Java Programming Language: Time Permanence Comparison with Other Languages: A Review

Rebin Abdulkareem Hamaamin[1,*], *Omar* Mohammed Amin Ali[2], *Shahab* Wahhab Kareem[3]

[1]Computer Science, College of Sciences, Charmo University, Chamchamal, Sulaimani, KRG, Iraq.
[2] Department of IT, Chamchamal Technical Institute, Sulaimani Polytechnic University,Chamchamal, Sulaimani, KRG, Iraq.
[3]Department of Technical Information Systems Engineering, Technical Engineering College, Erbil Polytechnic University Erbil, KRG, Iraq.

Abstract. Programming languages are necessary tools for teaching computer programming languages, so it is no surprise that deciding which one is the "right tool for the job" can be a contentious issue. To build solid foundations for such debates, we must first consider how different languages' features relate to one another. Many Java programmers do not think about how to make their programs run faster. Application developers should tune their applications before putting them into production. Application code tuning often results in significant performance gains. The proposed model intended to assist Java programmers in fine-tuning and improving Java -based applications. This article review work in time, the time required to execute some algorithm, i.e., speed of operation, flexibility in modifying some code, and performance, are all factors in the comparison operation. To evaluate the program is better; the same code used to compare the two. in this review, researcher work in terms of the time compare in Java with other language, before compare other language, show performance Java in time and runtime Java, after select some language to compare with Java, like C language, C++ language, C# language and Python, to compare Java performance with other language and illustrate best execution time performance. Comparing Java with other languages reveals that the latest JDK runs faster and executes tasks more swiftly than older versions of Java JDK.

1. Introduction

In April 1995, first encountered the Java programming language (JPL). In Amsterdam for a conference and was debating appropriate first programming languages with a group of American academics over breakfast. They discussed Java, a new language that was similar to C++ but had many of the "dirty" sections removed and some of the nice parts of languages as if Smalltalk added [1]. The Java white paper was obtained and read. I thought it looked good but, like other object-oriented languages before it, I did not think it would catch on. After all, though many people thought Smalltalk was nicer than C++, C++ still reigned supreme, and languages like Eiffel and Self were hardly used at all [2].

* Corresponding author: rebin.abdulkarim@chu.edu.iq

Many Java developers disagree about the development of the JDK Java platform's features. JDK stands for Java Development Kit. Using the Java programming language to create applications, applets, and components necessitates the use of JDK, which is a growing environment for them. Programs written in the Java programming language and running on the Java platform can be developed and tested by tools are involved in the JDK [3]. JDK comes in a variety of versions, such as JDK 5.0, 6.0, 7.0 and 8.0, which adds a range of new features to the Java programming language. Many developers nowadays use the Java programming language to build their projects. Java programmers spend about eight hours a day working with the language five days a week. Improvements in the functionality of the Java platform have an influence on both experienced programmers and newcomers to the language [4]. Java can be interpreted from many angles, which distinguishes it from many other programming languages, which can only be viewed as a programming language. Java, on the other hand, is more than a programming language [5]. We will look at a few different ways to categorize Java; Object Oriented Programming (OOP) is a programming paradigm that uses classes and objects. It is used to break down a software program into interchangeable code blueprints (called classes) that can be used to build individual instances of items. Object-oriented programming languages include JavaScript, C++, Java, and Python, to name a few [6]. User referring to the existence of system-provided objects rather than a specific development environment when I say programming environment. Unlike several other languages (including C++), Java has a built-in debugger. It has a broad (and standard) set of classes associated with it. These classes make Java extremely strong while also encouraging a consistent development style. Rather than progressing, you spend the majority of your time expanding the "method." spend most of your time extending the "system" rather than programming from scratch [7]. An operational setting The Java Virtual Machine is the operating environment in which all Java programs run. This is Java's personal computer, which has been ported to a number of different platforms and operating systems. As a result, Java can run on a variety of systems (without recompilation) [8]. This a non-garbage collected memory area and is one of the resource management constructs of the Real-Time Specification for Java (RTSJ). Programmers may directly control memory, for example, the nest memory regions, assign objects to those areas and free the region when not required by using a set of servant memory scopes found in the [RTSJ] [9]. There are two more types of memories in this memory paradigm introduced by Instantaneous Java, which are, eternal memory, and the scoped memory. Though each object has its memory decaying after the time given to it, all the objects that are assigned to the everlasting memory survive throughout the total runtime of the program. It does not control these artefacts otherwise; rather, it searches the limitless memory for reference to the garbage-collected heap, and can even alter them. [5]. In this article, researcher tries to show performance Java in time and time to execute program or code in Java compare other language, this language as compare with Java like C language, C++ language and Python language [10].

2. Java programming language permanence

Fine tuning of Java is somewhat an art form inside a art form. Getting decent information about Java performance optimization is tremendously difficult because of several reasons.
First of all, the system which you are developing and the material and hardware on which the system is built are the main considerations in performance tuning. The underlying libraries, your system, the Java virtual machine, and indeed, the hardware on which everything runs evolve over time. With this progression is a progression in the Java performance methodologies germane to such a progression. Second, a lot of misinformation has been circulating to the Java developers about which Java Virtual Machine and the Java compiler.

Unfortunately, many java developers suppose that the java compiler or even the virtual machine (VM) will optimize this code better than you can do [11].

This isn't always entirely true though. Where it is located matters: core algorithms, data formats, data structures, memory allocation, and input/output allocations! You know considerably more about your system, about its use, data structures, cycles, and other details than the Java compiler and JVM well therefore in many cases you can do the optimization superiorly than can they [12].

While there are many distinct techniques for optimising performance in Java programming, they usually belong to one of the following categories:While there are many distinct techniques for optimising performance in Java programming, they usually belong to one of the following categories:

• Reduce the number of steps (activities) that can be possibly be undertaken in a given project or on a definite work.

• This level is called match code to hardware such as a CPU, RAM, SSD, etc. also called mechanically compassion.

• Try and divide work into parallelism wherever possible and, if possible [13].

To lessen the amount of effort (operations) necessary for the overall task, it is necessary to exercise a decreased amount of time on a specific task. In some of these cases it also requires knowledge of how Java works within and at a low level to the hardware. At times you can augment the speed of working by finding a better algorithm – faster algorithm, a more efficient data structure – compact data structure or a fast data structure etc.

Such things as setting variables on 8-byte addresses for example 0, 8, 16 and so on, or storing data on the same cache page, means that we store data in close seminars to benefit from the serial memory access, or minimizing branching to optimize / or replacing it are some of the ways you can align code with the underlying hardware. [4].

One of the best examples of coordinating code and data with the functioning of the underlying hardware is minimising the distance between CPU and data. Processing of the data is typically accelerated by reducing the distance between the CPU and the data it works with. It takes less time to access data from the CPU registers than it does from the L1 cache, L2 cache, L3 cache, main RAM, disc, and, most of the time, a distant computer [14].

In certain situations, dividing a work into smaller, parallel-executable tasks or just carrying out several separate activities at once can significantly increase performance. When jobs can be simply divided into smaller sections or are already independent of one another, parallelization can result in significant performance gains. This is because modern CPUs are designed to have more and more parallel execution units, or cores. Another way to accomplish parallelization would be to have several computers collaborate to solve a problem instead of just one.

The key performance in Java is The Just-In-Time (JIT) compiler increases the efficiency of Java™ programmes by translating bytecodes to native machine code during execution. Groups in Java programmes are platform-neutral bytecodes that may be interpreted by a Java Vector Machine (JVM) on a variety of computer architectures. The JVM loads the class files at runtime, specifies the semantics of each bytecode, and performs the necessary computations. A Java programme runs quicker because it uses more CPU and memory during the interpretation process. A Java programme executes slower than a native application due to the increased processing and memory utilisation during interpretation. [11].

By converting bytecodes into native machine code during runtime, the JIT compiler enhances the efficiency of Java programmes. The JIT compiler is enabled by default. Upon compiling a method, the Java Virtual Machine invokes the compiled code of the method without interpretation. If the Java programme didn't need to use CPU time and memory, compiling every method may potentially enable it to run at a pace that would be comparable to a native application. [15].

3. Compare time performance of JPL to other program languages

This comparison of Java with other languages is based on reading some articles over a period and determining which language is faster when compared to others for the same data or program.

3.1 Comparison between Java and C programming languages in time

C is a procedural, low level and a compiled language of computer programming. Java is a versatile language that is categorized as object-oriented, high level, and interpreted language. For example, procedure in C language is equivalent to object in Java language Java is based on objects while C is based on functions [16]. For the purpose of this comparison, I got to the decision of doing a matrix multiplication using both of these two languages. The matrices are 2048 x 2048 in size (that is 8,589,934,592 multiplication s and addition s each) and filled with random number s between 0. 0 and 1. 0 (The difference in performance can be considered as a slight negative effect of using random values instead of the same matrices for Two languages method). Currently the researcher performed each experiment five times and then averaged the running time.

C programming language Code

```
#include <stdlib.h>

#include <stdio.h>

#include <time.h> #define n 2048 double A[n][n];
double B[n][n]; double C[n][n]; int main() {

   //populate the matrices with random values between
0.0 and 1.0    for (int i = 0; i < n; i++) {        for (int j =
0; j < n; j++) {

        A[i][j] = (double) rand() / (double) RAND_MAX;

        B[i][j] = (double) rand() / (double) RAND_MAX;

        C[i][j] = 0;        } }

   struct timespec start, end;    double time_spent;
//matrix multiplication

   clock_gettime(CLOCK_REALTIME, &start);

   for (int i = 0; i < n; i++) {        for (int j = 0; j < n;
j++) {           for (int k = 0; k < n; k++) {
C[i][j] += A[i][k] * B[k][j];

        }   } }

   clock_gettime(CLOCK_REALTIME, &end);

   time_spent = (end.tv_sec - start.tv_sec) + (end.tv_nsec
- start.tv_nsec) / 1000000000.0;    printf("Elapsed time in
seconds: %f \n", time_spent);    return 0;

}
```

Java programming language Code

```
import Java.util.Random;

public class MatrixMultiplication {     static int n = 2048;     static double[][] A = new
 double[n][n];
    static double[][] B = new double[n][n];
   static double[][] C = new double[n][n];
   public static void main(String[] args) {
      //populate the matrices with random values between 0.0 and 1.0
 Random r = new Random();
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
A[i][j] = r.nextDouble();
            B[i][j] = r.nextDouble();
            C[i][j] = 0;
         }
      }
      long start = System.nanoTime();         //matrix multiplication
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
for (int k = 0; k < n; k++) {
 C[i][j] += A[i][k] * B[k][j];
            }   }   }

      long stop = System.nanoTime();
   double timeDiff = (stop - start) * 1e-9;
      System.out.println("Elapsed time in seconds: " + timeDiff);
   } }
```

After run two programs can take this result show in Table (1).

Table 1. Compare time between Java and C programming languages elapsed time (seconds)

Itération	C programming language	Java programming language
1	196.83	85.81
2	300.66	82.38
3	199.15	83.02
4	195.11	86.8
5	198.68	85.05
Average	198.86	84.612

Upon analysing the findings presented in Table1, it becomes evident that Java programs exhibit superior performance in terms of execution time when compared to their counterparts written in the C programming language. Across all metrics measured, Java programs consistently demonstrate faster execution times. This observation underscores the efficiency and optimization of Java code execution in various scenarios. Such comparative analysis provides valuable insights into the relative strengths and weaknesses of different

programming languages, aiding developers in making informed decisions regarding language selection for their projects. [17] [18] [19] [30]

Fig.1. Java Language and C languages program

3.2. Time comparison between Java and C++ programming languages

In this section reviewing two articles about compare time between Java a C++, first article [20] Used two applications, one in C++ and the other in Java, were created. Apart from a few minor syntactical and functional variations, they are almost identical. Each program loads three files with 1000, 10,000, and 100,000 randomly ordered integers, respectively. These files are pre-made and contain the same values every time. Before testing, they are created at random by another small application.
Every file is tested three times, one for each run. Quicksort is used to sort the data after it is loaded. The dataset file is used to reset the data, and insertion-sort is then used to sort it once more. Next, a file containing the sorted data is stored. Time spent loading, sorting, and storing the data is stored, and each test is executed 50 times in a lengthy loop. Between the C++ and Java applications, 300 tests are executed in total. Based on the results, Java is quicker overall than C++ and loads two of the three datasets more quickly.

The same idea was used to generate two different programmes, but one was written in Java and the other in C++. The goal of every application is to be as similar as possible, although

there are some minor changes in syntax that will be discussed below. Both programmes were written by the same highly skilled programmer in both languages.

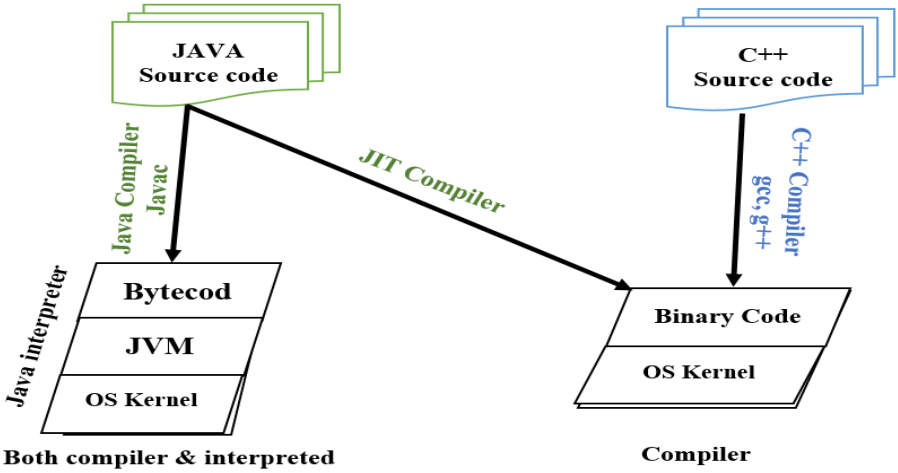

Fig.2. Compiled between Java and C++ programming languages

3.2.1. Methodology

This research compared two programming languages in terms of their ability to load, sort, and save increasing quantities of data using quantitative measurements. In both C++ and Java, two identical applications were developed. In both applications, loading, sorting, and saving are timed separately, and the results are then compared [20].

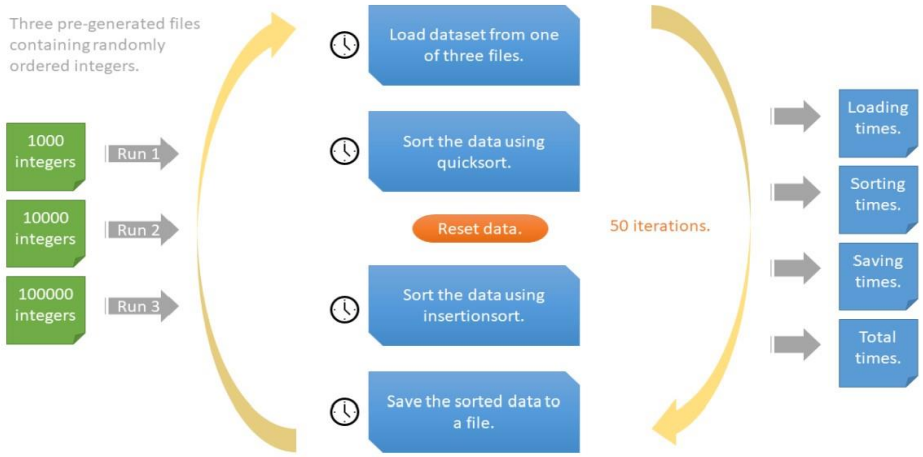

Fig.3. Methodology between Java and C++ programming languages [20].

3.2.2. Programming languages C++ and Java time libraries

The execution time of the C++ was measured using the ‘’c time’’ from the standard C++ c lib library. It contains the clock function and implements the clock type. These are then combined to come up with total figure of processor ticks right from the time the system was first started. The time of the start was subtracted from the time of the end, and the result gives the time taken to accomplish an event, for instance the loading of data. To do this, two times the current time was collected: Was involved in a flight landing or taking off once before the incident and once after the incident. The following is the excerpt how this was done precisely in C++ application code example below, considering one of the methods.

C++ programming language Code

```
start = std: clock();

loadFromFile(arr, SIZE, FILENAME);  duration = static_cast<long>(((std::clock() - start) /
(double)CLOCKS_PER_SEC) * 1000.0);
For the Java application, the standard “System” package was used which has a built-in time
function. To gather the time in milliseconds for a specific event, the
 “currentTimeMillis” function was used in the same way it was for C++. One measurement was
taken at the start of an event and one at the end, these values were then subtracted to get the
elapsed time. The exact way this was done for one of the functions is shown in the below example
from the Java application code.

  start = System.currentTimeMillis();  try

{ loadFromFile(FILENAME, arr, SIZE);

 } catch (IOException e1)

 { e1.printStackTrace();

 } duration = System.currentTimeMillis() - start;
```

Result in This Article: This Article Compares C++ And Java for all datasets in Loading and Sorting and Saving and Total time for all parts for each dataset, the result are shown in Table 2 and Table 3 below:

- C++ program language times

Table 2. C++ data to run program [20].

Dataset Size	Save	iSort	Qsort	Load	Total
1,000	1,64 Ms	0,1 Ms	0,04 Ms	0,9 Ms	**2,68** Ms
10,000	8,52 Ms	12,62 Ms	0,5 Ms	7,5 Ms	**29,14** Ms
100,000	16,94 Ms	1208,58 Ms	6,04 Ms	73,18 Ms	**1304,74** Ms

- Java program language time

Table 3. Java data to run program [20].

Dataset size	Save	iSort	Qsort	Load	Total
1,000	2,18 Ms	0,22 Ms	0,06 Ms	2,34 Ms	**4,8** Ms
10,000	8,08 Ms	12,48 Ms	0,76 Ms	6,44 Ms	**27,76** Ms
100,000	19,9 Ms	1216,06 Ms	7,66 Ms	51,28 Ms	**1294,9** Ms

In the last comparison in this article, java is faster, compared to C++ in Total time [20]. In [21] constructs a comparison between two major programming languages, Java and C++. The comparison operation involves the time required to execute some algorithms, i.e., speed of operation, flexibility in modifying some code, and performance. To evaluate whether the program is better, the same code is used to compare the two, which means the JDK used in this article old JDK compared to the first article because the resulting article says: Java is a powerful language. Even still, programmers prefer to utilise C++ even though it does not truly provide many implementation issues. Contractions indicate that there have been a few linguistic issues with the topic above. Every language design has some side effects or disadvantages that most people find objectionable. Compared to Java, which requires more time to run code, C++ is a really interesting language that gives us a raw experience of writing in a language with much more versatility.
In the final review between Java programming and C++ programming, can say that Java has more performance in C++ language for the last version in JDK and takes a longer time to execute the same program in Java language and C++ language [22] [23] [24].

Table 4. Program in Java and C++ programming languages

Java	C++
public class HelloWorld { *public static void main(String[] args) {* *System.out.println("Hello, World!");* *}}*	*#include <iostream>* *int main() {* *std::cout << "Hello, World!" << std::endl;* *return 0;}*

3.3. Time evaluation in Java and C# programming languages

The Self-Organizing Migrating Algorithm (SOMA), a very powerful evolutionary optimisation tool targeted at the same set of issues as genetic algorithms, is one of the two algorithms used for the comparison of the Java and C# programming languages. Two implementations of the algorithm were developed: one for the C# framework and the other for the Java framework. Our goal is to ascertain these two concurrent implementations of the algorithm's calculation time efficiency.
The main objective of this paper was therefore to analyze and compare two cases (Jurva and C Sharp). Table 4 synthesises the findings in the study. The Table proves that, by generating

evaluation results of 10 conditions, the asynchronous parallel SOMA developed in C# has higher performance in 7 conditions out of ten. For example, it was than twice as consequent in the case of Michaelewicz test function which is the most time consuming for the employed benchmark function. In contrast, only three examples illustrated that the SOMA asynchronously created in Java was somewhat faster than the previous one – the parallel. [25] [26].

Table 5. Compare Java and C# has better computational time

Evaluate the Function	**Java Single Thread Time [S]**	**Java Two Threads (Time [S])**	**C# Single Thread Time [S]**	**C# Two Threads (Time [S])**
Ackley	77.89	40.20	43.21	**21.81**
EggHolder	21.73	11.12	12.59	**6.69**
Griewangk	10.73	**5.93**	17.45	9.07
Masters	45.37	23.32	33.30	**16.67**
Michalewicz	92.62	46.80	42.81	**21.61**
Rana	37.98	19.36	20.60	**10.29**
Rastrigin	11.91	**6.74**	15.27	7.74
Rosenbrock	12.02	**11.16**	28,25	14.32
Schwefel	11.20	6.15	9,01	**4.60**
Sine Wave	84.96	43.57	61,93	**31.15**

3.4. The execution times of the programming languages Python and Java

This study compares Java and Python programming in terms of term time. To do this comparison, the identical programme was developed, ran, and timed in both Python and Java. The comparison's results are displayed in Table 6.

Table 6. Comparison between Java and Python programming languages in term time

Language Used for Programming	**Time in seconds**
Python Language	11.00
Java Language	19.00

Table 6 indicates that the Python programming language operates at a quicker speed than the Java programming language. This contrast of the two applications (Java JDK 6.0.50.13 and Python 2.7) utilised for compassion [27].

However, companies truly worry about the application's speed or performance when it comes to development, especially with the latest JDK version in Python compared to Java. Because Java comes with the Java Virtual Machine (JVM), code execution in Java is faster and the language is compiled. However, Python is slower than other languages since it is an interpreted language and decides the type of data at runtime. [28].

Despite Java being compiled language it runs faster than Python speed wise. It is an interpreted language which determines the class of data at the time of execution and there is a lack of automatic garbage collector that makes python slower than java. So, Java is faster than other languages like Python in terms of execution of a code. Discussed Below Are Some of the Basic Distinctions Between Python and Java Which are summarized in Table 6.

Table 7. General different between Java and Python [27] [28].

Java program	Python program
Compiler languages like Java	An Interpreted Language is Python.
The language used for programming is object-oriented.	It is a language for scripting.
Java uses statically typed input.	Python uses dynamic typing.

In this final, comparison Java and Python in new version Java faster time compare Python Language because Java is a compiled language but Python is an Interpreted Language [29].

Table 8. Write Program in Java and Python

	
public class HelloWorld { *public static void main(String[] args) {* *System.out.println("Hello, World!");* *}}*	*print("Hello, World!")*

4. Conclusion

Programming languages are used to regulate the operations of computer systems. Numerous programming languages are available now, and more are being created every day. These programming languages gain popularity among various programmers because there are frequently trade-offs between performance, convenience of learning and usage, and speaking power. This assessment starts out with demonstrating Java's performance in real time, which requires CPU time and memory to be used for JIT compilation. When the JVM boots up, thousands of methods are named. Compiling all of these techniques will have a significant impact on start-up time, even if the software eventually reaches extremely excellent peak efficiency. Second part is comparison between Java and C Language after used the specify matrix multiplication in each language after run program get the result time for each program, this time Java less compare C language to run program. Another comparison Java with C++ language, in a final comparison of Java programming language and C++ programming language, it can be concluded that Java has superior performance in the C++ language with the most recent version of the JDK and takes less time to run the same program in both languages. And in this article review Java language with C# language in term of time, for Self-Organizing Migrating Algorithm (SOMA), after recommend all time in specify table, can get same case Java is batter mean less time and some case C# is less time. Last

comparison Java with Python language, Java is faster compared than Python. After all, when comparing java to other languages, one can see that the new JDK is quicker and takes less time to execute, while some older versions of JDK in Java take longer to execute. Apart from time, Java programs have several advantages that can be used to choose Java over other languages. This article reviews work on Java output in time and compares it to other languages. Languages define a program or algorithm and run it in Java and other languages to see how long it takes. This analysis also shows other reviews in work to compare Java with other languages, demonstrating which language takes less time to run the program or algorithm.

5. Future Work

For future studies, we urge that other researchers utilise the most recent JDK versions to ensure consistency and relevance in comparing analysis. This will allow for fair comparisons of Java to other programming languages, ensuring that each is assessed based on the most current improvements. This technique will give useful insights on Java's strengths and shortcomings relative to other languages, allowing for more informed technological decisions.

Table 9. Systematic review on (time performance evaluation in Java programming languages)

Year	Author	Title	Journal / Publish	Type	Approach	Limitations (Problem)	Future Work	Remark
2002	Jos'e E. Moreiraa, Samuel P. Midkiffa, Manish Guptiaa, PengWua, George Almasia and Pedro Artigasb	NINJA: Java for high performance numerical computing	Scientific Programming 10 (2002) IOS Press	Article				
2015	Monica, N. Agu.1 and Blessing, O. Ogbuokiri.2, Benedette, O. Okwume.3	Comparison of python and java for use in instruction in first course in computer programming	Google Scholar	Article	Run Program in Java and Python	Python is better than Java		Python 2.7 and Java (dk 6.0.50.13)
2019	Suraj Sharma	Performance comparison of Java and C++	Faculty of Computing, Blekinge Institute of Technology, 37179 Karlskrona, Sweden	thesis	1-Constructing two identical applications for C++ and Java 2-Loading three increasingly large datasets into the applications 3-Running the applications and gathering and comparing the timings for each dataset	there are not many studies that compare library functions from different languages as was done in this study.	comparisons between similar languages	
2015	Vamsi Krishna Myalapalli and Sunitha Gelothi	High Performance JAVA Programming	2015 International Conference on Pervasive Computing (ICPC)	Article	elucidates miscellaneous techniques to escalate JAVA application program performance and can serve as an optimization tool for the JAVA application programmers.		We extend this work by introducing a tool which optimizes statements on fly, even before it is processed by Java Virtual Machine	
2014	Jan Kolek*, Roman Jasek	A Time Performance Evaluation of the Soma Asynchronous Parallel Distribution in Java and C#	25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM 2014	Article	implementations of the Self-Organizing Migrating Algorithm (SOMA).		we are going to suggest an experiment to prove this hypothesis.	
2012	As'ad Mahmoud Alnaser1, Omar AlHeyasat2, Ashraf Abdel-Karim Abu-Ein2, Hazem (Moh'd Said) Hamaleh1, Ahmed A. M. Sharadqeh2	Time Comparing between Java and C++ Software	Journal of Software Engineering and Applications, 2012, 5, 630-633 http://dx.doi.org/10.4236/jsea.2012.58072 Published Online August 2012 (http://www.SciRP.org/journal/jsea)	Article	the comparison operation includes the time needed to perform some algorithm i.e. speed of operation, flexibility to adjusting some code, and efficiency.			C++ is an interesting language that enable us to write codes easily with more flexibility and with little time needed to execute some code com- nation to Java

References

1. G. Bracha, M. Odersky, D. Stoutamire, and P. Wadler, "Making the future safe for the past," *ACM SIGPLAN Not.*, vol. **33**, no. 10, pp. 183–200, 1998, doi: 10.1145/286942.286957.
2. A. B. History, "of Time," no. April 1995, pp. 49–50, 1998.
3. J. M. Abdullah, M. A. Mohammed, and D. A. Muhammed, "International Journal of Multidisciplinary and Current Research Java 8 New Features Improvements and Complications," no. March, 2020, [Online]. Available: http://ijmcr.com.
4. T. Cramer, R. Friedman, T. Miller, D. Seberger, R. Wilson, and M. Wolczko, "Compiling Java just in time," *IEEE Micro*, vol. **17**, no. 3, pp. 36–43, 1997, doi: 10.1109/40.591653.
5. W. S. Beebee and M. Rinard, "An implementation of scoped memory for real-time java," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. **2211**, pp. 289–305, 2001, doi: 10.1007/3-540-45449-7_21.
6. A. Miyoshi, T. Kitayama, and H. Tokuda, "Implementation and evaluation of real-time Java threads," *Proc. - Real-Time Syst. Symp.*, pp. 166–175, 1997, doi: 10.1109/real.1997.641279.
7. K. Nilsen, "Real-time programming with Java technologies," Proc. - 4th IEEE Int. Symp. Object- Oriented Real-Time Distrib. Comput. ISORC 2001, pp. 5–12, 2001, doi: 10.1109/ISORC.2001.922812.
8. D. Theriault, "Issues in the Design and Implementation of Act 2," pp. 1–28, 1983.
9. C. Andreae, Y. Coady, C. Gibbs, J. Noble, J. Vitek, and T. Zhao, "Scoped types and aspects for real-time Java memory management," *Real-Time Syst.*, vol. **37**, no. 1, pp. 1–44, 2007, doi: 10.1007/s11241-007-9024-3.
10. C. A. Lo, Y. T. Lin, and C. C. Wu, "Which programming language should students learn first ? A comparison of Java and python," *Proc. - 2015 Int. Conf. Learn. Teach. Comput. Eng. LaTiCE 2015*, pp. 225–226, 2015, doi: 10.1109/LaTiCE.2015.15.
11. J. E. Moreira, S. P. Midkiff, M. Gupta, P. Wu, G. Almasi, and P. Artigas, "NINJA: Java for high performance numerical computing," *Sci. Program.*, vol. **10**, no. 1, pp. 19–33, 2002, doi: 10.1155/2002/314103.
12. M. Harkema, D. Quartel, B. M. M. Gijsen, and R. D. Van der Mei, "Performance monitoring of Java applications," *Proc. Int. Work. Softw. Perform.*, pp. 114–127, 2002, doi: 10.1145/584369.584388.
13. N. Togashi and V. Klyuev, "Concurrency in Go and Java : Performance analysis," *ICIST 2014 - Proc. 2014 4th IEEE Int. Conf. Inf. Sci. Technol.*, pp. 213–216, 2014, doi: 10.1109/ICIST.2014.6920368.
14. R. Dimpsey, R. Arora, and K. Kuiper, "Java server performance : A case study of building efficient, scalable Jvms," *IBM Syst. J.*, vol. 39, no. 1, pp. 151–174, 2000, doi: 10.1147/sj.391.0151.
15. J. Programming, "GSJ : Volume 8, Issue 5, May 2020, Online : ISSN 2320-9186," vol. 8, no. 5, pp. 1899–1913, 2020.
16. A. Shafi, B. Carpenter, M. Baker, and A. Hussain, "A comparative study of Java and C performance in two large-scale parallel applications," *Concurrency Computation Practice and Experience*, vol. 21, no. 15. pp. 1882–1906, 2009, doi: 10.1002/cpe.1416.

17. Brihadiswaren, G. "A performance comparison between C, Java, and Python." (2020).
18. J. Martin and H. A. Muller, “Strategies for migration from C to Java,” pp. 200–209, 2002, doi: 10.1109/.2001.914988.
19. G. P. Nikishkov, Y. G. Nikishkov, and V. V. Savchenko, “Comparison of C and Java performance in finite element computations,” *Comput. Struct.*, vol. 81, no. 24–25, pp. 2401–2408, 2003, doi: 10.1016/S0045-7949(03)00301-8.
20. S. Sharma, “Performance comparison of Java and C ++,” no. February, 2019.
21. A. M. Alnaser, O. AlHeyasat, A. A.-K. Abu-Ein, H. Hatamleh, and A. A. M. Sharadqeh, “Time Comparing between Java and C++ Software,” *J. Softw. Eng. Appl.*, vol. 05, no. 08, pp. 630–633, 2012, doi: 10.4236/jsea.2012.58072.
22. L. Gherardi, D. Brugali, and D. Comotti, “A Java vs. C++ performance evaluation: A 3D modeling benchmark,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7628 LNAI, pp. 161–172, 2012, doi: 10.1007/978-3-642-34327-8_17.
23. R. Vivanco and N. Pizzi, “Computational performance of Java and C++ in processing large biomedical datasets,” *Can. Conf. Electr. Comput. Eng.*, vol. 2, pp. 691–696, 2002, doi: 10.1109/ccece.2002.1013025.
24. R. A. Vivanco and N. J. Pizzi, “Scientific computing with Java and C++ : A case study using functional magnetic resonance neuroimages,” *Softw. - Pract. Exp.*, vol. 35, no. 3, pp. 237–254, 2005, doi: 10.1002/spe.633.
25. J. Kolek and R. Jasek, “A time performance evaluation of the SOMA asynchronous parallel distribution in Java and C#,” *Procedia Eng.*, vol. 100, no. January, pp. 1672–1677, 2015, doi: 10.1016/j.proeng.2015.01.542.
26. J. Bishop, R. N. Horspool, and B. Worrall, “Experience in integrating Java with C# and .NET,” *Concurr. Comput. Pract. Exp.*, vol. 17, no. 5-6 SPEC. ISS., pp. 663–680, 2005, doi: 10.1002/cpe.858.
27. B. Ogbuokiri, M. Agu, and O. B.O, “Comparison of python and java for use in instruction in first course in computer programming,” *Transylvanian Rev.*, vol. 24, no. 7, 2016.
28. G. Destefanis, M. Ortu, S. Porru, S. Swift, and M. Marchesi, “A statistical comparison of Java and python software metric properties,” *Proc. - 7th Int. Work. Emerg. Trends Softw. Metrics, WETSoM 2016*, pp. 22–28, 2016, doi :10.1145/2897695.2897697.
29. I. Cheng, P. Cheng, and S. Tsai, “Rehabilitation System,” pp. 660–661, 2015.
30. A. Varma and S. S. Bhattacharyya, “Java-through-C compilation : An enabling technology for Java in embedded systems,” *Proc. -Design, Autom. Test Eur. DATE*, vol. 3, pp. 161–166, 2004, doi :10.1109/DATE.2004.1269224.
31. qizi Sharopova, M.M., 2023. INTRODUCING" PROGRAM CONTROL OPERATORS" IN THE JAVA PROGRAMMING LANGUAGE. Multidisciplinary Journal of Science and Technology, 3(5), pp.222-231.
32. Lu, K.C. and Krishnamurthi, S., 2024. Identifying and Correcting Programming Language Behavior Misconceptions. Proceedings of the ACM on Programming Languages, 8(OOPSLA1), pp.334-361.
33. Do, N.V. and Mai, T.T., 2023, December. A Knowledge Representation Model for Designing the Knowledge Querying System in Programming Language C/C++.In 2023 RIVF International Conference on Computing and Communication Technologies (RIVF) (pp. 366-371). IEEE.

34. Kumar, A. and Goswami, M., 2023. Performance comparison of instrument automation pipelines using different programming languages. Scientific Reports, 13(1), p.18579.
35. O. Ali, O.M.A., Kareem, S.W. and Mohammed, A.S., 2022, February. Evaluation of electrocardiogram signals classification using CNN, SVM, and LSTM algorithm: A review. In 2022 8th International Engineering Conference on Sustainable Technology and Development (IEC) (pp. 185-191). IEEE. doi: 10.1109/IEC54822.2022.9807511.
36. Hamaamin, R.A., Wady, S.H. and Sangawi, A.W.K., 2022. COVID-19 Classification based on Neutrosophic Set Transfer Learning Approach. UHD Journal of Science and Technology, 6(2), pp.11-18. https://doi.org/10.21928/uhdjst.v6n2y2022.pp11-18
37. Thorgeirsson, S., Weidmann, T.B., Weidmann, K.H. and Su, Z., 2024, March. Comparing Cognitive Load Among Undergraduate Students Programming in Python and the Visual Language Algot. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (pp. 1328-1334).