International Journal of Software Engineering and Its Applications
Vol. 9, No. 11 (2015), pp. 79-90
http://dx.doi.org/10.14257/ijseia.2015.9.11.07

Impact and Comparison of Programming Constructs on JAVA and C# Source Code Readability

Aisha Batool[1], Muhammad Habib ur Rehman[2], Aihab Khan[1], and Amsa Azeem[1]

[1]Department of Computing and Technology, Iqra University, Islamabad, Pakistan
[2]Faculty of CS and IT, University of Malaya, Kuala Lumpur, Malaysia
[1]aishabatool_omsats@yahoo.com, [2]mhrehman@siswa.um.edu.my,
[1]aihabkhan@yahoo.com

Abstract

Software readability is a property that manipulates how easily a given piece of code can be read and understood. Since readability can affect maintainability, quality, reusability, understandability etc. Programmers are very concerned about the readability of code. For the good decision of selecting languages, it is necessary to know about the readability of languages. Many software constructs may affect readability of code. In this paper we have selected some constructs that affect readability property and we calculated their readability in C# and java PL. At the end we have also compared results for both languages to make decision easy for programmers to choose best one from both. Short snippets are taken from c# and java and for their readability; five readability indexes are used to get results.

Keywords*: Code readability, programming constructs, Readability index, Gunning fox index, SMOG*

1. Introduction

Code readability is the capability of programming code that makes it readable and understandable even for a nonprogrammer person [1], generally readability can be calculated by the ratio between number of lines of code and the number of comments that are for the understanding of human not for the computer. We may also say that if we can understand a program without searching for declarations and definitions of that language or the average rate of right answers about program in a specific time. It's clearly seems that readability is a attribute related to others,*e.g.*, robustness, maintainability, modifiability, complexity, understandability and reusability. Programming code that is more [2] readable is less complex, more robust, more understandable, reusable, modifiable [3, 4] and maintainable as well. Now a days software developer spend most of the time on evolving nad maintaining existing code, rather than generating new one. Readability is of significant importance and critical for software maintenance phase. Reading the code is first step in software maintenance. In early as well as current research, revealed that readability plays largest role in maintenance phase. The area of programming readability has supreme importance in software development and whole engineering process. Enough literature found on how to increase code readability, how to calculate and measure source code readability, how to build analytical models, how programming language's readability [13] affects software cost and economy. In this article we follow a different path; we explore the question of which constructs affect code readability and which language from c# and java is better in terms of readability.

The main goal of this paper is to find programming features and their impact on source code readability. Our contribution is important in a way that if will help developers to choose one of the both languages for their soft ware's according to their readability. As it's mentioned before that software readability is important for the economic value of SW too.

In this paper we will discuss some programming features that effect readability of code in Java and c#. Short snippets from java and c# code will be taken. Different metrics exist to calculate the readability of code. These metrics include ARI automated readability index, SMOG and the gunning fox index. By using these metrics we will find the readability of snippets from both languages. And at the end a comparison of both language's readability will be made on the basis of these results.

Rest of the paper is organized as Section 2: literature review of the relevant literature, Section 3describes problem statement of the research, Section 4 consists on methodology of work including the constructs that affect programming readability and metrics to measure readability of source code, Section:5 presents experimental results and rest finally conclusion and future work is given.

2. Literature Review

In 2010 Buse and Weimer constructed a readability tool that automatically measures readability [6]. Authors considered a number of human annotators for the judgment of selected code's readability. Selected code was snippets of java code. Results obtained from experts were compared with their proposed measure. Results shown that the measure's accuracy was 80%.Association between Two quality features and readability were studies, features were errors or faults and evolution. If selected is fine grained so the effect of code volume will be neglected. Author used java code snippets and read features in code line by line. Author said that the volume of code directly effects the readability feature of code.; as short code is easier to read as compared to large one. Readability attribute of the code depends on code's complexity and coding style as well.

Author in [7] developed an automatic system to increase code readability. He proposed that if blank lines are added in source code then it would improve code readability and also he located points from internal documentation. Author developed a tool for his proposed technique, which automatically gets java methods as input and returns a version segmented by blank lines after each meaningful block of code. This segmented version of code helps in code readability and also it is helpful for the internal comments that where to place. Evaluation results shown that the automated blank line's insertion is as effective as blank lines added by programers. It seemed by results that system uses vertical lines as programmers think it is better to use.

In [8], the author Wang, Xiaoran and Lori Pollock cleared the role of source code readability in the improvement of software quality. They said that code readability is important for the later stages of Software development life cycle *e.g.*, maintenance phase. As most of the cost of software development is expended on maintenance phase. Author gathered a number of open source snippets from internet and asked some programming experts to value the complexity of code. This is done on the basis of some programming constructs *e.g.*, keywords, loops etc. they also developed a tool that automatically measures code's readability, which's effectiveness is better than the human judgment.

Collar Jr, Emilio, and Ricardo Valerdi proposed that the source code readability effects the software cost [9]. It means that the improved code readability will decrease the time spent on code reading and it will decrease that software cost in each stage of the software development. Similarly less readability will improve the time of reading that will leads to the improved software cost. Results are presented with procedural languages, that shows how programming code can be analyzed with respect to the readability. Readability of code directly effects the time spend on understanding code during maintenance phase.

Maintenance phase is the most important and cost consuming phase of software development life cycle.

In [10] ARI is described, which is the metric for code readability measurement. In ARI (automated readability index) two factors for readability are used. One is the sentence difficulty which is established by calculating words per sentence [15]. And second is the word difficulty which refer to the letters per word

SMOG, Simple Measure of Gobbledygook was proposed in 1969 by G Harry McLaughlin [11]. SMOG is used for measuring code readability. This metric gives an estimated level of education needed for reading and understanding a piece of code. According to some others, SMOG stands for Robert Gunning's FOG. SMOG outputs are calculated by adding 3 in polysyllable count's square root.

Robbert Gunning introduced [12] another readability metric FOG. In FOG formula average length of sentences added to the hard word's percentage. Average length of sentences is calculated by dividing number of words by the total number of sentences.

3. Problem Statement

Programs must be written for people to read, and only incidentally for machines to execute [2]. A portion of code written by a programmer (author) must be understandable by current stakeholders, *e.g.*, the author's immediate team members (and even the author at a future time). But that is not all; the code must be understandable by future stakeholders, *e.g.*, rest of the programmers in the project or organization, especially programmers who might be hired in future. Code readability is important in terms of modification, understandability, maintainability and reusability. So readability factor increase quality of software. According to TIOBE programming community index java and C# languages are maximum used commercially, cause of being type safe. Therefore we must know the readability value of both languages so that we could be able to choose best one of both. In this way a comparison of C# and java readability is needed to make us able to make decisions.

4. Methodology

In this paper we will discuss some programming features that effect readability of code in Java and c#. To find the effect of these features Short snippets from java and c# code will be taken, and then the effect of these features on readability will be calculated using some metrics. Different metrics exist to calculate the readability of code. These metrics include ARI automated readability index, SMOG and the gunning fox index. By using these metrics we will find the readability of snippets from both languages. And at the end a comparison of both languages' readability will be made on the basis of these results. This will be helpful in making decision in selecting language for projects.

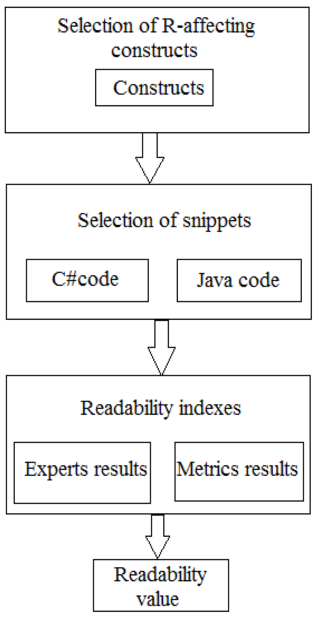

Figure 1. Approach workflow

Table 1. Constructs that Effect Code Readability

Meaningful names	Naming conventions
Comments	Comments
Spacing	Spacing
Indents	Indentation
Short scopes	Scopes
Line length distribution	Code Distribution
Identifier name length	Identifier Length
Arithmetic formulas	Complexity of formulas
Identifier frequency	Number of identifiers
If-else	Decision structures
Nested-if	
Switch	
While Loop	Repetition structures
For loop	
Do-while	
Nested loop	Nested Repetition
Recursive	Repetition
Arrays	Array Structures
Classes Distribution	Class Diagrams
Inheritance	

4.1 Code Readability Metrics

The following sub sections focus on ARI, SMOG and Gunning Fog metrics.

4.1.1. The Automated Readability Index (ARI): In ARI (automated readability index) two factors for readability are used. One is the sentence difficulty which is established by calculating words per sentence. And second is the word difficulty which refers to the letters per word. The equation for calculating ARI index is (see Eq. 1).

$$\text{ARI} = 4.71(\text{characters}) + 0.5\ (\text{words}) - 21.43 \qquad (1)$$

4.1.2. SMOG: The Simple Measure of Gobbledygook (SMOG) was proposed in 1969 by G Harry McLaughlin [11]. SMOG is used for measuring code readability. This metric gives an estimated level of education needed for reading and understanding a piece of code. According to some others, SMOG stands for Robert Gunning's FOG. SMOG outputs are calculated by adding 3 in polysyllable count's square root.

$$\text{SMOG} = 3 + \text{Square Root of Polysyllable Count} \tag{2}$$

4.1.3. The Gunning's Fog Index: Robbert Gunning introduced [12] another readability metric FOG. In FOG formula average length of sentences added to the hard word's percentage. Average length of sentences is calculated by dividing number of words by the total number of sentences.

$$\text{Grade Level} = 0.4\ (\text{ASL} + \text{PHW}) \tag{3}$$

4.1.4. Flesch-KincaidReadability Index: Flesch-Kincaid test results indicates the reading ease of the given metirial, if the value is high it means readability is high and if the output is less that means code is difficult to read. Flesch Reading Ease Score (FRES) test formula is given as [14].

$$206.835 - 1.015\left(\frac{total\ words}{total\ sentences}\right) - 84.6\left(\frac{total\ syllables}{total\ words}\right) \tag{4}$$

Table 2. Output Score Can be Interpreted According to these Criteria

Score	Notes
90.0 – 100.0	Easily understood by an average 11-year old student
0 – 70.0	Easily understood by 13- to 15-year old students
0.0 - 30.0	Easily understood by University graduates

4.1.5. Coleman-Liau Index: Meri Coleman and T. L. Liau designed another readability index similar as ARI but unlikely all others. This index focuses on the letters per word but not on the syllables. Opinions about accuracy of both varies. Formula for the Coleman–Liau index is following:

$$CLI = 0.0588L - 0.296S - 15.8 \tag{5}$$

where L and S are average number of letters and sentences.

5. Experimental Results

The Experiments are made on the java and C# source code snippets. SMOG, Gunning Fog, ARI etc metrics are used for getting readability index. Some constructs are not well fitted for the metrics readability index so that we have also arranged a survey. In this survey we have prepared a questionnaire and distributed to some programming experts with the request to fill that. In this questionnaire we mentioned all snippets used in experiments and their readability percentage is asked from those experts. Readability index results are presented in tabular form as well as in graphical form. Cause graphs gives more understandability and visibility.

5.1. Tabular Representation of Results: All metrics are applied on java snippets and C# as well. In the following table metric grades are mentioned, applied on different constructs of programming languages. Line length distribution and class distribution have maximum average readability grade, that shows high readability of these constructs. But comparing C# and java for these two constructs, C# have require more grade level as compared to java. Inheritance and overriding are two constructs with the highest readability. But comp[aring java and C# we found that C# have high readability grade level as compared to java. So According to the results, it is proved that java is more readable as compared to C#.

Table 3. Java Construct's Readability

Construct	FKGL Grade	Gunning fog Grade	CLI Grade	SMOG Grade	ARI Grade	FK Reading ease	Average
If-else	2	2.7	2.2	1.8	-3.6	92.9	1.0
Switch statement	1	2.3	3.5	1.8	-3.1	98.5	1.1
Nested if	4.2	6.8	-0.6	1.8	-0.7	95	2.3
For loop	4.2	6.8	-0.2	1.2	-0.5	95	2.4
While loop	1.5	3.6	0.5	1.8	-3.8	100	0.7
Do-while	1.3	3.1	3.9	1.8	-1.8	99.7	1.7
Nested loop	5.2	8	-2.6	1.8	-0.8	93.5	2.3
Comments	11.2	8	10.9	10.1	5.5	34.2	9.1
Arrays	4.5	7.2	-1.4	1.8	-0.9	94.6	2.2
Recursive	5.7	6.6	8.6	5	1.6	66.8	5.5
Inheritance	-0.1	2.3	-1.8	1.8	-7.3	106	-1.0
Overriding	-0.3	1.9	-1.5	1.8	-7.7	105	-1.2
Scope	0.9	2,9	4.2	1.8	-1.7	101	1.6
Class distribution	13.5	12.6	14.7	6	17	48.5	12.8
Arithmetic formula	3.3	3.1	9.2	3.2	1.3	80.9	4.0
Indents	4.3	5.4	10	4.1	0.3	68.1	4.8
Spacing	9.9	12.9	13.5	9.2	8.1	45.9	10.6
Line length distribution	15	16.4	20.1	11.6	15.3	16.1	15.7

Table 4. C# Constructs' Readability

Construct	FKGL Grade	Gunning Fog grade	CLI grade	SMOG grade	ARI grade	FK reading ease	average
If else	3.9	5.7	5.5	4.4	-0.8	79.9	3.7
Switch statement	4.4	7.2	5.3	5.4	0.1	79.8	4.5
Nested if	2.8	6	-0.9	1.8	-2	101	1.5
For loop	2.8	6	-0.5	1.8	-1.7	101	1.7
While loop	3.3	6.5	1.1	4.4	-1.9	92.4	2.7
Do while loop	3.1	6	5.1	4.4	0.3	90.9	3.8
Nested loop	3.5	6.8	-2.3	1.8	-2.1	100	1.5
Comments	8.6	9.7	6.5	8.3	5.9	67.3	7.8
Array	3.2	6.4	-1.8	1.8	-2.2	100.7	1.5
Recursive	8.4	10	9.8	7.2	4.1	52.9	7.9
Inheritance	1.7	5.7	-1.7	4.4	-6.8	94.8	0.7
Overriding	2	5.5	-0.4	4.3	-5.9	91.9	1.1
Scope	5.6	8.4	5.9	6	2.7	79.5	5.7
Class distribution	13.5	12.6	14.7	6	17	48.5	12.8

Arithmetic formula	7.5	8.6	11.6	6.6	5.1	57.3	7.9
Indents	6.9	9.2	12.4	5.1	2.3	49.5	7.2
Spacing	8.4	10.4	12.8	7.9	6.5	52.9	9.2
Line length distribution	20	21.2	21.6	15.6	21.5	-1.8	19.9

5.2. Graphical Representation of Results: We have presented readability calculation results in the graphical form to improve visibility. Graphs are designed using tabular information. As concluded in the tables section that JAVA is more readable as compared to C#, similarly it is clearly presented in the graphs that the C# code snippets require more grade level for reading and understandability. Java snippet's require less grade level and have high Flesch-Kincaid Reading Ease index. So according to the results it is concluded that the JAVA have more readability than C#.

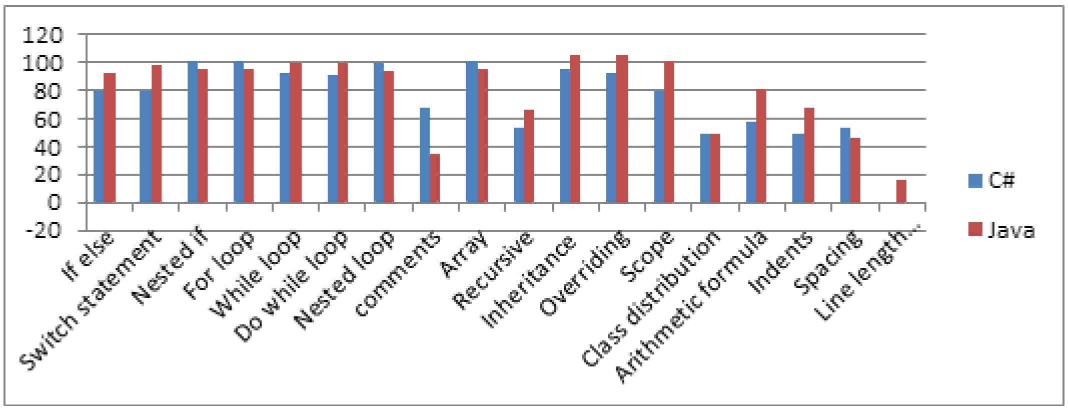

Figure 1. Flesch-Kincaid Reading Ease

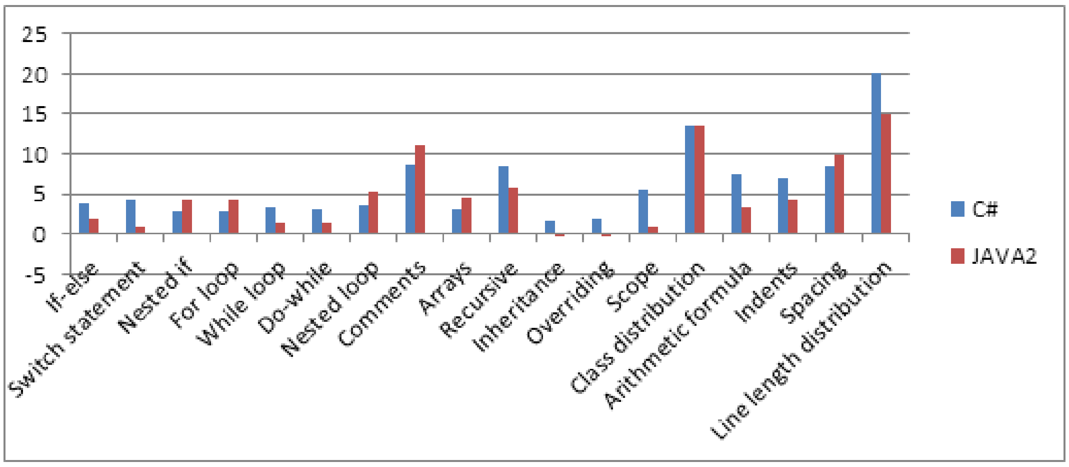

Figure 2. Flesch-Kincaid Grade Level

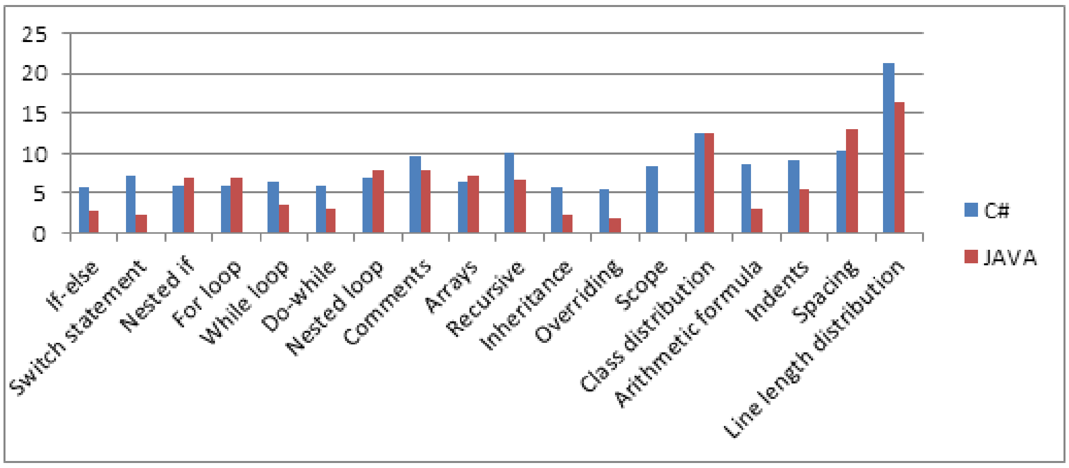

Figure 3. Gunning Fog Score Grade Level

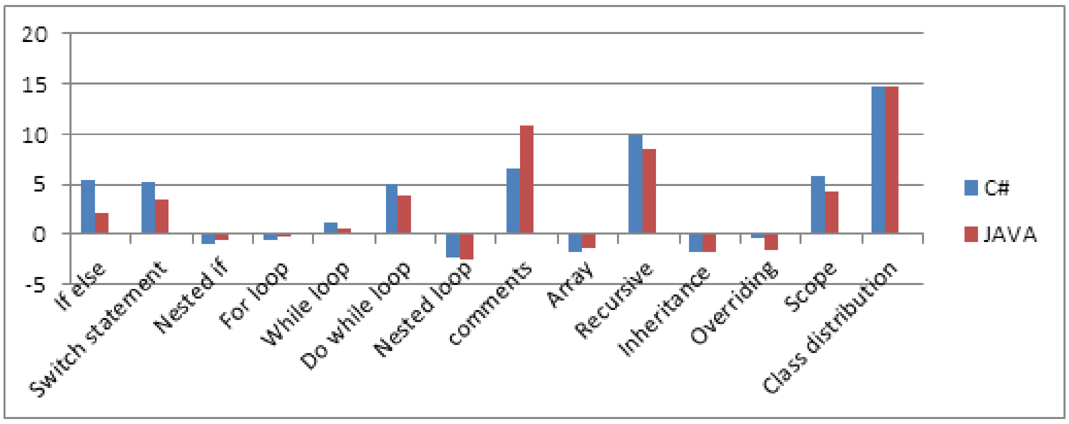

Figure 4. Coleman-Liau Index Grade Level

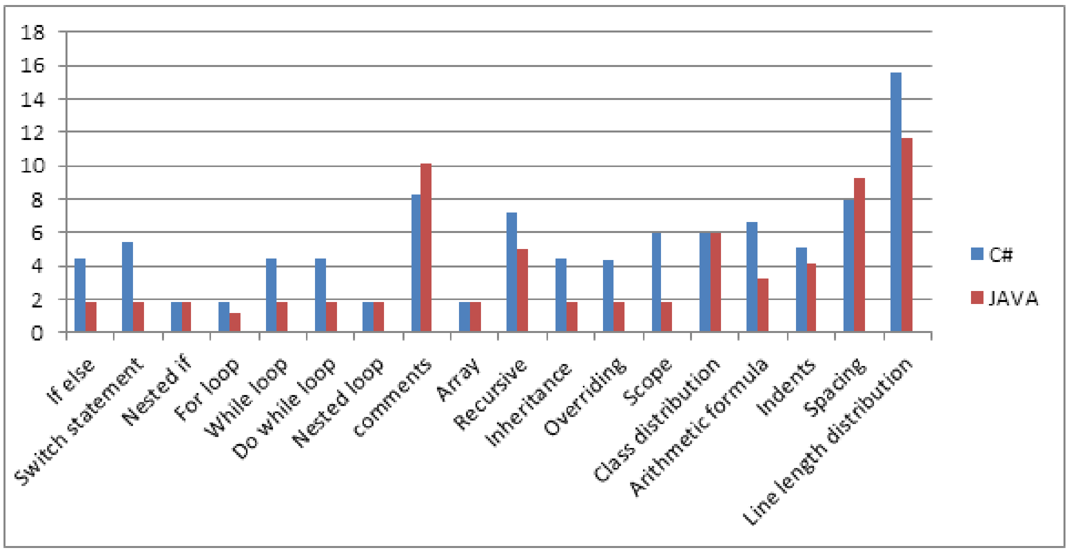

Figure 5. SMOG Index Grade Level

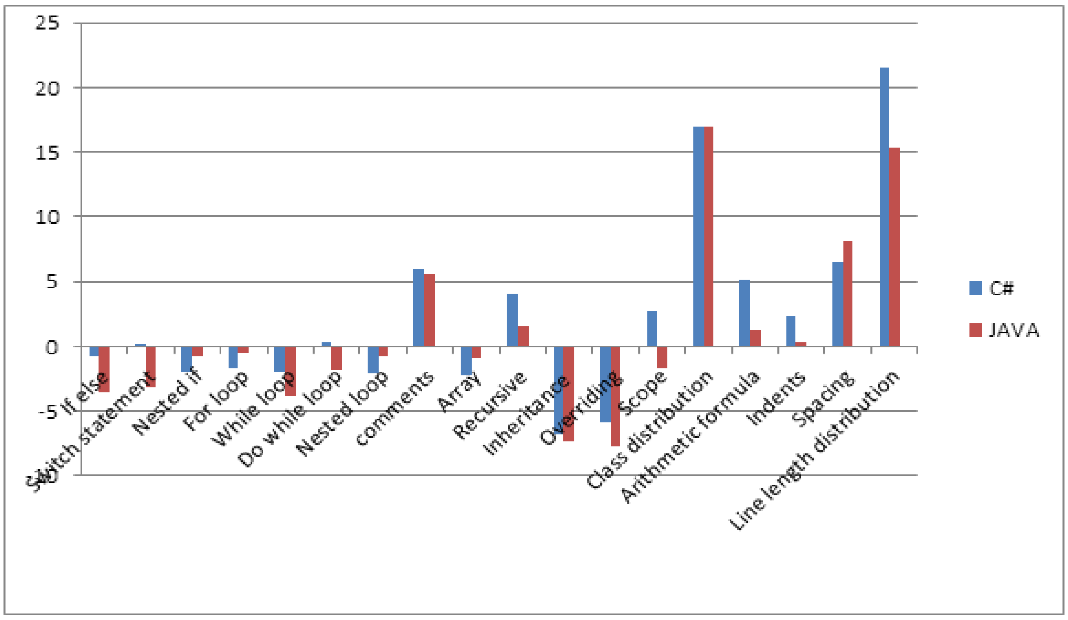

Figure6. Automated Readability Index Grade Level

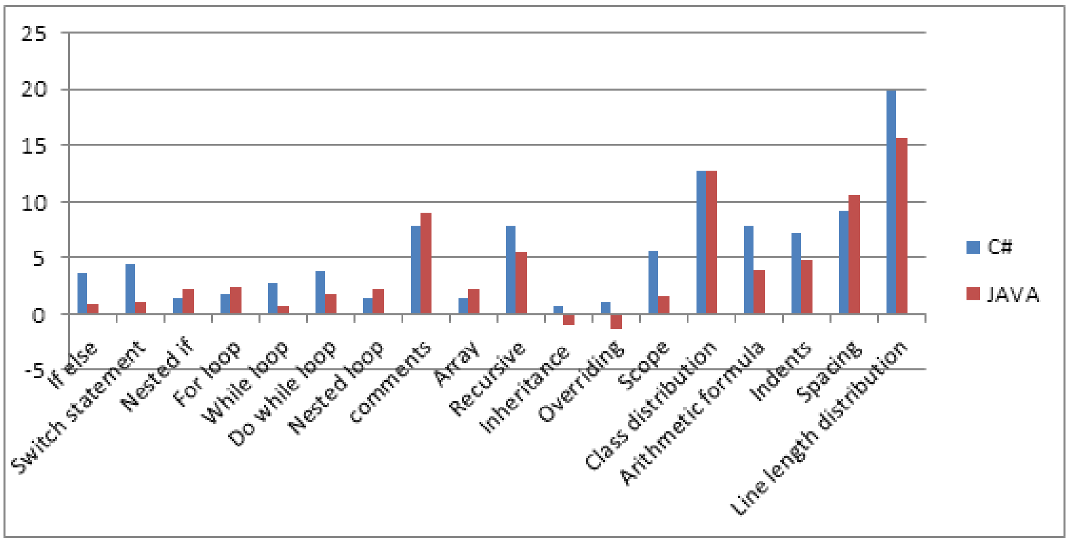

Figure 7. Average Grade Levels for C# and JAVA

Table 5. Results Taken from Programming Experts

Construct	C# readability in %	Java Readability in %
If else	60-80	100
Switch statement	60-80	80-100
Nested if	100	80-100
For loop	100	100
While loop	80-100	100
Do while loop	80-100	100
Nested loop	80-100	80-100
Comments	60-80	60-80
Array	80-100	80-100
Recursive	60-80	60-80

Inheritance	80-100	100
Overriding	80-100	100
Scope	60-80	60-80
Class distribution	40-60	40-60
Arithmetic formula	40-60	80-100
Indents	40-60	60-80
Spacing	40-60	40-60
Line length distribution	1-20	20-40

6. Conclusion and Future work

In this paper we have used some metrics to calculate readability of C# and java. We selected 22 features that may affect source code readability. For the calculation of readability index we have used five common readability metrics. And for the well trusted results we have also designed a questionnaire using these C# and java snippets. This questionnaire is distributed in 15 experts to get their opinion about readability percentage of both languages. According to the results, metrics shown that java is more readable programming language than C# and from the survey as well. For the further enhancement of the work, more constructs of the programming languages can be used for the experiments which may affect the programming language readability. There are many other languages which are becoming popular, so we may make comparison in different language’s readability for the good decision of programmers to choose best language for development.

References

[1] G. Amrulla, *et al.*,“A Survey of Improving Computer Program Readability to Aid Modification”, International Journal, vol. 3, no. 3, **(2014)**.
[2] Y. Uni, “Impact of Programming Features on Code Readability”, **(2013)**.
[3] T. Tenny, “Program readability: Procedures versus comments”, Software Engineering, IEEE Transactions on, doi>10.1109/32.6171, vol. 14, no. 9, **(1988)**, pp. 1271-1279.
[4] C. A. Cunha, J. L. Sobral and M. P. Monteiro, "Reusable aspect-oriented implementations of concurrency patterns and mechanisms", Proceedings of the 5th international conference on Aspect-oriented software development. ACM,**(2006)**.
[5] J. L. Elshoff and M. Marcotty, "Improving computer program readability to aid modification”, Communications of the ACM, vol. 25, no. 8,**(1982)**, pp. 512-521.
[6] R. P. L. Buse and W. R. Weimer, "Learning a metric for code readability", Software Engineering, IEEE Transactions on, vol. 36, no. 4, **(2010)**, pp. 546-558.
[7] P. Sivaprakasam and V. Sangeetha, "An accurate model of software code readability", International Journal of Engineering Research and Technology. ESRSA Publications,**(2012)**August, vol. 1, no. 6.
[8] X. Wang, L. Pollock and K. Vijay-Shanker, "Automatic segmentation of method code into meaningful blocks to improve readability", Reverse Engineering (WCRE), 2011 18th Working Conference on IEEE,**(2011)**.
[9] E. Collar Jr. and R. Valerdi,“Role of software readability on software development cost”,**(2006)**.
[10] “The Automated Readability Index (ARI)”, http://www.readabilityformulas.com/automatedreadability-index.php.
[11] “The smog readability formula”, http://www.readabilityformulas.com/smog-readability-formula.php.
[12] “The Gunning’s Fog Index (or FOG) Readability Formula”, http://www.readabilityformulas.com/gunning-fog-readabilityformula. Php.
[13] J. L. Elshoff and M. Marcotty, “Improving computer program readability to aid modification”, Communications of the ACM, vol. 25, no. 8,**(1982)**, pp. 512-521.
[14] http://www.mang.canterbury.ac.nz/writing_guide/writing/flesch.shtml.
[15] R. Namani1 and J. Kumar, “A New Metric for Code Readability”, IOSR Journal of Computer Engineering, vol. 6, Issue 6,**(2012)** November-December.

Authors

Aisha Batool, she is working on her MS thesis at the Department of Computing and Technologies, Iqra University, Islamabad, Pakistan. She is working on data analysis methods using artificial intelligence theories. Her research covers multiple disciplines within computer science and software engineering including mobile computing, image processing, artificial intelligences, and big data. She has two journal publications and overall she has more than 03 years' research experience in CS and IT.

Muhammad Habib ur Rehman, he is a third year PhD student at Department of Computer Systems and Technology, University of Malaya, Kuala Lumpur, Malaysia. He is working on big data mining systems for Internet of Things. His research covers a wide spectrum of application areas including smart cities, mobile social networks, Quantified self, mHealth and wearable assistive technologies among many others. The key research areas of his interest are: mobile computing, edge-cloud computing, Internetof Things, data mining, machine learning, and mobile distributed analytics.

Aihab Khan, he is assistant professor at Iqra University Islamabad Pakistan. His expertise is in Information Systems (Business Informatics), Computer Communications (Networks), Computer Security and Reliability. He has overall 45 publications.

Amsa Azeem, she is working on her MS thesis at the Department of Computing and Technologies, Iqra University, Islamabad, Pakistan, she is working on representation and quantification of non-functional requirement. Her research covers the software requirement engineering. Overall she has 01 year research experience in software engineering.