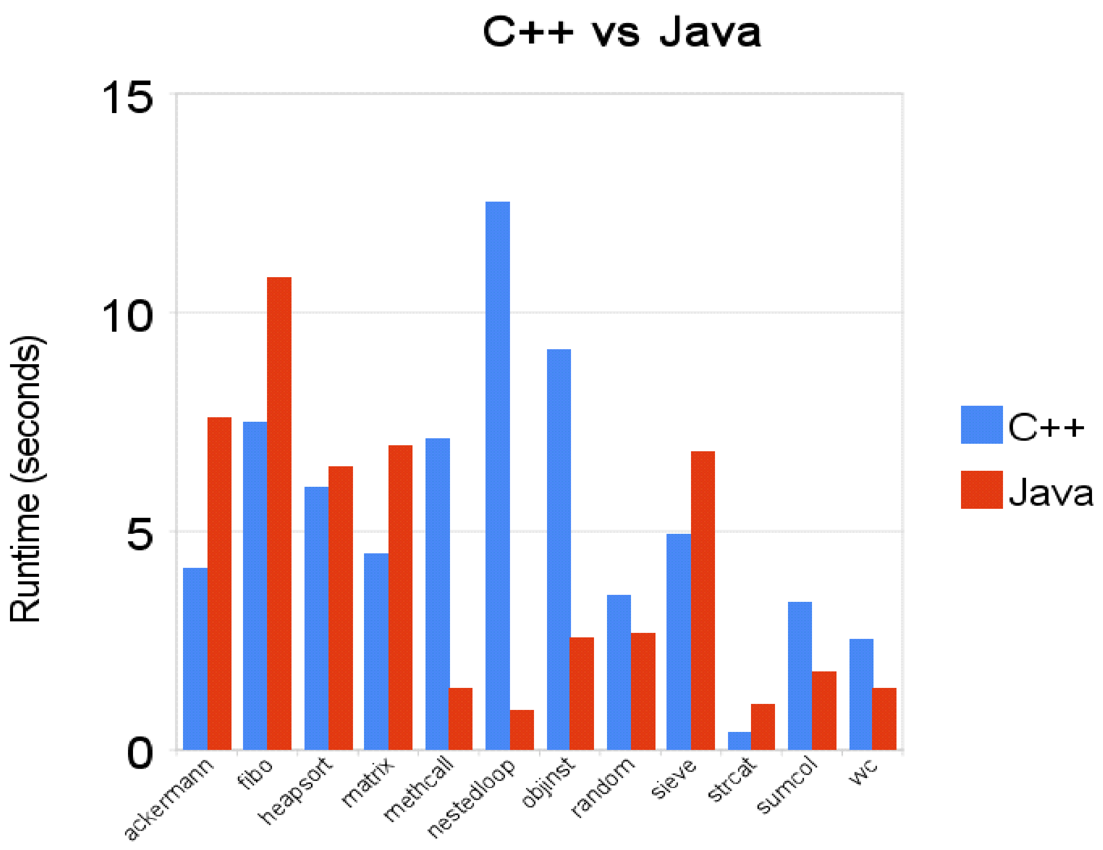


Figure 2. JAVA Runtime Vs C++

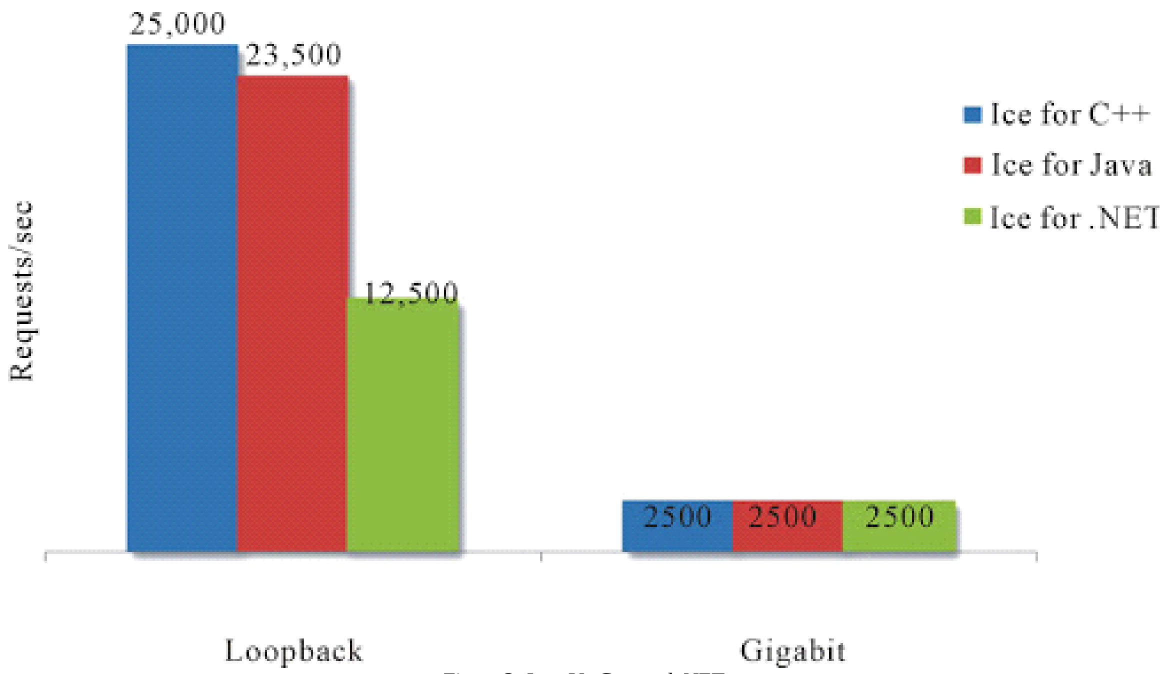


Figure 3. Java Vs C++ and .NET

II. Results and Discussion

From last comparing between Java and C++, the main thing that may reveal which software is better is the time to execute the same algorithm. So as an example con- sider the following Java code:

```
public class RealTime
{
    public void Do()//must complete in 500 µs
    {
        Clock c = new Clock; //might collect!
        // diddle with clock for 100 µs
    }
}
```

If this code is executed using Java and C++, Java takes 500 µs to be complete such algorithm. This is a typical constraint in a hard real time system. Those functions that call {Real Time. Do()}depend on the fact that it will take no longer than 500 µs to execute. While the same function takes just 450 micro seconds to be executed using C++. The goal of the experiment is to measure the time needed to execute the same code on both Java and C++.

Figure 1 show a time of execution comparison between Java and C++. It is clear that C++ is faster than Java which can be represented as:

TJava = 1.10* Tc++

TJava: time needed to execute some given code using Java SW.

TC++: time needed to execute some given code using C++ SW.

To compare the current study with previous studies, **Figure 2** comparing Java runtime for different algorithms with that of C++, it is clear that the Java runtime is more in almost algorithms than that of C++.

Figure 3 comparing between Java requests/sec and that for C++ and .NET, it is clear that the C++ requests per unit of time is more which told us that C++ is more efficient.

III. Conclusion

Java is a powerful language. While C++ has a relatively easy time to be learned, and will find that the programmers enjoy using it. It is noted that a few problems with the language in the above discussion. Language design always involves some disadvantages or shortcomings that displease someone. C++ is an interesting language that enables us to write codes easily with more flexibility and with little time needed to execute some code com- paring to Java.

References

1. P. Lutz "Comparing Java vs C/C++ Efficiency Differences to Inter-Personal Differences," Communications of the ACM, Faculty at fur Informatic University, Karlsruhe, 1999.
2. S. Peter, "Numeric Performance in C, C# and Java Peter Sestoft," IT University of Copenhagen, Copenhagen, 2010.
3. E. Dirk and F. Romain, "Rcpp: Seamless R and C++ Integration," 2010. http://dirk.eddelbuettel.com/
4. W. Peng, M. Sam, M. J. Moreira and G. Manish, "Effi-cient Support for Complex Numbers in Java," ACM 1999 Java Grande Conference, San Francisco, 12-14 June 1999, pp. 109-118.
5. J. E. Moreira, S. P. Midkiff, M. Gupta, P. V. Artigas, M. Snir and R. D. Lawrence, "Java Programming for High Performance Numerical Computing," IBM Systems Jour-nal, Vol. 39, No. 1, 2000, pp. 21-56. doi.org/10.1147/sj.391.0021

6. J. Glenn, C. M. Clement, S. Q. Snell and G. Vladimir, “Design Issues for Efficient Implementation of MPI in Java,” ACM 1999 Java Grande Conference, San Fran-cisco, 12-14 June 1999, pp. 58-65.

7. H. Michi and S. Mark, “Choosing Middleware: Why Performance and Scalability Do (and Do Not) Matter,” ZeroC, Inc., Palm Beach Gardens, 2011.

AUTHOR(S) PROFILE

Piyush Sudhakarrao Gondchawar, appeared for B.E. 3rd year in Computer Science & Engineering Department from Prof. Ram Meghe Institute of Technology & Research, Badnera. He had published papers in the field of Software’s like Networking, Coding Languages and Operating System etc. in International Journals along with that he had presented a ORAL Presentations in National Level Conferences and attended a National Level Conference on Developing Frontiers of Physics, Astronomy and Space Sciences held in Shri R.L.T. College of Science. Also, he is serving in Social Organizations. He had played State Levels in the sports like Table-Tennis, Squash and attended the V.C.A. camp in Cricket and now representing the Cricket Team of the College.