COMPARATIVE STUDY & ANALYSIS OF OBJECT ORIENTED PROGRAMMING LANGUAGES

Tejenderkaur H. Sandhu
Master of Engineering (M.E.) Scholar
Department of Computer Science & Engineering,
Sant Gadge Baba Amravati University, Amravati, India

Abstract : There are a variety of programming languages in the world. Many of them are in the category of Object Oriented Programming Languages. Each of them has its advantages and disadvantages. Comparison of programming languages is a one of the most important topic for debate among software developers. Many programming languages are designed, specified, and implemented every year to level up with programming paradigms, hardware evolution, etc. We are summarizing and comparing these languages on different parameters. We are illustrating the language differences using certain program that represents a common problem with an object-oriented characteristic. Based on our examples, we are analyzing which language supports which characteristic with more features. These measurements are intended to provide with an understanding of the approximate performance of the current language implementations. Such an understanding is useful because often performance considerations constrain what languages a programmer is likely to consider using on a particular project.

Keywords **– Object Oriented Language, Evaluation Criteria, Analysis of Language, Criteria, Compiler**

I. INTRODUCTION

Programming languages is one of the captivating and interesting field of study. Computer scientists tend to create new programming language. Many varieties of languages with more advance features and paradigm have been created in the last few years. Some languages fancy wide quality and have introduced new features. Each language has its advantages and drawbacks. With this variety of languages and their worldwide usage, software engineer and programmers should be familiar with the pros and cons of these which could be used to bring solution to their software and should be aware with the decisions they will make with it. The object-oriented programming paradigm provides a more intuitive way of programming, it also has complexities. This is due to the various complicated features that the model offers. OOPLs differ widely in the way they implement features that are associated with the object design. For example, some object oriented language support certain object oriented features which are not present in other object oriented languages.

II. THE PURPOSE OF THIS STUDY

The purpose of this study is to achieve the fact that there are no fixed or defined set of procedures or evaluation criteria for comparing any programming language which each other. We can define our own criteria with verified results, calculations and observation for analyzing the difference between these languages.

III. ANALYSIS OF PROBLEM

Since there are hundreds of programming languages existing nowadays, we can compare and analyze the efficient one. We can classify the representative characteristics of languages and make a broader view on them according to some certain criteria. Thus our research problem is aiming to compare and contrast object oriented languages according to certain characteristics with the purpose of determining the suitability and applicability of the languages for each criterion, distinguish them their pros and cons, evaluate and explore the related features on those languages, illustrate the best language usage for evaluated characteristics and also get the details of resources required for a particular language.

IV. PROPOSED STRATGEY

We are using different object oriented programming languages such as C++, Java, C#, Python & Ruby for comparison and analysis of the efficient one. We are determining the efficiency of the language on the basis of various characteristics. These characteristics will give us comparison and analysis on the languages we are using. In order to better compare the overall features of the languages under study, we will also consider different other criteria to illustrate how some languages outperforms others in a given criterion and the reason behind that.

The comparison of the languages is based on:

- Code Size – This feature gives the detail of number of lines generated for writing the program.

- Running Time - It will define the time taken by the program to represent the output after compilation.
- Space - This parameter gives the memory size required to store the program. Other sizes that can be calculated are Object size and Binary size after compilation.

We are using two groups of programs for evaluating the language where first group consists of two algorithms to be implemented such as factorial and sorting algorithm. For both of this algorithm we are considering five set of inputs.

Second group consists of implementing programs of object oriented features such as class & object, inheritance and polymorphism. We have performed comparison and analysis by implementing same programs in mentioned languages with our hardware and software environment. Following is the obtained result calculation and graphical analysis of the comparison.

Language	Compiler	Version
C++	G++	5.4.0
Java	Javac	1.8.0_191
C#	Mono	4.2.1
Python	Python	2.7.12
Ruby	Ruby	2.3.1

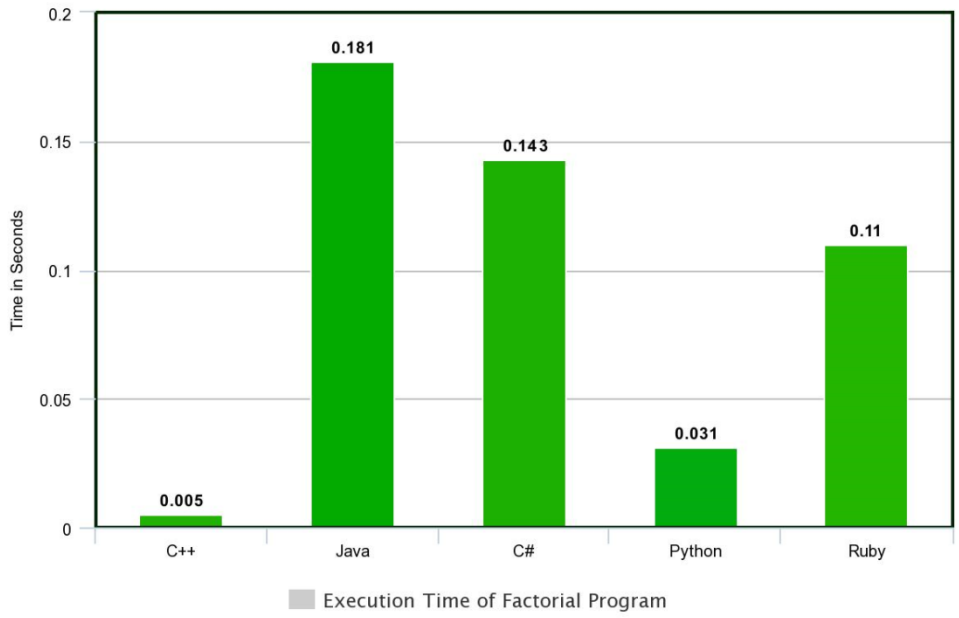
FIGURE 1: LANGUAGE COMPILER USED

V. RESULTS

Group 1: a) First we are taking factorial program as our input language for the entire five object oriented languages. In this we have taken 5 input samples to get an average value for each language thus given us the idea that which language takes how much time, memory usage, number of lines taken to implement same factorial algorithm program with same inputs.

Language	Test 1 (Input = 5)	Test 2 (Input = 10)	Test 3 (Input = 15)	Test 4 (Input = 20)	Test 5 (Input = 25)	Average
C++	0.005 s	0.005 s	0.005 s	0.005 s	0.005 s	0.005 s
Java	0.131 s	0.175 s	0.192 s	0.194 s	0.213 s	0.181 s
C#	0.131 s	0.141 s	0.147 s	0.148 s	0.152 s	0.143 s
Python	0.030 s	0.031 s	0.032 s	0.032 s	0.032 s	0.031 s
Ruby	0.101 s	0.106 s	0.111 s	0.115 s	0.119 s	0.110 s

FIGURE 2: CALCULATIONS OF FACTORIAL PROGRAM



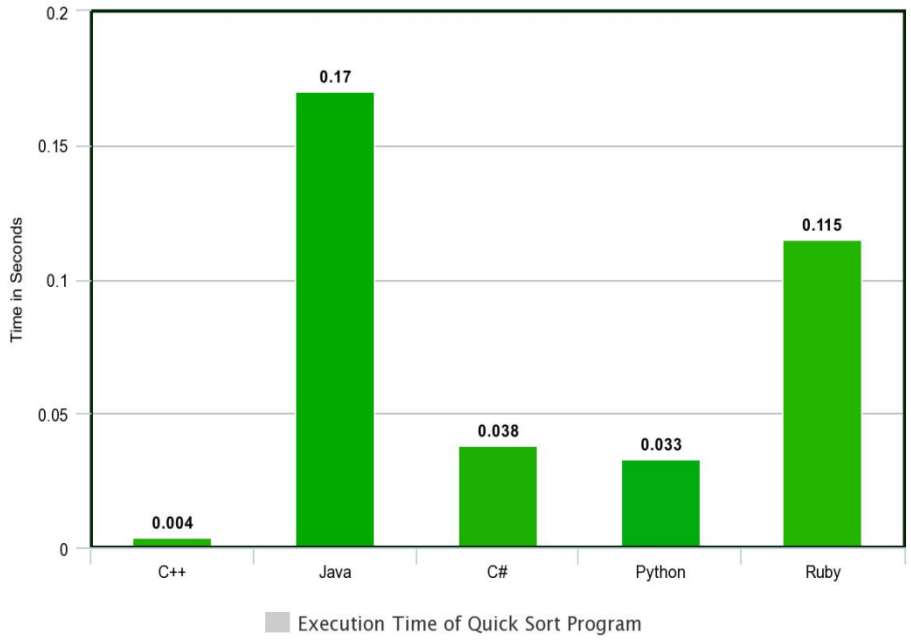
Language	Memory Consumption
C++	257 Bytes
Java	1055 Bytes
C#	3584 Bytes
Python	732 Bytes
Ruby	214 Bytes

FIGURE 3: CALCULATIONS OF FACTORIAL PROGRAM

Group 1: b) Second we are taking sorting program for quick sort as our input language for the entire five object oriented languages. In this we have taken 5 input samples to get an average value for each language thus given us the idea that which language takes how much time, memory usage, number of lines taken to implement same sorting algorithm program with same inputs.

Language	Test 1 (Input = 5)	Test 2 (Input = 10)	Test 3 (Input = 15)	Test 4 (Input = 20)	Test 5 (Input = 25)	Average
C++	0.004 s	0.005 s	0.005 s	0.005 s	0.005 s	0.004 s
Java	0.134 s	0.138 s	0.186 s	0.190 s	0.206 s	0.170 s
C#	0.029 s	0.036 s	0.039 s	0.044 s	0.044 s	0.038 s
Python	0.031 s	0.032 s	0.033 s	0.034 s	0.036 s	0.033 s
Ruby	0.106 s	0.112 s	0.118 s	0.121 s	0.121 s	0.115 s

FIGURE 4: CALCULATIONS OF SORTING PROGRAM



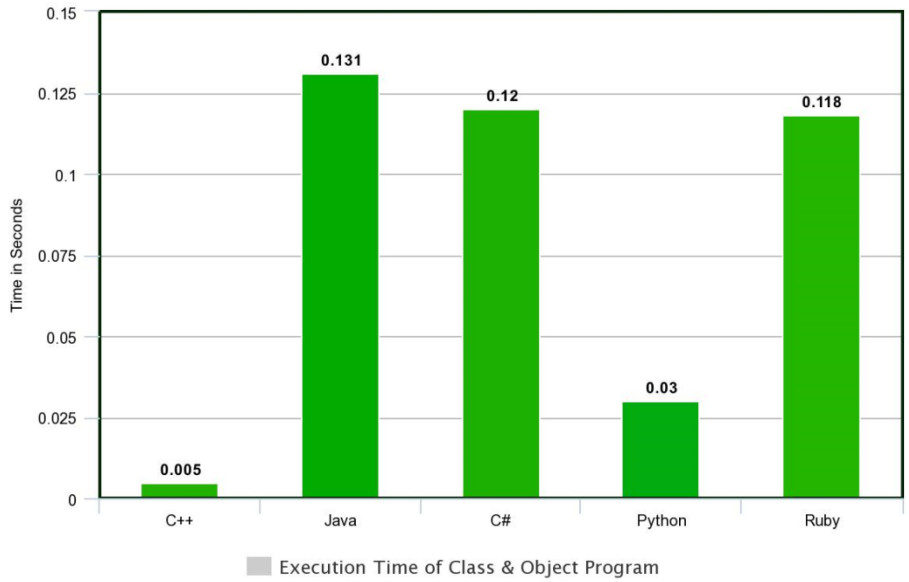
Language	Memory Consumption
C++	1650 Bytes
Java	1501 Bytes
C#	5120 Bytes
Python	1283 Bytes
Ruby	568 Bytes

FIGURE 5: CALCULATIONS OF SORTING PROGRAM

Group 2: a) Third we are taking class and object program as our input language for the entire five object oriented languages. In this we have implemented the same example in the entire five languages with same number of object thus giving us the idea that which language takes how much time, memory usage, number of lines taken to implement same program with same number of objects.

Language	Execution Time (s)	Memory Utilization
C++	0.005 s	526 Bytes
Java	0.131 s	624 Bytes
C#	0.120 s	3072 Bytes
Python	0.030 s	356 Bytes
Ruby	0.118 s	456 Bytes

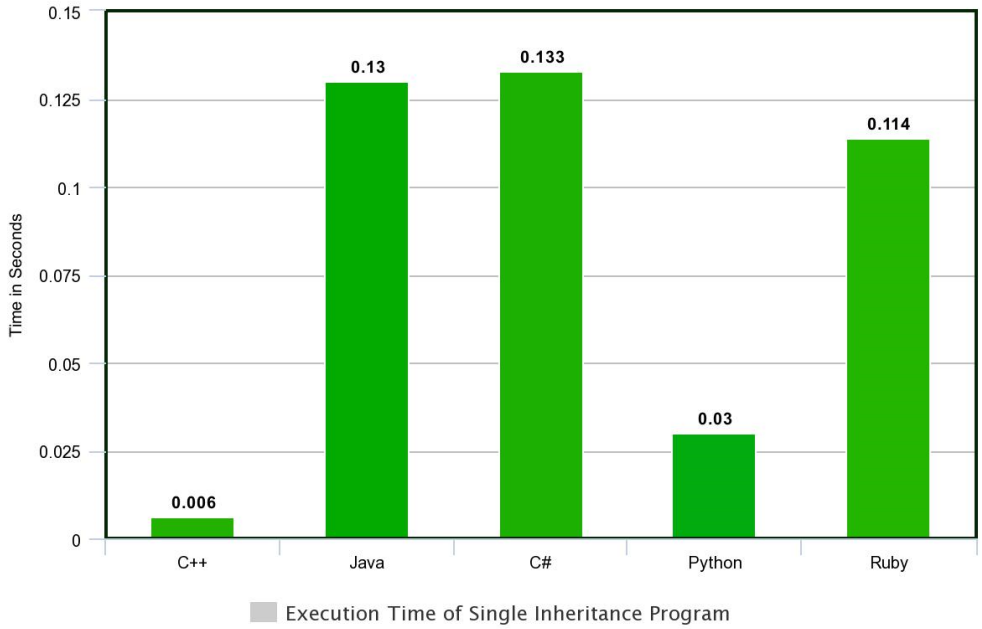
FIGURE 6: CALCULATIONS OF CLASS & OBJECT PROGRAM



Group 2: b) Fourth we are taking two types of inheritance (single & multilevel) program as our input language for the entire five object oriented languages. In this we have implemented the same example in the entire five languages with same number of object thus giving us the idea that which language takes how much time, memory usage, number of lines taken to implement same program with same number of objects.

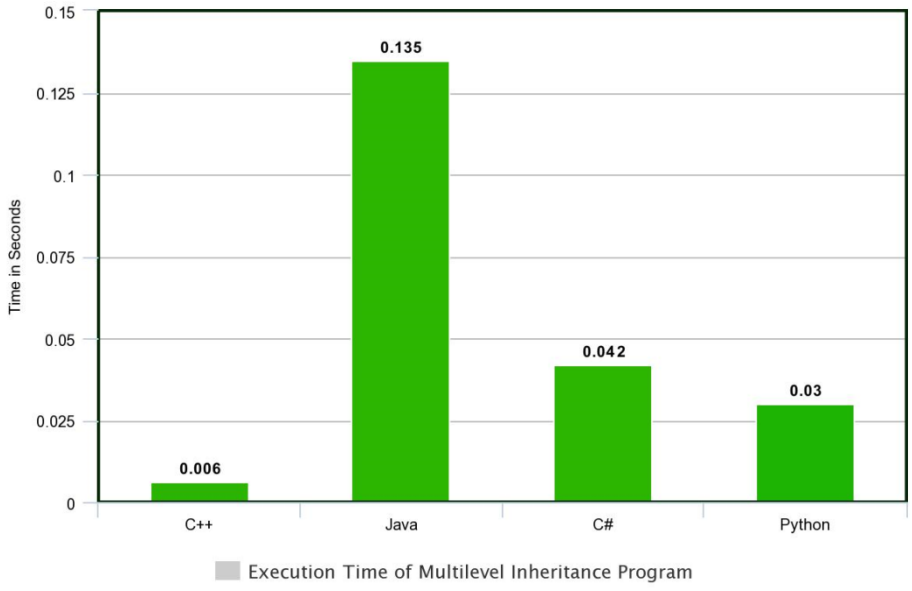
Language	Execution Time (s)	Memory Utilization
C++	0.006 s	333 Bytes
Java	0.130 s	350 Bytes
C#	0.133 s	3584 Bytes
Python	0.030 s	293 Bytes
Ruby	0.114 s	264 Bytes

FIGURE 7: CALCULATIONS OF SINGLE INHERITANCE PROGRAM



Language	Execution Time (s)	Memory Utilization
C++	0.006 s	504 Bytes
Java	0.135 s	381 Bytes
C#	0.042 s	3584 Bytes
Python	0.030 s	453 Bytes

FIGURE 8: CALCULATIONS OF MULTILEVEL INHERITANCE PROGRAM

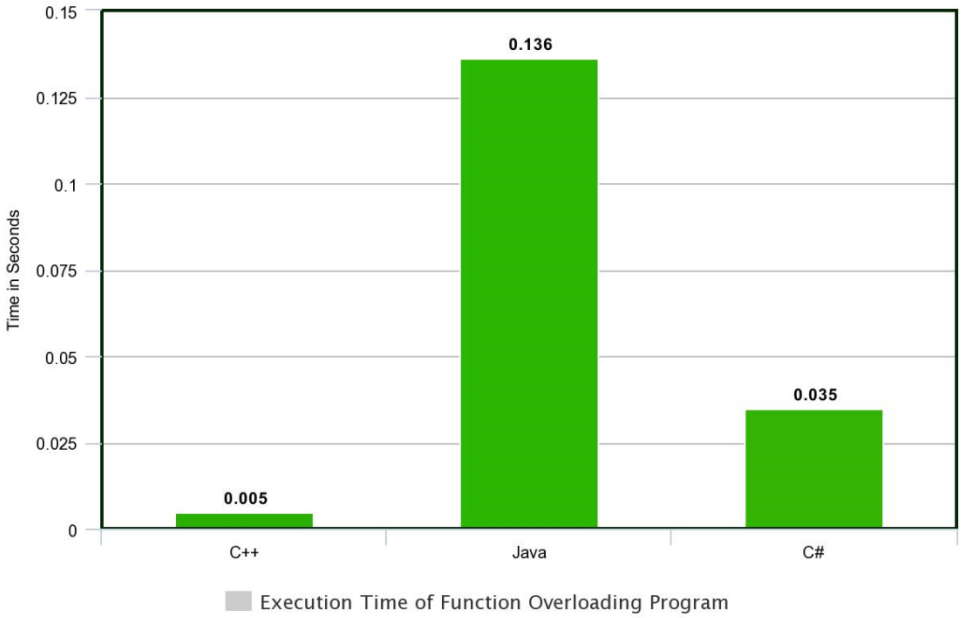


Group b: c) Fifth we are taking two types of polymorphism program (function overloading & method overriding) as our input language for the entire five object oriented languages. In this we have implemented the same example in the entire five languages with same number of object thus giving us the idea that which language takes how much time, object size allocated to program,

memory usage, number of lines and can the code be optimized or not; to implement same program with same polymorphism feature.

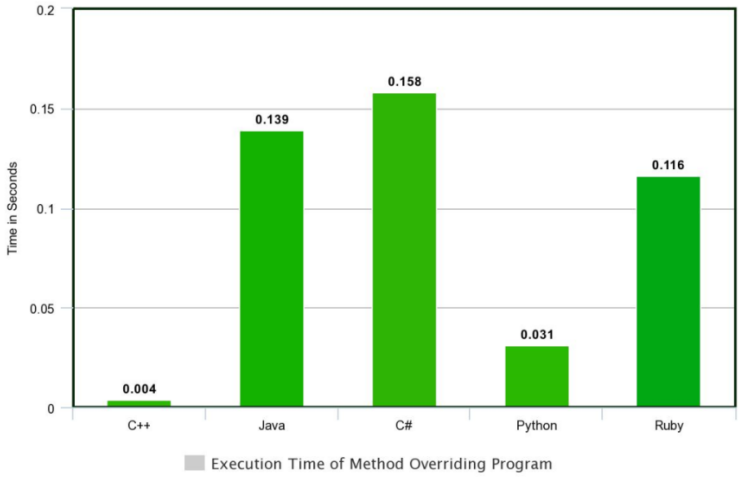
Language	Execution Time (s)	Memory Utilization
C++	0.005 s	307 Bytes
Java	0.136 s	815 Bytes
C#	0.035 s	3072 Bytes
Python	-	-
Ruby	-	--

FIGURE 9: CALCULATIONS OF FUNCTION OVERLOADING PROGRAM



Language	Execution Time (s)	Memory Utilization
C++	0.004 s	1101 Bytes
Java	0.139 s	882 Bytes
C#	0.158 s	3584 Bytes
Python	0.031 s	469 Bytes
Ruby	0.116 s	405 Bytes

FIGURE 10: CALCULATIONS OF METHOD OVERRIDING PROGRAM



VI. CONCLUSION

Object-oriented programming languages are used worldwide on many alternative projects and applications. Mastery of the object-oriented paradigm has become an essential part of any programmer's careers. The key features of the object-oriented paradigm (abstraction, encapsulation, inheritance, and polymorphism) have different flavors in the various OOPLs available to the users. There is still lot of work to be done not only to reach a common representation for these crucial features of OOPLs, but also to find appropriate ways to implement features like inheritance and polymorphism to avoid misuse.

VII. REFERNCES

[1] O. Nierstrasz "A Survey of Object-Oriented Concepts", 1989, pp 3–22.

[2] Robert Henderson, Benjamin Zorn "A Comparison of Object-Oriented Programming in Four Modern Languages", November 1994, pp 1077-1095.

[3] Lutz Prechelt "An Empirical Comparison of Seven Programming Languages", IEEE, October 2000, pp 1-7.

[4] D. J. Armstrong "The Quarks of Object-Oriented Development", Communications of the ACM, 49(2):123–128, Vol. 49 No. 2, February 2006.

[5] K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra "Empirical Study of Object-Oriented Metrics", Journal of Object Technology, Vol. 5, No. 8, November-December 2006, pp. 149-173.