Code Size

Definition : Lines of code (LOC) are the "source code" of the program, and one line may generate one machine instruction or several depending on the programming language. Almost every compiler shows total lines of code without specifying how many blank lines, comment lines, data declarations or headings.

[Executable Statement (ES) ignores comments, data declaration and heading. Delivered source instruction (DSI) includes data declarations and headings as source instructions but not comment and blank lines]

Measuring tecniques

- **Total size**, **LOC = NCLOC + CLOC**

  Here,

  NCLOC = Non-commented line of code

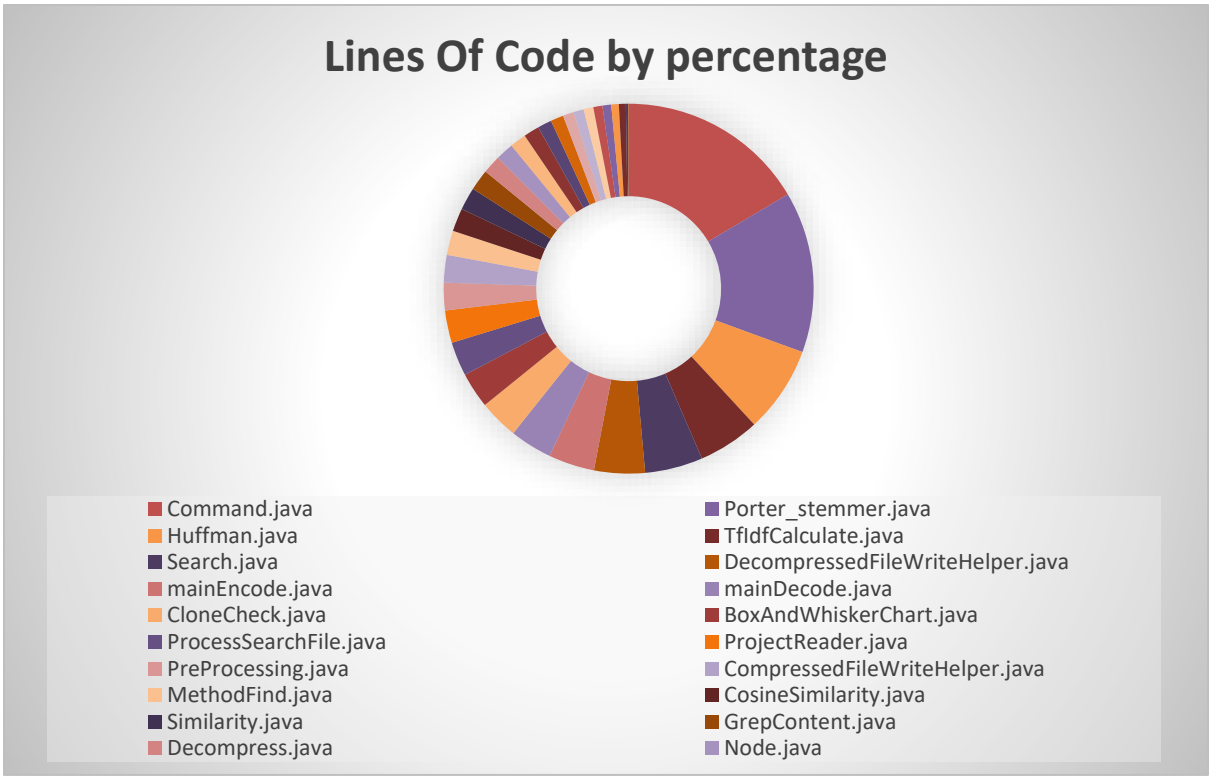
  CLOC = Commented line of code

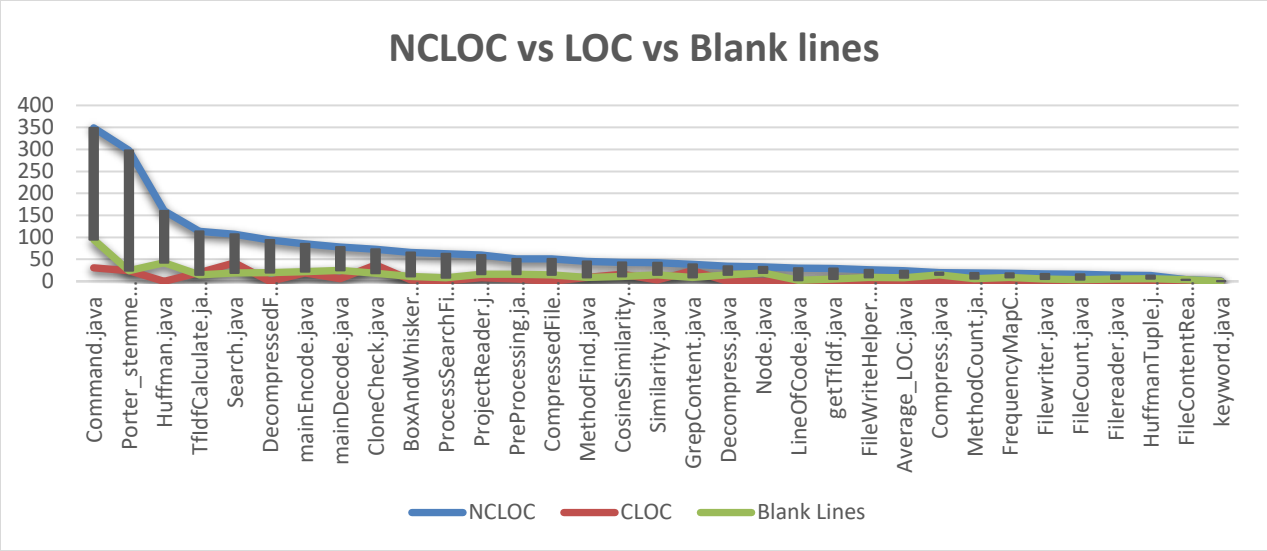
  - Measure procedure: Compiler and manual check.
  - Language : All.

Class Name	LOC (%)	NCLOC	CLOC	Blank_Lines	Total
Command.java	16.49 %	349	31	95	475
Porter_stemmer.java	14.08 %	298	24	24	346
Huffman.java	7.56 %	160	0	42	202
TfIdfCalculate.java	5.39 %	114	20	15	149
Search.java	5.06 %	107	42	19	168
DecompressedFileWriteHelper.java	4.44 %	94	0	20	114
mainEncode.java	4.02 %	85	19	22	126
mainDecode.java	3.69 %	78	6	25	109
CloneCheck.java	3.45 %	73	37	18	128
BoxAndWhiskerChart.java	3.12 %	66	3	11	80
ProcessSearchFile.java	2.98 %	63	1	8	72
ProjectReader.java	2.84 %	60	8	16	84
CompressedFileWriteHelper.java	2.41 %	51	0	15	66
PreProcessing.java	2.41 %	51	5	16	72
MethodFind.java	2.13 %	45	9	9	63
CosineSimilarity.java	2.03 %	43	16	11	70
Similarity.java	1.98 %	42	3	15	60

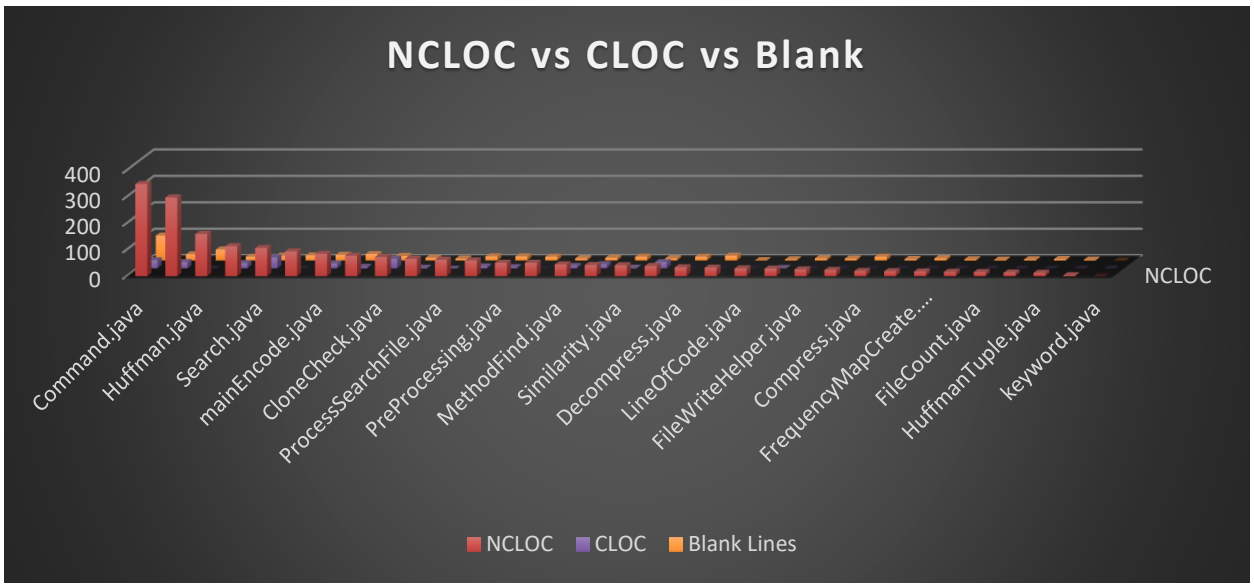
GrepContent.java	1.84 %	39	24	9	72
Decompress.java	1.61 %	34	0	15	49
Node.java	1.56 %	33	0	19	52
LineOfCode.java	1.42 %	30	0	2	32
getTfIdf.java	1.37 %	29	2	5	36
FileWriteHelper.java	1.23 %	26	0	9	35
Average_LOC.java	1.13 %	24	0	8	32
Compress.java	0.95 %	20	0	15	35
MethodCount.java	0.90 %	19	0	6	25
FrequencyMapCreate.java	0.85 %	18	0	9	27
Filewriter.java	0.80 %	17	0	5	22
FileCount.java	0.76 %	16	0	4	20
Filereader.java	0.66 %	14	0	5	19
HuffmanTuple.java	0.61 %	13	0	6	19
FileContentReader.java	0.19 %	4	0	4	8
keyword.java	0.05 %	1	0	0	1
Project : coding_helper	100.00 %	2,116	250	502	2,868

- **Total LOC = 2116 + 250 = 2366**
- **Average LOC per class = 2366/33 = 182**
- **Average NCLOC per class = 2116/33 = 64.121212**
- **Average CLOC = 250/33 = 7.5757**



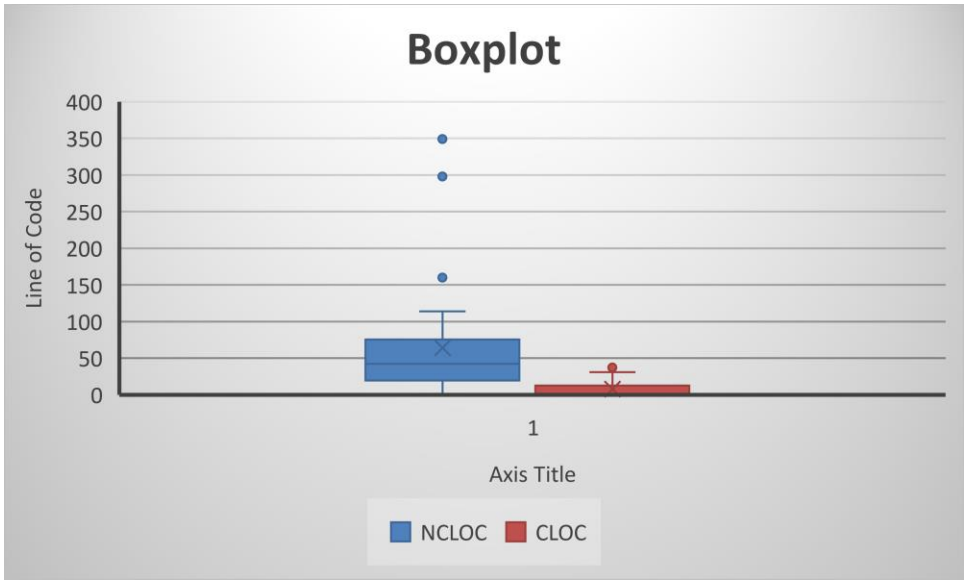


It is a 3D bar graph which show difference of line of codes by class name.



- **Measuring tool :** Rstudio, Microsoft excel, Sloc, Netbeans.

- **Boxplot:** It is a simple way of representing statistical data on a plot in which a rectangle is drawn to represent the second and third quartiles, usually with a vertical line inside to indicate the median value. The lower and upper quartiles are shown as horizontal lines either side of the rectangle.



- **Central Trendency and quartiles:** The tendency for the values of a random variable to cluster round its mean, mode, or median. Quartile defines range of data.

Measure	NCLOC	CLOC
Mean	164.121212	7.577575
Median	142	0
Mode	45	0
1st quartile	119.5	0
3rd quartile	175.5	12.5

- **Comment Density of program:** This measurement accomplished dividing commented line by line of codes.

CDoP = CLOC / LOC

- Measure procedure: programatic.
- Language : All.

Java program for density of comment in individual class

```
1   package software_metrics_lab_final;
2
3   import java.util.Scanner;
4
5   public class DensityOfProgram {
6
7     public static void main(String args[])
8   {
9   int x, y ;
0   float dop;
1   System.out.println("***   Density of Program   ***\n");
2
3   Scanner sc = new Scanner(System.in);
4   System.out.print("Enter the Commented Line Of Code: ");
5   x = sc.nextInt();
6   System.out.print("Enter the Non-commented Line Of Code: ");
7   y = sc.nextInt();
8   //dop = dop(x, y);
9   dop = (float)x/y;
0   System.out.println("Density of the program: " +dop);
1       System.out.println("\n\n");
2
3    }
4   }
```

```
Output - Software_Metrics_Lab_Final (run)
run:
***   Density of Program   ***

Enter the Commented Line Of Code: 31
Enter the Non-commented Line Of Code: 349
Density of the program: 0.08882522
```

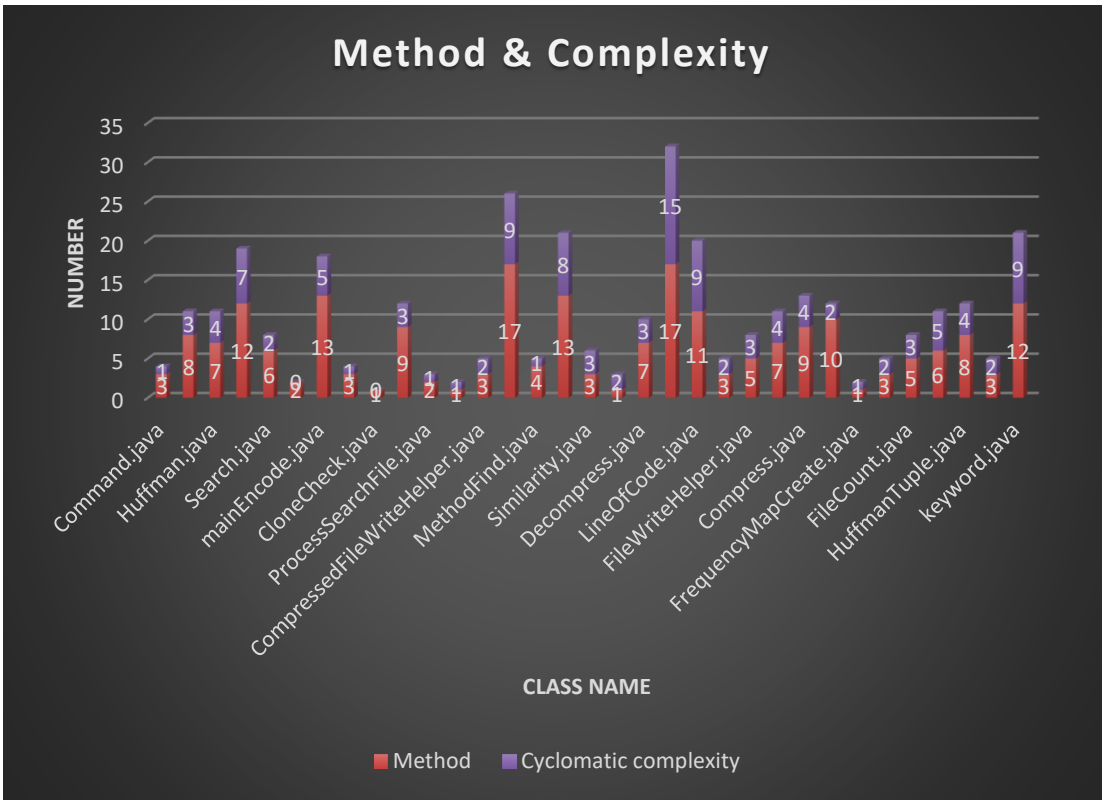
So, DCoP for command.java is 0.08882522

Total Density of comments: 6.7991631799163175

- **Number of Methods and cyclomatic complexity:** Method is a block of code in a program and complexity measure criticality of that method.

Class Name	Method	Cyclomatic complexity
Command.java	3	1
Porter_stemmer.java	8	3
Huffman.java	7	4
TfIdfCalculate.java	12	7
Search.java	6	2
DecompressedFileWriteHelper.java	2	0
mainEncode.java	13	5
mainDecode.java	3	1
CloneCheck.java	1	0
BoxAndWhiskerChart.java	9	3
ProcessSearchFile.java	2	1
ProjectReader.java	1	1
CompressedFileWriteHelper.java	3	2
PreProcessing.java	17	9
MethodFind.java	4	1
CosineSimilarity.java	13	8
Similarity.java	3	3
GrepContent.java	1	2
Decompress.java	7	3
Node.java	17	15
LineOfCode.java	11	9
getTfIdf.java	3	2
FileWriteHelper.java	5	3
Average_LOC.java	7	4
Compress.java	9	4
MethodCount.java	10	2
FrequencyMapCreate.java	1	1
Filewriter.java	3	2
FileCount.java	5	3
Filereader.java	6	5
HuffmanTuple.java	8	4
FileContentReader.java	3	2
keyword.java	12	9
Project : coding_helper	**215**	**121**

- **Average method per class = 6.51**
- **Average complexity per class = 3.67**



- **Halstead Approach**: Halstead's goal was to identify measurable properties of software, and the relations between them.
  - **Volume of Program , $V = N \times \log_2 \mu$**
  - **$N = N1 + N2$ & $\mu = \mu 2 + \mu 2$**

Here,

$\mu 1$ = Number of unique operators

$\mu 2$ = Number of unique operands

N1 = Total occurrences of operators

N2 = Total occurrences of operands

- Measure procedure: Programatic.
- Language : All.

#java program for halstead

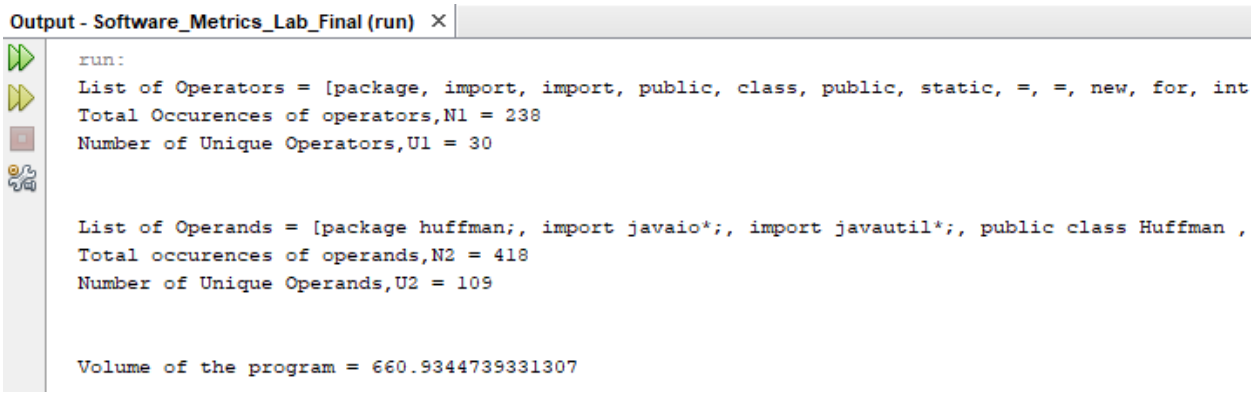
```java
package software_metrics_lab_final;

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Scanner;
import java.util.Set;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class HalsteadComplexity {

    static String java;

public static void main(String[] args){
//    public static void (Iterable<String> datatypes){}
//    public static void main(String[] args, Iterable<String> datatypes, BufferedReader reader)
//            throws FileNotFoundException {
        String[] keywords= {"public", "System", "out","println", "void", "class", "string", "int", "static", "abstract",
              "assert", "boolean", "break", "Byte", "Case", "Catch", "Char", "Const", "Continue", "Default", "Do", "Double",
              "Else", "enum", "Extends", "Final", "Finally", "Float", "For", "goto", "If", "implements", "import", "Instance
              "interface", "long", "native", "new", "package", "private", "protected", "return", "short", "strictfp", "super"
              "switch", "synchronized", "this", "throw", "throws", "transient", "try", "volatile", "while","+", "-", "*", "/"
              "--", "%", "==", "!=", "=", "<", ">", "<=", ">=", "&", "|","^", "~", ">>", "<<", ">>>", "&&", "||", "!", "=", "
              "*=", "/=", "%=", "<<=", ">>=", "&=","^=", "|=","(",")","{","}","[","]",".",",",";",":","\"","'","<","?","php",
              "$_POST","$_FILES","PATHINFO_EXTENSION","mysqli_query","SELECT"," FROM","WHERE","echo","alert","<script>","</sc
              "<!doctype html>","</html>","<head>","</head>","href","link","stylesheet","<title>","</title>","<style>","</sty
```

```java
        ArrayList<String> operators = new ArrayList<String>();
        ArrayList<String> operands = new ArrayList<String>();
        ArrayList<String> variables = new ArrayList<String>();
        int operatorCount = 0, operandCount = 0;

        try {
            File file = new File("C:\\Users\\User\\Desktop\\6th Semester\\LAB\\SMF\\coding_help new\\Huffman.java");
            Scanner input = new Scanner(file);

            int count = 0;
            boolean multiLineComment = false;
            while (input.hasNext()) {

                    String word = input.next()
                            .replaceAll("\\[", "")
                            .replaceAll("\\]", "")
                            .replaceAll("\\{", "")
                            .replaceAll("\\}", "")
                            .replaceAll("\\.", "");

                    for (String dd : word.split("\\(")) {
                        for (String keyword : keywords) {
                            if (dd.equalsIgnoreCase(keyword)) {
                                operators.add(dd);
                                operatorCount++;
                            }

                        }
                    }
                    count = count + 1;
                }
```

```
63
64              System.out.println("List of Operators = "+operators);
65
66              //int N1 = operatorCount;
67              //System.out.println("Total Occurences of operators = " +N1);
68              //System.out.println("Count = "+count);
            } catch (Exception e) {
                e.printStackTrace();
71          }
72
73                int N1 = operatorCount;
74                System.out.println("Total Occurences of operators,N1 = " +N1);
75
77              Set<String> hash_Set = new HashSet<String>(operators);
78              //System.out.println("operators uniques : "+hash_Set);
79
                int unique_operators=0;
                  for (String value : hash_Set)
81                {
82                unique_operators++;
83                }
84
85                int u1 = unique_operators;
86                System.out.println("Number of Unique Operators,U1 = "+u1);
87                System.out.println("\n");
88
89
90          String fileName = "C:\\Users\\User\\Desktop\\6th Semester\\LAB\\SMF\\coding_help new\\Huffman.java";
91          List<String> list = new ArrayList<>();
```

```
92
93              try (Stream<String> stream = Files.lines(Paths.get(fileName))) {
94
95                  list = stream
96                          .filter(line -> !line.contains("//"))
97                          .filter(line -> !line.isEmpty())
98                          .map(line -> line.replaceAll("\\}",""))
99                          .map(line -> line.replaceAll("\\{",""))
.00                       .map(line -> line.replaceAll("\\(",""))
.01                       .map(line -> line.replaceAll("\\)",""))
.02                       .map(line -> line.replaceAll("\\[",""))
.03                       .map(line -> line.replaceAll("\\]",""))
.04                       .map(line -> line.replaceAll("\\.",""))
.05                       .collect(Collectors.toList());
.06             } catch (IOException e) {
                    e.printStackTrace();
.08             }
.09
.10             for (String keyword : list) {
.11                operators.remove(keyword);
.12                 int x = operatorCount++;
.13             }
.14
.15                 System.out.println("List of Operands = "+list);
.16                 int N2 = operatorCount;
.17                 System.out.println("Total occurences of operands,N2 = " +N2);
.18
                    Set<String> hash_Set1 = new HashSet<String>(list);
.20                 //System.out.println("operators uniques : "+hash_Set1);
.21                    int unique_operands=0;
                       for (String value : hash_Set1)
```

```
                for (String value : hash_Set1)
                {
                unique_operands++;
                }

                int u2 = unique_operands;
                System.out.println("Number of Unique Operands,U2 = "+u2);
                System.out.println("\n");


            int N = N1 + N2 ;
            int u = u1 + u2;
            double v = N + Math.log(u);
            System.out.println("Volume of the program = "+ v +"\n\n");

        }
    }
```



It’s an automatic process, we only have to provide class path and then it will calculate result.

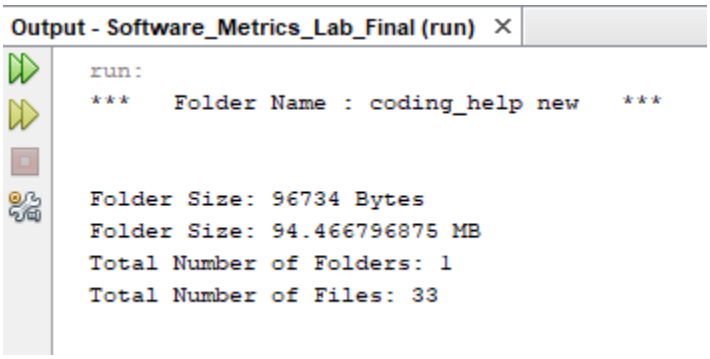
Huffman.java volume is 660.9344739 (largest class)

Total volume of the project : 14651

- **Byte of code**
    - Measure procedure: Programatic
    - Language : All.

#java program for file and folder size

```java
package software_metrics_lab_final;

import java.io.File;

public class GetFolderSize {

    int totalFolder = 0;
    int totalFile = 0;

    public static void main(String args[]) {
        String folder = "C:\\Users\\User\\Desktop\\6th Semester\\LAB\\SMF\\coding_help new";
        try {
            GetFolderSize size = new GetFolderSize();
            long fileSizeByte = size.getFileSize(new File(folder));
            System.out.println("\n\nFolder Size: " + fileSizeByte + " Bytes");
            double fileSizeMB = (float)fileSizeByte/1024;
            System.out.println("Folder Size: " +fileSizeMB+ " MB");
            System.out.println("Total Number of Folders: "
                + size.getTotalFolder());
            System.out.println("Total Number of Files: " + size.getTotalFile());
            System.out.println("\n\n\n");
        } catch (Exception e) {}
    }

    public long getFileSize(File folder) {
        totalFolder++;
        System.out.println("***   Folder Name : " + folder.getName()+"   ***");
        long foldersize = 0;
        File[] filelist = folder.listFiles();
        for (int i = 0; i < filelist.length; i++) {
            if (filelist[i].isDirectory()) {
                foldersize += getFileSize(filelist[i]);
            } else {
                totalFile++;
                foldersize += filelist[i].length();
            }
        }
        return foldersize;
    }

    public int getTotalFolder() {
        return totalFolder;
    }

    public int getTotalFile() {
        return totalFile;
    }
}
```



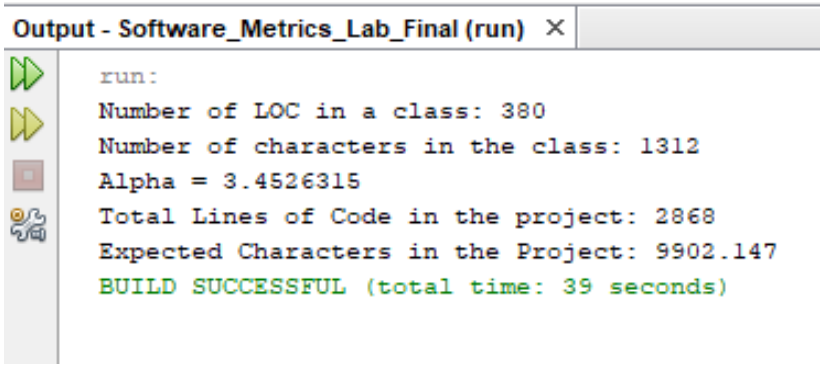
- **Number of characters**

  - **CHAR = α LOC**

Here, Alpha = Average character in a line of code

- Measure procedure: Programatic. [UNIX & Linux operating systems have the command **<wc>** to compute it]
- Language : APL, LISP, C, C++, Java.

```java
package software_metrics_lab_final;

import java.util.Scanner;

public class NumberOfCharacters {
    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

         System.out.print("Number of LOC in a class: ");
         int loc = scan.nextInt();

         System.out.print("Number of characters in the class: ");
         int ch = scan.nextInt();

          float alpha = (float) ch / loc;
         System.out.println("Alpha = "+alpha);

         System.out.print("Total Lines of Code in the project: ");
         int LOC = scan.nextInt();
         float total_char = alpha * LOC;

         System.out.println("Expected Characters in the Project: "+total_char);
    }
}
```



- **Number of override methods:** Methods which are re-implemented in other classes.
  - **NoORM = 25**

- **Number of overloaded methods:** Multiple methods can have the same name with different parameters
  - **NoOLM = 8**