SCRUTINIZE SOURCE CODE WITH THE INTENT TO TARGET AND UNMASKING THE CODE CLONES USING AUTOMATED TECHNIQUE.

Dissertation submitted in fulfilment of the requirements for the Degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

MANJIT KAUR

11502632

Supervisor

BHAVNEESH SOHAL

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

April 2017

TOPIC APPROVAL PERFORMA

School of Computer Science and Engineering

Program : P173::M.Tech. (Information Technology) [Full Time]

COURSE CODE : INT546 **REGULAR/BACKLOG :** Regular **GROUP NUMBER :** CSERGD0024

Supervisor Name : Bhavneesh Sohal **UID :** 16042 **Designation :** Assistant Professor

Qualification : _____ **Research Experience :** _____

SR.NO.	NAME OF STUDENT	REGISTRATION NO	BATCH	SECTION	CONTACT NUMBER
1	Manjit Kaur	11502632	2015	K1520	09646150946

SPECIALIZATION AREA : Program Methodology and Design **Supervisor Signature:** _____

PROPOSED TOPIC : Scrutinise source code with the intent to target and unmasking the code clones using automated technique.

Qualitative Assessment of Proposed Topic by PAC		
Sr.No.	Parameter	Rating (out of 10)
1	Project Novelty: Potential of the project to create new knowledge	7.33
2	Project Feasibility: Project can be timely carried out in-house with low-cost and available resources in the University by the students.	7.00
3	Project Academic Inputs: Project topic is relevant and makes extensive use of academic inputs in UG program and serves as a culminating effort for core study area of the degree program.	7.00
4	Project Supervision: Project supervisor's is technically competent to guide students, resolve any issues, and impart necessary skills.	6.67
5	Social Applicability: Project work intends to solve a practical problem.	7.00
6	Future Scope: Project has potential to become basis of future research work, publication or patent.	7.00

PAC Committee Members		
PAC Member 1 Name: Gaurav Pushkarna	UID: 11057	Recommended (Y/N): NA
PAC Member 2 Name: Mandeep Singh	UID: 13742	Recommended (Y/N): NA
PAC Member 3 Name: Er.Dalwinder Singh	UID: 11265	Recommended (Y/N): NA
PAC Member 4 Name: Balraj Singh	UID: 13075	Recommended (Y/N): Yes
PAC Member 5 Name: Harwant Singh Arri	UID: 12975	Recommended (Y/N): Yes
PAC Member 6 Name: Tejinder Thind	UID: 15312	Recommended (Y/N): NA
DAA Nominee Name: Kanwar Preet Singh	UID: 15367	Recommended (Y/N): Yes

Final Topic Approved by PAC: **Scrutinise source code with the intent to target and unmasking the code clones using automated technique.**

Overall Remarks: Approved

PAC CHAIRPERSON Name: 11011::Dr. Rajeev Sobti **Approval Date:** 26 Oct 2016

ABSTRACT

Today in this modern era of science, technology has developed its roots deep into the world, where most of the things are done with the help of automated tools and techniques to do more work in less time and with great efficiency. This is the case with software industry also. In software industries, a technique called software cloning has come into existence. Software cloning has various broad aspects, out of them; the shadow of light is thrown on one of the aspect called code cloning. In code cloning, some significant quantity of code as desired by the user is taken from some pre-existing code and copied into some another code. In short, it is a kind of copying or pasting of code where some desired code is copied from one source and pasted into another source. The code in which pasting is done is called the replica of original code. In other words, the code which contains the replicated code is called the clone. It leads to the fast development of the software systems. But despite of having so many boons like time saving technique, fast development of software systems, reuse, etc.; it also has some drawbacks as well like bug propagation, effect on maintenance, lack of originality ,plagiarism, effects software evolution, etc. [25]

So to overcome the problems related to cloning, now a day's clone detection has been an active research area. Lot of researches based of this has already been conducted till date to come up with efficient and effective clone detection techniques and tools to target code clones. To enhance code detection, various clone removal techniques are also there on which a parallel research work is going on along with clone detection. But the emphasis of this proposal in purely based on clone detection and its techniques and tools.[25]

DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation entitled " SCRUTINIZE SOURCE CODE WITH THE INTENT TO TARGET AND UNMASKING THE CODE CLONES USING AUTOMATED TECHNIQUE" in partial fulfilment of the requirement for the award of Degree for Master of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Bhavneesh Sohal. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

Manjit Kaur

R.No 11502632

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the M.Tech Dissertation entitled "**SCRUTINIZE SOURCE CODE WITH THE INTENT TO TARGET AND UNMASKING THE CODE CLONES USING AUTOMATED TECHNIQUE"**, submitted by **Manjit Kaur** at **Lovely Professional University, Phagwara, India** is a bonafide record of her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

Bhavneesh Sohal

Date:

Counter Signed by:

1) **Concerned HOD:**

   HoD's Signature: _____

   HoD Name: _____

   Date: _____

2) **Neutral Examiners:**

   External Examiner

   Signature: _____

   Name: _____

   Affiliation: _____

   Date: _____

   Internal Examiner

   Signature: _____

   Name: _____

   Date: _____

ACKNOWLEDGEMENT

I find myself highly blessed to get this opportunity to acknowledge all those people who are directly or indirectly help me in my dissertation work. I would like to extend my warm thanks towards all of them. With their help, cooperation and guidance only, that today I am successfully able to come up with my proposal.

To start with, I would first like to thanks the almighty for his endless grace and mercy upon me that always show me the way or ray of hope in my dark or tough times during my dissertation work when I found it hard and facing difficulties, finding the best proposal to be work upon. I am highly thankful to God for that entire he did for in the past, doing in the present and will also does in the future.

I am highly grateful to Mr. Dalwinder Singh, Head of Department, School of Computer Science and Engineering for giving me all the facilities that are needed for successful completion of the thesis. I am also thankful to all other faculty members and staff for their kind support.

Next I would like to thanks my guide Mr. Bhavneesh Sohal for giving me such an interesting topic to be worked upon. My research journey in this field till date is really knowledgeable and enjoyable.

Then I would like to extend my great thanks towards Ms. Sandeep Kaur for giving me direction and advices on regular basis that how I should conduct my study in this field. I would like to thanks her again for giving me her precious time.

Last but not the least,I would like to thanks my parents for all the hard work, they did for me to make me to reach at this level.

TABLE OF CONTENTS

CONTENTS **PAGE NO.**

LIST OF FIGURES

LIST OF TABLES

Checklist for Dissertation-III Supervisor

Name: _____ UID: _____ Domain: _____
Registration No: _____Name of student:_____

Title of Dissertation:
_____

- ☐ Front pages are as per the format.
- ☐ Topic on the PAC form and title page are same.
- ☐ Front page numbers are in roman and for report, it is like 1, 2, 3…….
- ☐ TOC, List of Figures, etc. are matching with the actual page numbers in the report.
- ☐ Font, Font Size, Margins, line Spacing, Alignment, etc. are as per the guidelines.
- ☐ Color prints are used for images and implementation snapshots.
- ☐ Captions and citations are provided for all the figures, tables etc. and are numbered and center aligned.
- ☐ All the equations used in the report are numbered.
- ☐ Citations are provided for all the references.
- ☐ **Objectives are clearly defined.**
- ☐ Minimum total number of pages of report is 50.
- ☐ Minimum references in report are 30.

Here by, I declare that I had verified the above mentioned points in the final dissertation report.

Signature of Supervisor with UID

CHAPTER 1
INTRODUCTION

Modern world is the era of science and technology due to which many new technologies have been introduced at different times. Internet is one of the results of this. Today lot of things are managed online via internet with the help of automated tools, programs, techniques; which surprised people that how things were carried on, when there was no internet. Before this invent, people focused more on reading books, magazines, newspaper, etc to gain knowledge. But now people shift their focus towards the internet to fetch any kind of knowledge. Now wikipedia's, journals and websites are available on internet which provides good and rich amount of knowledge to the people [25]. In a nutshell, internet provides the people with ample of opportunities to do work in more sophisticated manner. It can be said to be an ocean of new technologies, knowledge and many more, to learn many new things out of it. But these all are the one side of coin that how the introduction of internet and advancements related to it, opens new opportunities for people. Now looking at the other side of the coin, these new advancements of internet make people tedious and weary. People now are too much dependent on internet that they start copying and pasting the things to accomplish their tasks instead of learning or grasping them in mind. They are not brainstorming their minds. According to software and technology terms, this duplicity achieved by making copying or pasting of things which could lead to lack of originality is referred to as "cloning". This copying of codes will lead to copyright infringements of original work of the authorized persons. This has been a question from many years in the mind of researchers who dedicated their research in this field of cloning that whether cloning is a legal or an illegal exercise. Then the answer to this is cloning is not illegal if it is done with the permission of authorized person. For instance, reusing the code in the software development is an efficient method to reuse the design or requirements; which will save lot of time and cut costs in developing large software products. So to reuse the required things, the owner must be asked about copying his code. In another case also, cloning can't be illegal if content present on some website or on some journal is for free, but that too leads to the absence of

one's originality and creativity. Cloning imparts many cons despite of having many pros, which can be easily justifiable to the fact that "Every rose has its thorns".[25]

1.1 CLONING AND SOFTWARE CLONING

When we think about the cloning, the concept or idea that comes to our mind is duplication of something. It creates the picture of two things in which one thing is same or similar with respect to the other thing in the one way or the other, which we can said to this process as a "cloning"[25]. According to software and technology terms, this duplicity or similarity can be achieved by making copying or pasting of things which could lead to lack of originality and introduction of duplicity is referred to as "software cloning". In software engineering terms, this cloning can be achieved in two ways: Model Cloning and Code Cloning. Model cloning deals with cloning of design, structure or model whereas Code cloning deals with the cloning of part or whole of the source code of the software. Our main aim or emphasis lies fully on code cloning.[25]

1.2 CLONE TERMINOLOGIES

Clone terminologies give the concise explanation of the meaning of some clone definitions or phrases. [39][25]

1. **Code Fragment**- A simple code snippet that comprises of some lines of code is referred to as code fragment. It can be acknowledged through name of file and line number.
2. **Code Clones**- These are the unit of source code as a part or as a whole which are the copied or duplicate form of other code.
3. **Clone Pairs**- The two parts of the code are said to be form the clone pairs if on the basis of some parameters of similarity, they are found to be same. Both the pieces of code should have something in common or similar in order to form the clone pairs.
4. **Clone Class**-When multiple code parts other than just two parts, are similar with each other, then that forms a block of clones known as clone class.
5. **Clone class Family**-It groups all clone classes bearing same or similar area .It is referred to as clone class family.

1.3 REASON OF CLONING

There are various reasons of cloning such as: - [10][39][25]

1. Reuse- It is an efficient method to reuse the design code and its requirements. Hence, it saves time and reduces cost.
2. Templates- Some programming paradigms have encouraged the use of templates.
3. Coincidence- Sometimes different developers unintentionally write the same piece of codes without knowing other's code.
4. Large codes- Fear of writing large codes also encourage programmers to copy the code.
5. Complexity of the system- Difficulty in understanding large systems also promotes copying the existing functionality and logic.

1.4 ADVANTAGES OF CLONING

Cloning has certain advantages which motivate the people to follow this process. [10][39][25]

- Lack of Knowledge about language- Some programmers doing well at one language while not so good in others. This is due to the lack of knowledge of the programming language.
- On-time software development- In software development, scheduling of tasks to each developer has been assigned to complete the work on time or to meet the deadlines.
- Fast method- It is considered as a fast method for developing software systems.

1.5 DISADVANTAGES OF CLONING

Cloning has certain disadvantages also which refrains the people to follow this process. [10][39] [25]

- Effect on maintainability- Cloning has an adverse effect on maintainability as it invites more maintenance cost.
- Bug -propagation- It also leads to bug propagation from original code to the copied one.
- Effect software evolution- It becomes hurdle for better evolution of software systems as it has bad impact on designing and many other areas of software.

- Lack of originality- Often developers copy some part of code from some websites and paste them in their source code. This process of copying and pasting results in loss of originality.

1.6 TYPES OF CLONES

There are four types of clones related to code cloning namely exact clone, parameterized or syntax clone, near-miss clone and semantic clone.[3][39][25]

1. Exact clone- Type1 clone
2. Parameterized clone- Type2 clone
3. Near-miss clone-Type3 clone
4. Semantic clone-Type4 clone

- TYPE 1 clone (Exact clone) - Exact clones are the clones which look like an original code. These clones can be easily detected with the help of simplest clone detection technique like text based, token based, etc. The difference comes only in the blank spaces or in the comments. These can be easily detected by text based techniques and tools like SDD[3][39], LCS [3] , Dup for string matching[3], etc.
- TYPE 2 clone (Renamed/parameterized clone) - Renamed clones are the clones where variations come in the name of literals, keywords, variables, etc. These clones can be detected by techniques called token based, metric based, etc. Various tools are also implemented by developers to detect these types of clones. These tools are CLAN [29], MCD-Finder [23][24], etc.
- TYPE 3 clone (Near-miss clone) - In these types of clones, changes persist in code in the form of addition, deletion and modification of statements. These clones can be detected by the techniques called tree based and tools called Deckard [21], CloneDr [3][10], etc. These clones can be easily detected by tree based techniques, where sub trees of AST are being compared with each other.
- TYPE 4 clone (Semantic clone) - In these clones, function or behavior of the clone remains same but syntax or coding of program is different. These types of clones are detected by using graph based techniques and tools like Scorpio [3][10], Duplix[3][10], etc.

1.7 CLONE DETECTION

It is the process of finding or detecting clones in code. It is used to find clone pairs in programs based on similarity. There are various advantages of finding clones so as to detect the bugs at the earlier stage**.** There are various steps that are involved in clone detection like preprocessing, transformation, match detection, formatting, etc. In addition to this, many techniques are used which are applied to detect the clones efficiently with the help of tools [3][10] such as cloneDr, Nicad, Deckard, CC-Finder [10], etc. [25]

1.8 ADVANTAGES OF CLONE DETECTION

Clone detection plays an important role in detecting code clones. [10][39] [25]

- Software analysis- It is very useful for software analysis and understanding of software evolution.
- Bug Detection- It finds bugs in the program so that they are not propagated from one program to another.
- Understand ability- It enhances program understand ability and reduces program size.
- Plagiarism Detection- It is having biggest advantage in detecting plagiarism in order to protect copyrighted content from being copied.

1.9 STEPS IN CLONE DETECTION

Clone Detection process is a series of steps that are taken in order to unmask or detect code clones. These series of steps comprises of [10] [25]:-

1. Pre-processing
2. Transform
3. Match Detection
4. Formatting
5. Post Processing
6. Aggregation

1. Pre-processing

1.1 Remove unnecessary parts- All the source code which seems to be irrelevant should be discarded in the comparison phase. For e.g., if the tool is not language independent, then different languages needed to be separated from code like separating sql from java code [10] [25].

1.2 Determine source units- The remaining code obtained after removing the uninteresting code will be bifurcated into a set of disjoint fragments and these fragments are known by the name "source units". [10] [25]

1.3 Determine comparison units/granularity-Based on the comparison technique used by the tool, these source units are further divided into much smaller units. [10] [25]

2. Transform

It converts source units which are needed for comparison, into some intermediate state. All the techniques, except text based, require a transformation of the source units. This transformation is also referred to as 'extraction' according reverse engineering community. This transformation can be achieved in two ways: extraction and normalization. [10] [25]

2.1 Extraction [10]- It is further bifurcated into 3 sub-categories namely tokenization, parsing, control and data flow analysis.

2.1.1 Tokenization[10] [25]- In this approach, source units are converted into tokens based on lexical protocols or procedures and these tokens are arranged in token sequences, after the removal of blank spaces and comments, for comparison.

2.1.2 Parsing[10] [25]- Here, entire source code is parsed to generate an AST (Abstract Syntax Tree) and then source units from AST's which are needed are shown in the form of sub trees. To figure out clones, these sub trees need to be compared.

2.1.3 Control and data flow analysis[10] [25]- In this approach, PDG (Program Dependency Graph) generated tools create PDG graphs in which nodes

represent statements whereas edges represent data and control dependency. To lay out a comparison, sub graphs of PDG's are compared.

2.2 Normalization [10] [25] - This is an optional step to eliminate differences based on comments, whitespaces, etc. This can be achieved in many ways like by normalizing the identifiers where all identifiers in source code are replaced by the single identifier, pretty-printing, etc.

3. Match Detection

In this phase, transformed units obtained from the transformation phase are passed into some comparison algorithm and then compared to find a proper match. The output contains a list of matches in the transformed code which represents the clone relations in the form of clone pairs, clone classes and clone family. Certain comparison approaches include hashing, suffix trees, etc. [10] [25]

4. Formatting

In match detection, clone pair list is obtained for transformed code but in this phase, the list is further converted into another pair of list that matches with the original code base. [10] [25]

5. Post Processing

In this phase, clones are filtered or ranked using manual analysis or automated heuristics. In manual analysis, false positive clones are filtered out by a human expert. Automated heuristics is based on length, diversity,

Frequency and other characteristics of clones in order to rank or filter out clone candidates automatically.[10] [25]

6. Aggregation

This is the last step of the clone detection process. It refers to proper analysis and data contraction. The detected clone pairs are combined to form clone classes and clone family.[10] [25]

1.10 TECHNIQUES IN CLONE DETECTION

Following are the techniques available for clone detection. [10][39] [25]

1. Text Based
2. Graph Based
3. Metric Based
4. Token Based
5. Tree Based
6. Hybrid

1. **Text Based** [10] [25]- This technique compares two code fragments line by line. This technique is only for Type 1 clones. It doesn't consider any renaming of variables and any syntactical or semantically changes. It provides high accuracy. But it is not highly efficient to detect any other kinds of clones. Many researchers come up with new tools and technologies like Johnson et al. proposed a fingerprinting technique for detecting text based clone fragments. Another tool is DUPLOC [10] which is devised by Ducasse et al. It is a language independent tool which requires no parsing of source code i.e. it is directly imposed on source code to detect clones. Similar Data Detection (SDD)[3] tool detects clones in systems of large size.

2. **Graph Based** [8][10] [25]- This technique uses program dependency graph (PDG). It is good for detecting semantically similar clones. Semantically similar clones are those clones which are syntactically different but show similar behavior or perform same function. In other words, it can detect type 3 and type 4 clones efficiently. PDG are directed graph which determines two types of dependencies namely data dependency and control dependency which exists between statements of the source code. Tools under this technique are Duplix, Scorpio [3], etc. Duplix tool is proposed by Krinke et al. It finds maximum similar sub graphs from the transformed source code. Scorpio is stated by Higo and Kusumoto et al [3]. In this tool, two way slicing is introduced i.e. forward slicing and backward slicing. If clone is not detected in the forward slicing, it can be detected in backward slicing. There is another tool GPLAG [3]which is proposed by Liu et al. It is a PDG based plagiarism detection tool and algorithm on the basis of program dependence.

3. **Metric Based** [25] - It is a straight forward technique. There are various types of metrics namely class, object-oriented, layout, method, control, etc.[8][10][24]. All these types follow a different metrics. Metric based approach is more scalable technique and gives accurate results for large software systems. It contains structural information only. So it is good for finding syntactic clones i.e. Type 1, Type 2 and Type 3 clones. CLAN [20] is a tool advocated by Mayrand et al. In this technique, AST of a source code can be collected to compare metrics based on it. There is another similar technique introduced by Kontogiannis et al. that applies dynamic programming on metrics. MCD-FINDER [23][24] is proposed by perumal et al. In this technique, Fingerprinting approach is used for clone detection in source code.

4. **Token Based** [25]- In this technique, there is a formation of lexical tokens [8][10][24]. It is good for detecting type 1 and type 2 clones. It gives fast response and is considered to be more efficient as compared to text based but also gives many false positives. It extracts tokens out of the source code with the help of lexical analysis and based on this token sequence is formed. There is a method called "functor" that maintains the order of tokens. Tool called CP-Miner [15] is based on data mining approach. It makes use of frequent item set mining which is helpful in bugs and structural clone detection. Likewise, CC-Finder [43][35] tool devised by Kamiya et al. is used to find identical subsequences with suffix matching algorithm. It detects clones in languages like C, C++, Java, COBOL, etc. Similarly, a tool which is known as LSC-Miner [37] detects clones in large source codes. Basically, it is a multilinguistic tool that is used to detect clones in more than one language. It is implemented in VisualBasic.Net.

5. **Tree Based** [8][10] [25]- This technique is based on Abstract Syntax Tree (AST) which is obtained after converting the source fragments into some intermediate form. It is efficient for detecting type 1, type 2 and type 3. It is a heavy weight technique and requires a sub tree comparison. Under this technique, various tools and methods are proposed by the researchers to detect clones that fall under type 1, 2 and 3 categories. A popular tool called CLONEDR [3] is used to fetch near miss clones. A tool called DECKARD [21] is more scalable and accurate tool than any other tool. It is a language independent tool which is proposed by Ling Xiao Jiang.

6. **Hybrid** [25]- This technique is the combination of various other techniques like tree, text, token, metric, graph [8][10][23]. A tool named HCDETECTOR merges PDG based technique and metrics based technique. It only works for java language and does its execution on java byte code (.class) which is the intermediate stage of java source code (.java), rather than on original code itself. Another hybrid technique [23] uses the combination of metrics and token based approach. The tool used in the technique MCD-Finder and CC-Finder.

1.11 CLONE DETECTION TOOLS

Given below are the clone detection tools. Although there lot many tools available for this purpose; out of those few are discussed below [3] [25]:-

Table 1.1: Textual Approaches

Tool	Comparison Method	Complexity	Clone Type
Dup	Suffix-tree	O(n+m)	Type1,Type2
Duploc	Dynamic Pattern Matching	O(n2)	Type1,Type2
Nicad	LCS	O(n2)	Type1,Type2, Type3
SDD	N-neighbor distance	O(n)	Type1,Type2, Type3

Table 1.2: Lexical Approaches

Tool	Comparison Method	Complexity	Clone Type
CCFinder	Suffix-tree based On token matching	O(n) where n is size	Type1,Type2
CP-Miner	Frequent subsequence mining	O(n2) where n is number of code lines	Type1,Type2
FRISC	Suffix array	N/A	Type1,Type2,Type3

Table1. 3: Semantic Approaches

Tool	Comparison Method	Complexity	Clone Type
Gplag	Isomorphic PDG Subgraph matching Algorithm	NP-Complete	Type1,Type2,Type3
Funaro et.al	Texual comparison	N/A	Type1,Type2,Type3

Table 1.4: Syntactical Approaches

Tool	Comparison Method	Complexity	Clone Type
CloneDr	Tree matching Technique	O(n) where n is number of AST nodes	Type1,Type2
Mayrand et.al	21 function metrics	Polynomial complexity	Type1,Type2,Type3
Kodhai et.al	Metrics	Textual comparison	Type1,Type2
Abdul-El-Hafiz, et.al	Metrics	Data mining clustering algorithm	Type1,Type2,Type3

CHAPTER 2
LITERATURE SURVEY

In this chapter literature review related to clone detection techniques which is given by different researchers in this field of study has been discussed. Many of them started theirjourney in this field of research by just giving review or survey related to discussions based on cloning or clone detection, impact of cloning practices on software, clone detection techniques and tools, their comparison, etc. While others come up with their proposals along with their implementations, which could make this field more interesting and active field for research .In this chapter, literature review based on different approaches of clone detection such as token based, metric based, hybrid and research based on multi-language clone detection have been discussed. [25]

In one of the survey or review given by Dhavleesh Rattan et.al [10] in theyear 2013, they have made a systematic review in this field of clone detection. Their systematic review helps many researchers who want to go for research in this field, to learn more about cloning in software projects, its pros and cons, types of clones found in source codes, clone detection process, its advantages, techniques used in clone detection, tools, etc. This paper acts as a base for further study in this area of cloning and clone detection.

Another survey is given by Abdullah Sheneamer [3] in the year 2016. This survey is based on clone detection techniques where a tool, techniques and their comparison with each other has been discussed. A proper survey has been given about each tool and technique. Moreover, all the related techniques which are already proposed by the researchers in this field have also discussed. Clone detection process and categories are also discussed in detail.

2.1 SURVEY ON METRIC BASED APPROACH

Metric based approach is more scalable technique and gives accurate results for large software systems[3][10]. It contains structural information only. So it is good for finding syntactic clones i.e. Type 1, Type 2 and Type 3 clones. Moreover, it is a straight forward

technique. There are various types of metrics namely class, object-oriented, layout, method, control, etc. Many proposals related with this, have been proposed by different researchers at different times.

Sushma et.al [42] in the year 2016 gives another proposal based on this technique. They try to impose the metric based approach to detect module or method level clones only. They implemented their tool named JSCCD (JS Code Clone Detector) in java language and this tool could detect the code clones in java language only. They used 7 metrics in their proposed technique. Their technique can found the type1 and type2 clones.

Sukhpreet Kaur et.al [41] has given a proposal in the year 2015. In their proposal Object-oriented metrics have been taken for carrying out the experiment on clone detection. In that metrics like DIT, NOC, WMC, LCOM, etc are taken.

Kanika Raheja [24] proposed a metric based technique in the year 2014 where a tool named MCD-Finder has been taken to calculate the metrics in java program. In this, instead of applying metrics on direct java code or any other transformed code, it uses java byte code for a metrics to be applied on. Moreover, java byte code is platform independent and represents the unified structure of the code. This technique was also able to detect some semantic clones. In the proposed methodology, this technique was used independently without any combination with other techniques. In this, a class level metric and a function level metric has been taken out.

K. Vidhya et.al [22] gives a proposal in 2014 where they detected the higher level clones such as file clones, directory clones, etc. between two object oriented languages C++ and Java. For implementing their technique they have used metric based technique. In this approach Metric based technique has been used twice. They first applied metrics on methods and then later on a file.

Geetika et.al [13] proposed a Metric based technique in year 2014 which can detects 15 Metrics in all. The implementation of the tool was formulated in PHP language. The tool accepts only .csv (comma separated values) files. In their proposed technique, a File level Metrics and Method level Metrics have been computed.

Salwa K. Abd-El-Hafiz [2] gives another proposal in 2012 regarding the Metric based approach. He uses this technique of metrics with data mining concept. In this approach first the metrics were calculated for all the functions and then on the basis of that, a popular data mining clustering algorithm called fractal clustering has been used to make small sized clusters. These small sized clusters were built or implemented on the basis of similar metric values i.e. all the similar or same metrics were put together into the one cluster. These clusters were thereafter used to formulate clone classes.

Zhu o LI et.al [46] advocated a proposal in 2010 in which detection of clones can be found with the help of metric space based technique. Metric space is a set where definition of distance between elements is defined within the set. For similarity measures, this technique uses distance within metric space. This technique is an advanced version of metric based technique. It uses the parameters of scalability, accuracy and flexibility to judge the performance of the system.

Doaa M. Shawky et.al [11] gives another proposal in 2010. In their proposal, they used all possible permutations and combinations of metrics in a sequential order to form the clusters having the highest similarity measure within them. They concluded that using the optimal sequence of metrics, they got 100% precision. They also concluded that if there is a ranking function that can assign weights to the metrics then it can increase the precision of clone detection in metric based technique.

Ettore Merlo [29] advocated other proposal in 2007 based on metric technique, detects plagiarism in university projects. In his technique, he uses CLAN tool which is used to calculate metrics and then creates the clusters to measure the cloning ratio and cloning percentage. He concluded that CLAN is a small memory, fast, accurate but conservative tool. Also it is a robust tool in terms of parsing and analyzing the code. It has good performance in relation to the speed.

Jean Mayrand et.al[20] in theyear 1996 gives a proposal where 21 metrics are used for automatically detecting duplicate and near-duplicate function clones in large software systems. The tool used in this technique to fetch metrics is popularly named as Datrix. This tool was considered useful in improving the maintainability of the system by

removing and managing the source code of the system by removing the functional clones. They carried out the comparison for detecting functional clones on the basis of 4 metrics or parameters namely name of the functions, Layout, Expression and control flow. They had taken 5 layout metrics, 5 expression metrics and 11 control flow metrics.

With the proposed techniques discussed above, the use and application of the metric based technique has not come to an end. Apart from the above discussed metrics such as method, layout, control, class and object oriented level metrics, the various other metrics such as product metrics and process level metrics can also be taken into consideration.

Table 2.1: Summary of Metric based Approaches

Year	**Name of Author**	**Type of Metrics/scope**	**No of Metric Used/Language**	**Tool**
2016	Sushma et.al	Method level Metrics	7Metrics /java	JSCCD
2015	Sukhpreet Kaur	Object-oriented Metrics		
2014	Geetika	File level & Method level	15	PHP
2014	Kanika Raheja	Metrics from Transformed code	14Metrics/ java	MCD-Finder
2012	Abd-El-Hafiz	Function level Metrics		Data mining
2010	Zhu o LI et.al			Metric space Algorithm
2007	Ettore Merlo	Detects Plagiarism in university projects.		CLAN tool
1996	Jean Mayrand	Detect functional clones	21	Datrix

2.2 SURVEY ON TOKEN BASED APPROACH

This is a technique where source code needs to be converted into sequence of tokens with the help of lexical analysis as detection of clones with this method cannot be possible without the transformation of source code into lexical tokens[3][10]. These tokens represent the transformed state of source code on which matching algorithm has been applied. Different proposals are proposed by many researchers with novel or enhanced techniques.

Rajnish Kumar [33] proposed a technique in 2014 in which he detects the clones with the help of program slicing. In that they get the program slices from the source code and then retrieve the tokens corresponding to that program slices. In the end they compare these tokens to get the cloned code in the source code. In their technique, they also worked for the detection of non-contiguous clones. The aim of their proposal is to detect the type1, type2 and type3 clones.

Qing Qing Shi et.al [32] proposed a technique in the year 2013.They implemented a tool named SaCD which has efficiently detects clones in the languages named C, C++ and java. In the proposed technique, a suffix array algorithm has been used in order to detect clones.

Saif Ur Rehman and Kamran Khan [37] proposes a technique in 2012 where they take some source code whose clone detection they want to found and then transformed that source code into tokens which would be stored in some two dimensional array. For comparison they assign some hash value to these tokens and compare the hash values to detect clones. They implemented their technique in prototype tool called LSC-Miner which detects code clone in large source code written in multiple languages.

Yang Yuan et.al [44] proposes another technique based on this approach in 2012. This technique makes use of Boreas tool. This tool uses a novel counting based characteristic-matrix for pattern representation. In the proposal this technique has compared with other tool named Deckard. They concluded that Boreas can detect the clone at a faster rate because of the fact that it uses two similarity functions i.e. Cosine similarity function and Proportional similarity function.

Khurram Zeeshan Haider et.al [26] gives a proposal in 2010. They main aim of their proposal was to detect Plagiarism in source code. To accomplish this, they use a greedy string tiling algorithm for finding Plagiarism. They conduct their implementation in two phases. In first phase, parsing of source code and pre-tokenization has been conducted to get tokens and then in the second phase greedy string tiling algorithm has been composed.

Hamid Abdul Basit [15] gives a technique in 2009. They use the concept of Data Mining to accomplish their implementation. Their aim was to detect higher level clones in software. They used the tool named clone Miner that implements their technique. They have conducted certain case studies to assess the scalability and usefulness of their proposed technique. The tool has written in c++ and has its own token based simple comparison algorithm. This tool uses the concept of FIM (Frequent itemset mining).

Harjot Kaur and Manpreet Kaur [16] proposed a technique in 2014. In their technique, They tried to detect clones in class diagrams. For that they first converted the class diagrams into XML format.After obtaining XML format, tokens are applied on XML format.Then these tokens are compared with the help of suffix array algorithm.

Hiroaki Murakami [18] proposed a new methodology in 2012 in order to improve the existing token based techniques.For that Folding repeated instructions mechanism is use at the preprocessing step to standardized the repeated instructions in the source code. Then suffix array based algorithm is used to compare the tokens in order to achieve code clones.

Table 2.2: Summary of Token based Approaches

Year	**Author**	**Tool/Technique/Language**
2014	Rajnish Kumar	Program Slicing
2014	Harjot Kaur	Suffix array
2013	Qing Qing Shi	Suffix array
2012	Saif Ur Rehman	LSC-Miner/ Multiple Language

2012	Yang Yuan et.al	Boreas
2010	Khurram-Zeeshan Haider	Greedy String Tiling Algorithm
2010	Hamid Abdul Basit	Data Mining(FIM)/Clone Miner

2.3 SURVEY ON HYBRID APPROACH

Hybrid approaches are the combinations of techniques such as metric, token, graph, etc. Various researchers come up with the proposals where this approach has been satisfied.

Aritra Ghosh et.al[5] proposed a technique in 2017 with hybrid combination of Graph and Metric based techniques. The size of the metrics in the proposed technique is 13 and the types of Metrics included are control flow metrics, class metrics and Function Metrics. In the proposed technique the Metric based technique has been applied at the secondary step whereas this technique has been applied even at the preliminary step also by most of the proposed techniques. This technique is able to found the clones in the java language only.

Jai Bhagwan et.al[19] comes up with the proposed technique in the year 2016 which comprises of the combination of Metric based and text based technique. A total of 11 Metrics have been computed in the technique. The technique in order to improve the results uses the Levenshtein Distance. In this technique a Metric based technique has been implemented at the first step and then Textual based comparison has been laid on. The implementation was carried on the tool called Netbeans.

Deepali et.al[9] in the year 2016 comes up with the proposal which consists of the hybrid combination of Metric and Token based technique. The no of metrics the proposed technique has computed are 15 and these metrics were calculated with the help of the tool called Source Monitor. The Token technique uses the Hash algorithm to compute the tokens efficiently.

Muneer Ahmad et.al [30] proposed a technique in 2016 in which they used the combination of Metric and text based techniques. They have used 10 metrics for detecting potential clones and then applied text based string matching algorithm to verify that whether the detected potential clones are the actual clones or not. The platform they have used to carry out their implementation was Netbeans and the implementation was carried in java language.

Manpreet Kaur et.al [28] uses this hybrid technique in the year 2015 with the combination of metrics and Text based. In their proposal they have detected type1, type2 and type3 clones efficiently. In their proposed technique they have applied metric based approach on the functions or methods which they obtained from source code. Their proposed technique could detect clones in C++ language only. To conduct their experiment they have used Visual Studio 2010 and .Net framework version 4.0.

Egambaram Kodhai[12] proposes a technique in 2014 where the tool called Clone Manager is used to detect clones in languages written in c and java. The implementation of tool has itself done in java language. To make it a light weight hybrid, this technique was a combination of metric and text based technique. A total of 16 metrics have been used in their proposed technique. To check the results, the tool is checked against Bellon's dataset.

Surbhi Sonika[39] in the year 2014 proposes a technique comprises of the combination of graph based and metrics based. The tool named HCDetector has been used here for graph based technique. This proposed technique detects clones only for java language and the matching algorithm works only on java byte code. Moreover, java byte code is platform independent and represents the unified structure of the code. Here graph based technique has implemented as a preliminary step to detect the potential clones and then metric based technique has been applied to found out the actual clones.

Himanshu et.al [17] gives the technique in 2014 which was a combined weighted approach bearing combinations of text analysis, token analysis and statistical analysis. The distinguishing ground that makes this hybrid technique different from other hybrid

techniques is that it assigns a weight to every approach while in other approaches no weight mechanisms are used with the approaches.

Akshat Agrawal [4] gives another hybrid approach in the year 2013 which was be formed by the combination of token based and text based approaches. This approach was given by. Their main aim behind this proposal was to overcome the problem encountered in the Boreas tool which was not able to detect type 3 clones. Generally code clones of type3 are not detected by token approach and only type1 and type2 clones are detected by this approach so they used text based approach in combination with token approach so as to detect type 3 clones as well. Their proposed technique was able to detect clones in C language only.

Kanika Raheja [23] has given a technique in 2013 which is composed of metric based and token based approach. The proposed technique is used to detect clones only in java language. The comparison has been performed directly on java byte code rather than on any other transformed code because java bytecode represents the unified state and moreover it is a platform independent. In this proposed technique a total of 14 metrics have been used. For calculating the metrics a tool named MCD-Finder has used and for a token based approach a tool named CC-Finder has been used. The metric based approach has been used as a preliminary step to detect the potential clones and later on token based approach has been applied to calculate the actual clones.

Yogita Sharma[38] in the year 2011 has given a proposal where hybrid technique with the combination of metric and text based approach has been proposed. In that 34 metric values are counted and then text based approach has been applied to obtain actual results.

Table 2.3: Summary of Hybrid Approaches

Year	**Author**	**First Approach/ Tool/Technique**	**Second Approach /Tool/Technique**	**Language/tool**
2017	Aritra Ghosh	Graph	Metric(13)	Java
2016	Jai Bhagwan	Metric(size-11)	Text	Netbeans

2016	Deepali	Metric(size-15)	Token	HashAlgorithm
2016	Muneer Ahmad et.al	Metric(size-10)	Text	
2015	Manpreet Kaur et.al	Metric	Text	C++
2014	Egambaram Kodhai	Metric(size-16)	Text	C, Java
2014	Surbhi Sonika	Graph/HCDetector	Metric	Java
2013	Akshat Agrawal	Token	Text	C
2011	Kanika Raheja	Metric(MCDFinder)	Token(CCFinder)	Java

2.4 SURVEY ON MULTIPLE LANGUAGE

There are so many studies which show that many researchers come up with the proposals where clone detection can be initiated or performed on multiple languages. Detection of clones on multiple languages makes it language independent.

Muneer Ahmad et.al [30] proposed a technique which was able to detect clones in object oriented and platform independent language java along with web based language such as JSP (Java Server Pages), asp.net, PHP and html. Moreover this is a hybrid technique based on the combination of metric based and text based approaches.

K. Vidhyaet.al [22] gives a technique in 2014 to detect the higher level clones such as file clones, directory clones, etc. between two object oriented languages C++ and Java. For implementing their technique they have used metric based technique. In this approach Metric based technique has been used twice. They first applied metrics on methods and then later on a file.

S.Mythili et.al [36] proposes another language independent approach in 2012. They tried to detect method level clones in source code with the help of Robin Karp fingerprinting algorithm. The system used in this proposal for carrying out the implementation was WordNet.

Saif Ur Rehman and Kamran Khan [37] gives a technique in 2012 where they take some source code whose clone detection they want to found and then transformed that source code into tokens which would be stored in some two dimensional array. For comparison they assign some hash value to these tokens and compare the hash values to detect clones. They implemented their technique in prototype tool called LSC-Miner which detects code clone in large source code written in multiple languages.

Muhammad Younas et.al[45] proposes another technique in 2011.They has used the tool named Clone Analyzer which is a token based tool to detect structural and simple clones. It detects clones in languages written in Java and C.

CHAPTER 3
SCOPE OF STUDY

People now are too much dependent on internet that they start copying and pasting the things from one place to another to accomplish their tasks. They are not grasping or brainstorming the things in mind. According to software and technology terms this duplicity which is the resultant of copying and pasting of things can lead to many consequences such as the introduction of code clones. [25]

Recently, code clones have received much attention from many researchers in the field of software engineering and clone detection is an active research area and work has been carried out on a larger scale in detection of clones. Clones are code fragments that are identical or similar to other code fragments in the source code, and they are generated for various reasons, such as copy-and-paste operations, to reduce the time and effort of the software developer, etc. As it has been said that "Each coin has two sides", so this is the one side of the coin. [25]

On the other side, cloning can lead to copyright infringements of original work of the authorized persons. It also affects software evolution as it become hurdle for better evolution of the software systems and has bad effect on designing and many other areas of software. It has been said that the presence of clones makes software maintenance more difficult as it invites more maintenance cost because cloning unnecessarily increases program size and complexity. Moreover it invites more maintenance effort because if bug fixing has been made on one code, it needs to be made on other codes as well where the codes are copied. It also decreases the quality of the software like readability and other problems due to bugs' propagation. [25]

Therefore, code cloning is the major issue in the industrial point of view. As the number of projects is increasing in this digital world; there is major challenge to verify the contents and identify the line of codes. So to overcome such problems a scalable and efficient clone detection tool and technique is needed that will unmask as much number of clones as possible so as to find the bugs at the earliest which prevent the bugs from

being travel from one source to other, detecting plagiarism that will prevent the copyrighted content from being copied, to increase program understandability and for better evolution of the software products. [25]

CHAPTER 4
PRESENT WORK

4.1 PROBLEM FORMULATION

People today are so much dependent on Internet that they start copying and pasting the things to accomplish their tasks instead of learning or grasping them in mind. They are not brainstorming their minds. According to software and technology terms, this duplicity achieved by making copying or pasting of things which could lead to lack of originality is referred to as “cloning”. This process leads to the generation of code clones. This copying of codes will lead to lack of copyright infringements of the original of original work of the authorized persons. [25]

There are many other problems which are associated with cloning such as it can impose bad effect on maintainability due to which sometimes it invites more maintenance cost. Other problems associated with this are it can lead to bug propagation from one software system to another. It also affects better software evolution. [25]

All these factors give invitation to the need of some clone detection tools and techniques which gives assistance in detecting the clones with high efficiency and accuracy.

4.2 OBJECTIVES OF STUDY

The main objectives of this thesis are:-

1. To propose a refined hybrid technique for efficient code clone detection using Metric analysis and Tokenization.
    a. To calculate Class level metrics and Function level metrics for input files.
    b. To compare calculated metrics to find potential clones.
    c. To perform Tokenization on pre-processed code.
    d. To implement Suffix array matching algorithm to find actual clones.
2. To implement proposed technique for multiple languages like Java (Object oriented language) and Asp.net (Web based language).

4.3 RESEARCH METHODOLOGY

Recently, code clones have received much attention from many researchers in the field of software engineering and clone detection is an active research area and work has been carried out on a larger scale in detection of clones. The detailed study of literature survey helps to understand that there are various approaches which provide a great help in detecting or unmasking code clones in source codes. These techniques can be text based, token based, metric based, tree based, graph based, etc. But there are certain advantages and disadvantages that are associated with these techniques.

Text based techniques do not require any transformation of source code and can be applied directly to the source code. Hence they can detect only type1 clones but with great accuracy and precision. Token based technique requires lexical analysis and transformation of source code to be applied on. This is a fast technique and can detect type 2 clones also. Metric based approach is an accurate and straightforward approach and can detect type 3clones as well. PDG and tree based approaches can detect type3 and type4 clones but these are complex and expensive techniques.

This thesis comes up with the proposal where hybrid technique is proposed which is the combination of metric based and token based technique. To implement a hybrid technique three alternatives are there which gives an idea that in what way the combination of two different approaches has been used.

1) Metric technique at a preliminary stage and Token technique at a secondary stage.

2) Token technique at a preliminary stage and Metric technique at a secondary stage.

3) Metric technique and Token technique are applied together.

In the proposed methodology, these approaches are work in the sequential manner. The Metric based technique has been applied at a preliminary stage to detect the potential clones and then Token based approach has been applied. The motivation behind using the Metric approach at the preliminary stage is as following:

- Metric based technique is good and useful if clone detection has been applied in large software projects. In these cases, rather than working on source code directly, metrics are used to detect clones.
- There might be the cases that no clone exists in the source code while detecting the clones with token based approach. This can lead to wastage of memory, time and effort on converting source code into tokens. So this is another factor that supports the use of metric based technique at the preliminary stage.
- Metric based approach decreases the complexity and simple to use.

So in the proposed methodology the metrics based approach can be refined by increasing the size of metrics and type of metrics to detect more efficient and précised clones that will more accurately describe clones so that further a token based technique can be applied on it.

Given below is the step by step explanation of the proposed technique that how it works as a hybrid technique using Metric and Token based approach.

4.3.1 METRIC APPROACH

a) Take two source code files with the help of the proposed tool.
b) Calculate the values of some predefined metrics of these two files and display these values in the tool itself. These metrics are also stored in excel file.
c) Compare the metric values calculated from the source files to detect potential clones.
d) To detecting the potential clones, threshold value has been set in the tool which is 3. If threshold value less than 3, then no potential clones exists in the code.
e) If the threshold value comes out to be 3 or greater than 3, then only the potential clones exists in the code and the tool will work further for Token approach.
f) Apply second approach [Token approach] to calculate actual clones.

In the existing hybrid technique of Metric and Token based technique, a total of fourteen metrics have been computed. In the proposed technique, a total of twenty metrics will be calculated. These metrics will include two types of metrics namely Function level Metrics and Class level metrics. Function level metrics are fetched from the functions

included in the source code and class level metrics computed metrics from the classes included in the source code.

Given below is the list of the metrics which proposed technique will compute.

1. Lines of Code(LOC)
2. No of private variable
3. No of public variable
4. No of protected variable
5. Total no of variables
6. No of loop controls
7. Redirect statements
8. No of conditional statements
9. Total no of assignment
10. No of Private functions
11. No of Public function
12. No of Protected function
13. Total no of functions
14. Function name
15. No of local variables
16. No of function calls in function
17. No of arguments passed in function
18. No of loop controls in function
19. No of return statements
20. No of conditional statements in function

4.3.2 TOKEN APPROACH

After obtaining a Potential clones from Metrics approach, the tool will further executed for token approach. In Token approach following steps has been followed out.

a) Apply Tokenization approach on pre-processed code so as to fetch a transformed code(Token string) from the pre-processed code.[6][14]

- For Tokenization, no external lexical analyser or Tokenizer has been used to transforms the source code into the token string.
- A unique numeric ID is manually assigned against each Token class.
- From all the source files, a single large token string is generated or obtained.
- Blank spaces, comments and blank lines are ignored during Tokenization.
- All the repeated token segments in the token string represent a code clones, code clone classes of various types.
- The end product obtained after this step is Tokenized code.

Given below is the way that how token formulation will take place by assigning unique numeric ID's.[6][14]

Table 4.1: Token Id Assignment

CLASS OF TOKEN	ID
Operators	
+	1
/	2
…	…
Identifiers	9
Keywords	
Private	11
Public	12
Protected	13
…	…
Punctuation symbols	
[	18
]	19
(	20
…	…
Data types	
int	31
Float	32
double	33

Table 4.2: Example of Tokenization

Eg:- public int division (int c, int d) {
 	
int a; return c / d;
 	
}
 	
Token string of above code is given below:-
 	
12 31 9 20 31 9 25 31 9 21 27 31 9 26 14 9 1 9 26 28

b) To find the repeated token segments or clones within the files, a space efficient Suffix array algorithm has been put in use to find or locate the non-extendible repeats in the token string. Non extendible token repeats are those repeats which are not always followed or preceded by same symbols.

Suffix array computes repeated token sequences with the time complexity of O (N)[32]. Moreover it is a simpler algorithm. The space consumption while constructing the suffix array is five times lesser than other algorithms such as Suffix tree, etc. [32]

In order to calculate the repeated clones or repeating token strings in two source code files, these files need to be integrated together in order to get the single file. Then after obtaining the single source code file, suffix array algorithm has been applied on it. Suffix array provides a complete set of repeats. These repeats are mapped with the original two source files to trace the locations of these repeated clones.

c) As suffix array returns full sets of non-extendible repeats along with location, hence the proposed tool directly aggregated clone classes based on that .This was not possible in the already existing techniques as in those there is a requirement of some additional post processing step.[6][14]

Given below is the example of how non extendible repeats or clones are computed manually:-[6][14]

Table 4.3: Example of manually computed clones

P I G H L H J Y U O H L G H Y U K

So, a complete set of non-repeating substrings are:-

GH- 3, 13

HL- 4 , 11

U- 9, 16

In this example, as G is always followed by H .In the same way L is also always proceeded by a same symbol H. So it is an extendible repeat. But they are not extendible in the pairs of GH and HL. Therefore, GH and HL are no-extendible repeats or clones.

These clones are computed automatically by making the use of suffix array algorithm.[1]

Table 4.4: Example of automated computation of clones using suffix array

I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
S	W	O	R	K	K	W	O	R	K	K	X	R	R	K	W	O
S.A	4	0	5	8	9	3	13	11	3	4	2	12	2	7	11	1
LCP	-1	1	4	2	2	0	2	3	1	1	0	3	4	2	1	0
Rank	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

In this demonstration, i -> index

S->String of characters

S.A-> Array of suffixes arranged in a lexicographical order.

LCP->Array contains longest common prefix Length.

It can be denoted as LCP[i] = LCP (S.A [i-1], S.A[i]) where $0 < i \leq p-1$.

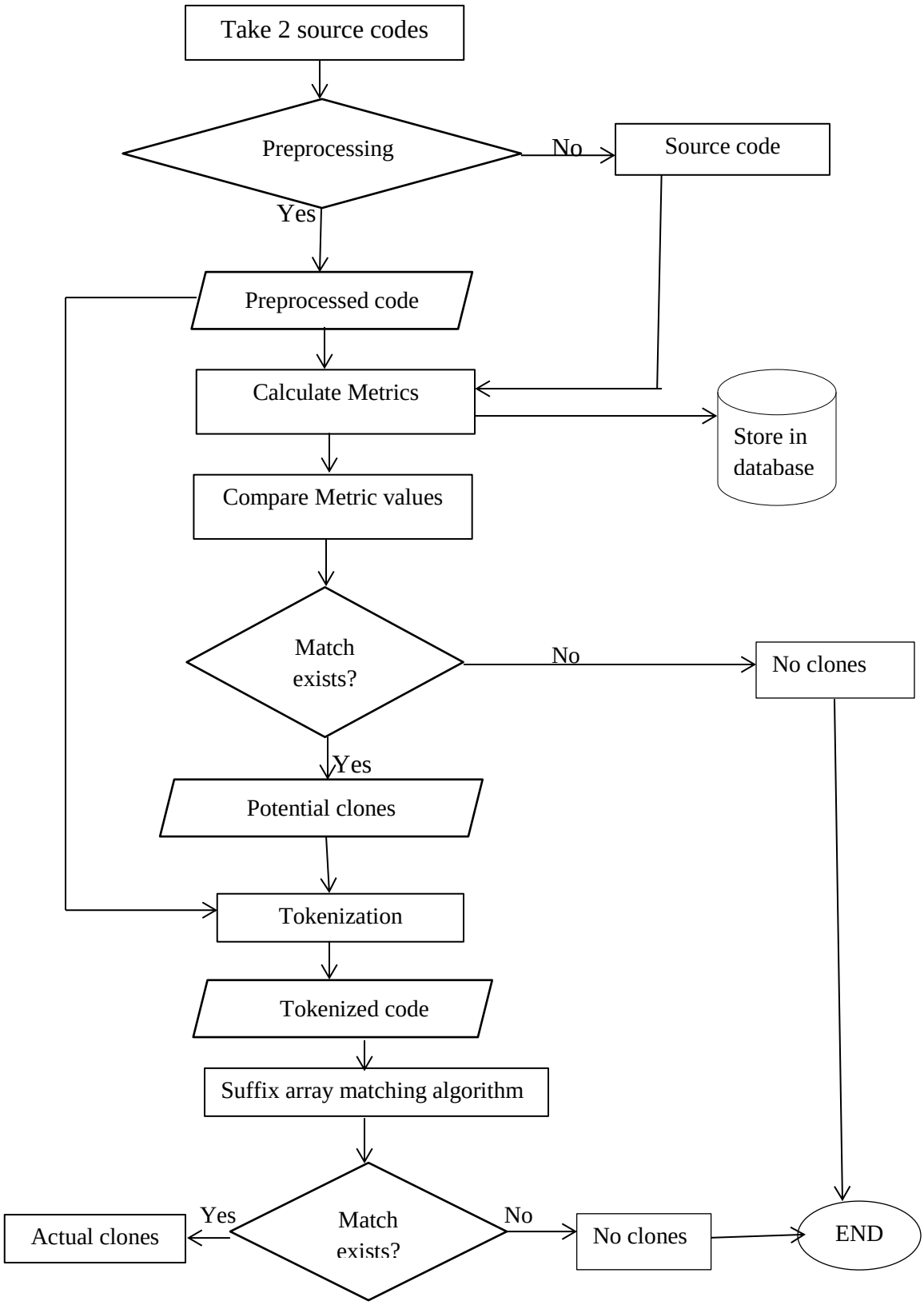
Now, the repeating sequences or subsequences from the source code can be calculated out by following procedure:-

1.) If LCP[i]<LCP[i+1], then repeats or clones may occur at the corresponding positions denoted by S.A[i] and S.A[i+1] and these repeats act as a clone pair in the source code. For example, LCP[2]<LCP[2], which means that repeats may occur at positions given by S.A[2] and S.A[2] i.e. repeats occurs at 0^{th} and 5^{th} position.

2.) If LCP[i]=LCP[i+1],then repeats may sustained at positions given by S.A[i] and S.A[i+1].

3.) If LCP[i]>LCP [i+1], then there are no clones found. For example, LCP[2]>LCP[2], which means no repeats occur at positions given by S.A[2] and S.A[2] i.e. no repeats occurs at 5^{th} and 8^{th} position.

Flowchart of Proposed Technology



CHAPTER 5
RESULTS & DISCUSSION

5. 1 IMPLEMENTATION

The implementation of the tool for proposed technique has been carried out in Java language. The software being used for carried out the implementation of tool is Netbeans8.1. The tool detects the clones in Java language and Asp.net language. This technique can detect Type1, Type2 and Type 3 clones.

Given below are the screenshots which shows how the detection of clones can take place in :-

- Java language
- Asp.net language.

5.1.1 FOR JAVA LANGUAGE (TYPE 1 CLONE)

The clone of this type is also called exact clones. In this type of clone, the cloned code is exactly same as that of the original code. The only change that makes difference is the presence of comments in the cloned code. [25]

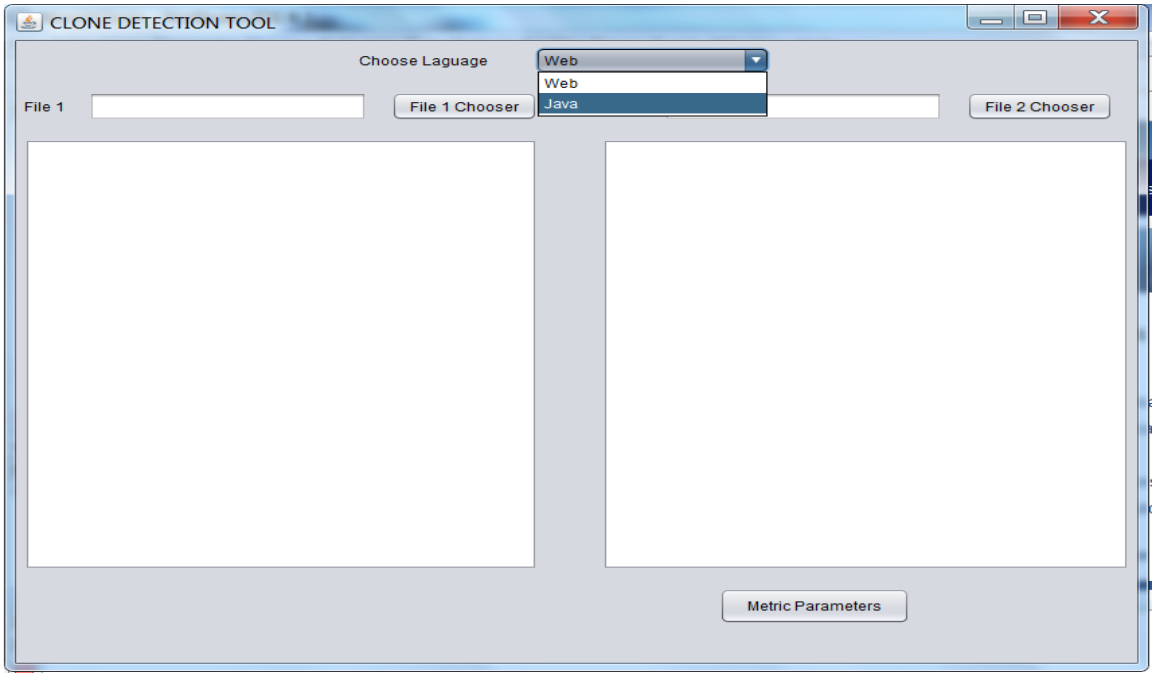


Figure 5.1 : Clone Detector Tool

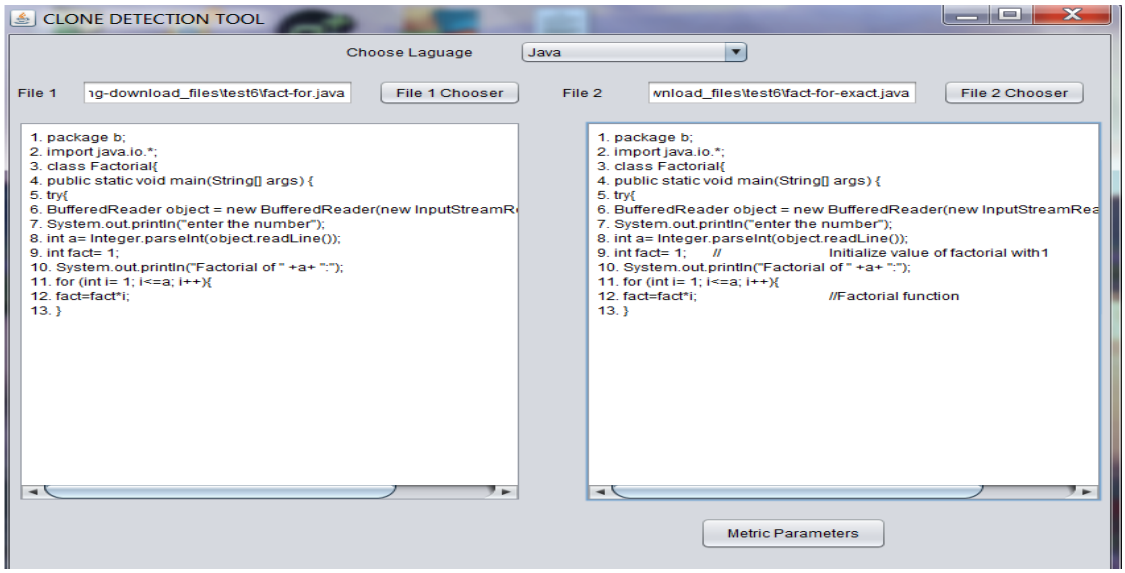


Figure 5.2 : Input File chooser

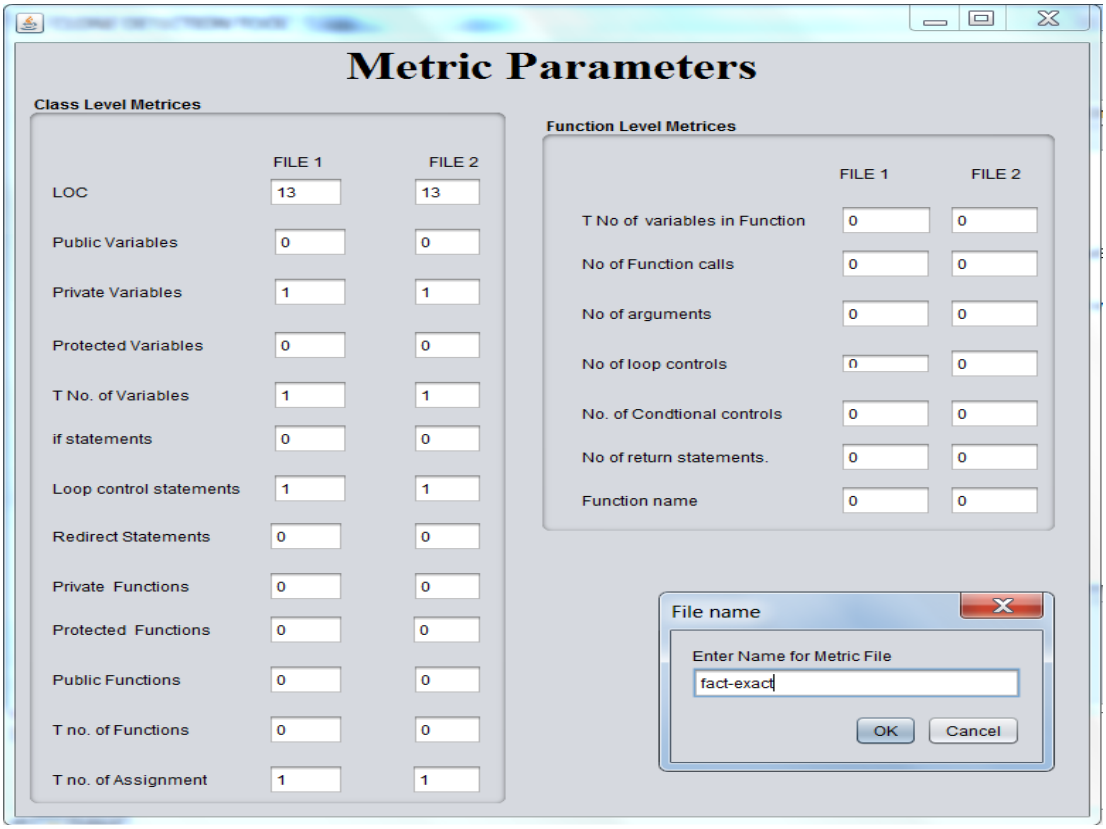
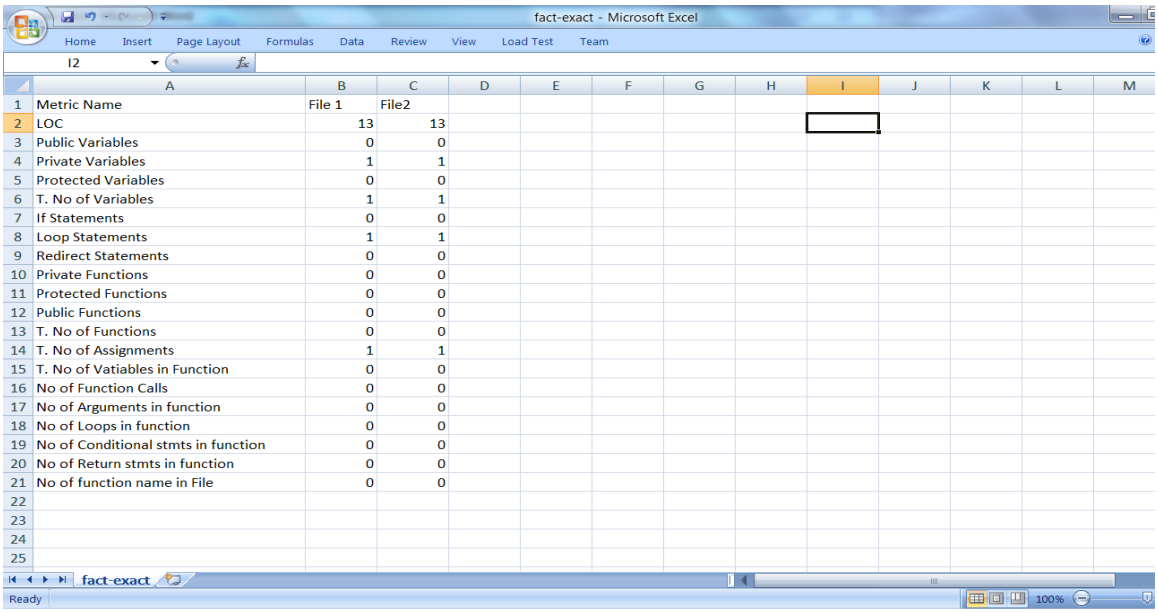


Figure 5.3: Metric calculation for Java -Type1 clones

Figure 5.3 shows the calculated Class level and Function level metrics for input files as shown in Figure5.2



	A	B	C
1	Metric Name	File 1	File2
2	LOC	13	13
3	Public Variables	0	0
4	Private Variables	1	1
5	Protected Variables	0	0
6	T. No of Variables	1	1
7	If Statements	0	0
8	Loop Statements	1	1
9	Redirect Statements	0	0
10	Private Functions	0	0
11	Protected Functions	0	0
12	Public Functions	0	0
13	T. No of Functions	0	0
14	T. No of Assignments	1	1
15	T. No of Vatiables in Function	0	0
16	No of Function Calls	0	0
17	No of Arguments in function	0	0
18	No of Loops in function	0	0
19	No of Conditional stmts in function	0	0
20	No of Return stmts in function	0	0
21	No of function name in File	0	0

Figure 5.4: Metrics stored in excel

Figure 5.4 shows the storage of calculated Metrics in Database (excel)

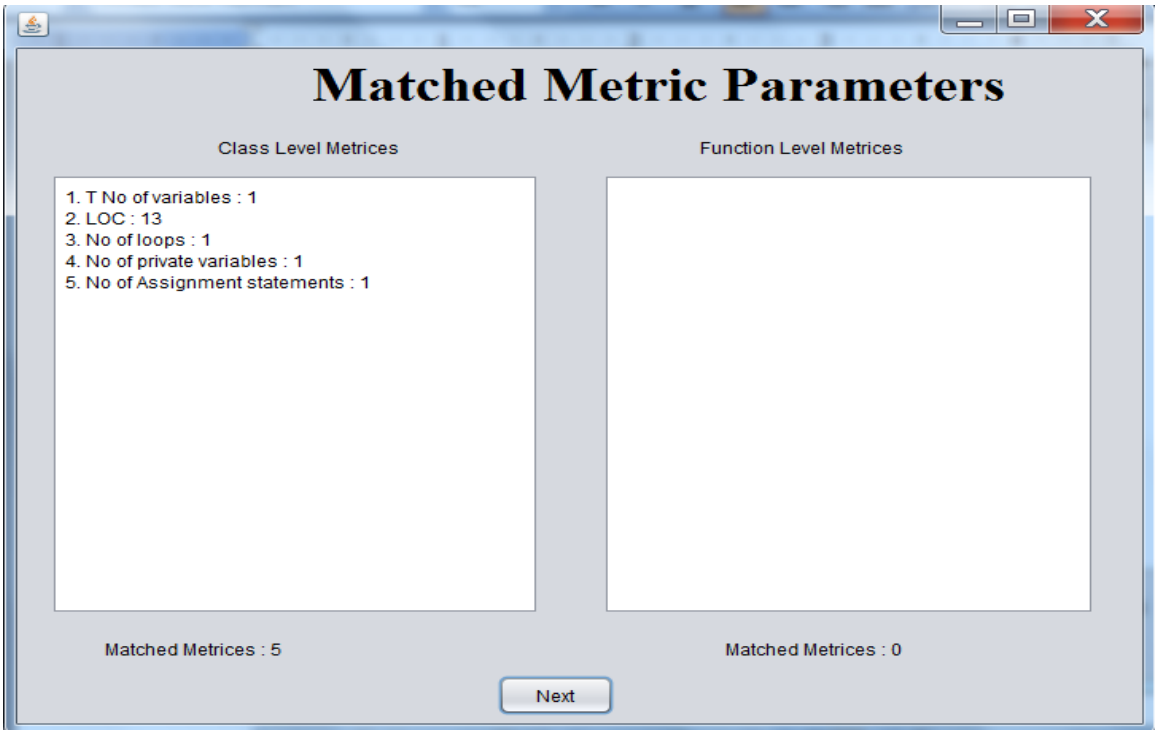


Figure 5.5: Matched parameters for Type1 clone.

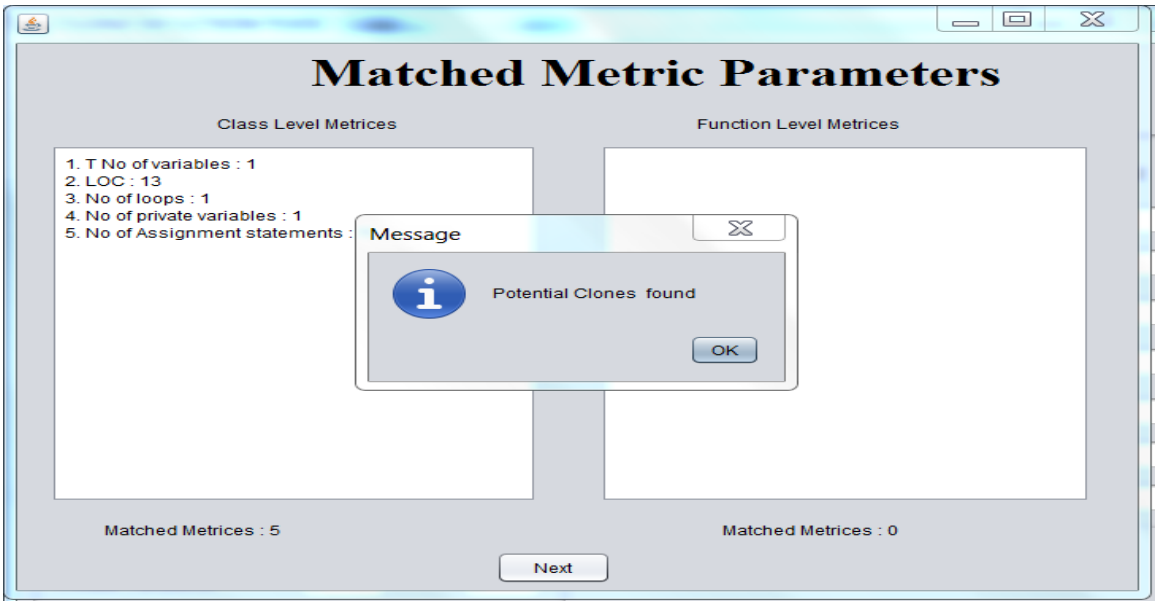


Figure 5.6: Potential clones

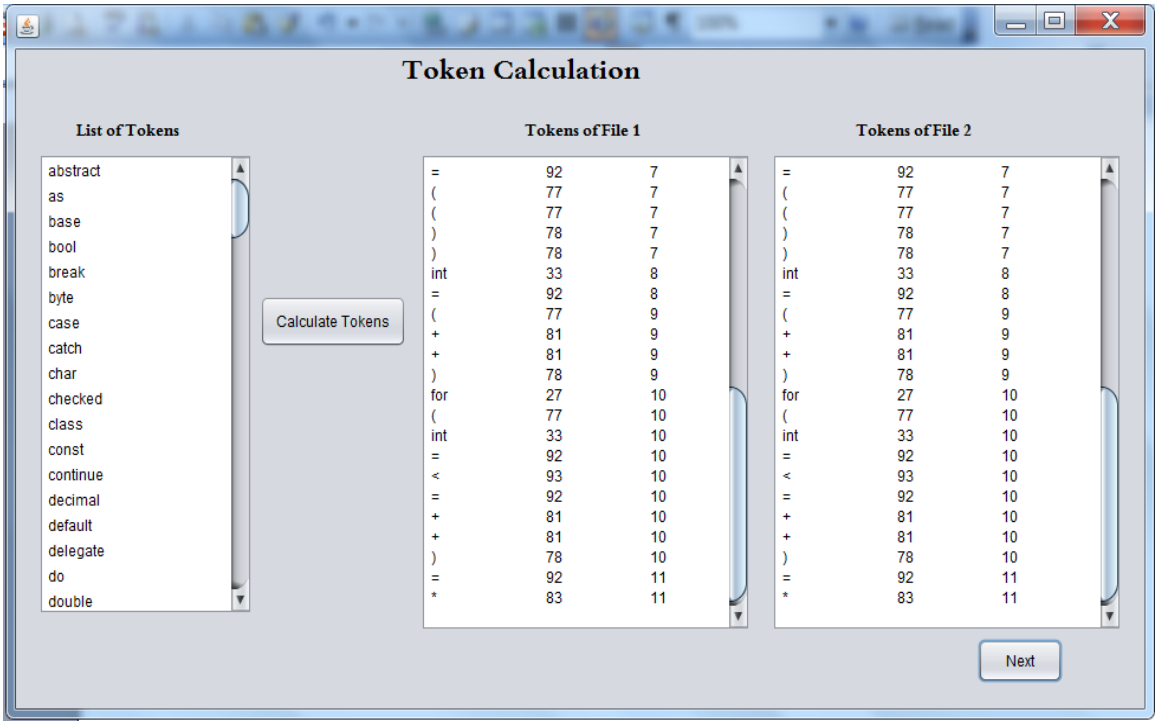


Figure 5.7: Token calculation

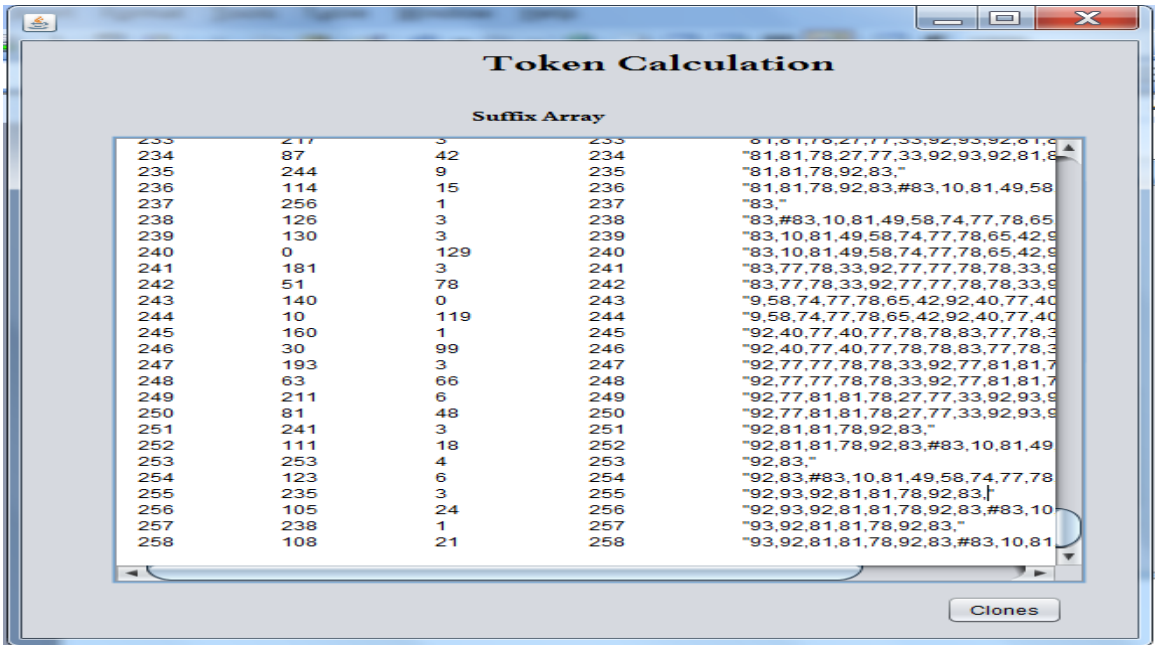


Figure 5.8: Suffix Array execution

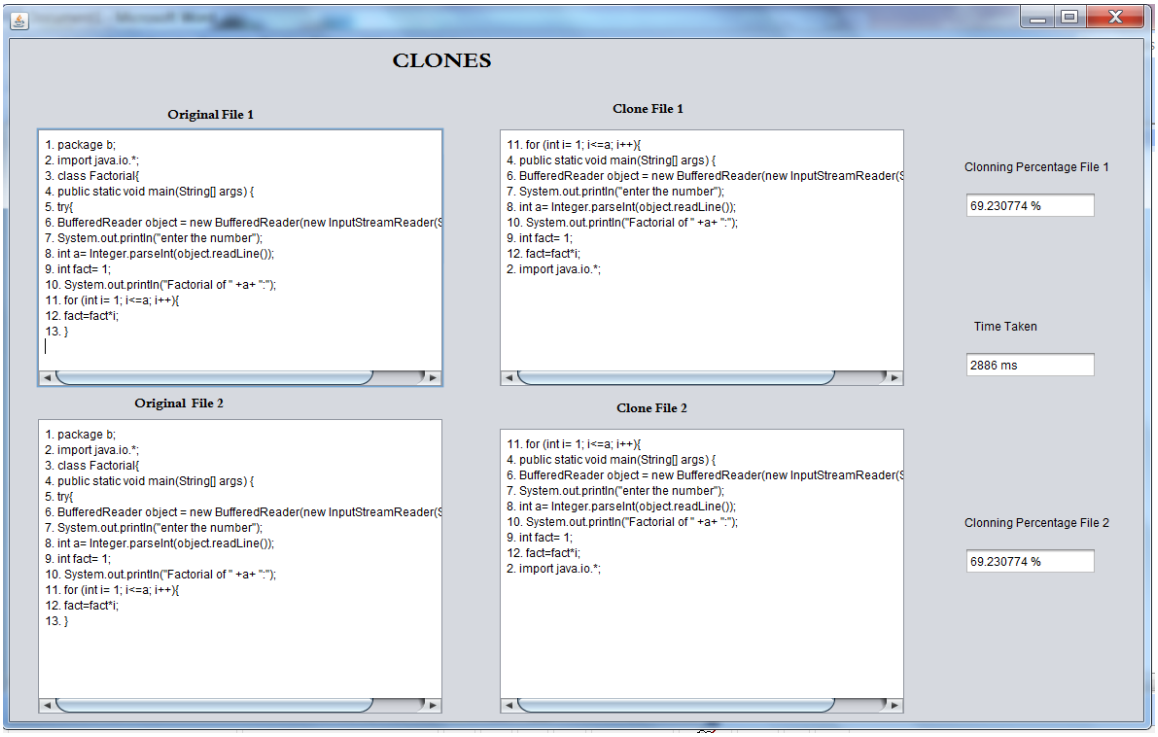


Figure 5.9: Type1 clones detected for Java

Figure 5.8 shows the execution of Suffix array for Token matching and Figure 5.9 shows the detection of clones w.r.t the original files along with the cloning % and Time taken(for computing clones) .

5.1.2 FOR JAVA LANGUAGE (TYPE 2 CLONE)

In this type of clone, the cloned code is not same but exactly similar to the original code due to slight renaming in variables, literals, identifiers, etc. [25]

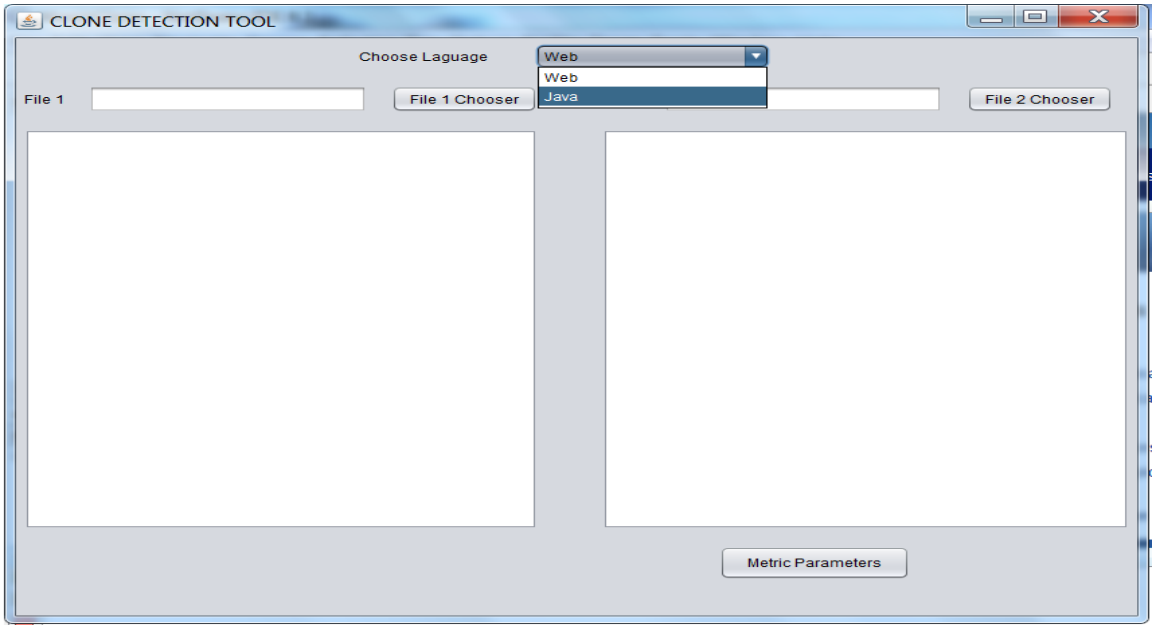


Figure 5.10: Clone detector Tool

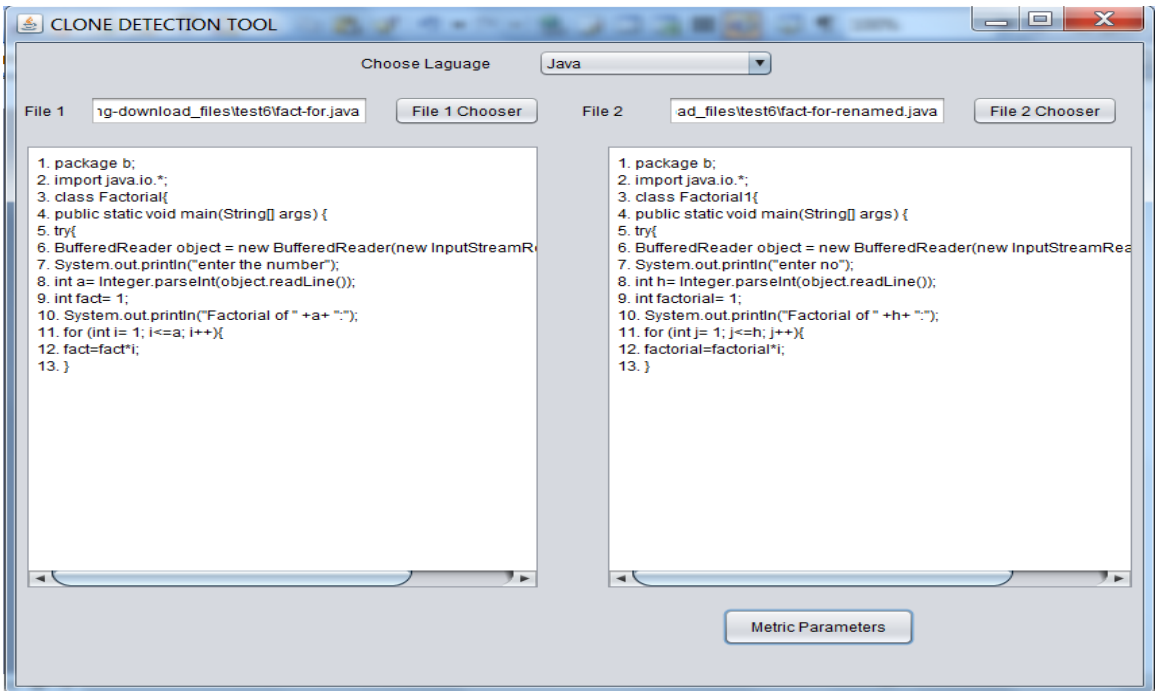


Figure 5.11 : Input File chooser

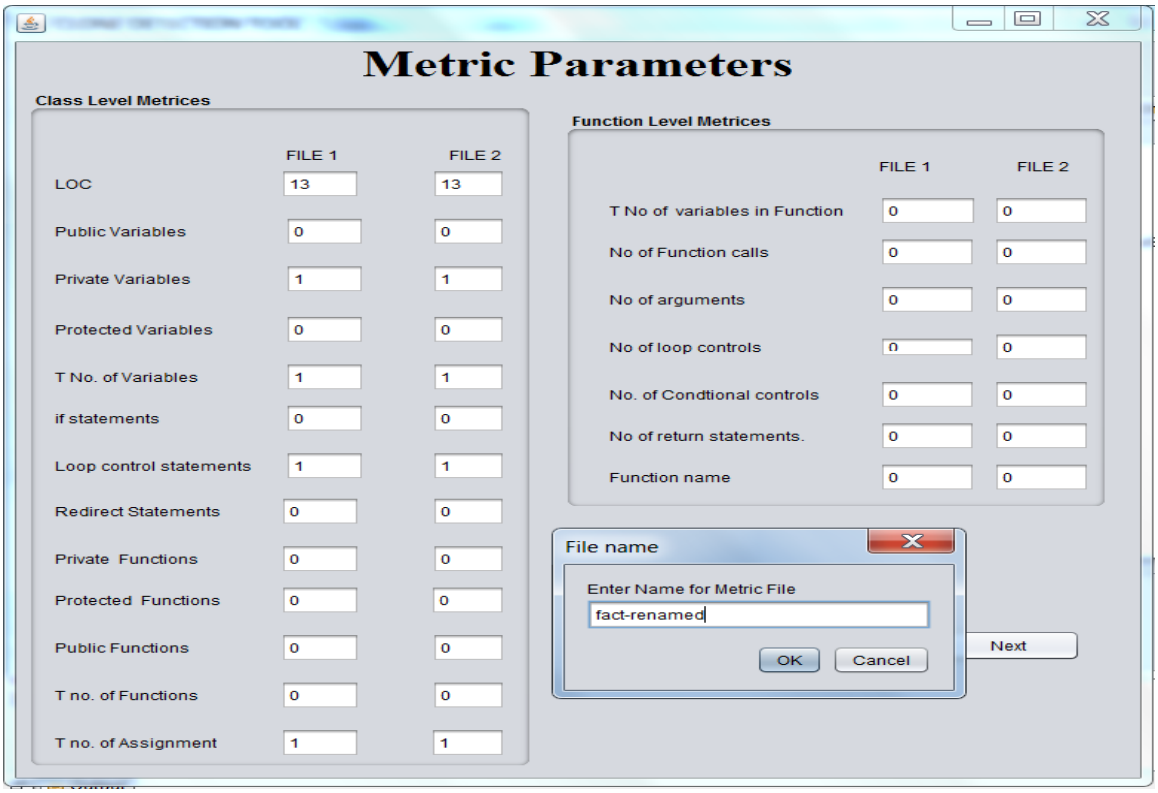


Figure 5.12: Metric calculation for Java-Type2 clones

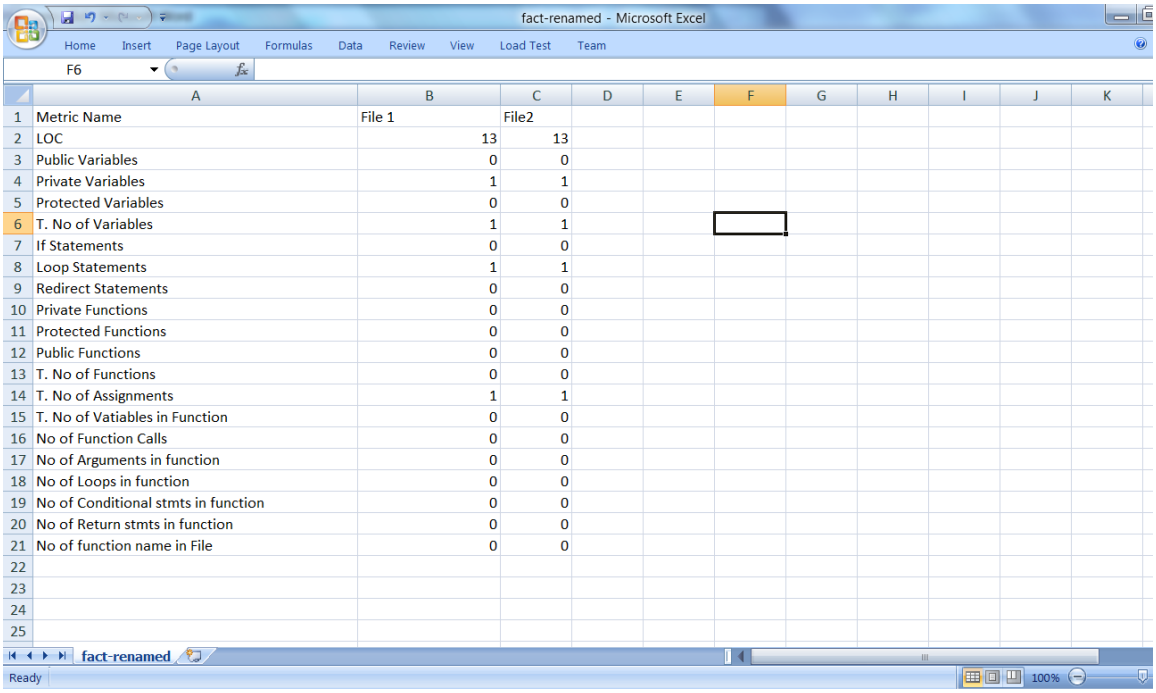


Figure 5.13: Metrics stored in Excel

Figure5.13 shows the storage of metrics in excel file.

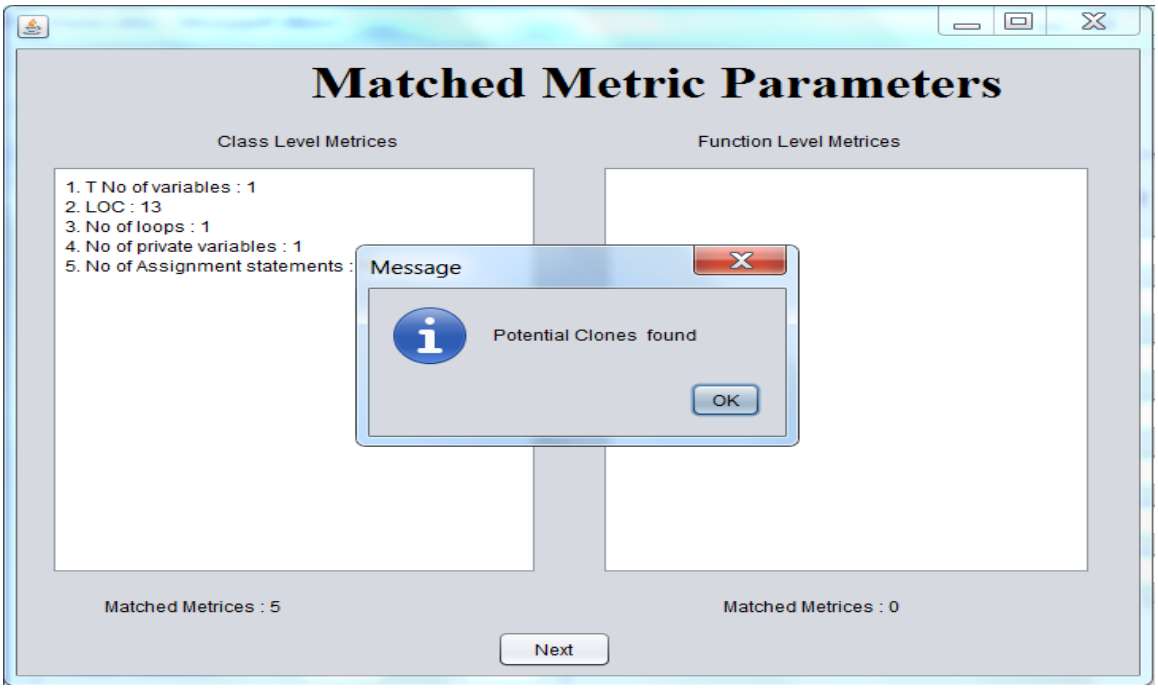


Figure 5.14: Potential clones found for matched parameters

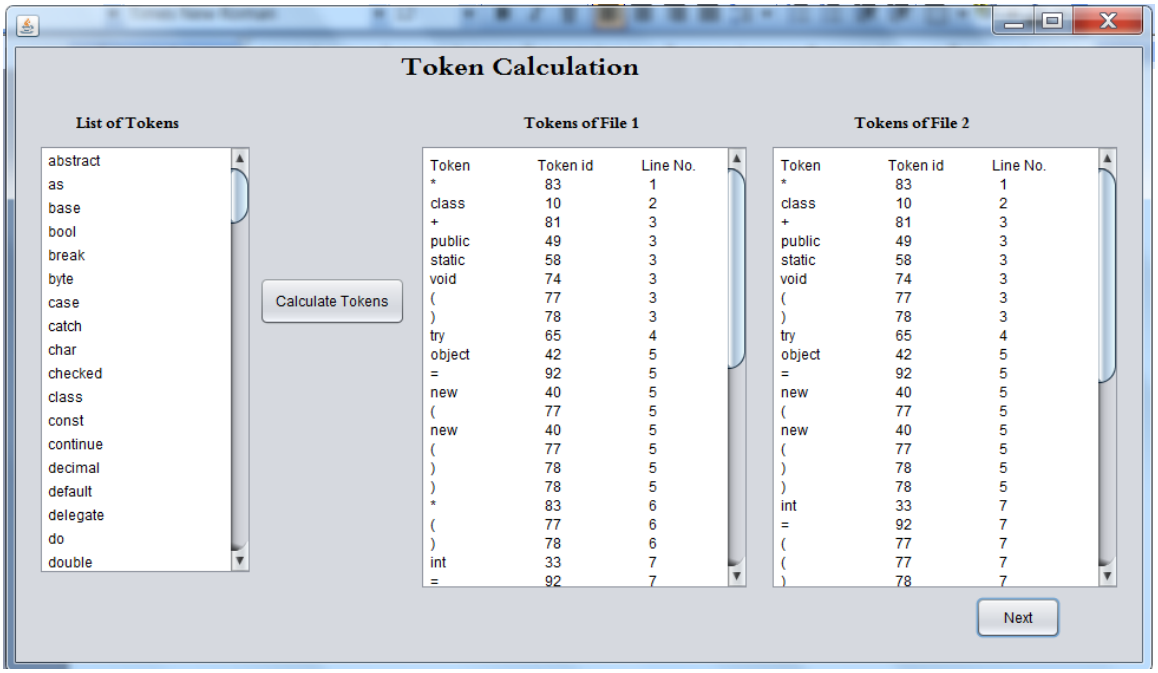


Figure 5.15: Token calculation

Once the Potential clones are found by Metrics Approach, then only the tool will work further for Token based Approach to detect Actual clones in 2 source files.

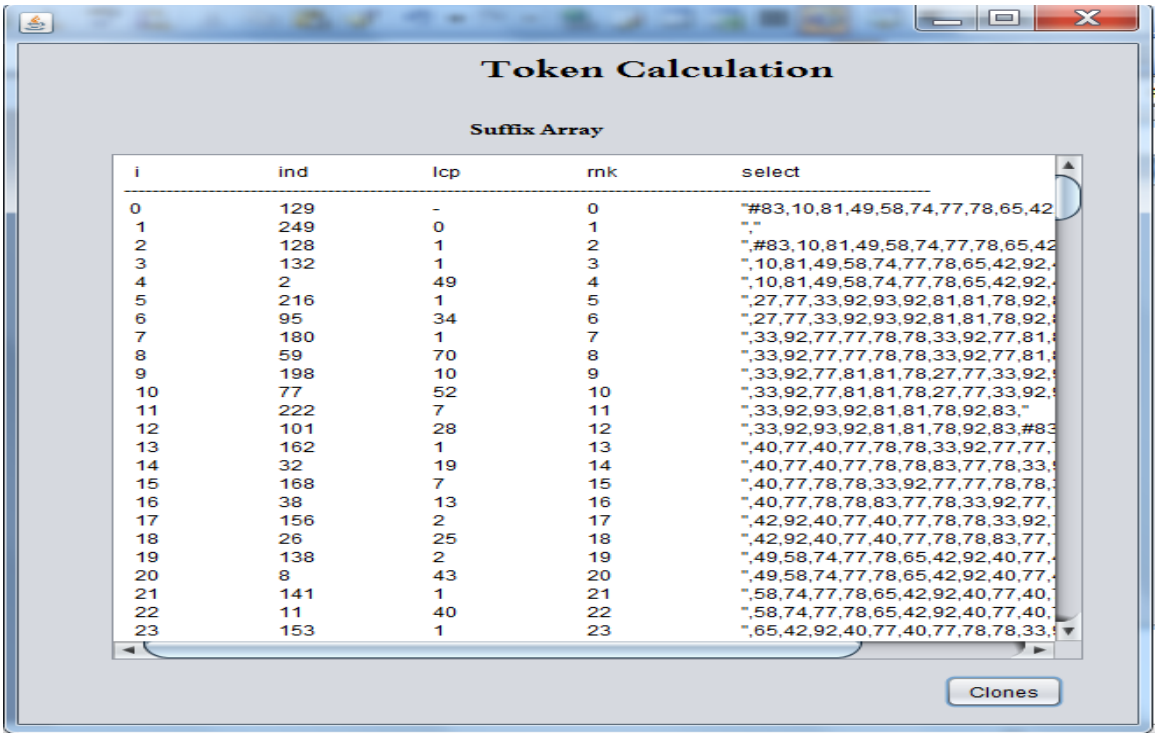


Figure 5.16: Suffix array execution

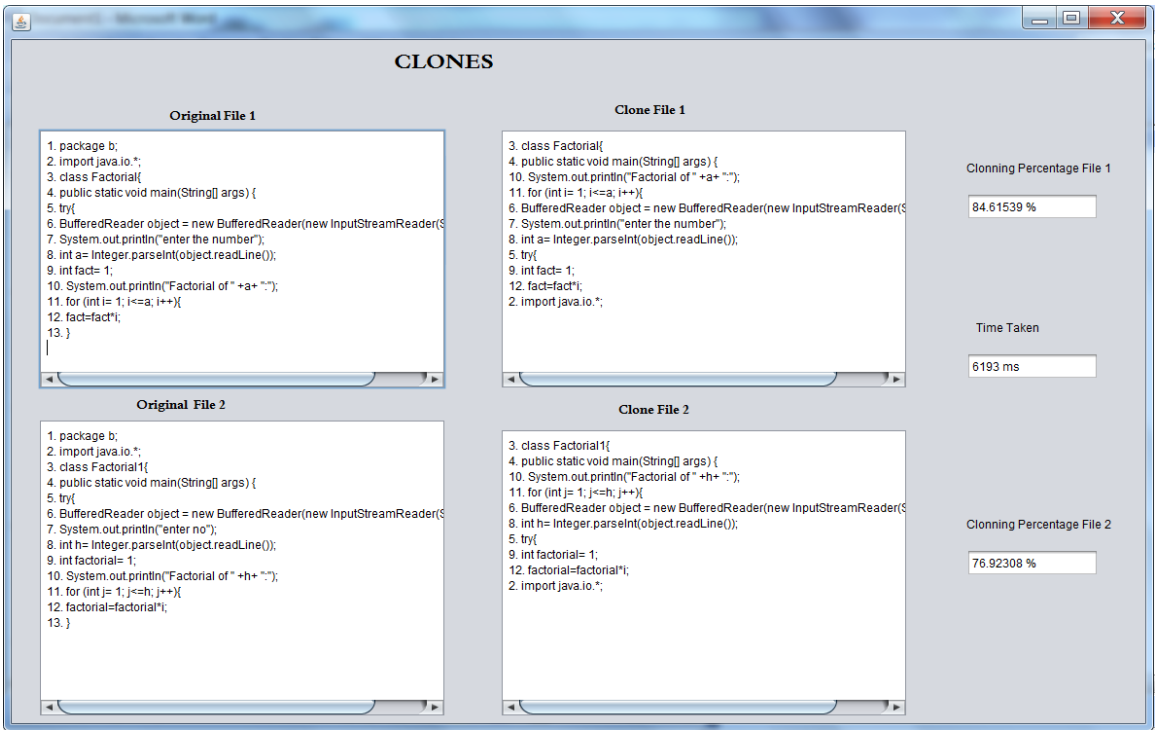


Figure 5.17: Type 2 clones detected for Java

Figure5.16 shows that how Suffix array works and Figure5.17 shows the computation of clones.

5.1.3 FOR JAVA LANGUAGE (TYPE 3 CLONE)

In this type of clones, there is addition, deletion and modification of the statements. [25]

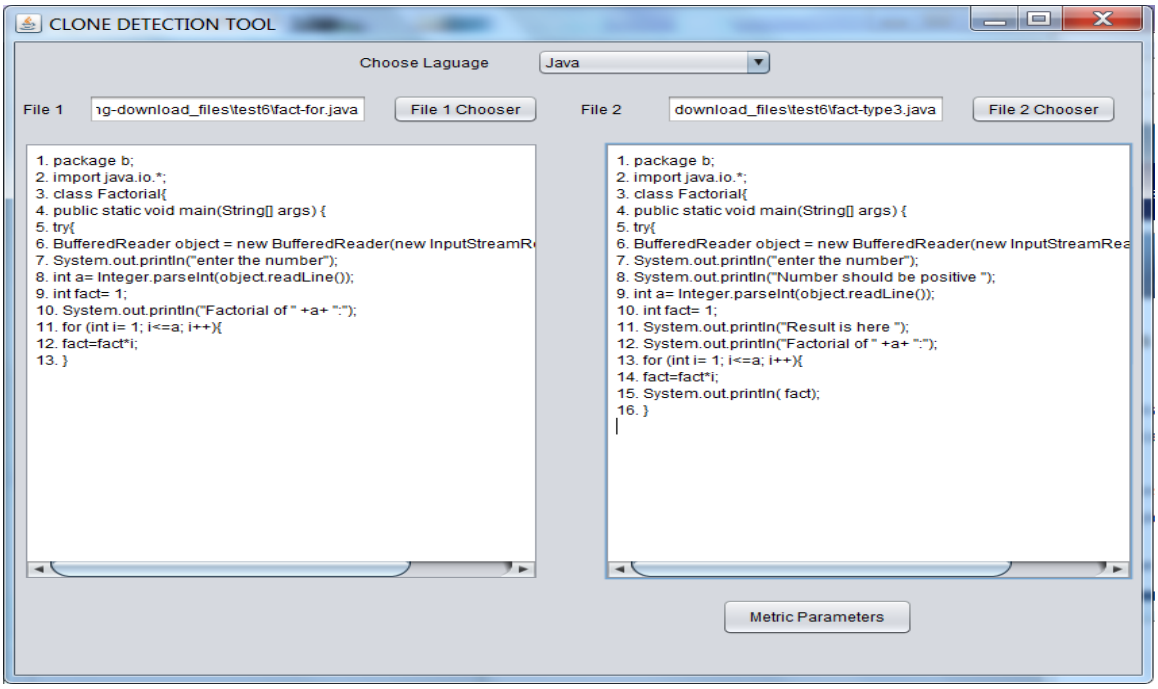


Figure 5.18 :Input File chooser

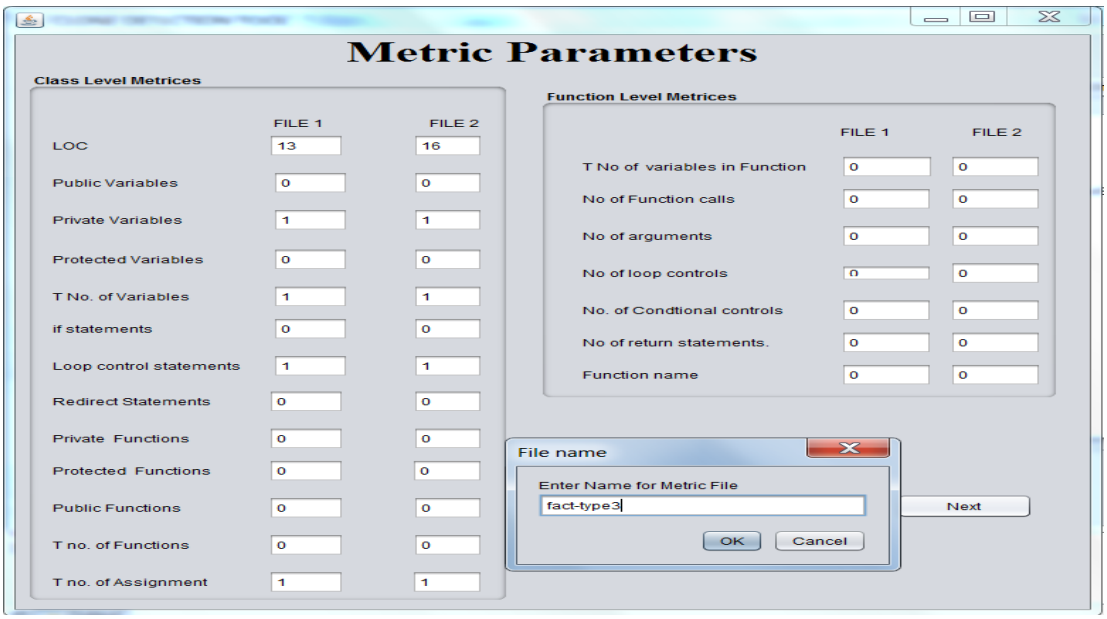
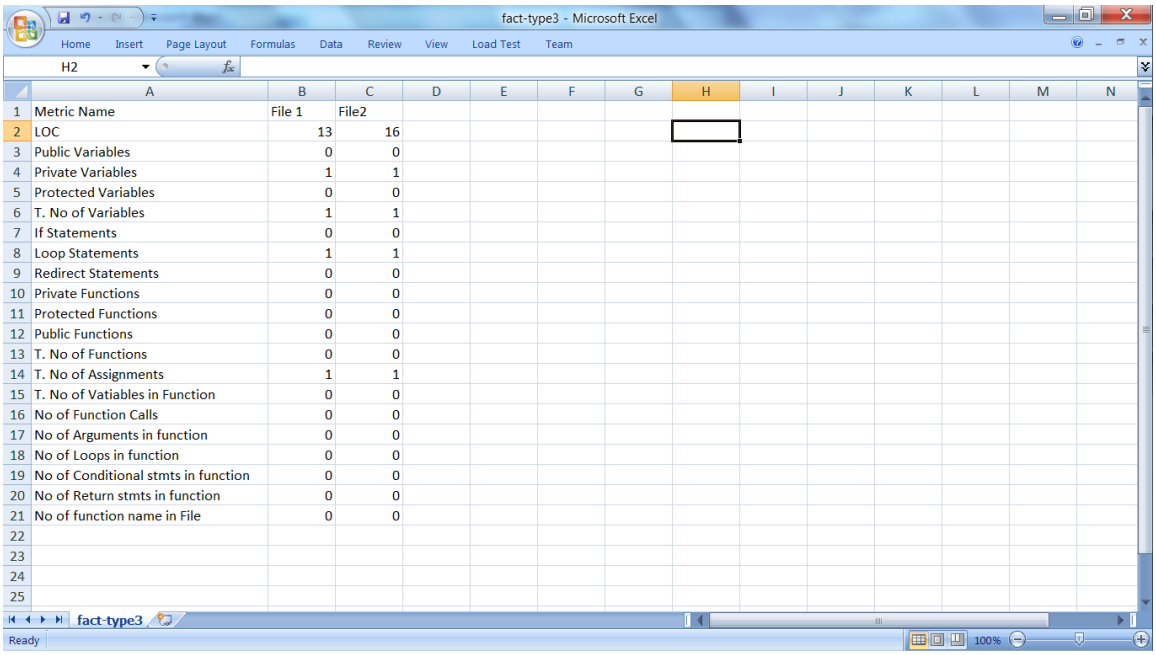


Figure 5.19: Metric Calculation for java-Type3 clones.



	A	B	C
1	Metric Name	File 1	File2
2	LOC	13	16
3	Public Variables	0	0
4	Private Variables	1	1
5	Protected Variables	0	0
6	T. No of Variables	1	1
7	If Statements	0	0
8	Loop Statements	1	1
9	Redirect Statements	0	0
10	Private Functions	0	0
11	Protected Functions	0	0
12	Public Functions	0	0
13	T. No of Functions	0	0
14	T. No of Assignments	1	1
15	T. No of Vatiables in Function	0	0
16	No of Function Calls	0	0
17	No of Arguments in function	0	0
18	No of Loops in function	0	0
19	No of Conditional stmts in function	0	0
20	No of Return stmts in function	0	0
21	No of function name in File	0	0

Figure 5.20 : Metrics stored in excel

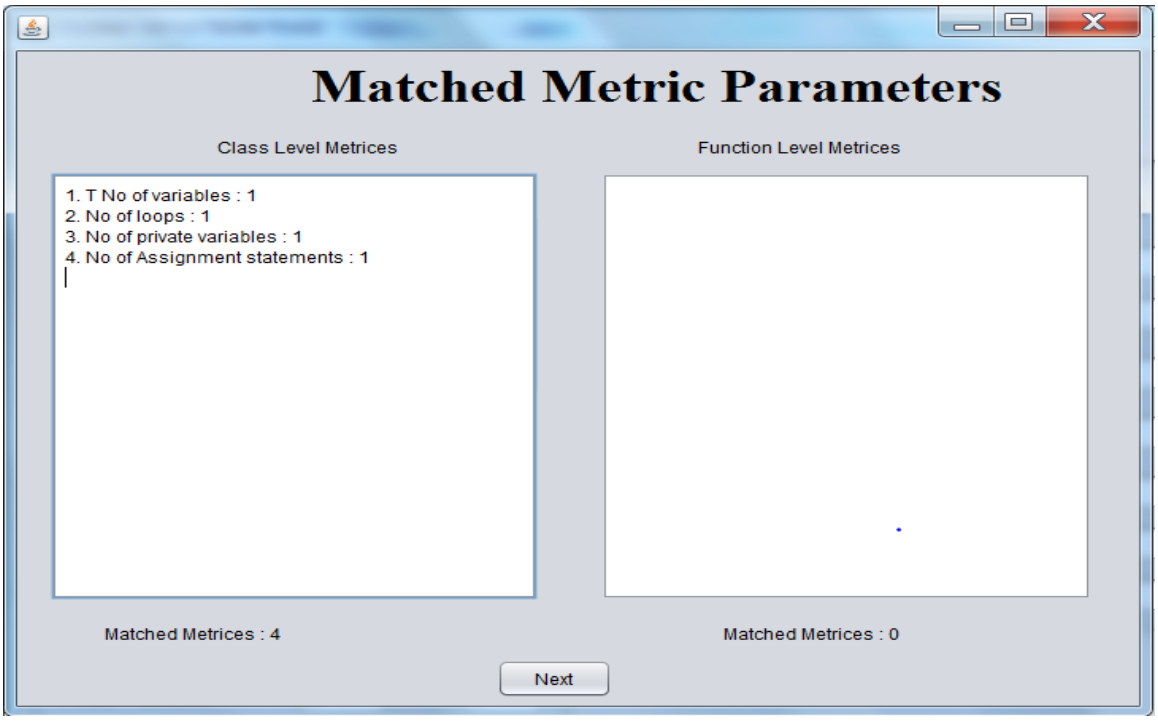


Figure 5.21: Matched parameters for Type 3 clones

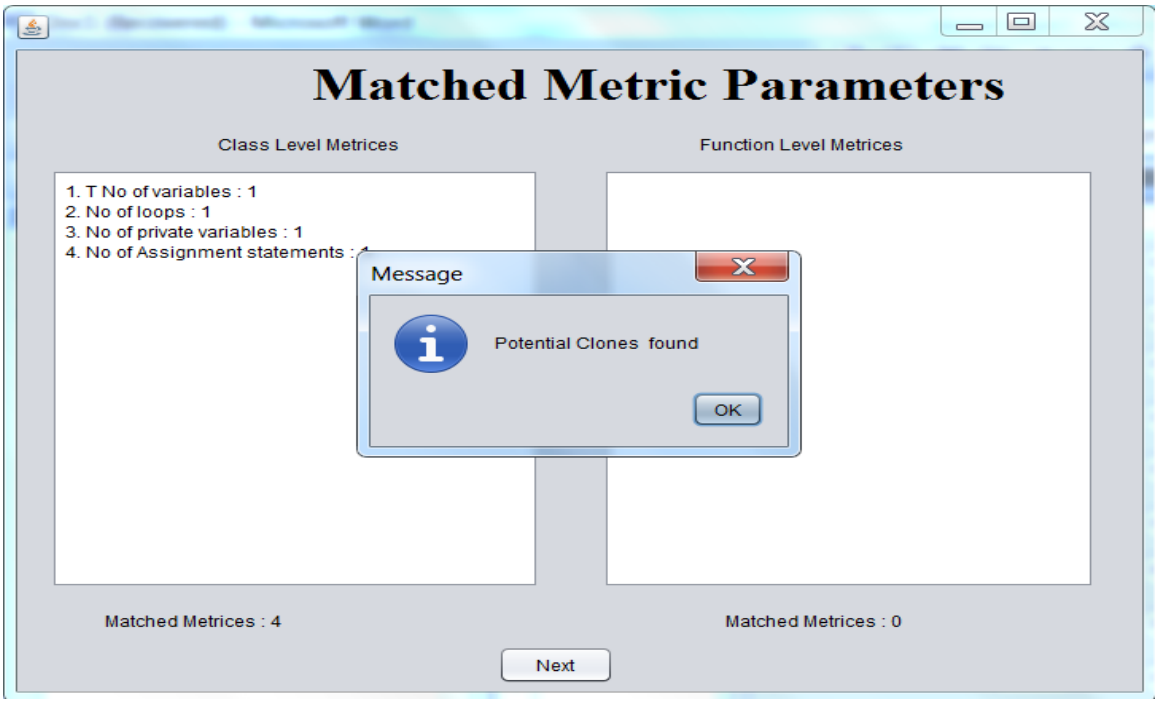


Figure 5.22 : Potential clones

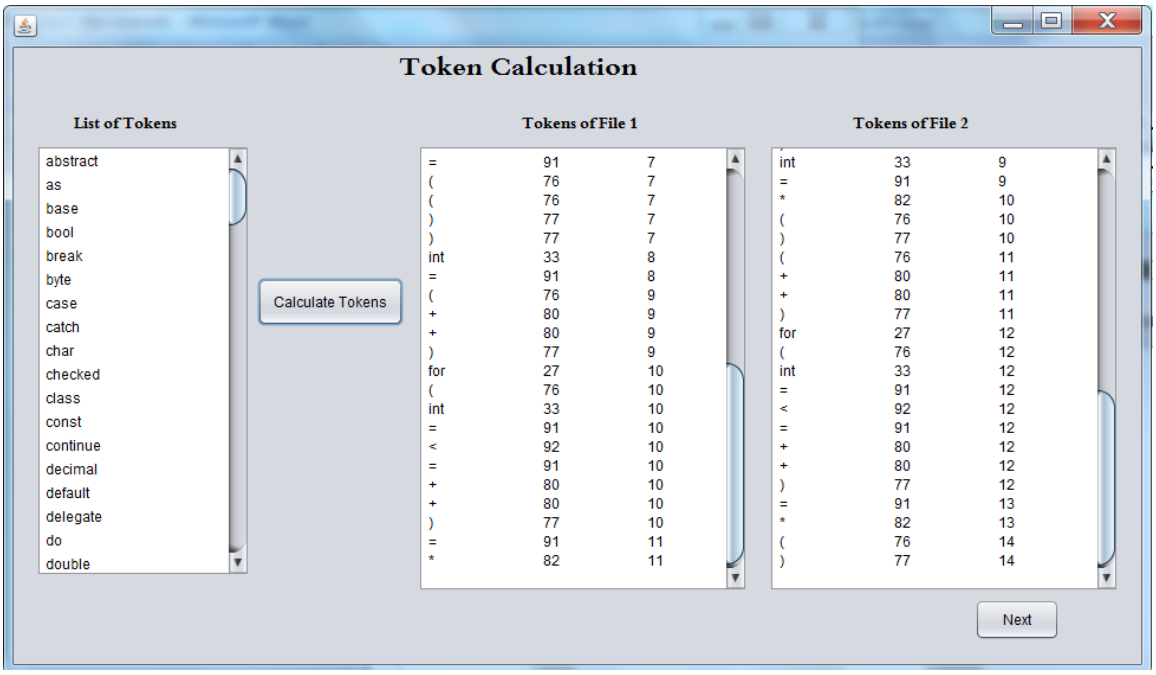


Figure 5.23: Token calculation

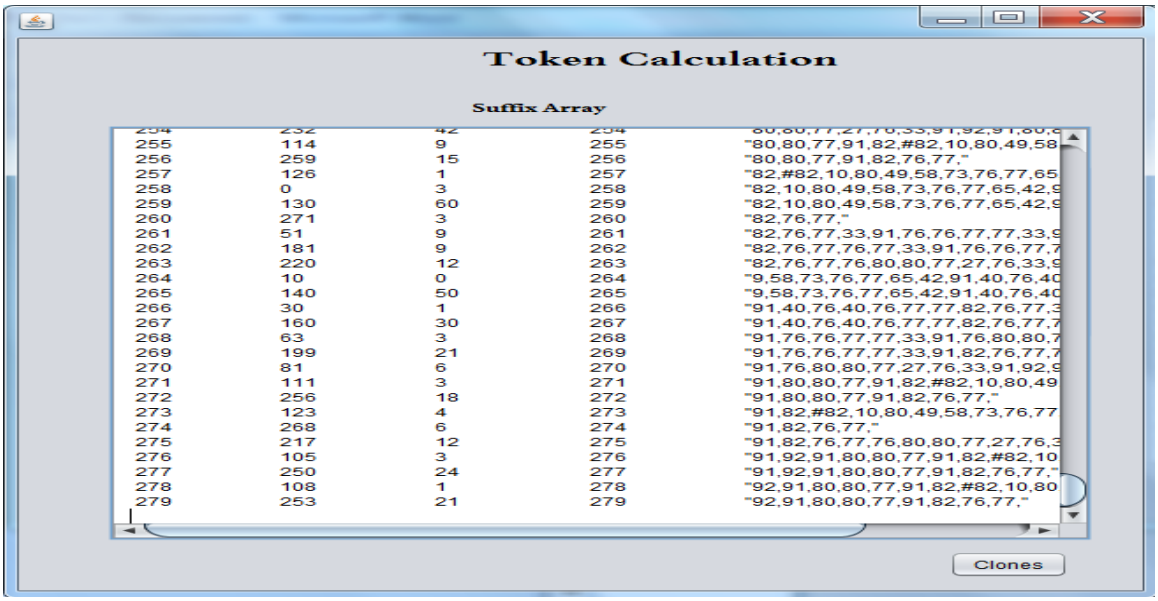


Figure 5.24 : Suffix Array execution

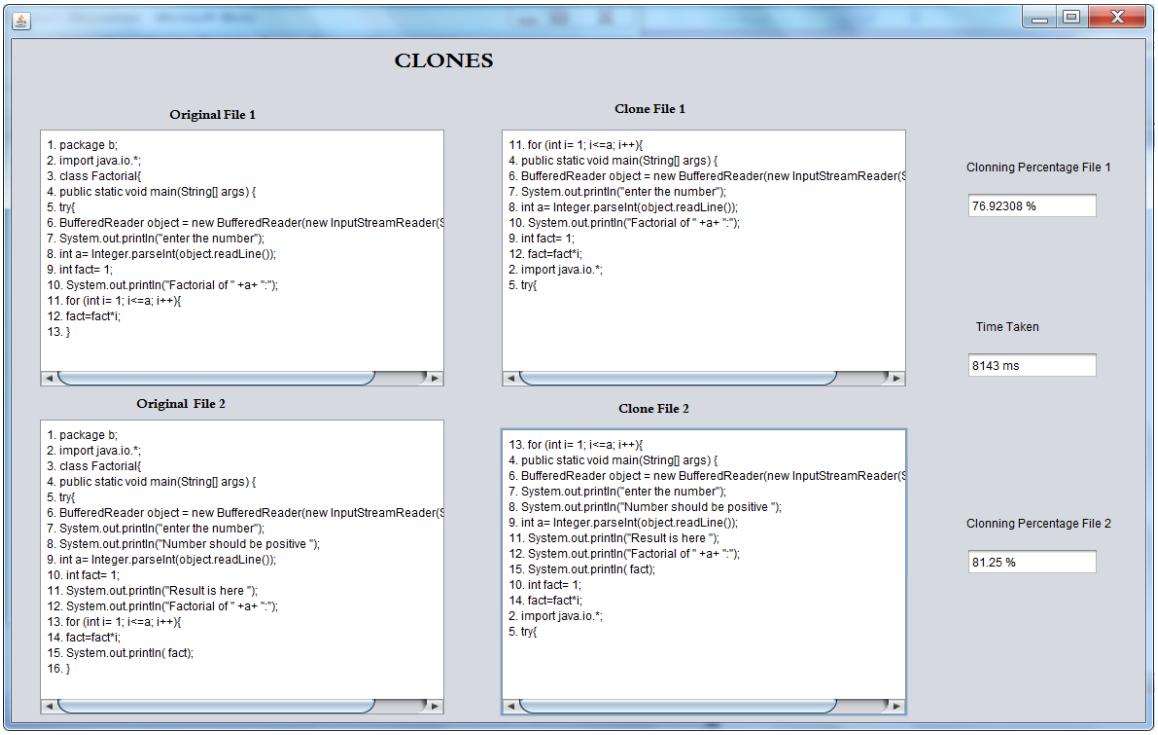


Figure 5.25: Type 3 clones detected for java

Above Figures shows the detection of clones for Java language. Figures given below shows the detection of clones in Asp.net language.

5.1.4 FOR ASP.NET LANGUAGE (TYPE1 CLONES)

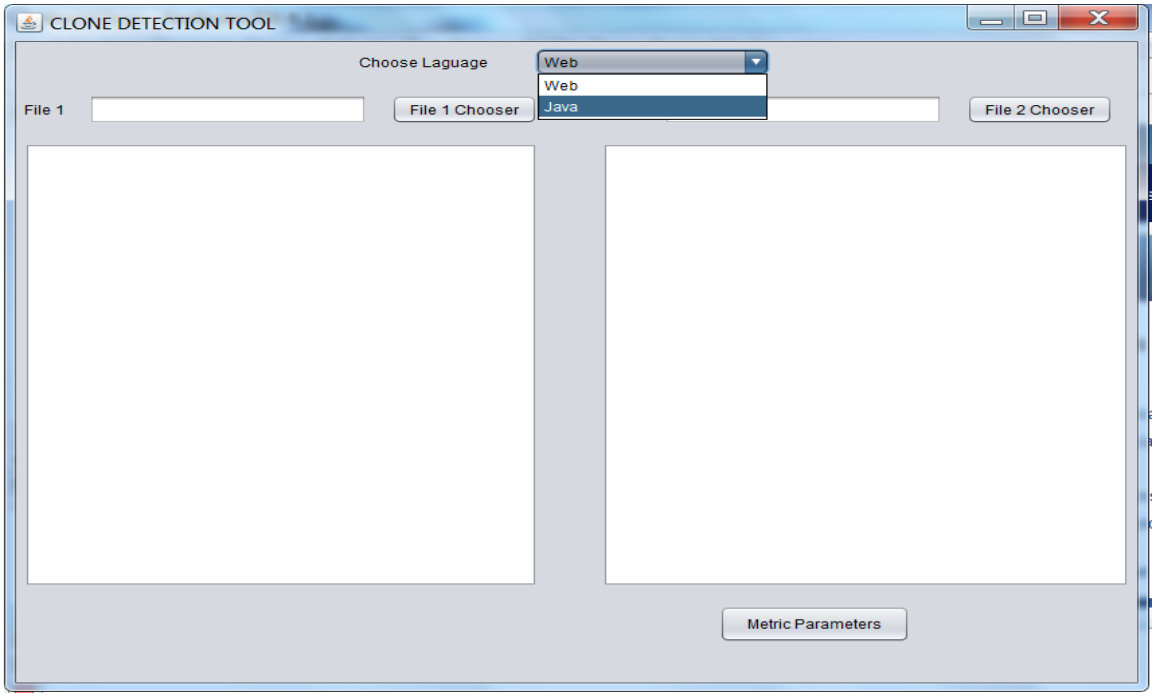


Figure 5.26 :Clone Detector tool

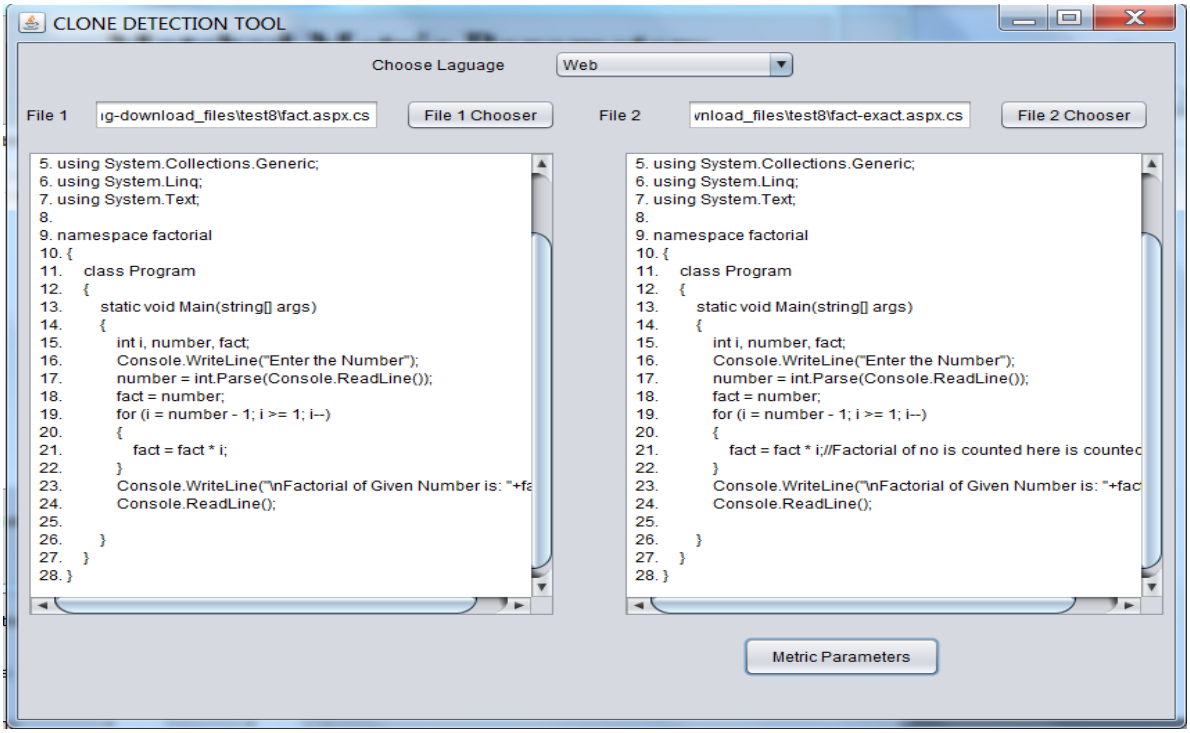


Figure 5.27 : Input File chooser

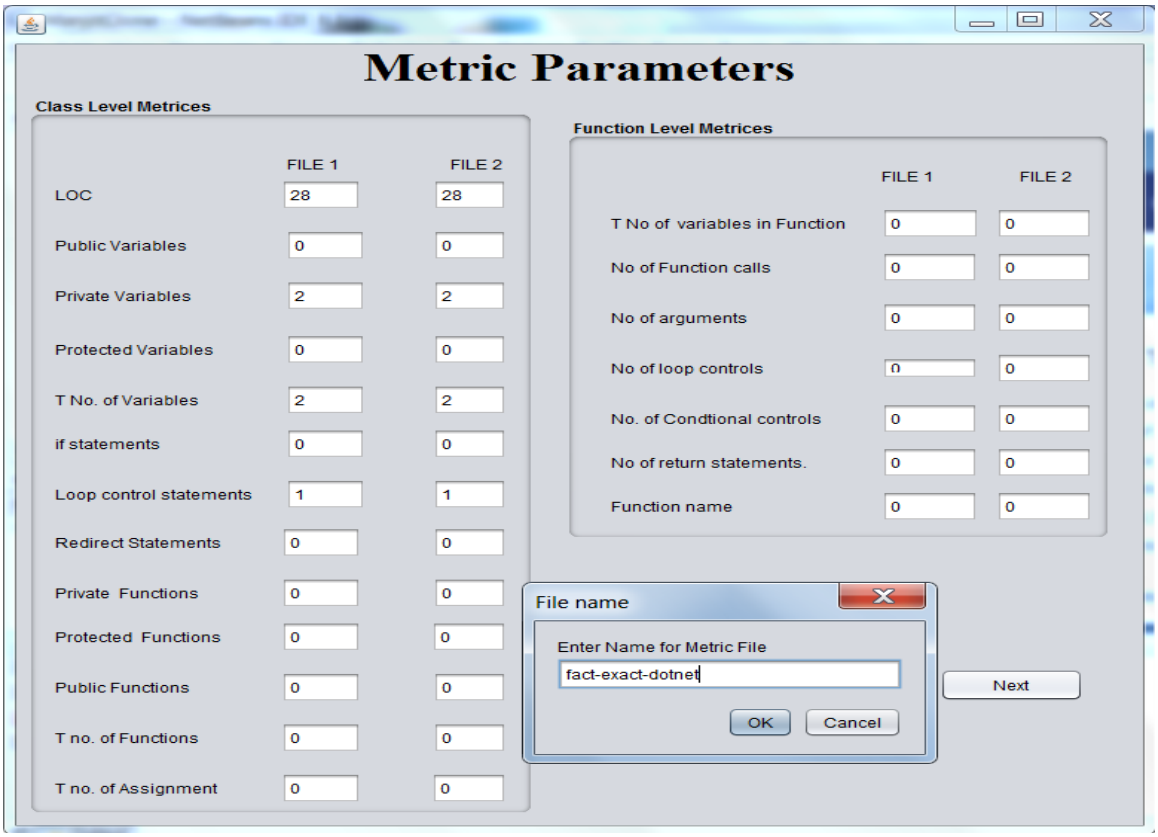


Figure 5.28 :Metric calculation for Asp.net-Type1 clones.

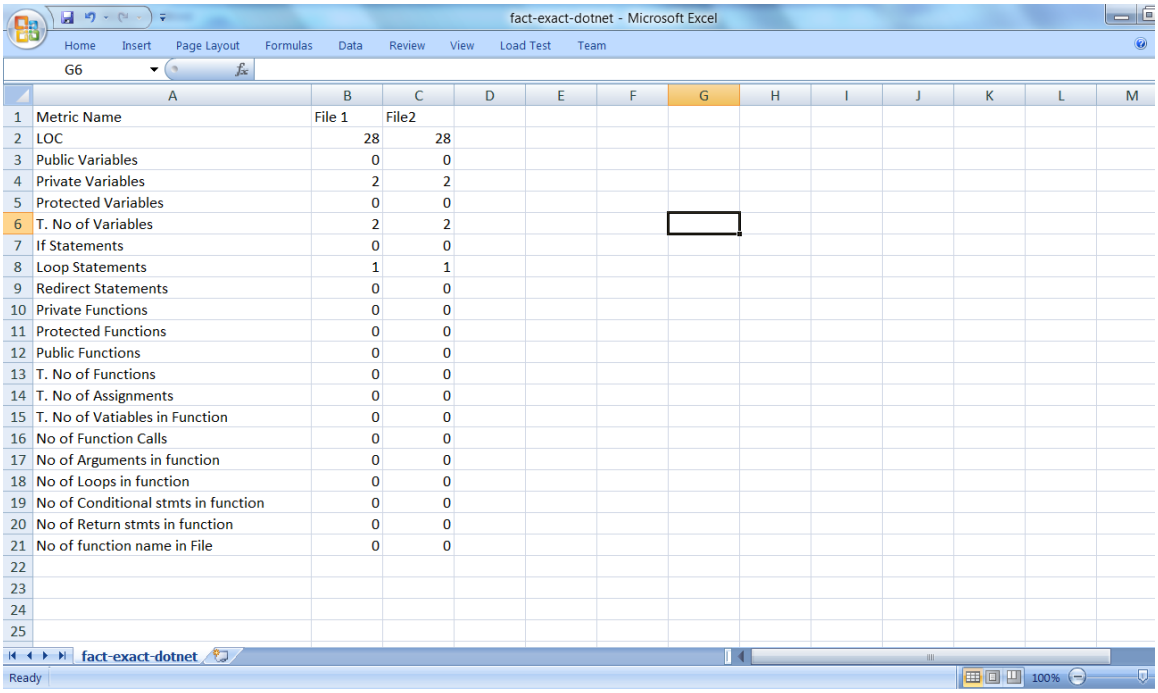


Figure 5.29 :Metrics stored in excel

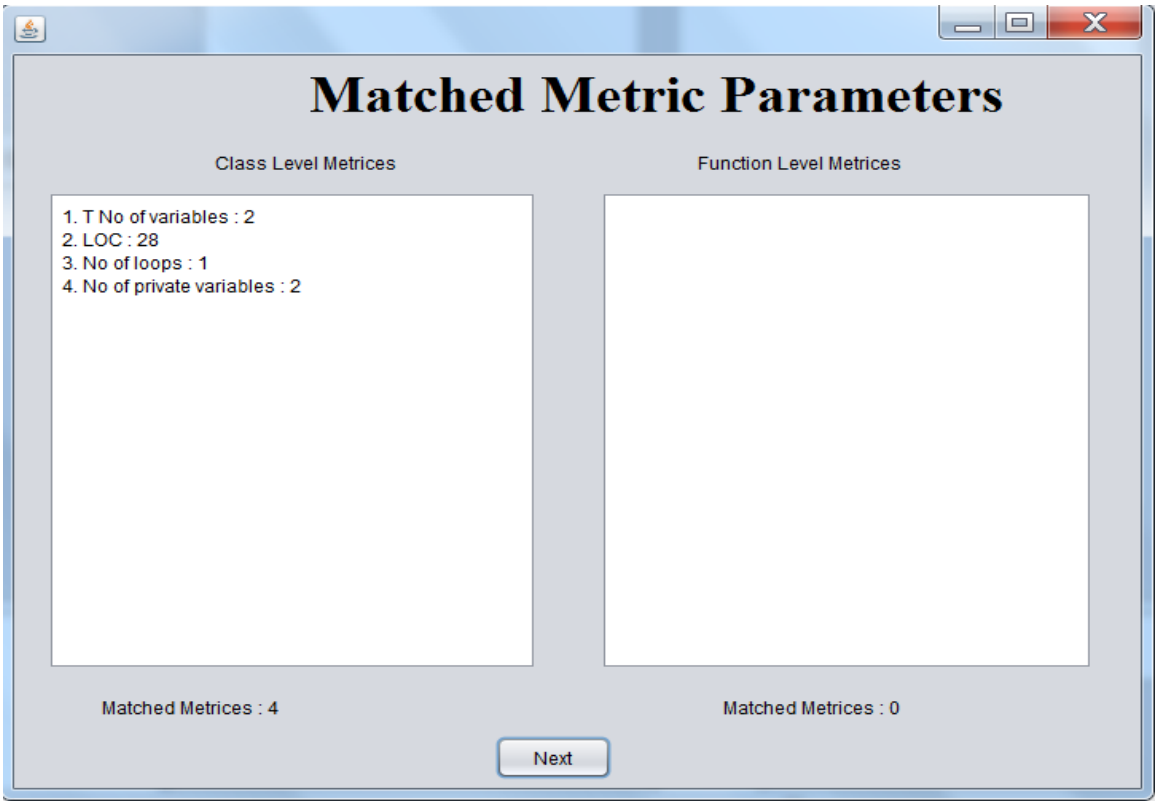


Figure 5.30 :Matched Parameters for Type1 clones.

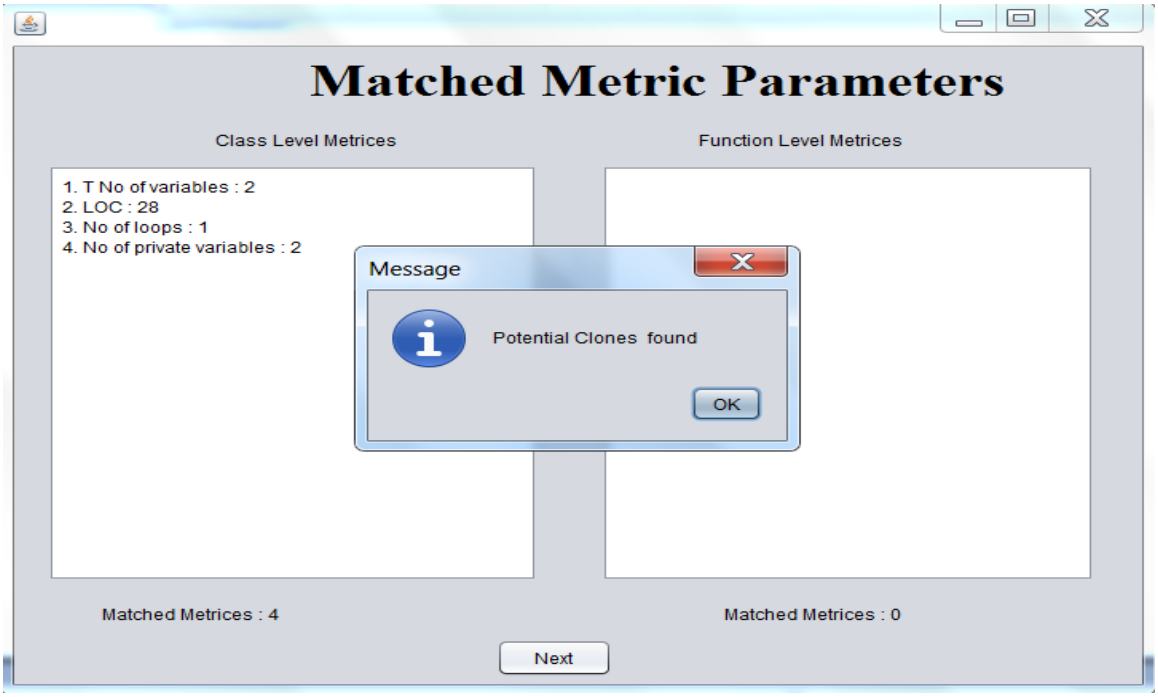


Figure 5.31:Potential clones

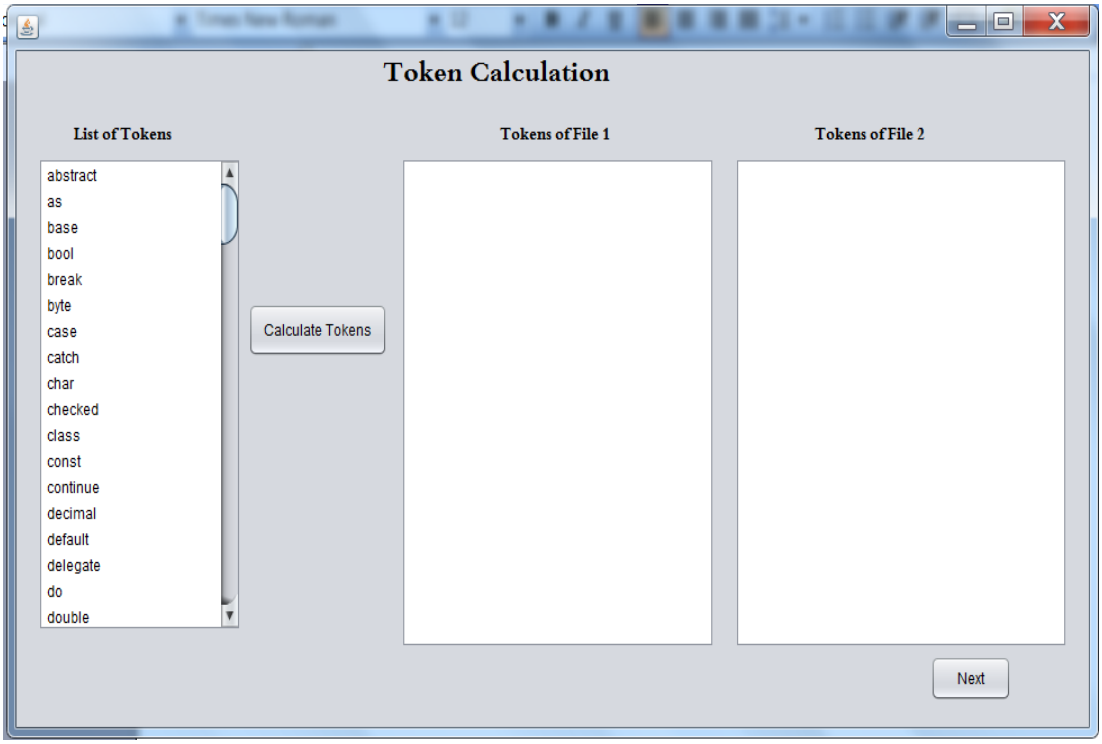


Figure 5.32: Token calculator window

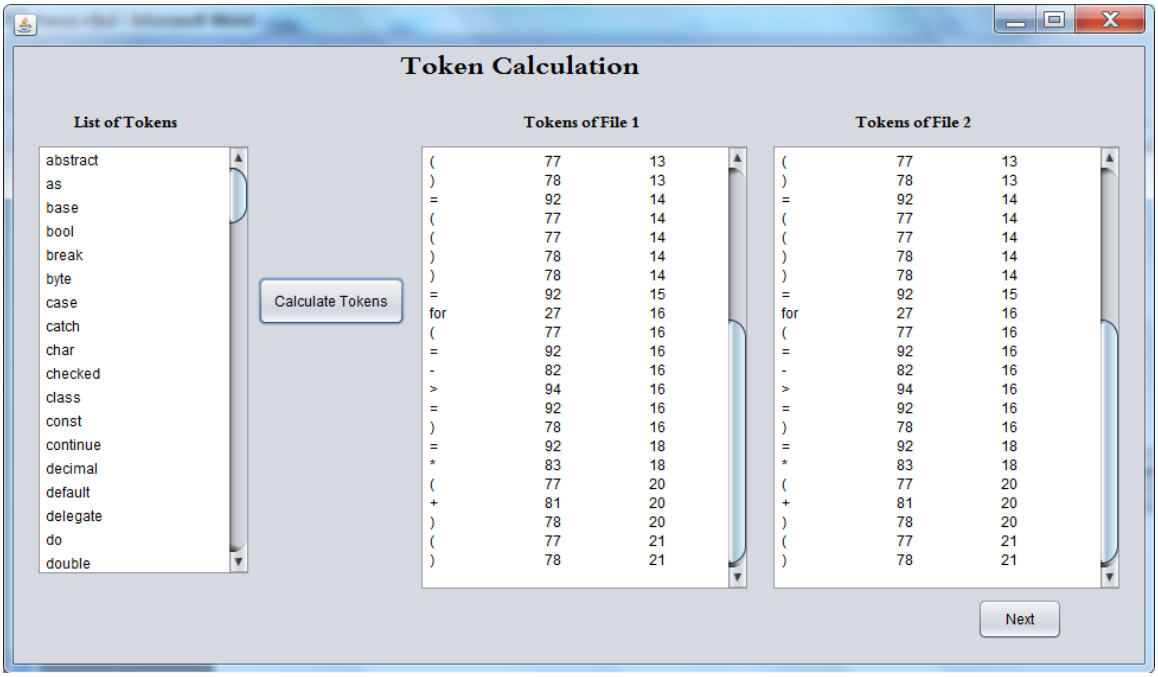


Figure 5.33: Token calculation

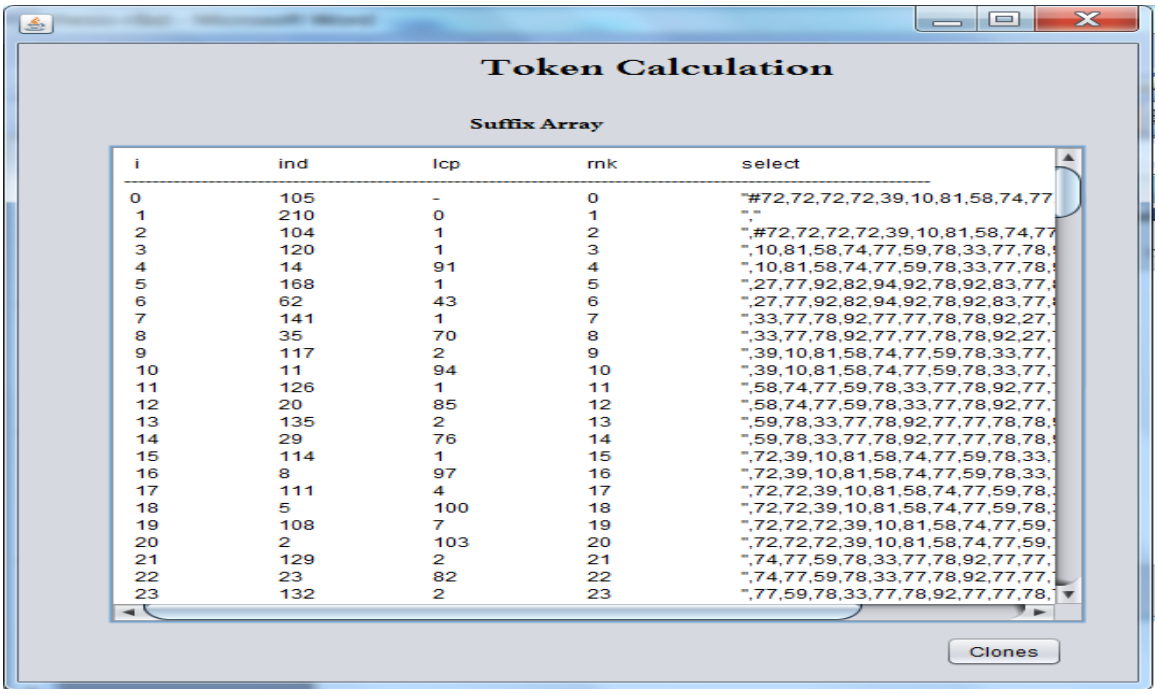


Figure 5.34: Suffix array execution

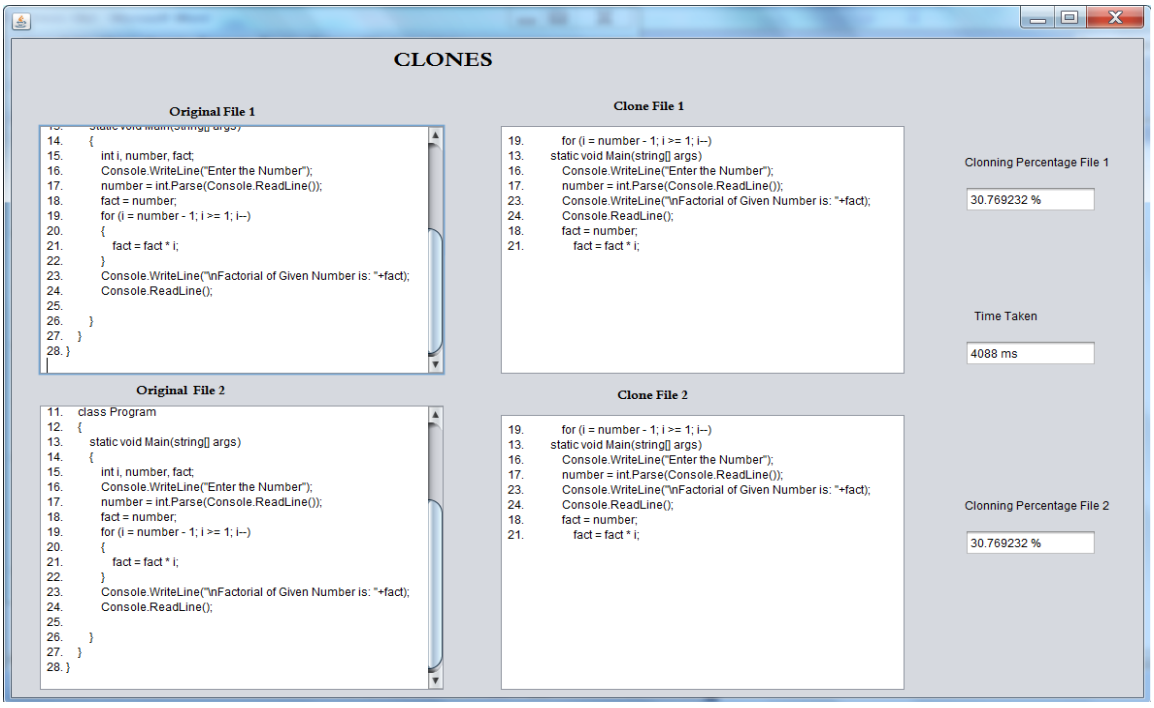


Figure 5.35 :Type1 clones detected for Asp.net

5.1.5 FOR ASP.NET LANGUAGE (TYPE2 CLONES)

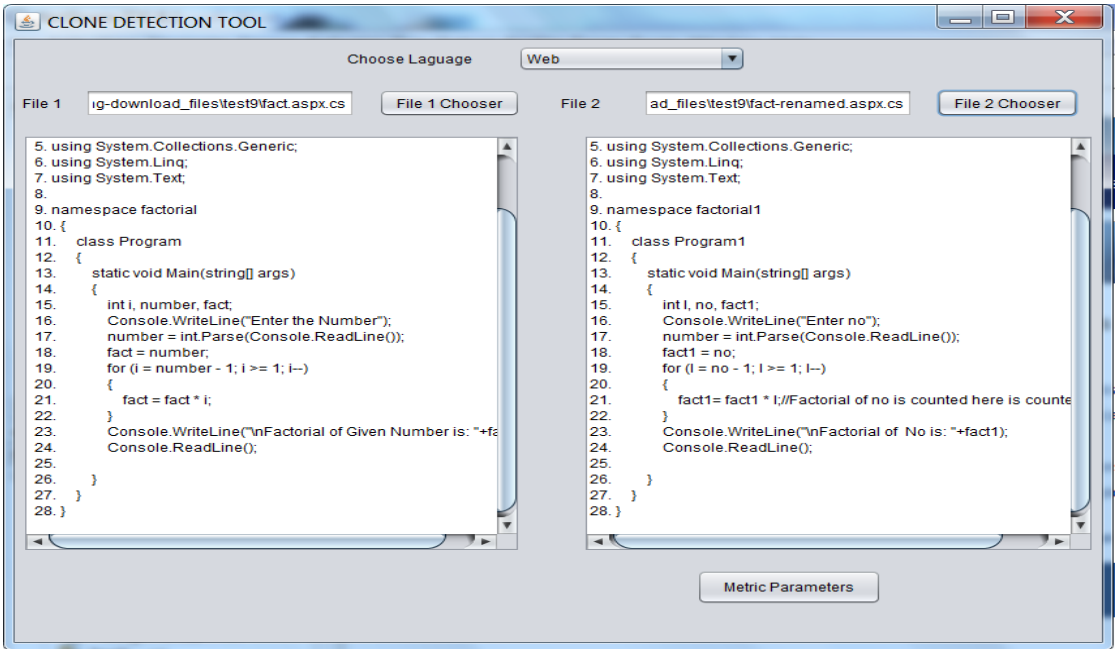


Figure 5.36 : Input File chooser

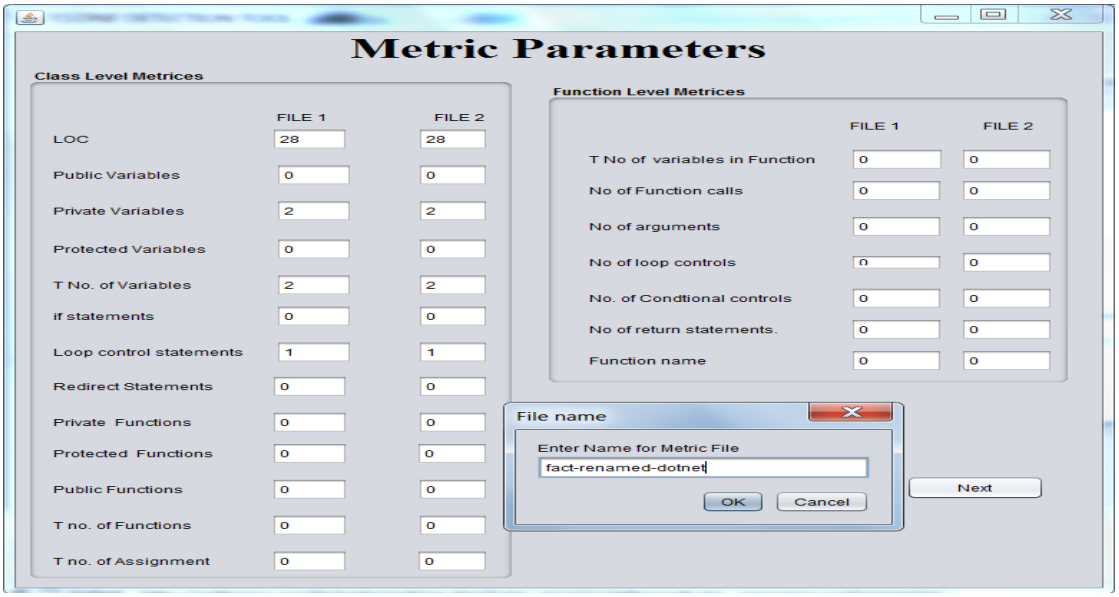
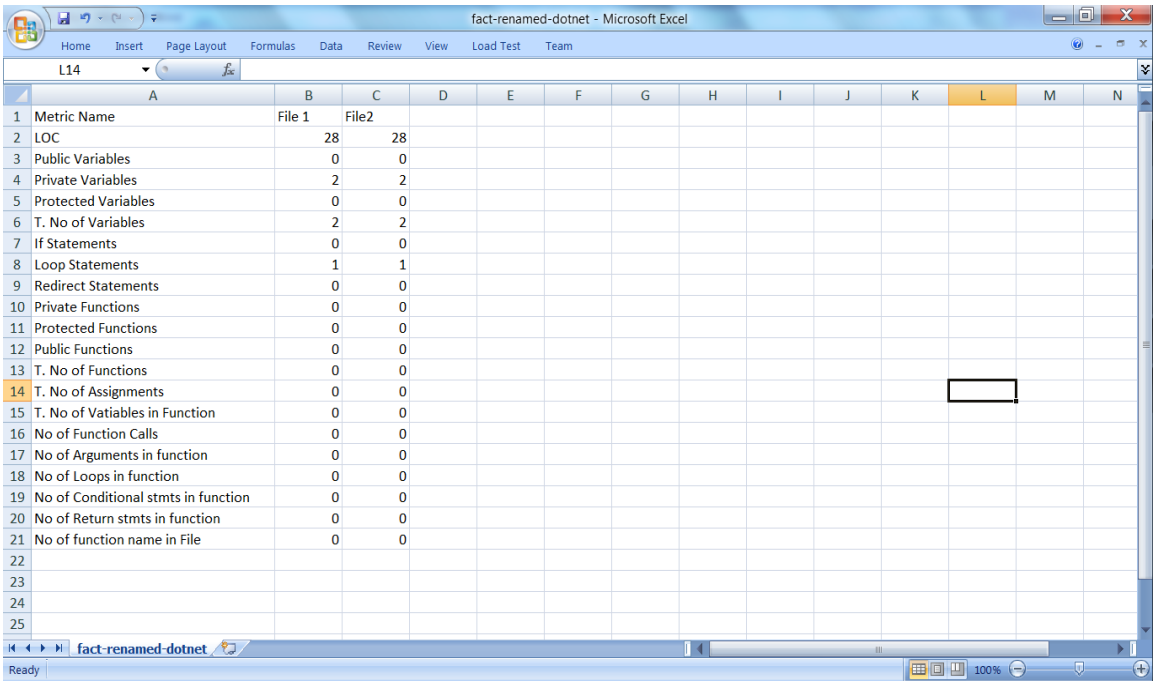


Figure 5.37:Metric calculation for Asp.net-Type2 clones.



	A	B	C
1	Metric Name	File 1	File2
2	LOC	28	28
3	Public Variables	0	0
4	Private Variables	2	2
5	Protected Variables	0	0
6	T. No of Variables	2	2
7	If Statements	0	0
8	Loop Statements	1	1
9	Redirect Statements	0	0
10	Private Functions	0	0
11	Protected Functions	0	0
12	Public Functions	0	0
13	T. No of Functions	0	0
14	T. No of Assignments	0	0
15	T. No of Vatiables in Function	0	0
16	No of Function Calls	0	0
17	No of Arguments in function	0	0
18	No of Loops in function	0	0
19	No of Conditional stmts in function	0	0
20	No of Return stmts in function	0	0
21	No of function name in File	0	0

Figure 5.38 :Metrics stored in excel

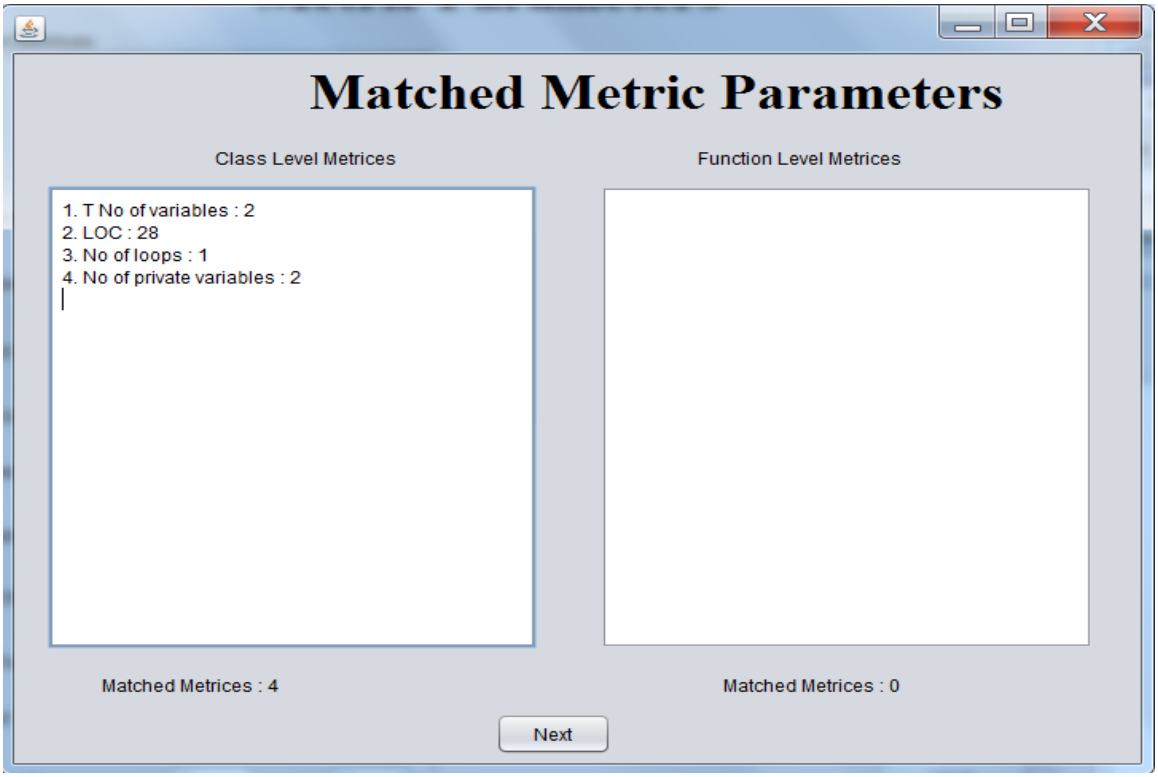


Figure 5.39 :Matched parameters forType2

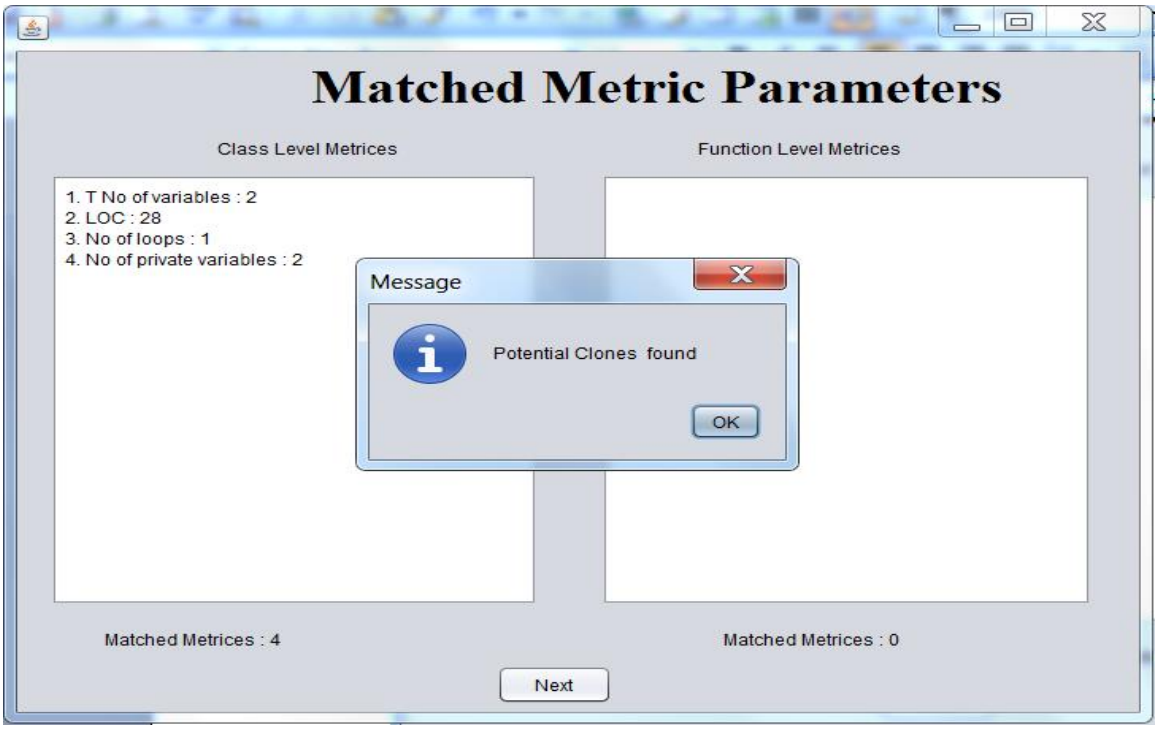


Figure 5.40:Potential clones

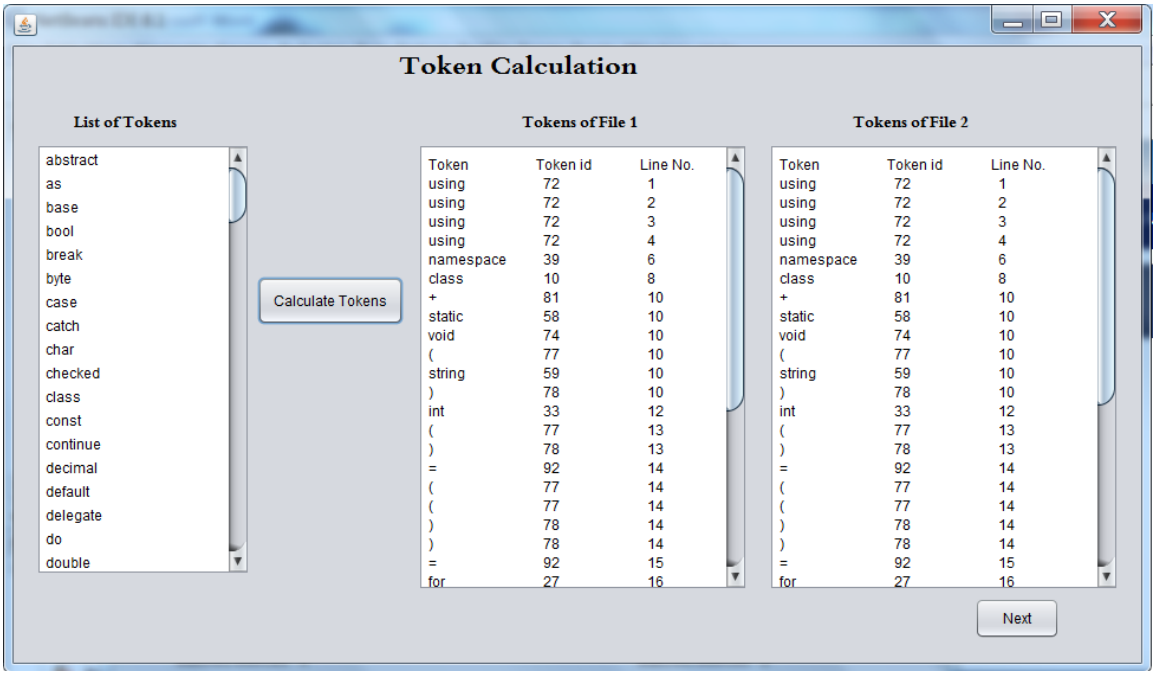


Figure 5.41:Token calculation

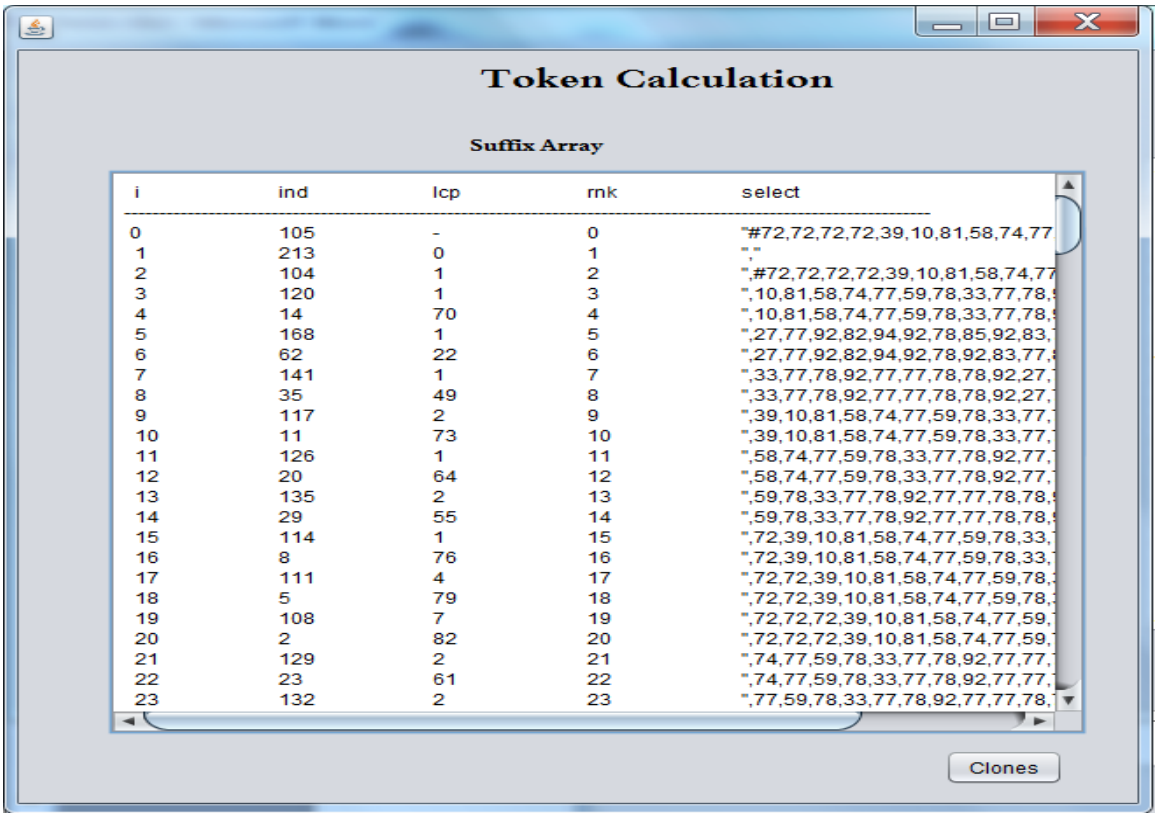


Figure 5.42: Suffix array execution

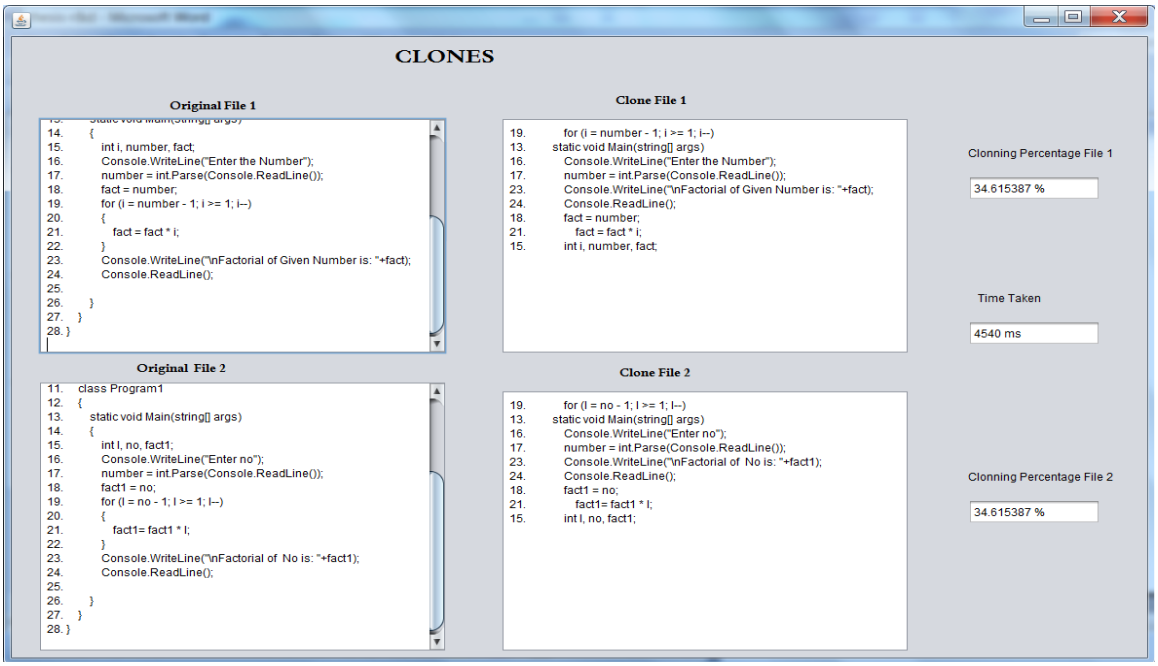


Figure 5.43 :Type2 clones detected forAsp.net

5.1.6 FOR ASP.NET LANGUAGE (TYPE3 CLONES)

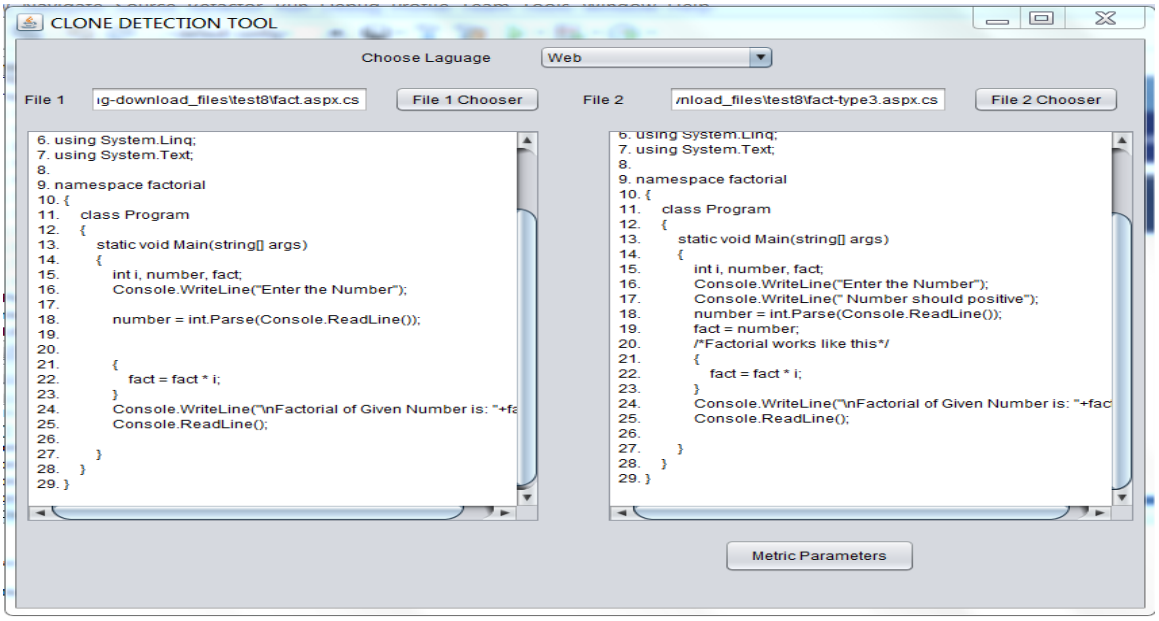


Figure 5.44 : Input File chooser

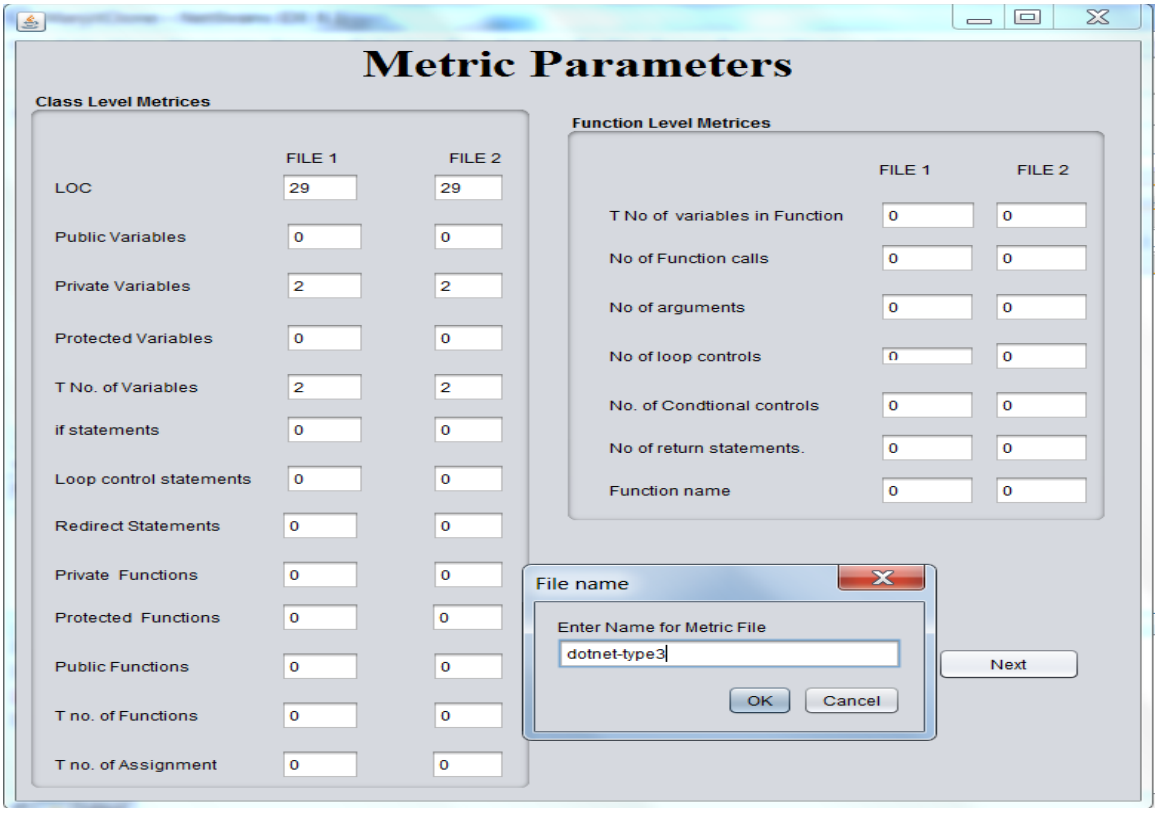
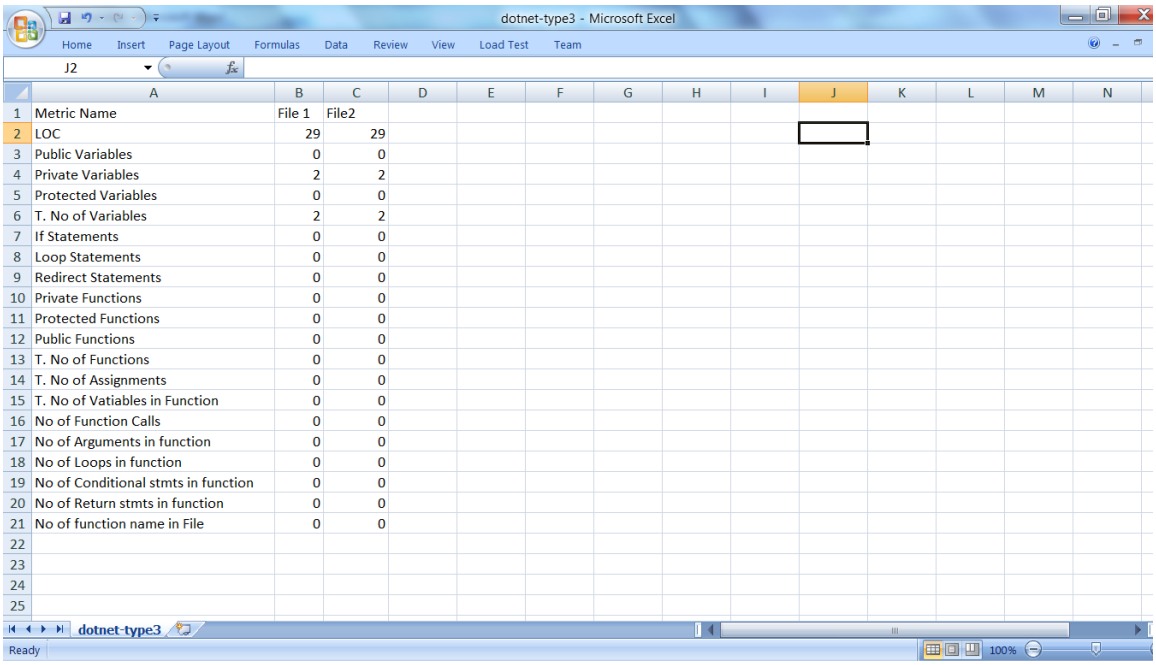


Figure 5.45 :Metrics calculation for Asp.net-Type3 clones



	A	B	C
1	Metric Name	File 1	File2
2	LOC	29	29
3	Public Variables	0	0
4	Private Variables	2	2
5	Protected Variables	0	0
6	T. No of Variables	2	2
7	If Statements	0	0
8	Loop Statements	0	0
9	Redirect Statements	0	0
10	Private Functions	0	0
11	Protected Functions	0	0
12	Public Functions	0	0
13	T. No of Functions	0	0
14	T. No of Assignments	0	0
15	T. No of Vatiables in Function	0	0
16	No of Function Calls	0	0
17	No of Arguments in function	0	0
18	No of Loops in function	0	0
19	No of Conditional stmts in function	0	0
20	No of Return stmts in function	0	0
21	No of function name in File	0	0

Figure 5.46 :Metrics stored in excel

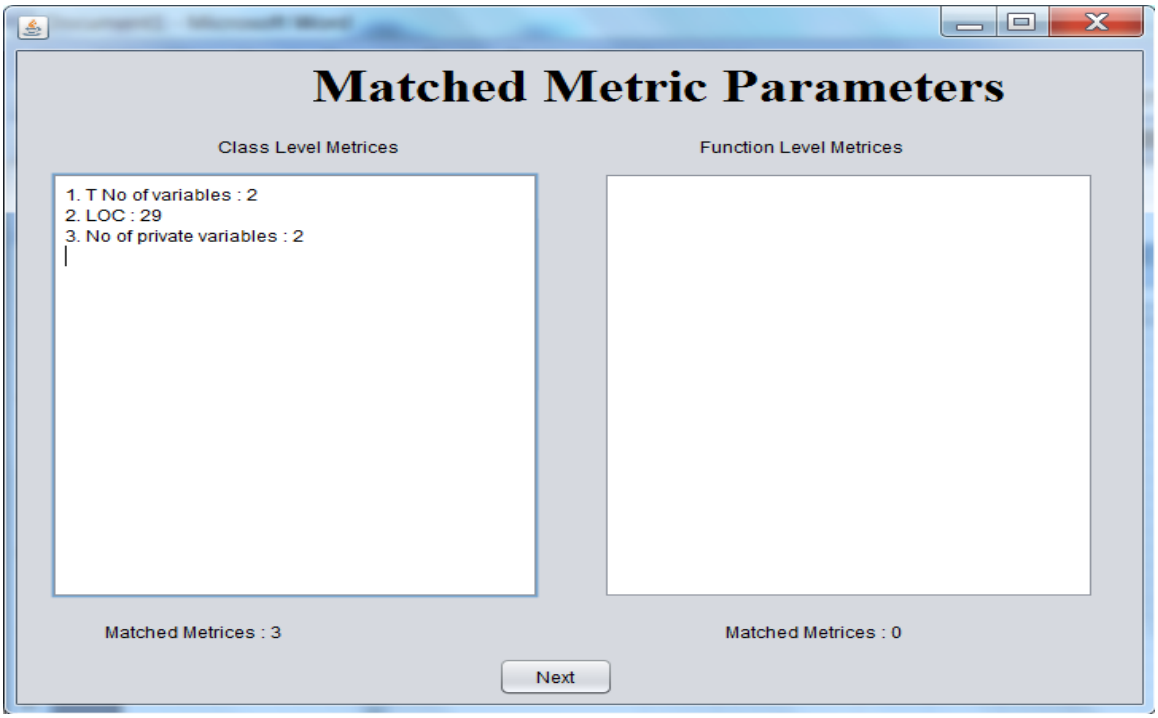


Figure 5.47 :Matched parameters for Type3 clones.

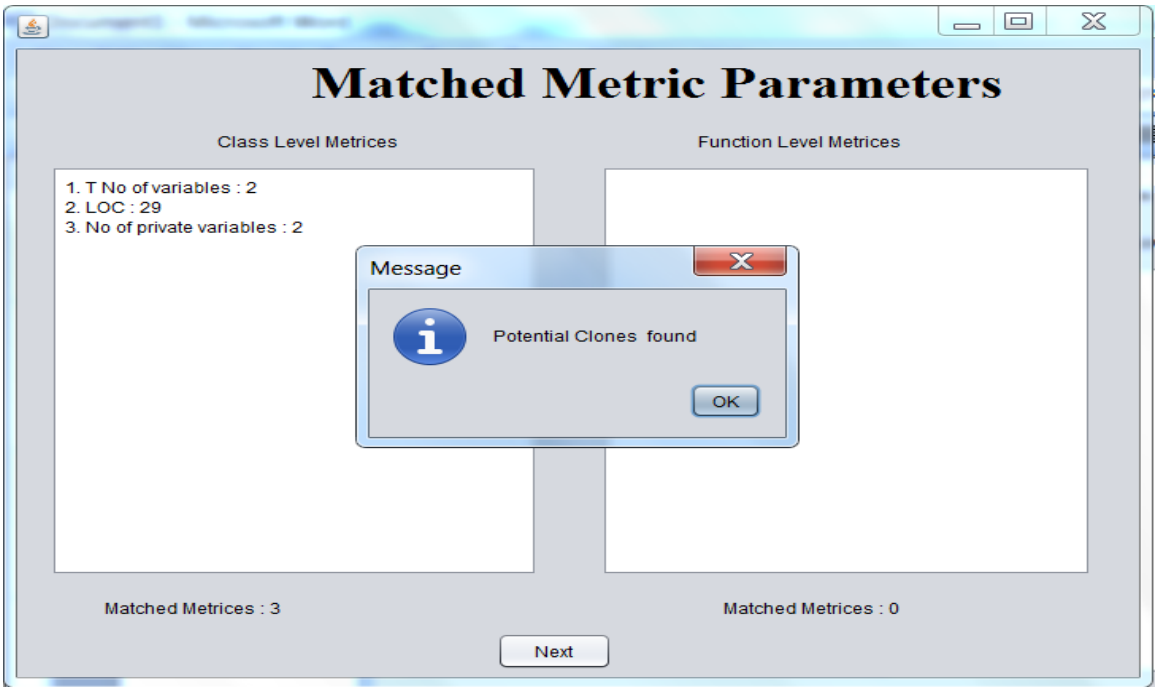


Figure 5.48 :Potential clones.

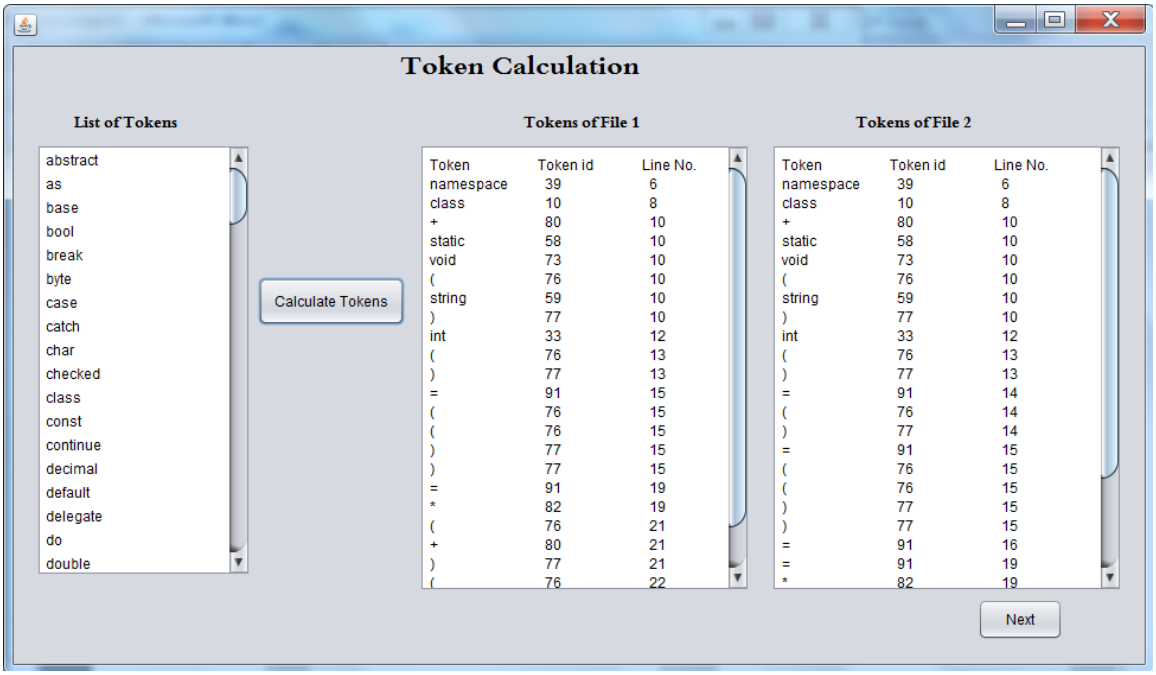


Figure 5.49: Token calculation

Figure5.49 displays the Tokenization of each character of source files by assigning Token ID's.

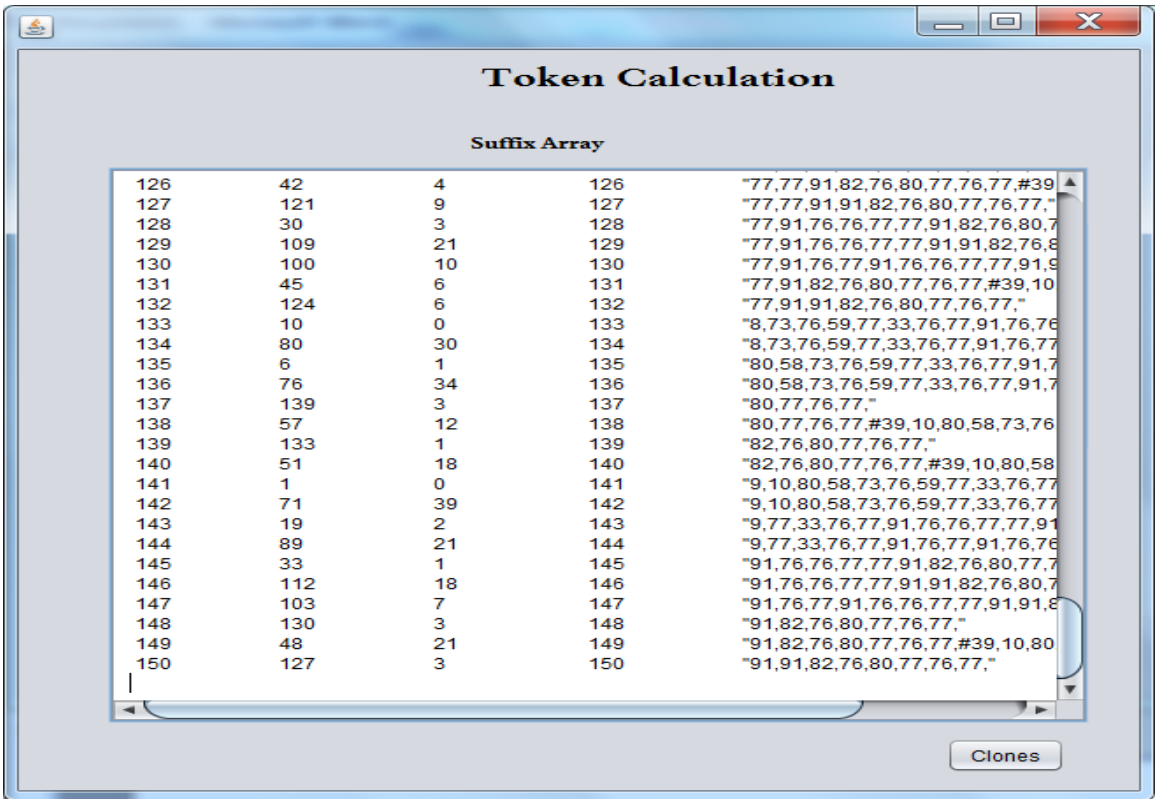


Figure 5.50:Suffix array execution

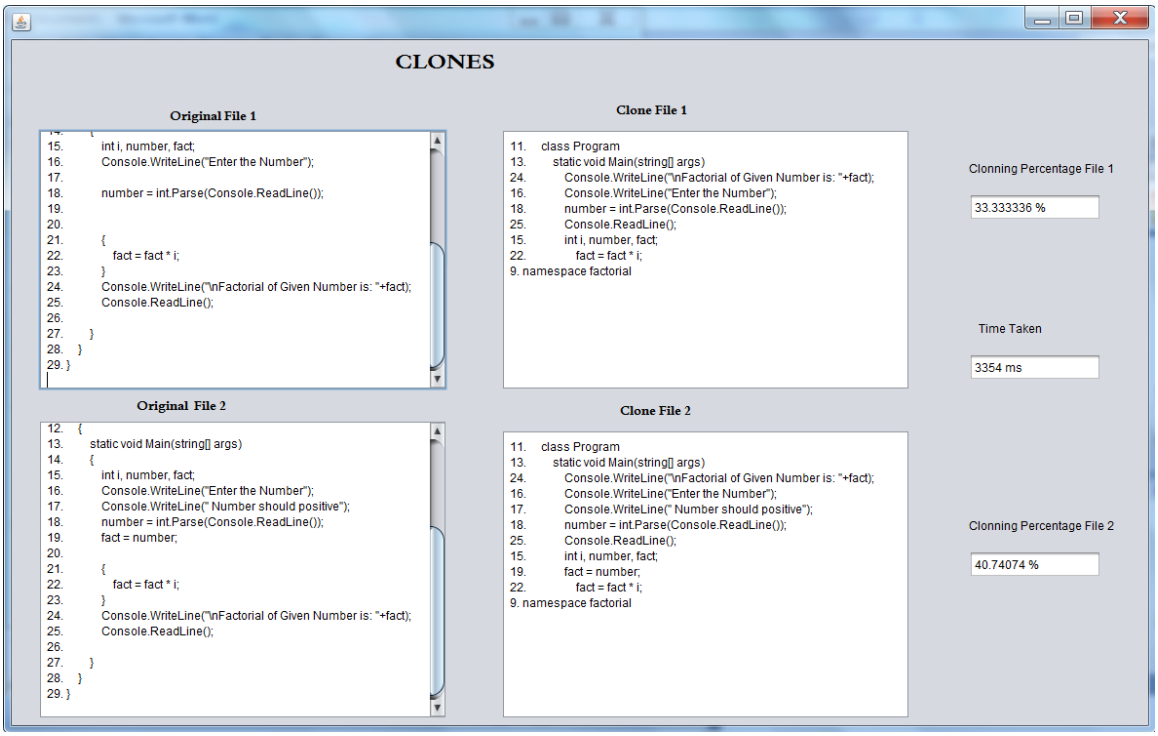


Figure 5.51 :Type3 clones detected for Asp.net

5.2 RESULTS & DISCUSSIONS

5.2.1 RESULTS BY METRIC BASED APPROACH

Table 5.1: Class level metrics for tested programs

	a	b	c	d	e	f	G	h	i	J	k	L	M
Fact-for.java	13	0	1	0	1	0	1	0	0	0	0	0	1
Fact-exact.java	13	0	1	0	1	0	1	0	0	0	0	0	1
Fact-renamed.java	13	0	1	0	1	0	1	0	0	0	0	0	1
Fact-type3.java	16	0	1	0	1	0	1	0	0	0	0	0	1
Fact.aspx.cs	28	0	2	0	2	0	1	0	0	0	0	0	0
Fact-exact.aspx.cs	28	0	2	0	2	0	1	0	0	0	0	0	0
Fact-renamed.aspx.cs	28	0	2	0	2	0	1	0	0	0	0	0	0
Fact-type3.aspx.cs	29	0	2	0	2	0	0	0	0	0	0	0	0
BinarysearchcharArray.java	54	0	0	0	0	3	5	0	0	0	0	0	0
BinarysearchByteArray.java	54	0	0	0	0	3	5	0	0	0	0	0	0
CRC32-ExtractZip.java	90	0	2	0	2	0	1	0	0	0	0	0	1
Deflater-compressarray.java	113	0	1	0	1	0	3	0	0	0	0	0	2
Adler32.java	36	0	2	0	2	0	1	0	0	0	0	0	2
Insertion.java	30	0	2	0	2	0	5	0	0	0	0	0	3
Selection.java	32	0	2	0	2	1	6	0	0	0	0	0	2
Insertion_sort.aspx.cs	48	0	3	0	3	1	5	0	0	0	0	0	2
Selection_sort.aspx.cs	42	0	3	0	3	1	6	0	0	0	0	0	1
Prime-renamed.aspx.cs	35	0	1	0	1	3	2	0	0	0	0	0	0
Exam-mgt.java	59	0	2	0	2	1	4	0	0	0	2	2	0
Exam-mgt-modified.java	76	0	2	0	2	1	4	0	0	0	2	2	0

a- LOC
b- Public variable
c- Private Variable
d- Protected Variable
e- Total no of variables
f- If statements
g- Loop control statements
h- Redirect statements
i- Private functions
j- Protected functions
k- Public functions
l- Total no of functions
m- Total no of Assignment

Table 5.2: Function level Metrics for tested program

	n	**o`**	**p**	**q**	**r**	**S**	**T**
Fact-for.java	0	0	0	0	0	0	0
Fact-exact.java	0	0	0	0	0	0	0
Fact-renamed.java	0	0	0	0	0	0	0
Fact.aspx.cs	0	0	0	0	0	0	0
Fact-exact.aspx.cs	0	0	0	0	0	0	0
Fact-renamed.aspx.cs	0	0	0	0	0	0	0
BinarysearchcharArray.java	0	0	0	0	0	0	0
BinarysearchByteArray.java	0	0	0	0	0	0	0
CRC32-ExtractZip.java	0	0	0	0	0	0	0
Deflater-compressbytearray.java	0	0	0	0	0	0	0
Adler32.java	0	0	0	0	0	0	0
Insertion.java	0	0	0	0	0	0	0

Selection.java	0	0	0	0	0	0	0
Insertion_sort.aspx.cs	0	0	0	0	0	0	0
Selection_sort.aspx.cs	0	0	0	0	0	0	0
Prime.aspx.cs	0	0	0	0	0	0	0
Prime-renamed.aspx.cs	0	0	0	0	0	0	0
Exam-mgt.java	0	6	12	3	0	0	0
Exam-mgt-modified.java	0	3	12	3	0	0	0

n- Total no of variables in function
o- No of Function calls
p- No of arguments
q- No of loop controls
r- No of conditional controls
s- No of return statements
t- Function name

5.2.2 RESULTS BY TOKEN BASED APPROACH

Table 5.3: Results byToken based Approach

Sno	File1	File2	Cloning % in file1	Cloning % in file 2	Time Taken (in ms)
1	Fact-for.java	Fact-exact.java	69.23	69.23	4134
2	Fact-for.java	Fact-renamed.java	84.61	76.92	6365
3	Fact.aspx.cs	Fact-exact.aspx.cs	30.76	30.76	2949
4	Fact.aspx.cs	Fact-renamed.aspx.cs	34.61	34.61	3884
5	BinarysearchcharArray.java	BinarysearchByteArray.java	29.03	32.25	1223
6	Insertion_sort.aspx.cs	Selection_sort.aspx.cs	34.78	47.5	6208
7	Exam-mgt.java	Exam-mgt-mdified.java	71.18	64.47	40997
8	Prime.aspx.cs	Prime-renamed.aspx.cs	41.75	43.75	8050
9	Fact-for.aspx.cs	Fact-type3.aspx.cs	33.33	40.74	3354
10	Fact-for.java	Fact-type3.java	76.92	81.25	8143

5.2.3 EXISTING TECHNIQUE vs PROPOSED TECHNIQUE

Table 5.4: Existing technique vs proposed enhanced technique

Parameters	**Existing Technique**	**Proposed Technique**
Language	Java	Java + Asp.net
Size of Metrics taken	14	20
Tool used in Token Approach	CC-Finder	Tool created in NetBeans 8.1 using Java language.
Data Structure used in Token Approach Tool	Suffix Tree	Suffix array
Memory Space Utilization	Suffix Tree consumes more memory. Hence less Memory efficient.	Suffix array is more memory or space efficient. (Consumes 5 times less memory than that of suffix tree)[1]
Time complexity	O(n) [1]	O(n) [1]
Type of clones detected	Type1 and Type2	Type1,Type2 and Type3
Output	Require extra preprocessing step to form clone pair and clone classes.[6][14]	Directly form clone pairs based on the output.[6][14]

CHAPTER 6

CONCLUSION & FUTURE SCOPE

It can be concluded that cloning is in great demand today apart from its various shortcomings. It has proven to be an advantageous process in fast development of the Software systems to meet the deadlines or to complete the work on time, etc. It is considered as a great boon to industries. Also, on the other side various tools and techniques have been proposed to detect the clones, wherever required, to overcome the various pitfalls released by cloning like bug propagation, maintenance costs, etc. Further detection process comprises of various levels such as preprocessing, transformation, match detection and so on. These techniques and tools can detect various types of clones according to their efficiency and ability. [25]

The proposed enhanced technique is a hybrid technique which is the combination of Metric based and token based technique. This technique is able to detect clones in multiple languages instead of just one. This technique can detect clones in object oriented language (Java) and web oriented language (Asp.net). This technique is efficient in detecting Type1, Type2 and Type3 clones. Furthermore, this is a fast and precise technique.

For future work, this technique will be further enhanced to detect Type4 clones. It can also be enhanced in such a way that it can detect clones for languages such as C++, C, PHP, etc. Clone removal techniques can also be added with this technique in order to further enhance it.

REFERENCES

[1] G. Y. Munina Yusufu, "Efficient Algorithm for Extracting Complete Repeats from Biological Sequences," *International Journal of Computer Applications,* vol. 128, pp. 33-37, 2015.

[2] S. K. Abd-El-Hafiz, "A Metrics-Based Data Mining Approach for Software Clone Detection," *36th International Conference on Computer Software and Applications,* pp. 35-41, 2012.

[3] J. K. Abdullah Sheneamer, "A Survey of Software Clone Detection Techniques," *International Journal of Computer Applications,* 2016.

[4] S. K. Y. Akshat Agrawal, "A Hybrid-Token and Textual Based Approach to Find Similar Code Segments," in *4th ICCCNT*, Tiruchengode, 2013.

[5] Y. L. Aritra Ghosh, "An Empirical Study of a Hybrid Code Clone Detection Approach on Java Byte Code," *GSTF Journal on Computing (JoC) Vol.5 No.2,* pp. 34-45, 2017.

[6] H. A. Basit, "Efficient Token Based Clone Detection with Flexible Tokenization," in *ACM*, 2004.

[7] P. Bhatta, "Hybrid Technique for Software Code CloneDetection," *International Journal of Computers and Technology,* pp. 97-102, 2012.

[8] J. R. C. K. Chanchal K Roy, "Comparison and evaluation of clone detection techniques and tools," *ELSEVIER,* pp. 470-495, 2009.

[9] Deepali, "Hybrid approach for Detecting Code Clone by Metric and Token based comparison," *International Journal of Advanced Research in Computer Science,* pp. 297-302, 2016.

[10] R. B. M. S. Dhavleesh Rattan, "Software clone detection: A systematic review," *ELSEVIER,* pp. 1165-1199, 2013.

[11] A. F. A. Doaa M. Shawky, "An Approach for Assessing Similarity Metrics Used in Metric-based Clone," 2010.

[12] S. K. Egambaram Kodhai, "Method-level code clone detection through LWH (Light Weight Hybrid) approach," *Journal of Software Engineering Research,* 2014.

[13] Geetika, "Detection of Potential Clones from Software using Metrics," *International Journal of Advanced Research in Computer Science and Software Engineering,* pp. 964-

969, 2014.

[14] S. J. P. F. S. T. J. Hamid Abdul Basit, “Efficient Token Based Clone Detection with FlexibleTokenization.,” in *ACM*, 2004.

[15] S. J. Hamid Abdul Basit, “A Data Mining Approach for Detecting Higher-Level Clones in Software,” *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,*, vol. 35, pp. 497-514, 2009.

[16] M. K. Harjot Kaur, “Detecting Clones in Class Diagrams Using Suffix Array,” *International Journal of Engineering and Advanced Technology (IJEAT)*, pp. 243-246, 2014.

[17] Himanshu, “Combined Weighted Approach to Detect Code Cloning,” *International Journal of Computer Science and Mobile Computing*, pp. 297-302, 2014.

[18] K. H. Y. H. H. I. a. S. K. Hiroaki Murakami, “Folding Repeated Instructions for Improving Token-based Code Clone Detection,” *2010 IEEE 12th International Working Conference on Source Code Analysis and Manipulation*, pp. 64-73, 2012.

[19] K. P. Jai Bhagwan, “Design and Analysis of a Hybrid Technique for Code Clone Detection,” *International Journal of Advanced Research in Computer and Communication Engineering*, pp. 380-385, 2016.

[20] C. L. M. M. Jean Mayrand, “Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics,” pp. 244-253, 1996.

[21] L. Jiang, “DECKARD: Scalable and Accurate Tree-based Detection of Code Clones∗”.

[22] N. S. D. R. K. Vidhya, “Cross Language Higher Level Clone Detection-Between Two Different Object Oriented Programming Language Source Codes,” in *ICIDRET*, 2014.

[23] R. K. T. Kanika Raheja, “An Efficient Code Clone Detection Model on Java Byte Code Using Hybrid Approach,” Patiala, 2013.

[24] R. T. Kanika Raheja, “An Emerging Approach towards Code Clone Detection:Metric Based Approach on Byte Code,” *International Journal of Advanced Research in Computer Science and Software Engineering*, pp. 881-888, 2013.

[25] M. Kaur, “Review on Software Cloning and Clone Detection,” in *Interenational Conference on Intelligent Circuits and Systems(ICICS 2016)*, Phagwara, 2016.

[26] T. N. S. u. D. A. J. Khurram Zeeshan Haider, “Efficient Source Code Plagiarism Identification Based on GreedyString Tilling,” *International Journal of Computer Science*

and Network , pp. 204-210, 2010.

[27] P. Kodhai.E, "Clone Detection using Texual and Metric Analysis to figure out all Types of clones," in *Proceedings of the International Joint Jounal Conference on Engineering and Technology*, 2010.

[28] M. L. Manpreet Kaur, "Code Clone Detection Using Function Based Similarities and Metrics," *International Journal of Emerging Research in Management &Technology,* pp. 156-159, 2015.

[29] E. Merlo, *Detection of Plagiarism in University Projects Using Metrics-based Spectral Similarity,* Canada, 2007.

[30] M. D. Muneer Ahmad, "A Novel Approach for Code Clone DetectionUsing Hybrid Technique," *International Journal of Advanced Engineering, Management and Science (IJAEMS),* pp. 1408-1411, 2016.

[31] T. T. Nguyen, "ClemanX:Incremental Clone Detection tool for evolving Software," in *IEEE* , Vancouver, 2009.

[32] L. P. Z. F. l. M. a. D. S. L. Qing Qing Shi, "A Novel Detection Approach for Statement Clones," *IEEE,* pp. 27-30, 2013.

[33] P. S. Rajnish Kumar, "Token based clone detection using program slicing," *Int.J.Computer Technology & Applications,* pp. 1537-1541, 2014.

[34] R. E. Roxas, "Automation generation of Plagiarism Detection among students programs," in *IEEE Transactions on Software Engineering*, 2006.

[35] H. K. K. Rupinder kaur, "Evaluation of Token Based Tools on the basis of Clone Metrics," *International Journal of Advanced Research in Computer Science and Electronics Engineering,* pp. 145-150, 2012.

[36] D. S.Mythili, "A Language Independent Approach for Method Level Clone Detection Using Fingerprinting," *International Journal of Advanced Research in Computer Science*, 2012.

[37] K. K. Saif Ur Rehman, "An Efficient New Multi-Language Clone Detection Approach from Large Source Code," in *International Conference on Systems, Man, and Cybernetics*, Korea, 2012.

[38] Y. Sharma, "Hybrid Technique for object oriented software clone detection," 2011.

[39] S. Sonika, "HCDETECTOR: A HYBRID APPROACH TO DETECT CODE CLONES IN

JAVA PROGRAMS," Patiala, 2014.

[40] R. K. A. K. Stefan Bellon, "Comparison and Evaluation of Clone Detection Tools," in *IEEE Transcations on Software Engineering*, 2007.

[41] S. S. Sukhpreet kaur, "CODE CLONE DETECTION AND ANALYSIS USING SOFTWARE DESIGN OBJECT ORIENTED METRICS," *International Journal of Computer Science and Communication Engineering,* vol. 4, no. 3, 2015.

[42] J. B. Sushma, "A Novel Metrics Based Technique for Code Clone Detection," *International Journal Of Engineering And Computer Science ,* pp. 18221-18224, 2016.

[43] S. K. I. Toshihiro Kamiya, "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,* vol. 28, pp. 654-670, 2002.

[44] Y. G. Yang Yuan, "Boreas: An Accurate and Scalable Token-Based Approach to Code Clone Detection," *ACM,* pp. 286-289, 2012.

[45] M. Younas, "Evaluating Clone Detection Technique in Multiple Language Programs," *International Journal of Science and Advanced Technology,* pp. 107-112, 2011.

[46] J. S. Zhu o LI, "A Metric Space Based Software Clone Detection Approach," 2010.

PAPER PUBLICATION

[1] M. Kaur, "Review on Software Cloning and Clone Detection," in *Interenational Conference on Intelligent Circuits and Systems(ICICS 2016)*, Phagwara, 2016.

Fact.java

```
package b;
import java.io.*;
class Factorial{
public static void main(String[] args) {
try{
BufferedReader object = new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter the number");
int a= Integer.parseInt(object.readLine());
int fact= 1;
System.out.println("Factorial of " +a+ ":");
for (int i= 1; i<=a; i++){
fact=fact*i;
}
```

Fact-exact.java

```
package b;
import java.io.*;
class Factorial{
public static void main(String[] args) {
try{
BufferedReader object = new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter the number");
```

```
int a= Integer.parseInt(object.readLine());
int fact= 1;      //          Initialize value of factorial with1
System.out.println("Factorial of " +a+ ":");
for (int i= 1; i<=a; i++){
fact=fact*i;                //Factorial function
}
```

Fact-renamed.java

```
package b;
import java.io.*;
class Factorial1{
public static void main(String[] args) {
try{
BufferedReader object = new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter no");
int h= Integer.parseInt(object.readLine());
int factorial= 1;
System.out.println("Factorial of " +h+ ":");
for (int j= 1; j<=h; j++){
factorial=factorial*i;
}
```

Fact-type3.java

```
package b;
import java.io.*;
class Factorial{
```

```
public static void main(String[] args) {
try{
BufferedReader object = new BufferedReader(new InputStreamReader(System.in));
System.out.println("enter the number");
System.out.println("Number should be positive ");
int a= Integer.parseInt(object.readLine());
int fact= 1;
System.out.println("Result is here ");
System.out.println("Factorial of " +a+ ":");
for (int i= 1; i<=a; i++){
fact=fact*i;
System.out.println( fact);
}
```

fact.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace factorial
{
   class Program
   {
      static void Main(string[] args)
```

```
        {
            int i, number, fact;
            Console.WriteLine("Enter the Number");

            number = int.Parse(Console.ReadLine());

            {
                fact = fact * i;
            }
            Console.WriteLine("\nFactorial of Given Number is: "+fact);
            Console.ReadLine();

        }
    }
}
```

Fact-exact.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace factorial
{
    class Program
```

```
    {
        static void Main(string[] args)
        {
            int i, number, fact;
            Console.WriteLine("Enter the Number");
            number = int.Parse(Console.ReadLine());
            fact = number;
            for (i = number - 1; i >= 1; i--)
            {
                fact = fact * i;//Factorial of no is counted here is counted here
            }
            Console.WriteLine("\nFactorial of Given Number is: "+fact);
            Console.ReadLine();

        }
    }
}
```

Fact-renamed.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace factorial1
```

```
{
   class Program1
   {
      static void Main(string[] args)
      {
         int l, no, fact1;
         Console.WriteLine("Enter no");
         number = int.Parse(Console.ReadLine());
         fact1 = no;
         for (l = no - 1; l >= 1; l--)
         {
            fact1= fact1 * l;//Factorial of no is counted here is counted here
         }
         Console.WriteLine("\nFactorial of  No is: "+fact1);
         Console.ReadLine();

      }
   }
}
```

Fact-type3.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace factorial
{
   class Program
   {
      static void Main(string[] args)
      {
         int i, number, fact;
         Console.WriteLine("Enter the Number");
         Console.WriteLine(" Number should positive");
         number = int.Parse(Console.ReadLine());
         fact = number;
         /*Factorial works like this*/
         {
            fact = fact * i;
         }
         Console.WriteLine("\nFactorial of Given Number is: "+fact);
         Console.ReadLine();

      }
   }
}
```