Project Name: ChitChat

Submitted By: Md Al Adnan(MUH1825008M)

Code Size

Total Counting lines of code (LOC)
Definition: Counting number of physical lines (including blank lines, comment lines).
Measurement Way: Programmatic
TOTAL LOC- 3630
Total Commented lines of code (CLOC)
Definition: Counting the commented lines of code. Comment can be single line comment (//) and multi-line comment (/**/).
Measurement Way: Programmatic
TOTAL CLOC- 259
Total Non commented lines of code (NCLOC)
Definition: Counting all lines excluding blank lines and commented lines of code. Called effective lines of code.
Measurement Way: Programmatic
TOTAL NCLOC- 2662
Total Density of comments
Definition: Can be derived by -CLOC / (NCLOC + CLOC)
Measurement Way: Programmatic
TOTAL TDC- 7.134986225895316
Total Number of bytes of computer storage
Definition: Number of bytes used in the computer storage for the program text.
Measurement Way: Programmatic
TOTAL Byte-131328
Total Number of characters
Definition: Number of characters in the program text.
Measurement Way: Programmatic

TOTAL Characters- 125438

Average number of characters per line

Definition: Calculating average number of characters per line. Can be done by total number of characters / LOC

Measurement Way: Programmatic

Average number of characters per line: 34.555

Total Blank Line

Definition: Blank lines, space characters and tabs make programs easier to read.

Measurement Way: Programmatic

Total Blank Line: 709

Project Cyclomatics complexity

Definition: Calculated by v(F) = 1 + d (d is the number of decision nodes where decision nodes are if...else, do...while, while, for loops)

Measurement Way: Programmatic

Project Cyclomatics complexity : 280

Total Halstead's Program Volume

Definition: A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands. Halstead's metrics are included in a number of current commercial tools that count software lines of code.

Measurement Way: Programmatic

Total Halstead's Program Volume: 2997557

Average Halstead's Program Volume per class

Measurement Way: Programmatic

Halstead's Program Volume per class : 115290

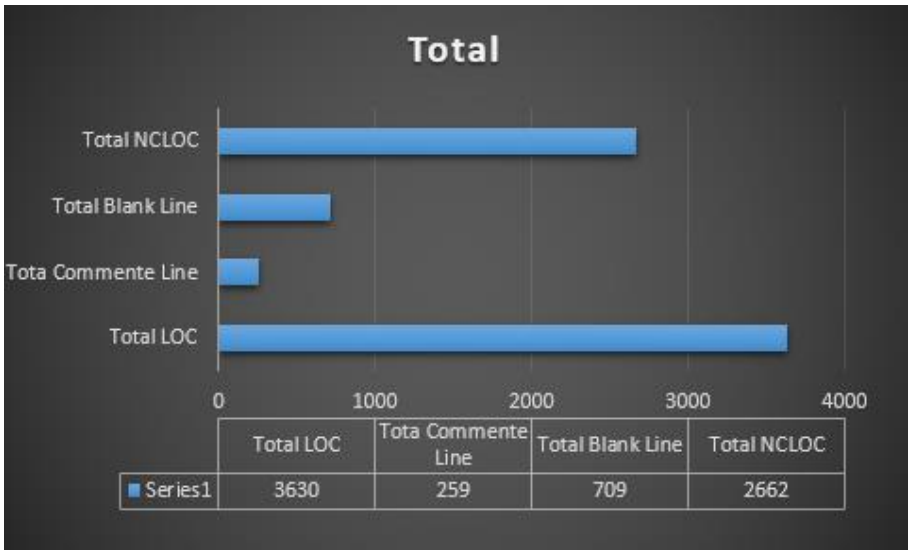
Total Density of comments

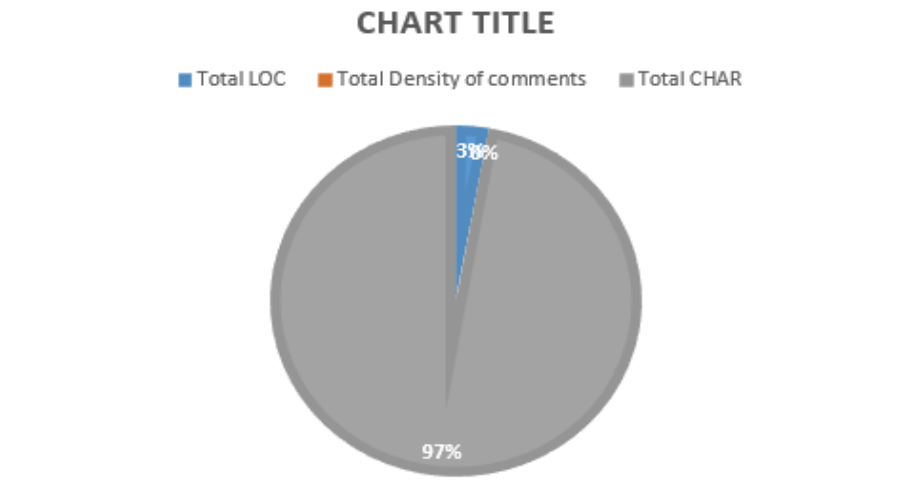
Definition: Comment density is the percentage of comment lines in a given source code base, that is, comment lines divided by total lines of code.

Measurement Way: Programmatic

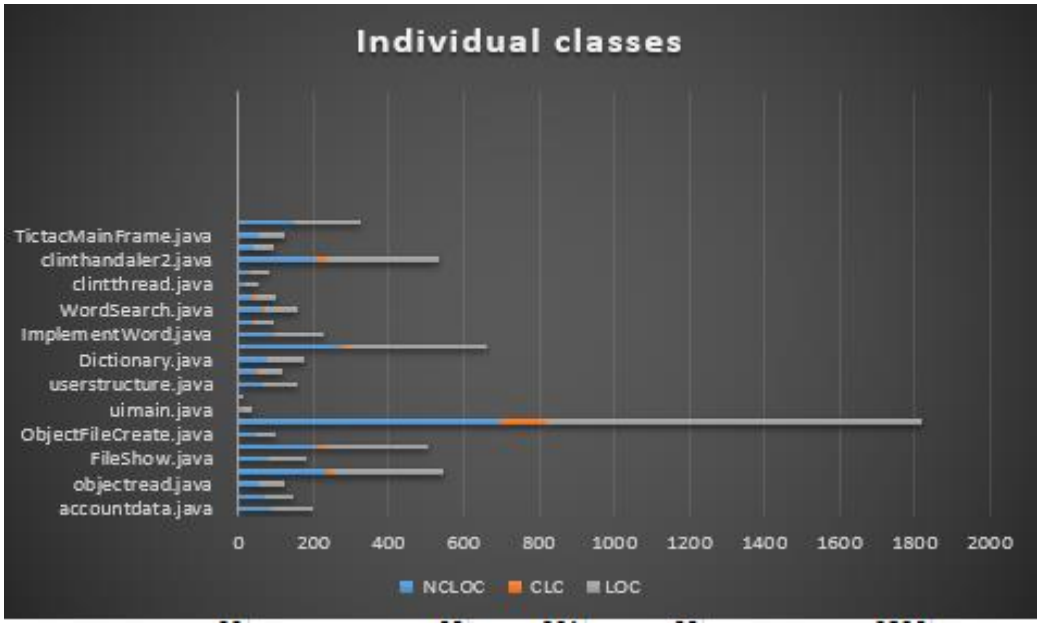
Total Density of comments: 7.134986225895316

Graphical Representation





Individual Classes Vs Metrics Value



accountdata.java

LOC: 109
Commente Line: 0
Blank Line: 19
Data Declaration: 12
NCLOC: 90
Density of comments: 0.0
Number of bytes: 3309
Total CHAR: 3091
Number of methods 17
Class Cyclomatics complexity: 13
Halstead;s Program Volume : 3180

==========

objectread.java

LOC: 67
Commente Line: 0
Blank Line: 14
Data Declaration: 5
NCLOC: 53
Density of comments: 0.0
Number of bytes: 1859
Total CHAR: 1725
Number of methods 10
Class Cyclomatics complexity: 1
Halstead;s Program Volume : 8414

chathistory.java

LOC: 79
Commente Line: 0
Blank Line: 11
Data Declaration: 11
NCLOC: 68
Density of comments: 0.0
Number of bytes: 2161
Total CHAR: 2082
Number of methods 10
Class Cyclomatics complexity: 8
Halstead;s Program Volume : 6230

==========

createaccountWindow.java

LOC: 299
Commente Line: 16
Blank Line: 52
Data Declaration: 21
NCLOC: 231
Density of comments: 5.351170568561873
Number of bytes: 12260
Total CHAR: 11662
Number of methods 21
Class Cyclomatics complexity: 17
Halstead;s Program Volume : 27296

FileShow.java

LOC: 100
Commente Line: 4
Blank Line: 19
Data Declaration: 6
NCLOC: 77
Density of comments: 4.0
Number of bytes: 3164
Total CHAR: 3064
Number of methods 9
Class Cyclomatics complexity: 8
Halstead;s Program Volume : 32248

ObjectFileCreate.java

LOC: 55
Commente Line: 1
Blank Line: 10
Data Declaration: 6
NCLOC: 44
Density of comments: 1.8181818181818181
Number of bytes: 1460
Total CHAR: 1405
Number of methods 4
Class Cyclomatics complexity: 1
Halstead;s Program Volume : 57330
==========

uimain.java

LOC: 23
Commente Line: 3
Blank Line: 11
Data Declaration: 1
NCLOC: 9
Density of comments: 13.043478260869565
Number of bytes: 477
Total CHAR: 431
Number of methods 1
Class Cyclomatics complexity: 0
Halstead;s Program Volume : 117990
==========

userstructure.java

==========

LOC: 88
Commente Line: 0
Blank Line: 21
Data Declaration: 20
NCLOC: 67
Density of comments: 0.0
Number of bytes: 1727
Total CHAR: 1639
Number of methods 16
Class Cyclomatics complexity: 0
Halstead;s Program Volume : 121104
==========

loginwindow.java

LOC: 275
Commente Line: 24
Blank Line: 48
Data Declaration: 14
NCLOC: 203
Density of comments: 8.727272727272728
Number of bytes: 11372
Total CHAR: 10822
Number of methods 18
Class Cyclomatics complexity: 6
Halstead;s Program Volume : 55062

personalchatroomui.java

LOC: 1000
Commente Line: 123
Blank Line: 181
Data Declaration: 146
NCLOC: 696
Density of comments: 12.3
Number of bytes: 46211
Total CHAR: 44211
Number of methods 103
Class Cyclomatics complexity: 75
Halstead;s Program Volume : 117207
==========

useraccountlistener.java

LOC: 6
Commente Line: 0
Blank Line: 2
Data Declaration: 1
NCLOC: 4
Density of comments: 0.0
Number of bytes: 104
Total CHAR: 98
Number of methods 1
Class Cyclomatics complexity: 0
Halstead;s Program Volume : 118134
==========

DeleteListedWord.java

==========
LOC: 63
Commente Line: 4
Blank Line: 10
Data Declaration: 2
NCLOC: 49
Density of comments: 6.349206349206349
Number of bytes: 2047
Total CHAR: 1984
Number of methods 9
Class Cyclomatics complexity: 5
Halstead;s Program Volume : 124263
==========

Dictionary.java

```
==========
LOC: 98
Commente Line: 1
Blank Line: 22
Data Declaration: 14
NCLOC: 75
Density of comments: 1.0204081632653061
Number of bytes: 2549
Total CHAR: 2451
Number of methods 9
Class Cyclomatics complexity: 8
Halstead;s Program Volume : 128259
==========
```

ImplementWord.java

```
==========
LOC: 126
Commente Line: 5
Blank Line: 26
Data Declaration: 13
NCLOC: 95
Density of comments: 3.968253968253968
Number of bytes: 3901
Total CHAR: 3775
Number of methods 11
Class Cyclomatics complexity: 5
Halstead;s Program Volume : 171690
==========
```

WordSearch.java

```
==========
LOC: 86
Commente Line: 7
Blank Line: 15
Data Declaration: 8
NCLOC: 64
Density of comments: 8.13953488372093
Number of bytes: 2702
Total CHAR: 2616
Number of methods 10
Class Cyclomatics complexity: 5
Halstead;s Program Volume : 179650
==========
```

clintthread.java

```
LOC: 32
Commente Line: 1
Blank Line: 12
Data Declaration: 9
NCLOC: 19
Density of comments: 3.125
Number of bytes: 640
Total CHAR: 576
Number of methods 2
Class Cyclomatics complexity: 0
Halstead;s Program Volume : 183750
```

DictionaryMainFrame.java

```
==========
LOC: 371
Commente Line: 17
Blank Line: 80
Data Declaration: 7
NCLOC: 274
Density of comments: 4.5822102425876015
Number of bytes: 12125
Total CHAR: 11754
Number of methods 53
Class Cyclomatics complexity: 37
Halstead;s Program Volume : 164190
==========
```

ListOfWord.java

```
==========
LOC: 52
Commente Line: 4
Blank Line: 12
Data Declaration: 6
NCLOC: 36
Density of comments: 7.6923076923076925
Number of bytes: 1451
Total CHAR: 1399
Number of methods 9
Class Cyclomatics complexity: 1
Halstead;s Program Volume : 174580
==========
```

clintsocket.java

```
==========
LOC: 55
Commente Line: 9
Blank Line: 9
Data Declaration: 8
NCLOC: 37
Density of comments: 16.363636363636363
Number of bytes: 1669
Total CHAR: 1559
Number of methods 2
Class Cyclomatics complexity: 0
Halstead;s Program Volume : 182650
==========
```

file_name_transfer.java

```
LOC: 44
Commente Line: 5
Blank Line: 9
Data Declaration: 5
NCLOC: 30
Density of comments: 11.363636363636363
Number of bytes: 1170
Total CHAR: 1082
Number of methods 4
```

Class Cyclomatics complexity: 0

Halstead;s Program Volume : 185860

clinthandaler2.java

==========

LOC: 301
Commente Line: 29
Blank Line: 66
Data Declaration: 66
NCLOC: 206
Density of comments: 9.634551495016613
Number of bytes: 9806
Total CHAR: 9204
Number of methods 52
Class Cyclomatics complexity: 42
Halstead;s Program Volume : 202420

==========

TictacMainFrame.java

==========

LOC: 69
Commente Line: 4
Blank Line: 15
Data Declaration: 2
NCLOC: 50
Density of comments: 5.797101449275362
Number of bytes: 2158
Total CHAR: 2089
Number of methods 6
Class Cyclomatics complexity: 3
Halstead;s Program Volume : 209920

==========

serversocket.java

==========

LOC: 55
Commente Line: 2
Blank Line: 14
Data Declaration: 14
NCLOC: 39
Density of comments: 3.6363636363636362
Number of bytes: 1808
Total CHAR: 1698
Number of methods 4
Class Cyclomatics complexity: 4
Halstead;s Program Volume : 205440

==========

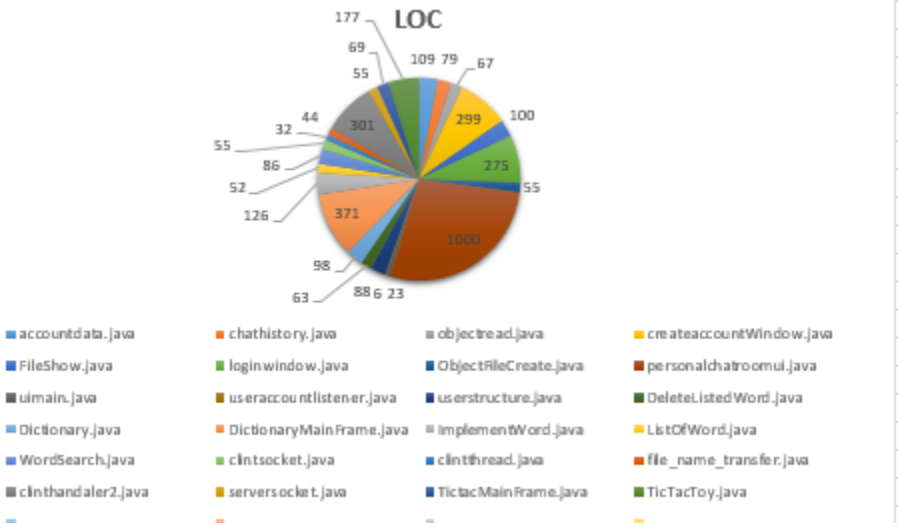
TicTacToy.java

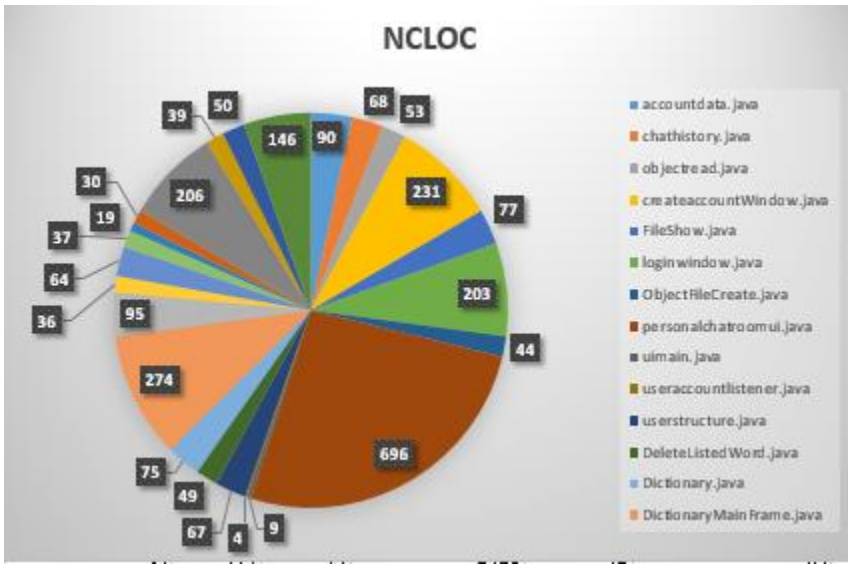
==========

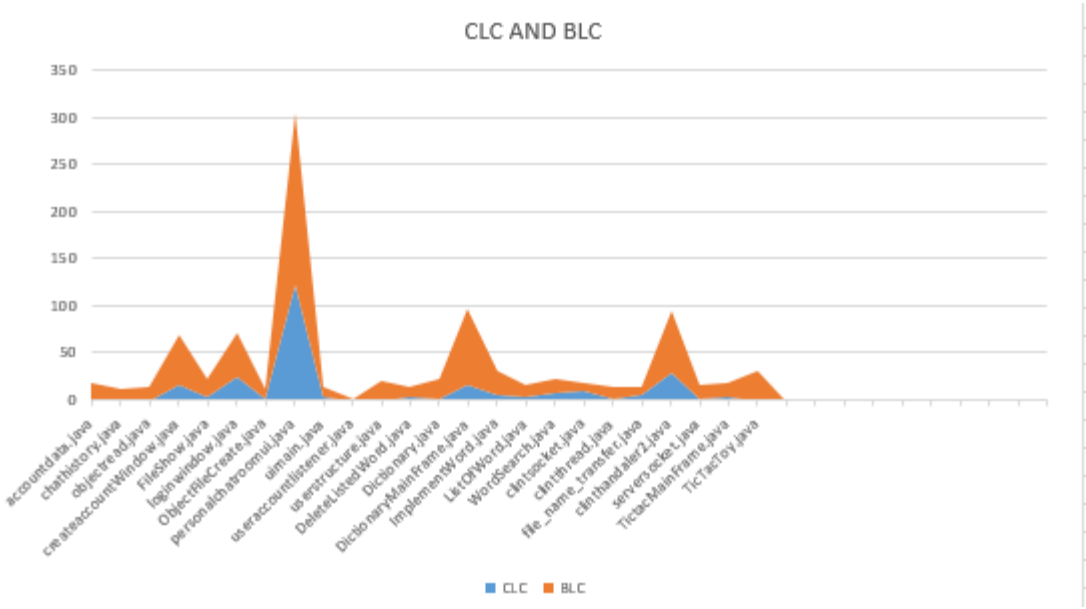
LOC: 177
Commente Line: 0
Blank Line: 31
Data Declaration: 22
NCLOC: 146
Density of comments: 0.0
Number of bytes: 5198
Total CHAR: 5021
Number of methods 25
Class Cyclomatics complexity: 41
Halstead;s Program Volume : 220690

Graphical Representations

Class Name	NCLOC	CLC	BLC	LOC	DD	TotalBytes	Methods	Cyclomatic_complex	Halstead's Volume
accountdata.java	90	0	19	109	12	3309	17	13	3180
chathistory.java	68	0	11	79	11	2161	10	8	6230
objectread.java	53	0	14	67	5	1859	10	1	8414
createaccountWindow.j	231	16	52	299	21	12260	21	17	27296
FileShow.java	77	4	19	100	6	3164	9	8	32248
loginwindow.java	203	24	48	275	14	11372	18	6	55062
ObjectFileCreate.java	44	1	10	55	6	1460	4	1	57330
personalchatroomui.jav	696	123	181	1000	146	46211	103	75	117207
uimain.java	9	3	11	23	1	477	1	0	117990
useraccountlistener.jav	4	0	2	6	1	104	1	0	118134
userstructure.java	67	0	21	88	20	1727	16	0	121104
DeleteListedWord.java	49	4	10	63	2	2047	9	5	124263
Dictionary.java	75	1	22	98	14	2549	9	8	128259
DictionaryMainFrame.ja	274	17	80	371	7	12125	53	37	164190
ImplementWord.java	95	5	26	126	13	3901	11	5	171690
ListOfWord.java	36	4	12	52	6	1451	9	1	174580
WordSearch.java	64	7	15	86	8	2702	10	5	179650
clintsocket.java	37	9	9	55	8	1669	2	0	182650
clintthread.java	19	1	12	32	9	640	2	0	183750
file_name_transfer.java	30	5	9	44	5	1170	4	0	185860
clinthandaler2.java	206	29	66	301	66	9806	52	42	202420
serversocket.java	39	2	14	55	14	1808	4	4	205440
TictacMainFrame.java	50	4	15	69	2	2158	6	3	209920
TicTacToy.java	146	0	31	177	22	5198	25	41	220690


LOC
177
69
55
109
79
67
299
100
275
55
1000
6
23
88
63
98
371
126
52
86
55
32
44
301
accountdata.java
chathistory.java
objectread.java
createaccountWindow.java
FileShow.java
loginwindow.java
ObjectFileCreate.java
personalchatroomui.java
uimain.java
useraccountlistener.java
userstructure.java
DeleteListedWord.java
Dictionary.java
DictionaryMainFrame.java
ImplementWord.java
ListOfWord.java
WordSearch.java
clintsocket.java
clintthread.java
file_name_transfer.java
clinthandaler2.java
serversocket.java
TictacMainFrame.java
TicTacToy.java


NCLOC
68
53
39
50
146
90
231
77
30
206
19
37
203
64
44
36
95
274
696
75
49
67
4
9
accountdata.java
chathistory.java
objectread.java
createaccountWindow.java
FileShow.java
loginwindow.java
ObjectFileCreate.java
personalchatroomui.java
uimain.java
useraccountlistener.java
userstructure.java
DeleteListedWord.java
Dictionary.java
DictionaryMainFrame.java



Design Size

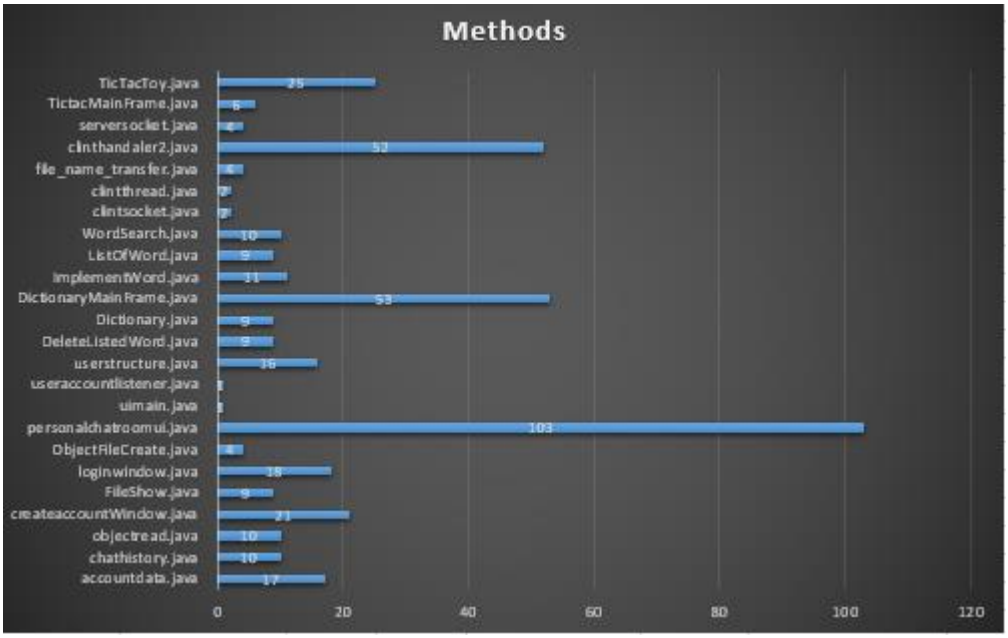
Number of Classes
Definition: Counting the number of classes used.
Type: Programmatic
Number of Classes: 26

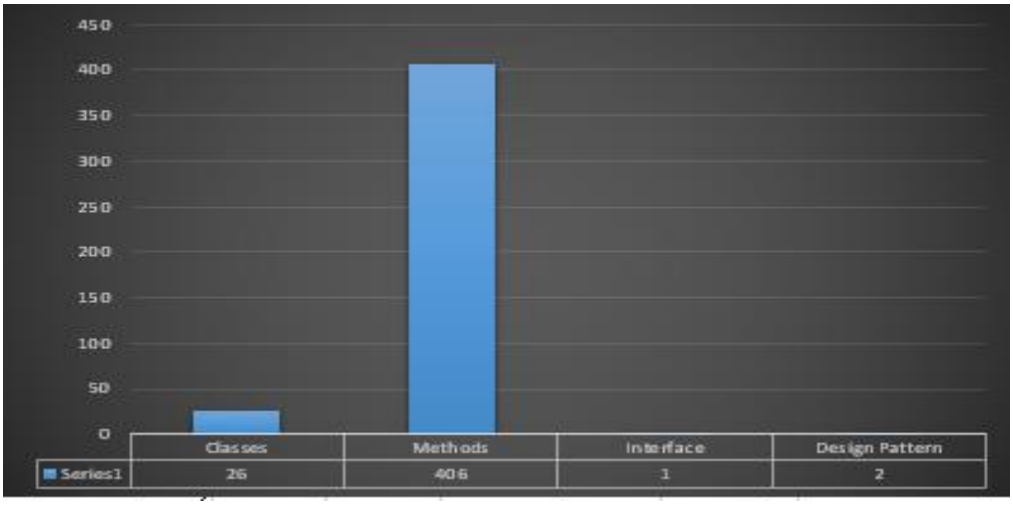
Number of Interfaces
Definition: Counting the number of classes used.
Type: Programmatic
Number of Interfaces : 1

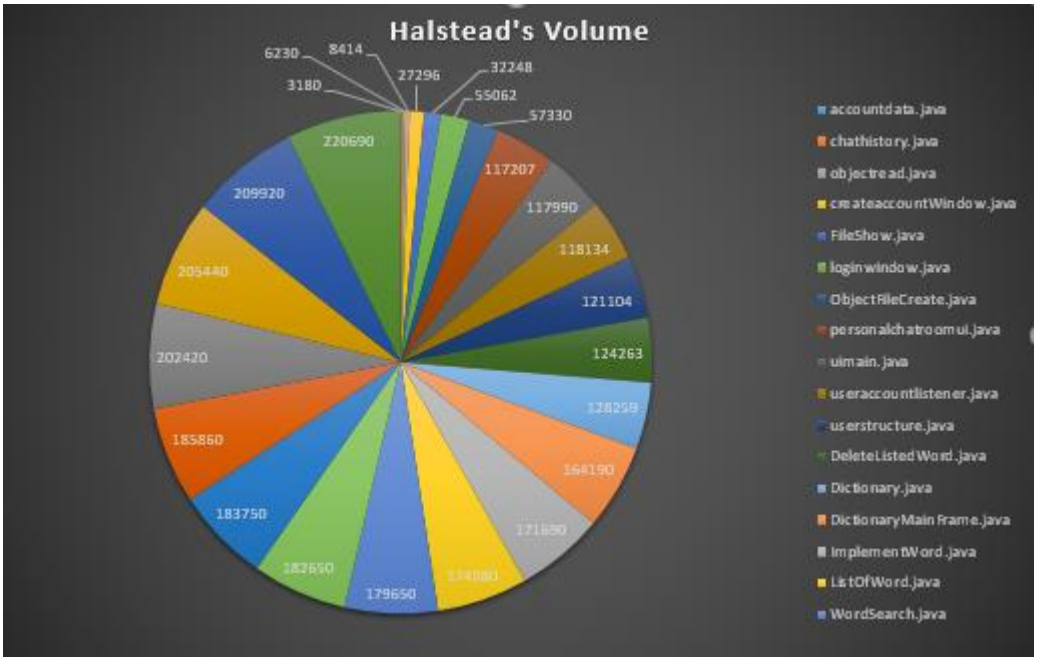
Number of design patterns
Definition: Number of design pattern used in the design.
Type: Manual
2

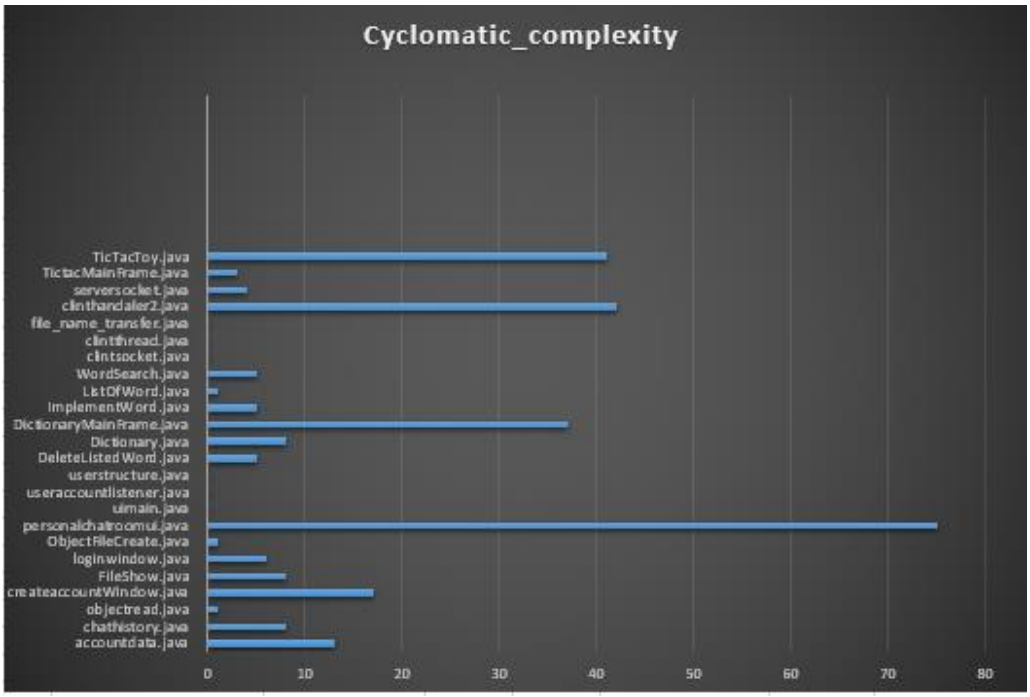
Weighted or methods average per class:
Definition: Calculate the weight of a method using cyclomatic complexity and finding out the weight of method for a class.
Type: Programmatic
Weighted or methods average per class: 15

Total Number of Packages:
Definition: A package in Java is used to group related classes.
Type: Programmatic
Total Number of Packages: 7
Total Number of Methods:
Definition: Number of methods use in a java class.
Type: Programmatic
Total Number of Methods: 406






Halstead's Volume
6230
8414
27296
32248
3180
55062
57330
220690
117207
209920
117990
118134
205440
121104
124263
202420
128259
185860
164190
183750
171690
182650
179650
accountdata.java
chathistory.java
objectread.java
createaccountWindow.java
FileShow.java
loginwindow.java
ObjectFileCreate.java
personalchatroomui.java
uimain.java
useraccountlistener.java
userstructure.java
DeleteListedWord.java
Dictionary.java
DictionaryMainFrame.java
ImplementWord.java
ListOfWord.java
WordSearch.java


Cyclomatic_complexity
TicTacToy.java
TictacMainFrame.java
serversocket.java
clinthandaler2.java
file_name_transfer.java
clintthread.java
clintsocket.java
WordSearch.java
ListOfWord.java
ImplementWord.java
DictionaryMainFrame.java
Dictionary.java
DeleteListedWord.java
userstructure.java
useraccountlistener.java
uimain.java
personalchatroomui.java
ObjectFileCreate.java
loginwindow.java
FileShow.java
createaccountWindow.java
objectread.java
chathistory.java
accountdata.java
0
10
20
30
40
50
60
70
80

Requirement analysis and specification size

SRS Requirements and specification documents generally combine text, graphs, and special mathematical diagrams and symbols. These document can consist of a mixture of text and diagrams.

Techniques:

Use case diagrams:

Number of use cases, actors, and relationships of various types.

- Measure procedure: Manually.

Use case:

Number of scenarios, size of scenarios in terms of steps, or activity diagram model elements.

- Measure procedure: Manually.
-

Number of actor	3
Number of Scenario	10
Number of Functional Req	10

Use Case	Tourist	Guide	Admin	No. of Actors	No. of Relationships	No. of Scenarios	Actors
Find Tour	1	1	0	2	1	6	Tourist, Guide
Search by Location	1	1	0	2	3	5	Tourist, Guide
Find Guide	1	1	0	2	1	3	Tourist, Guide
Check Profile	1	1	0	2	2	3	Tourist, Guide
Login as Tourist	1	0	0	1	1	1	Tourist
Log in as Guide	0	1	0	1	1	1	Guide
Confirm Guide	1	1	0	2	1	3	Tourist, Guide
Check Feedback	1	1	0	2	1	2	Tourist, Guide
Give Feedback	1	1	0	2	1	2	Tourist, Guide
Register as a Tourist	1	0	0	1	1	2	Tourist
Register as a Guide	0	1	1	2	1	2	Guide, Admin

Scenario	Size of Scenario (Steps)	Activity Diagram Model Elements	Functional Req.	Non-Functional Req.
SC1	4	7	2	3
SC2	2	6	4	2
SC3	3	7	3	4
SC4	3	6	4	2
SC5	3	5	2	1
SC6	3	6	1	5
SC7	4	5	3	6
SC8	3	7	4	2
SC9	4	7	4	2
SC10	4	6	2	4

A	B	C	D
Associated Usecase	**Actors**	**No. of Relationship**	**No. of Scenarios**
Find Tour	Tourist	1	6
Find Tour	Guide	1	6
Search by Location	Tourist	3	5
Search by Location	Guide	3	5
Find Guide	Tourist	1	3
Find Guide	Guide	1	3
Check Profile	Tourist	2	3
Check Profile	Guide	2	3
Login as Tourist	Tourist	1	1
Log in as Guide	Guide	1	1
Confirm Guide	Tourist	1	3
Confirm Guide	Guide	1	3
Check Feedback	Tourist	1	2
Check Feedback	Guide	1	2
Give Feedback	Tourist	1	2
Give Feedback	Guide	1	2
Register as a Tourist	Tourist	1	2
Register as a Guide	Guide	1	2
Register as a Guide	Admin	1	2

Function Points of Tour guide

FPs are intended to measure the amount of functionality in a system as described by a Software Requirement Specification. Computing FPs we first compute an unadjusted function point count (UFC).

UFC depends on five factors.

1. External inputs
2. External outputs
3. External inquiries
4. External files
5. Internal files

$$\text{UFC} = \sum_{i=1}^{15} (\text{Number of items of variety } i) \times (\text{weight}_i)$$

First, find out the Function Point Complexity Weights

Item	Simple	Average	Complex
External input	3	4	6
External output	4	5	7
External inquires	3	4	6
External files	7	10	15

Internal files	5	7	10

Find out all the factors for the tour guide project.

- External Inputs

Item	Complexity	Weight	In Total
Tourists find out tour guide	simple	3	Simple ->4 Average ->3 Complex->2
Tourists search for a spot	simple	3	
Tourist gives the registration info	simple	3	
Guide gives the registration info	average	4	
Tourist provides feedback to guide	complex	6	
Tourists search for a specific spot	simple	3	
Change tourist profile	average	4	
Guide change his/her profile	average	4	
Tourists confirm a guide	complex	6	9

- External output

Item	complexity	weight	In total
Tourist profile create	simple	4	Simple->3 Average->2 Complex->3
Guide profile create	complex	7	
Tourists show the guide list	simple	4	
Tourists show the tour spot list	simple	4	
Tourist profile update	average	5	
Guide profile update	average	5	
Guide confirms successfully	complex	7	
Guide shows the tourist feedback	complex	7	8

- External Inquiries

Item	complexity	weight	In total
The system can manage tour guide	complex	6	Simple->1 Average->2 Complex->2
Guide Confirmation fee manage by system	complex	6	
Manage most visited tourist spot	average	4	
Calculate tour guide cost	average	4	
Calculate system fee in advance	simple	3	5

- External Files

Item	complexity	weight	In total
Provide trusted guide info file	average	10	Average->1 Complex->1
System fee per tour	complex	15	2

- Internal File

Item	complexity	weight	In total
Guide info file	simple	5	Simple->1 Average->2 Complex->1
System fee file	average	7	
Image directory	average	7	
System document	complex	10	4

Put all the value in UFC formula

UFC = {(3A +4A+6A)+(4B+5B+7B)+ (3C+4C+6C)+ (7D+10D+15D) + (5E+7E+10E)}

=(3*4+4*3+6*2)+(4*3+5*2+7*3)+(3*1+4*2+6*2)+(7*0+10*1+15*1)+(5*1+7*2+10*1)

=48+43+23+25+29

UFC =168

Technical Complexity Factor

F_1 Reliable backup and recovery	F_2 Data communications
F_3 Distributed functions	F_4 Performance
F_5 Heavily used configuration	F_6 Online data entry
F_7 Operational ease	F_8 Online update
F_9 Complex interface	F_{10} Complex processing
F_{11} Reusability	F_{12} Installation ease
F_{13} Multiple sites	F_{14} Facilitate change

Each component/subfactor rated 0-5

- 0 means the subfactor is irrelevant
- 3 means the subfactor is average
- 5 means the subfactor is essential

$$\text{TCF} = 0.65 + 0.01 \sum_{i=1}^{14} F_i$$

We know that,

$$\text{FP} = \text{UFC} \times \text{TCF}$$

It seems reasonable to assume that F3, F5, F9, F11, and F13 are 0, that F1, F2, F6, F7, F8, F12 and F14 are 3, and that F4 and F10 are 5.

We calculate the TCF

TCF = 0.65 + 0.01(21 + 10) = 0.96

Now measure the functional points FP,

FP=UFC*TCF

=168*0.96

=161.28

Measuring Internal Structure Attributes

Cyclomatic complexity
Definition: Calculated by v(F) = 1 + d (d is the number of decision nodes where decision nodes are if...else, do...while, while, for loops)
Type: Manual
:48