

CERTIFICATION OF APPROVAL

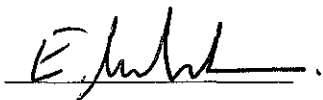
'Plagiarism Detection Application'

by

Ntabane Mamphofore Jan
7181

A Project Dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION COMMUNICATION TECHNOLOGY)

Approved by,



(Dr Etienne Schneider)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
July 2007

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

Student Name : *NTABANE MAMPHOFORE JAN*

Student ID : *7181*

Date : *12th November, 2007*

Abstract

The purpose of this abstract is to give a complete idea of the project being undertaken. Part 1 of this project deals mainly with the research value of the project. The project name is plagiarism Detection system or application. Methodologies and procedure of reaching this goal are included in the report as well. The motive behind the development of this application is to diminish plagiarism in University Technology Petronas (UTP). After reading this report a clear understanding of scope of study should be fully comprehended. The scope of study is researching about functionality of components needed to be integrated to complete a full plagiarism detection system. Behaviors of these components will also be studied to enable the student to make modification were possible to suit UTP while making it easy to use.

Acknowledgement

I would like to extend my deepest gratitude to my Supervisor Dr Etienne Schneider for his understanding, constant support and encouragement. He made this endeavor interesting and eye-opening for me. His patience is greatly admired.

Also I am very grateful to one of the important characters who had my best interest at heart to make sure I never doubt my potential is Mr. Dicky Ekklesia . Thank you for giving me your time, valuable opinions and advises.

I would like to express my gratitude to my roommate, Mr. Ernest Danke for bearing with my late night work and his support and constructive criticism, for without him as the user for my system improvement wouldn't have materialized.

In the mist of all my achievements I would like to extend my deepest gratitude to my source of inspiration, my family. I am very grateful to my cornerstone ms Dinah Ntabane. Although we are oceans apart their love and support are the reason I wake up everyday to do what I can do best.

Last but not least I would like to thank Prof. Jubert for his approach and advice on the interface design.

Table of Contents

Certification of Approval	i
Certification of Originality.....	ii
Abstract	ii
Acknowledgement.....	iv
Table of Contents	v-vi
List of Figures	vii
CHAPTER1: Introduction.....	1
1.1 Background of Study	1
1.2 Problem Statement	2
1.3 Objective of Study	2
1.4 Objective of the Application	3
1.5 Scope of Study	3
CHAPTER 2: Literature Review	5
2.1 What is Plagiarism	5
2.2 Levels of Plagiarism.....	6
2.3 Method Used to Detect Plagiarism	7
2.4 Algorithm Used in Plagiarism Detection.....	8
2.5 Existing Softwares	10
2.5.1 JPlag.....	10
2.5.2 My drop Box	11
2.5.3 Software Integrity Diagnosis System (SID).....	11
2.5.4 YAP3.....	11
2.5.5 SIM.....	12
2.5.6 MOSS.....	12
CHAPTER 3: Methodology Used in Study	13
3.1 Spiral methodology	13
3.2 Research methodology	14
3.2.1 Reading articles.....	14
3.2.2 Getting Information from Science Forums	14

3.3 Study Available Applications	16
3.3.1 WCopyfind 2.6.....	16
3.3.2 Cheat Guru	18
3.3.4 Asking Lecturers for clarification.....	19
3.4 Project Activities.....	19
3.4.1 Components Gathering.....	19
3.4.2 Code Analysis	19
3.4.3 Literature review	19
3.4.4 Interface Design	20
3.4.5 Action Inserted	20
3.4.6 Software Updates	20
3.4.7 The Progress of the Project's time line	21
3.5 Project Implementation	22
CHAPTER 4: Results and Discussion	24
4.1 Result	24
4.2 Discussion.....	26
CHAPTER 5: Conclusion and Recommendations.....	28
5.1 Conclusion	28
5.2 Recommendations.....	28
REFERENCES.....	29
APPENDICES	29-49

List of Figures

1. Figure 1:Four-Stages plagiarism detection model	6
2. Figure 2: The “Greedy String Tiling” algorithm [Wise, 1993].....	8
3. Figure 3: Part of a JPlag results display page for a pair of programs	9
4. Figure 4: A similar code segment detected by SID	10
5. Figure 5: Opening web page of MOSS results.	11
6. Figure 6: depiction of waterfall Methodology	12
7. Figure 7: plagiarism detection Progress timeline.....	28
8. Figure 8: WCopyfind 2.6 detection systems	16
9. Figure 9: Guru Cheat detection software.	17
10. Figure 10: The interface for the application.....	20
11. Figure 11: Plagiarism detection Timeline Second Part.....	29
12. Figure 12: Process Flow of Plagiarism detection applications	23
13. Figure 13 : User Interface with user inputs.....	24
14. Figure 14: Plagiarism Detection Application’s output.....	24
15. Figure 15: Help window for the output.....	26

List of Tables

Table 1.1: Plagiarism spectrum of program.....	6
--	---

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

Plagiarism detection application is an application intended to deter student from plagiarism activities. To agree not to do something you have to know what that thing is so that you don't do it. To make sure people know what it is not to be done this editorial will also encompasses the definition of plagiarism and methods of checking plagiarism. In most university students get away with plagiarism all the time which won't do them good later in life. Beside the fact that it is a crime to do plagiarism, it also unethical and not forgetting mind deteriorating. This lead to lot professionals who are really not qualified but they have the document to prove that they are worthy to be called postgraduate. 66% of 16,000 students from 31 prestigious U.S. universities have cheated at least once, says 1991 Rutgers University study. 36% of undergraduates have admitted to plagiarizing written material, says 1997 Psychological Record study. Cheating on campus increased an estimated 744% from 1993 to 1997, says University of California-Berkley officials . If the rate of plagiarism is quite high in those universities, one can imagine the rate in UTP. Amongst the sectors to be discussed the following will be discussed as listed:

1. Identify Problem
 - Objective
 - Scope of study
2. Literature review
 - ↓ What is Plagiarism
 - ↓ Method used to Detect Plagiarism
 - ↓ Algorithms Used for plagiarism Detection
 - ↓ Existing Softwares
3. Methodology Used in study
4. conclusion
5. Reference

1.2 Problem Statement

The topic itself gives one the idea of why the system has to be developed without reading the problem statement, nevertheless as a request, I will explain in detail the problem that arise due to plagiarism in University Technology Petronas. The mission and vision of UTP involves producing students who are innovative and creative. This statement got compromised when students start copying other student's work diminishing originality which means upon graduating 50% of their knowledge is not true or fake. Lectures do not encourage this but they don't have a mechanism to see where some assignments are being replicated. As such when they get student's work from there they can evaluate how many students did understood, erroneously not taking into consideration plagiarism. This system is to aid lectures to spot the work that are repeating and can have students who don't understand to ask for help

1.3 Objectives of study

↓ Establish the requirements of a detection application

The application has the requirements it must meet to be regarded as a detection application. As there are other application already in use, aligning the requirements of this application to comply with what other system already in use would be a good practice.

↓ Determine the standards of a detection application

For a detection application to be regarded as the best, it has to have the standard that best detection application have. Research what are the functionalities that the application must have to reach the set standards.

↓ Analyze the already in use applications

Analyze the application in use based on what is the common algorithm to be used and what are the difference between them .Analyzing the pros and cons of the available algorithms. Do a research about the vulnerability of each application.\

1.3.1 Objectives of the Application

↓ Deter plagiarism in UTP

Knowing that your assignment is going to be fully scrutinized, it is unlikely to commit plagiarism act. Most people commit and/ or brake rules if they know for sure that they won't get caught. I know that UTP does not have hard core criminals it is just students desperate to make it to the industrial world.

↓ Assist Lectures in evaluating

The application will assist lectures in evaluating the level of understanding of the whole class by the result they get from the test and assignments. Lectures will easily be able to spot the students who needs help and take necessary step.

↓ Encourage originality practice

Reinforcement is one of the techniques used to ensure that people get use to what they are supposed to do. Upon the implementation of this application in UTP student will have to get use to submitting their original work even when the application is no longer in use they will still know that there is still a way that they can be analyzed should their work have any suspiciousness of plagiarism.

1.4 Scope of Study

The application is design to cater for student doing programming in UTP, meaning it will be searching for comparison between programming assignments. Plagiarism detection is a very complex process and trying to start writing programs and codes for the whole application will take three to four years based on the research that was done. The scope of study is to study components used in the plagiarism detection system. Factors about the system's components are their behavior, history and functionality then gather them from the internet finally integrate them. Integration of the components

might spawn lot of errors which will consume time but with lot of research supervisor's guide it is doable.

Creating packages was one of the skills learn during Java programming, after all the packages have been gathered a user interface will be designed to suit UTP's standards and culture.

CHAPTER TWO

LITERATURE REVIEW

2.1 What is Plagiarism?

Plagiarism is defined by the Cornell University Honor Council as "the acts of passing off as one's own the ideas or writings of another." In the Appendix to the Honor Council pamphlet called "Acknowledging the Work of Others" (which is used by permission of Cornell University), three simple conventions are presented for when you must provide a reference[1]

1. If you use someone else's ideas, you should cite the source.
2. If the way in which you are using the source is unclear, make it clear.
3. If you received specific help from someone in writing the paper, acknowledge it.

Examples of Plagiarism.[1]

↓ word-for-word plagiarism

When material is taken directly from a book, article, speech, statement, remarks, the Internet, or some other source, the writer must provide proper attribution. In this example, no credit is given to the author.

↓ The footnote without quotation marks

↓ The paraphrase

Even if the author's exact language is not used, a footnote is required for material that is paraphrased.

↓ The mosaic: Having the original idea or passage but with copied material woven throughout the passage.

↓ The "apt phrase"

Taking one phrase from the author and use it as your own.

Definition from Programming perspective.

A program plagiarism can be defined as a program which has been produced from another program by copying or by using a small number of source code modifications. [13] Plagiarized programs are usually produced in a short time without understanding

than level 4 in a plagiarism spectrum [13]. Most students tend to use levels from 0-2 which are totally unacceptable.

2.3 Method used to detect plagiarism

Automated Methods of Plagiarism Detection

There are three different methods of plagiarism detection.

✚ Quiz methods

Another method of detecting plagiarism is quizzing students about their written work. A student who has produced their own paper should be familiar with its contents and should

be able to answer questions about it.[2]

✚ Writing style methods

Many faculty members detect plagiarism by observing writing styles. Sometimes a paper seems to be too professionally written to have been prepared by a student. Another clue is a sudden shift in writing styles. Ironically, the Copy/Paste process that makes plagiarism so easy also betrays the crime because students forget to reformat the text into a uniform font.[2]

✚ Comparison with original sources [2]

Chunking Methods

Metrics based plagiarism monitoring

A Plagiarism Detection Using a Syntax-Tree

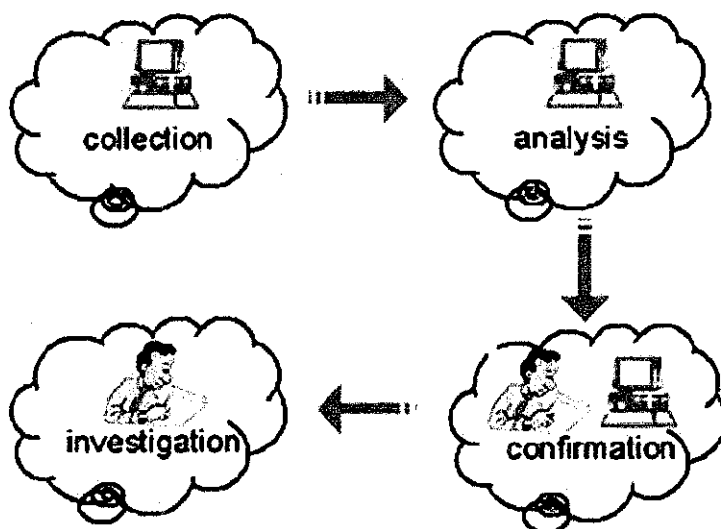


Figure 1 Four –Stage plagiarism Detection Model

2.4 Algorithms Used in Plagiarism Detection

The Running Karp-Rabin Matching and Greedy String Tiling (RKR-GST)

Algorithm 1 Compare a File Against an Existing Collection[4]

```

1 p = 1 // the first token of Q
2 WHILE p · q i ° + 1
3   find Q[p::p + ° i 1] from the suffix array
4   IF Q[p::p + ° i 1] was found
5     UpdateRepository
6     p = p + °
7   ELSE
8     p = p + 1
9   FOR EVERY file Fi in the collection
10    Similarity(Q; Fi) = MatchedT okens(Fi)=q

```

Algorithm 2 Update the Repository [4]

```
1 Let S be the set of matches of  $Q[p:::p + \circ ; 1]$ 
2 IF some of the strings in S are found in the same file /* collision of type 1 */
3   leave only the longest one
4 FOR every string M from the remaining list S
5   IF M doesn't intersect with any repository element
6     insert M to the repository
7   ELSE IF M is longer than any conflicting rep. element /* collision of type 2 */
8     remove all conflicting repository elements
9     insert M to the repository
```

```
0 Greedy-String-Tiling(String A, String B) {
1   tiles = {};
2   do {
3     maxmatch = M;
4     matches = {};
5     Forall unmarked tokens  $A_a$  in A {
6       Forall unmarked tokens  $B_b$  in B {
7         j = 0;
8         while ( $A_{a+j} == B_{b+j}$  &&
9             unmarked( $A_{a+j}$ ) && unmarked( $B_{b+j}$ ))
10          j ++;
11         if ( $j == maxmatch$ )
12           matches = matches  $\oplus$  match(a, b, j);
13         else if ( $j > maxmatch$ ) {
14           matches = {match(a, b, j)};
15           maxmatch = j;
16         }
17       }
18     }
19     Forall match(a, b, maxmatch)  $\in$  matches {
20       For  $j = 0 \dots (maxmatch - 1)$  {
21         mark( $A_{a+j}$ );
22         mark( $B_{b+j}$ );
23       }
24       tiles = tiles  $\cup$  match(a, b, maxmatch);
25     }
26   } while (maxmatch > M);
27   return tiles;
28 }
```

Figure 2: The “Greedy String Tiling” algorithm [Wise, 1993].

2.5 Existing Softwares

The recommended solution to be used in understanding the result produced by each software, one needs to comprehend the concept of their Algorithm.

2.5.1. JPlag

JPlag is a java application that was developed for plagiarism detection(see figure 2 below)

How it works

1. All program source codes to be compared are parsed (or scanned, depending on the input language) and converted into token strings.
2. These token strings are compared in pairs for determining the similarity of each pair. The method used is basically “Greedy String Tiling” [Wise, 1993]: During each such comparison, JPlag attempts to cover one token string with substrings (“tiles”) taken from the other file as well as possible. The percentage of the token strings that can be covered is the similarity value. The corresponding tiles are visualized in the HTML pages.[5]

Matches for 132207 & 792145
93%
INDEX - HELP

132207 (93%)	792145 (93%)	Tokens
Jumpbox.java(39-177)	Jumpbox.java(9-154)	149
Jumpbox.java(184-214)	Jumpbox.java(166-196)	27
Jumpbox.java(216-349)	Jumpbox.java(200-327)	108
Jumpbox.java(345-354)	Jumpbox.java(337-352)	12
Jumpbox.java(391-448)	Jumpbox.java(374-428)	49

```
public void paint (Graphics g) {
    // System.err.println("paint()");
    // Use update() to display the offscreen buffer.
    update(g);
}

/**
 * Update canvas
 */
void updateCanvas ()
{
    offDimension = dim;
    offImage = createImage(dim.width, dim.height);
    offGraphics = offImage.getGraphics();
    offGraphics.setBackground(Color.white);
    offGraphics.fillRect(0, 0, dim.width, dim.height);
    offGraphics.setColor(Color.black);
    offGraphics.drawRect(0, 0, dim.width, dim.height);
    offGraphics.drawRect(0, 0, dim.width, 0);
    drawImage();
}

/**
 * Repaint canvas if it was modified
 */
synchronized public void update (Graphics g) {
    // System.err.println("update()");
    Dimension dim = getSize();
    // Is the offscreen buffer still valid?
    if ( offGraphics == null)
        || (dim.width != offDimension.width)
        || (dim.height != offDimension.height) ) {
        // Repaint it
        updateCanvas ();
    }
    // copy the offscreen buffer into the game area
    g.drawImage(offImage, 0, 0, this);
}

/**
 * Handle mouse drags.
 */
public void mouseDragged(MouseEvent e) {
    mouseMoved(e);
}
```

Figure 3: Part of a JPlag results display page for a pair of program

2.5.2. My Drop Box

My Drop Box is a family of products for e-earning that includes the world's leading plagiarism prevention service and a comprehensive Course management Toolset.

2.5.3 SID: A Software Integrity Diagnosis System

It detects similarity between programs by computing the shared information between them. It was originally an algorithm developed for comparing how similar or dissimilar genomes are [15]. It was then realized that this algorithm could be extended to many other applications including finding chain letter history and detecting plagiarism.

```
cheng
class LStack implements Stack { // LStack
    private Link top; // Point to top element

    public LStack() { setup(); } // Create empty stack
    public LStack(int sz) { setup(); } // Create stack of size sz

    private void setup() { top = null; } // Initialize stack

    public void clear() { top = null; } // Remove objects from stack

    public void push(Object it) // Push object onto stack
    { top = new Link(it, top); }

    public Object pop() // Pop object from stack
    { Assert.notNull(top, "Empty stack");
      Object o = top.element();
      top = top.next();
      return o;
    }

    public Object topValue() // Get top value
    { Assert.notNull(top, "Empty stack");
      return top.element();
    }

    public boolean isEmpty() // Return true if empty
    { return top == null; }
} // class LStack
/*
More.java
*/

Layde310
private void setup() {
    top = null;
    size = 0
} // Initialize stack

public void clear() {
    top = null;
    size = 0
} // Remove Objects from stack

public void push(Object it) // Push
{ top = new Link(it, top);
  size++;
}

public Object pop() // Pop
{ Assert.notNull(top, "Empty stack");
  Object o = top.element();
  top = top.next();
  size--;
  return o;
}

public Object topValue() // Get top value
{ Assert.notNull(top, "Empty stack");
  return top.element();
}

public boolean isEmpty() // Return true if empty
{ return top == null; }

public int getSize() {
    return size;
}
```

Figure 4: A similar code segment detected by SID.

2.5.4 YAP3:

Which is developed by M.J. Wise is a similarity evaluation system using structural matrix method. YAP is a system for detecting suspected plagiarism in computer programs and other texts submitted by students YAP3, the third version of YAP, focusing on its novel underlying algorithm - Running-Karp-Rabin Greedy-String-Tiling (or RKS-GST), whose development arose from the observation with YAP and other

systems that students shuffle independent code segments. YAP3 is able to detect transposed subsequences, and is less tempered by phony additional statements. The paper concludes with a discussion of recent extension of YAP to English texts, further illustrating the flexibility of the YAP approach.[16]

2.5.5 SIM.

The SIM plagiarism detection system compares token sequences using a dynamic programming string alignment technique [6]. The SIM plagiarism detection system [1] converts the source programs into token strings, then compares the strings using dynamic programming string alignment techniques like those used in DNA string matching [17].

2.5.6 MOSS:

MOSS stands for "Measure Of Software Similarity." It is a system developed in 1994 by Alex Aiken, associate professor of computer science at UC Berkeley. [8]

Moss Results				
Sun Mar 14 15:24:02 PST 1999				
Options -l c -m 10				
[Text Report] [How to Read the Results] [Tips] [FAQ] [Contact Moss] [Submission Scripts] [Credits]				
File 1	File 2	Tokens Matched	Lines Matched	
mike_wolf.c (79%)	mike_fox.c (80%)	463	139	
bill_smyth.c (86%)	bill_smith.c (86%)	456	133	
jane_white.c (59%)	jane_blanco.c (68%)	354	111	
john_doe.c (100%)	john_deer.c (100%)	220	49	

Any errors encountered during this query are listed below.

Figure 5: Opening web page of MOSS results.

CHAPTER THREE

METHODOLOGIES USED

3.1 Spiral Methodology

The perfect methodology for this project would be Spiral Methodology since the student and managerial control.[10]

Benefits of Spiral model

- ↓ the methodology iterates over the processes of think a little, plan a little, implement a little, then test a little
- ↓ he spiral methodology is an incremental improvement on the waterfall methodology
- ↓ It allows for feedback to each team the complexity of each requirement.
- ↓ There are stages where mistakes in the requirements can be corrected.
- ↓ The end user gets a peek at the results and can feedback information
- ↓ It is easy to make changes.

Drawbacks of Spiral Model

- ↓ The spiral methodology has no governors to control oscillations.
- ↓ The length or number of cycles grows unbounded.
- ↓ There are no constraints on the requirement team to "get things right the first time"
- ↓ There are no firm deadlines
- ↓ Cycles continue with no clear termination condition.
- ↓ During implementation the developer may be chasing a continuously changing architecture and dynamic product requirements.

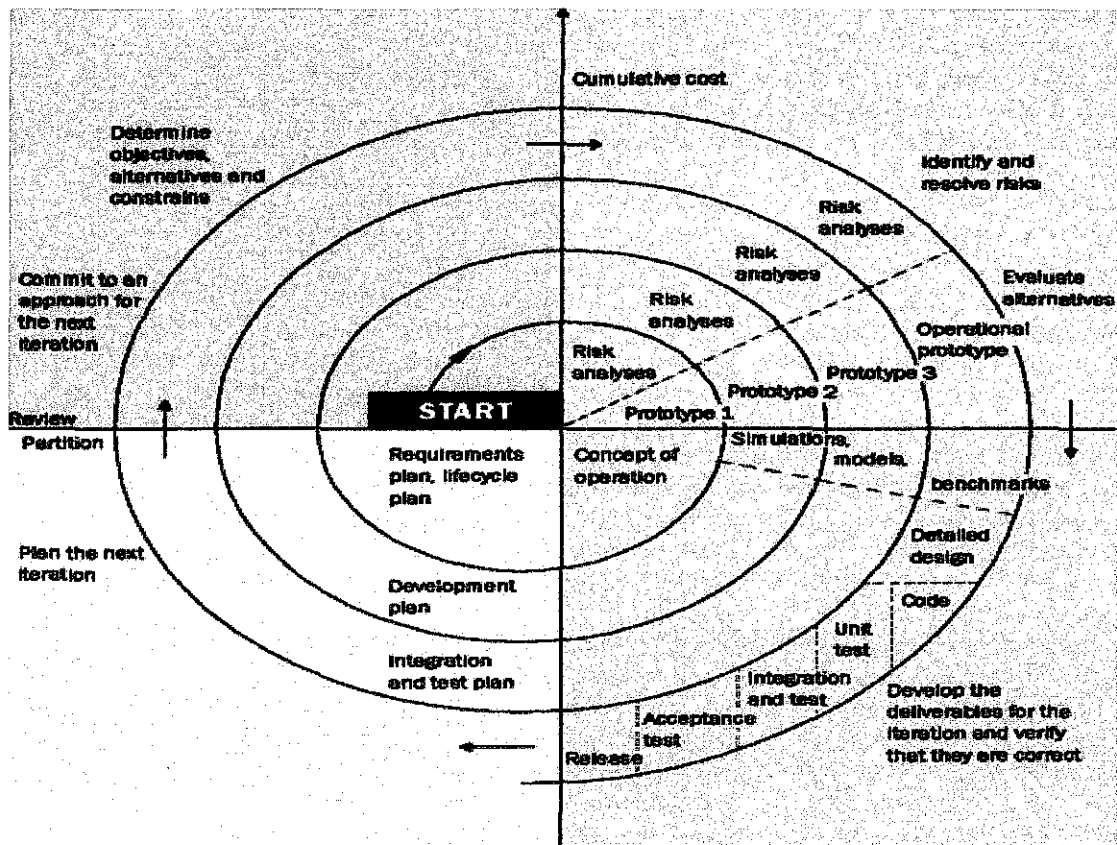


Figure 6: depiction of Spiral Methodology

3.2 Research Methodology

3.2.1 Reading articles

Reading more articles pertaining to plagiarism it is discovered that most online /internet systems consider the plagiarism detection application on a larger scale thus making it a full plagiarism detection system that covers all aspects of plagiarism. The main aim of studying the literature review was to derive a feature that makes this plagiarism detection application being developed unique and different. As stated in most of the articles they only provide high level of their architecture as a result keeping the user in total oblivion.

3.2.2 Getting information from Computer Science forums

One of the most effective medium when it comes to gathering information to contribute to the project reaching towards completion is posting your problem in science forums.

Rules from forums promote the idea that they are there to help with problem not to do you work. The forums is composed of various expert of programmers and they will require a clear idea of your problem and give you suggestion whilst they both constructively criticize each other's ideas giving the poster heads up on his/ her pit falls or vulnerability to his / her application. Listed below is a list of ideas collected from forums pertaining to this project:

- ✚ Most file compare algorithms use a line as a basic unit with the ability to ignore (or more normally to treat as a single space) any white space. Once any filtering of this form has been done they then look for an exact match. I don't think this Sort of comparison can really detect plagiarism.

Suggestion: If someone has plagiarized a bit of code but changed all the variable names and reformatted the code then how will you detect this? You will need to treat source code tokens as the basic comparison unit and then compare (fuzzy?) tokens.

This does not sound easy to me .

by Roger sun Developer Network

- ✚ My guess is that, by compiling the files, you can identify these changes by going a comparison score on the generated-bytecode. Using \$strings can get that for you, then compare the edit-distance (Levenshtein distance) on the bytecode.

All this gets you is a probability of cheating – it is up to the user to examine (□ctually look at files where) the scores where the distance is beneath some arbitrary threshold. Then you can adjust the threshold based on some 'training' (read – neural ntwk).

by rafeal

From this forum makes options to solution unlimited and helps the developer to think outside the box.

3.3 Studying available Applications

Most plagiarism detection system that compare bytes code are web based and users have to use the internet to access them or they need reasonable network connection. Standalone application in most articles and the ones being user tested after installation handles plagiarism from plain text only point of view which when it comes to byte codes they can easily be fooled. Amongst the softwares that are standalone and consider text format input are Cheat gurus, Wcopyfind 2.6 and Eve.

3.3.1 Wcopyfind 2.6

Wcopyfind2.6 is a stand alone software that consider plain text only and it has a lot of inputs from the user as a result it will confuse the user since vast number of result can be obtained although this sound like a good thing for analysis the lecturer might not have time to rigorously asses those outputs. Program examines a collection of document files. It extracts the text portions of those documents and looks through them for matching words in phrases of a specified minimum length. When it finds two files that share enough words in those phrases, Wcopyfind 2.6 generates html report files. These reports contain the document text with the matching phrases underlined.

- ✚ **What Wcopyfind 2.6 can do:** It can find documents that share large amounts of text. This result may indicate that one file is a copy or partial copy of the other, or that they are either copies or partial copies of a third document.
- ✚ **What Wcopyfind 2.6 cannot do:** It cannot search for text that was copied from any external source, unless you include that external source in the documents you give to Wcopyfind 2.6. It works on only purely local data—it cannot search the web or internet to find matching documents. If you suspect that a particular outside source has been copied, you must create a local document containing that outside material and include this document in the collection of documents that you give to Wcopyfind 2.6. (see figure 2)

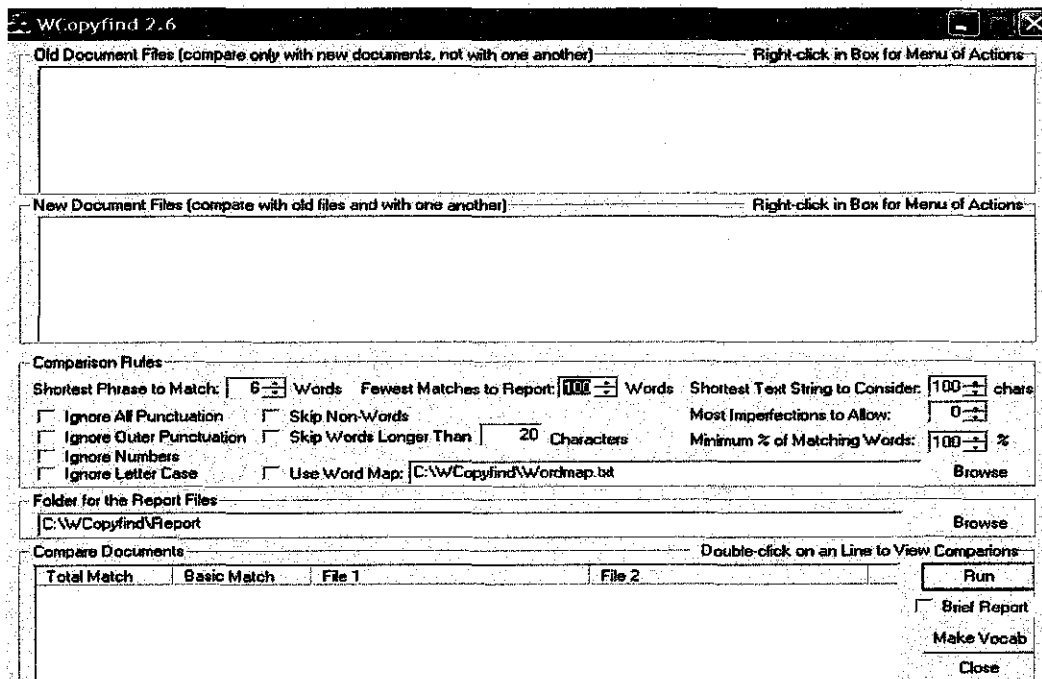


Figure 8 : Wcopyfind 2.6 detection systems

Steps of how Wcopyfind 2.6 is used.

1. Select a file you need to be detected
2. Select a folder where you saved all submitted files
3. Run the application
4. Get result in

Advantages

- ✚ This application is just a Stand alone application file , there is no need to install
- ✚ No need for a server
- ✚ Can generate a brief report upon request in form of html.

Disadvantages

- ✚ Detects plain text only
- ✚ Have many options which might lead to undesirable outputs
- ✚ Complicated to a novice user

3.3.2 Cheat Guru

Cheat Guru is using the specialized Plagiarism Detection software to prevent instances of plagiarism. Furthermore, they have developed the special client module and made this software accessible to their customers. Many companies claim to utilize the tools of such kind, few of them do and none of them offer their Plagiarism Detection software to their customers. They have created a server that does the comparison for their customers.



Figure 9: Guru Cheat detection software.

Advantages

- ✚ User friendly interface
- ✚ Straight forward
- ✚ Protect user from technical side.

Disadvantages

- ✚ If internet connection is lost the application hang there is no way to resume it
- ✚ You don't choose your own collection of files to run I against with.
- ✚ It uses materials from the net only not other submitted by students
- ✚ It only compares plain text.

6.1.4 Asking Lecturers for clarification

Seeking clarification from the lectures is one of the method one can acquire information fast whilst uncovering new leads to other unlimited information. The advantages of consulting lecturers is they share a bit of their experience and often give you word of encouragement when any is needed. The new professor from Pakistan has shared his views on interface design and the right methodology when it comes to interface design and what are the basic requirements of a good interface.

3.4 Project Activities

3.4.1 Component Gathering

The components collected are java since comparing to the other programming languages is friendly and has lot of resources.

- JDK1.6
- JWSD 2.0
- APACHE ANT
- DIFFJ

3.4.2 Code Analysis

Initially the packages and classes collected were meant to work towards creating a simple simulation of the system. It was later discovered that the system thought to be of the same purpose as the one planned to be developed has deviated a little bit from the project's goal. The code shows that the system depends on the second party which is located remotely. The ultimate goal of this project is to create an application that functions similar to the prior mentioned (Wcopyfind 2.6), the only discrepancy is that the application in this project will have better user interface and will compare bytes code instead of just plain text. All Source codes of Wcopyfind 2.6 have been collected even though not analyzed yet (Microsoft C++ workspace).

3.4.3 Literature Review

The literature review was redone to find a way to cater for the system requirements. The "Plaggie" is a java developed an application that was developed to detect plagiarism between java source codes. The available application at present cater for Linux operating

system meaning there is no complete packaged that was assembled for windows .The system however is compatible with windows as well it is just the mater of creating a MAKE file that caters for windows. Java language is a language that can operate across most operating system.

3.4.4 Interface Design

Although the functionalities of “Plaggie” are somehow close to the application being developed, Plaggie does not have an interface. It runs from the command line. The documentation of the application mentioned the reason behind this feature. The interface is meant to make the system easy to use whilst leaving the functionality of the application in tact. The Interface should encompass nonfunctional factors like UTP colors and logo to resemble the UTP culture. The diagram below(figure10) depicts the proposed interface for the system. Although only the basic functions are being looked at the interface will merge with the Plaggie functionalities when the MAKE file issue is being resolved.

3.4.5 Action inserted

The actions included in the interface are not core functionality buttons but the ones to help user navigate the application for example status bar and the help file. When the user clicks the browse button or any other button the status bar will display the function being performed by the user. The browse button is used to search for the file that needs to be scanned against the other files that are already submitted and saved in a folder. The source button is used to point to the location of the folder where all submitted files resides.

3.4.6 Software Updates

- ✚ Jbuilder 4.0
- ✚ Photoshop
- ✚ Diffj
- ✚ Net beans
- ✚ Jdk 1.6.0_07
- ✚ Make file (still in process)

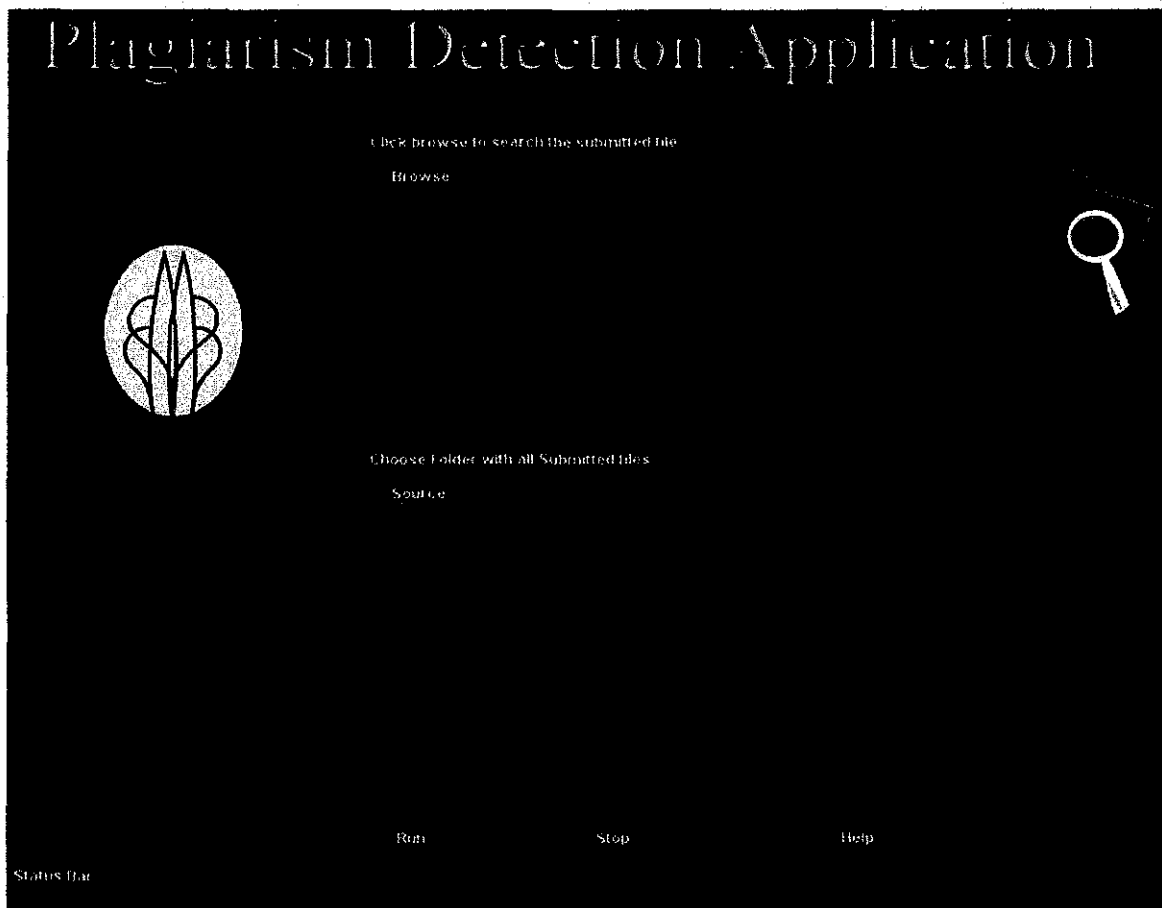


Figure 10: The interface for the application

3.4.7 The progress of this project's time line

The time line shows the repeat of literature review to the realization of the first literature review deviating from the system requirements. The new literature review was done. The softwares that thought to be the tools for implementation had to be changed as well but the change is not completely alien to the original requirements. The feasibility research has to follow the literature review that helps to find the successful application available comparing them with developed application taking into consideration the tools available

3.5 Project Implementations

Process Flow of Plagiarism detection application

As shown in figure (12). The basic flow of the application is not complex provided that the internal operations of the application are not justified.

Phase One

Step 1: click browse to locate a file being submitted. This event will trigger call the JfileChooser class to generate a fileChooser window. The user will select a file to be submitted and then click open. The event of clicking open will call the submission. Java class which will get the path of the file whilst displaying the path in the text field.

Step 2: Click source to find the folder which holds previously submitted files. This event will trigger the JfolderChooser class to generate a folder chooser window, prior to the user selecting the folder. The directorySubmission.java class will be executed which will get the path of the folder and display its path in the text field.

Phase Two

Step1: Click run button to execute the application. While the application is running the user can click help button without affecting the application. The run button will execute the main class which sends commands to other multiple classes via the Configuration. Java class . When the application complete processing the result it will execute the Report.java class which will generate a HTML file with result or Text report and display the result on the interface.

Step 2: click Stop button, this button might be used in a circumstances whereby the user loaded the wrong file or choose the wrong folder and there is no point of waiting for the undesired results.

Step 3: Help button it is used to help with the navigation of the system and explain the features of the interface

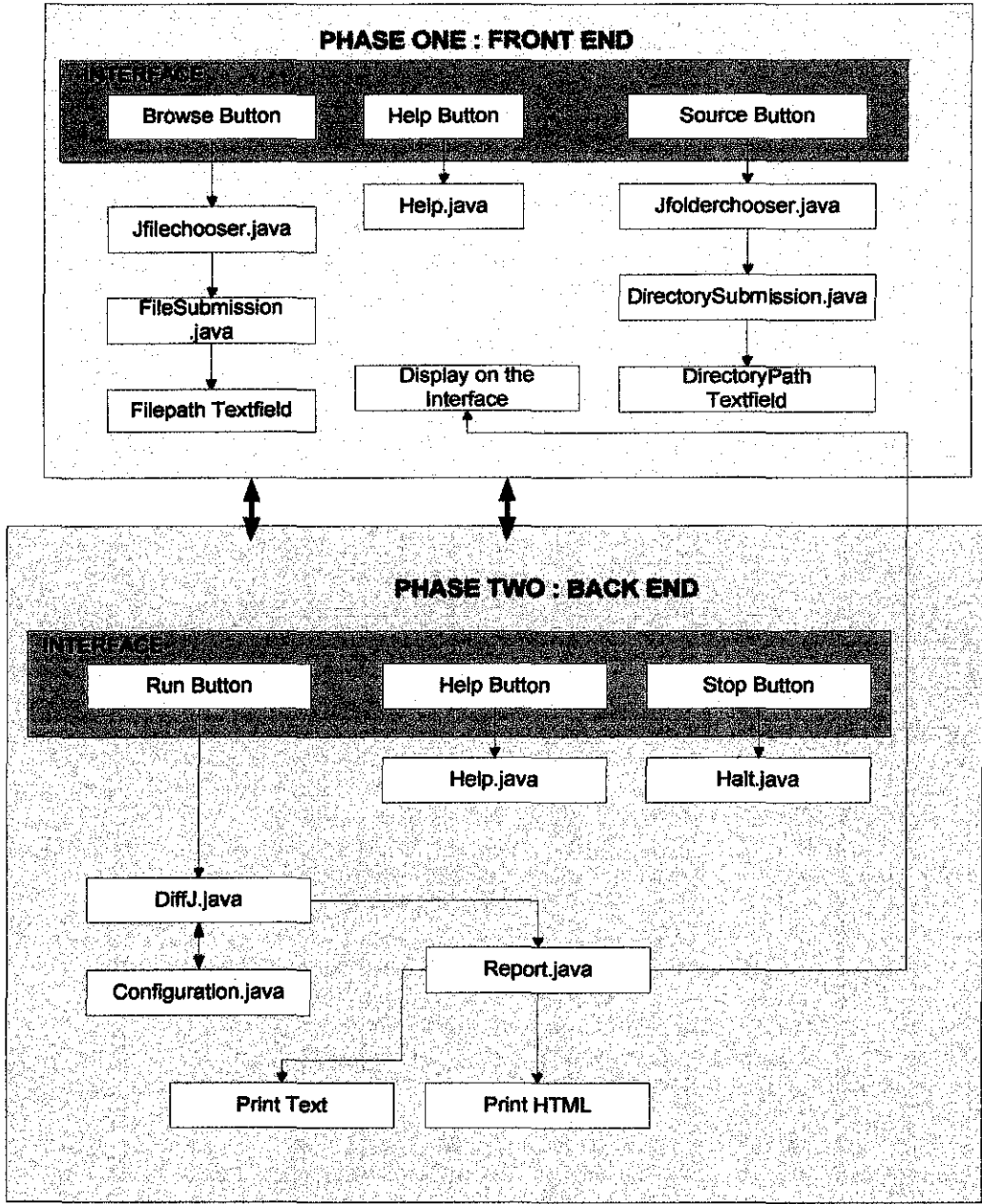


Figure 12: Process Flow of Plagiarism detection applications

CHAPTER FOUR

RESULT AND DISCUSSION

4.1 Results

The plagiarism detection application developed uses the differences found on the code to find similarities amongst the code in the same destination folder. If the application does not return any difference or changes it means that the code or the assignment has the replica of it already submitted. The application can expose the following changes:

- ✚ packages renamed
- ✚ imports added and removed
- ✚ types (classes and interfaces) added and removed
- ✚ methods added and removed
- ✚ fields added and removed
- ✚ code changed within methods and constructors, and for field initializers
- ✚ access changed
- ✚ changes in method/ctor throws list

The plagiarism detection application can accept a submitted java files and compare it with the other files already submitted by other students. The application can only detect java files in the destination folder (as shown in figure 13 below), other formats will not be displayed or even be considered when the application is executed.

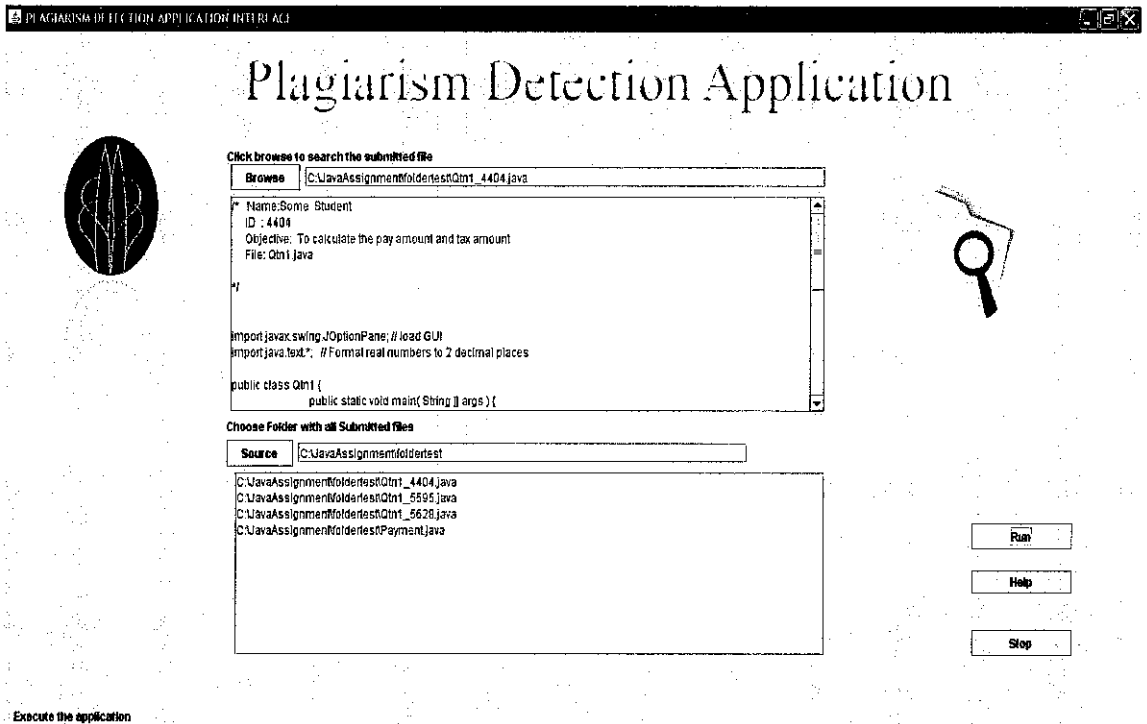


Figure 13: User Interface with user inputs

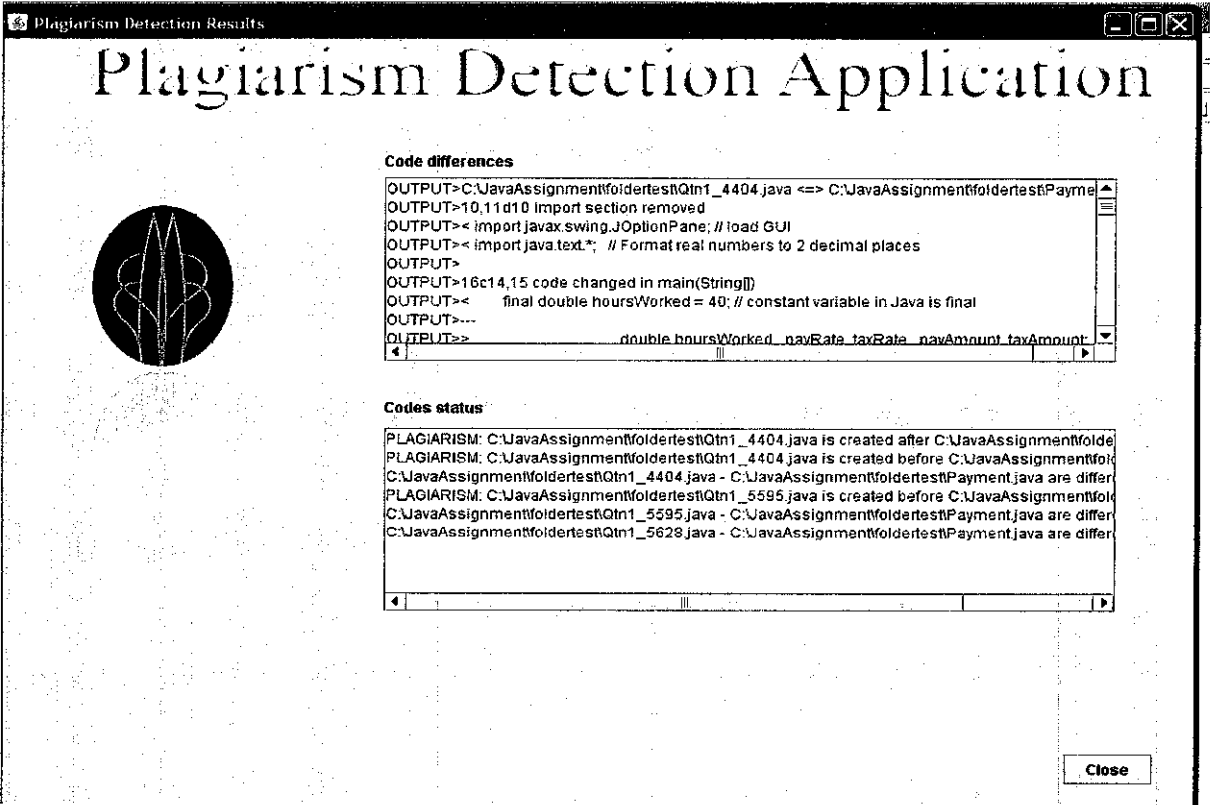


Figure 14: Plagiarism Detection Application's output

As shown in figure 14, the application returns the differences among the files within the destination folder whilst displaying the status of all the files in the folder. As the submitted file might be the same as the newly submitted file the application can also show which file was created first. If the result shows that few files are plagiarized it is advised to remove them from the destination folder.

The application only compare with the same class name, as such the files submitted might require the student to have a standard class name. The reason behind this logic is that when the system has so many functions/ task that require a single action, it decreases the cohesiveness of the system. Consequently the system will not be reliable because any error within either task will hinder the operation of other task as they are intertwined. Converting each class to have the same name will require to be executed separately as it is also a major function. As such to overcome this encumbrance setting the class name before the files are compared was considered.

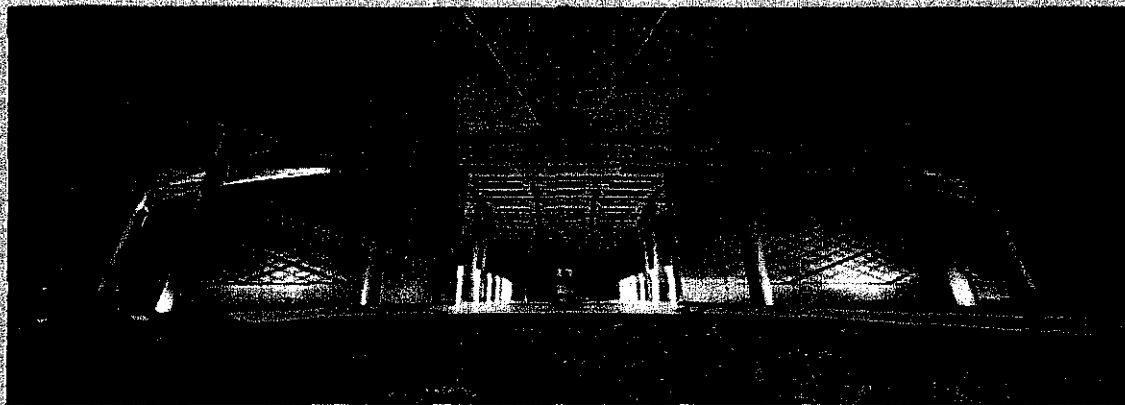
4.2 Discussion

Most plagiarism detection systems function are based on the chances that the student changed variable names or jumbled the code like spaghetti to confuse the lecturers. The other approach taken was looking at the way the application could be improved was doing a survey based on hypothetical questions conveyed verbally among my peers and my juniors assuring them that those questions are just for killing time and nothing much. In entering the mind of a plagiarizer beside my own, it was found that it is most likely that the original file or assignment is created first. In situations were both assignments were created on the same day the copy would have been modified recently.

Students in UTP do not put too much effort when they plagiarize. The same pattern and behavior between plagiarizer and the source is classic, meaning that it is not complex. The student who knows their work they are unlikely to procrastinate with their assignment leading to their file assignment possessing the properties of being created first. The students who plagiarize wait till the last minute and then in panic they plagiarize not considering the fact that the file's properties will show that it was

completed in unrealistic time. Based on this finding it is thought that it will be a contribution to also insert a function that return date created and modified .

Figure 15 depicts the window generated by the help button in the output window. Although the system has the right functionality to detect plagiarism the output can be quite puzzling for the first time user and for the novice user who has no programming background but it is highly unlikely since this is for java programming lecturers. The system uses a format that is of assembly language for the computer to understand it as such few standard symbols of the the output need s to be explained.



Symbol	Definitions
c	<u>C</u> hange
d	D e lete
a	A d
-b	Ignore blanks //flags
-l	Ignore cases //flags
>	Add the line displayed
<	Remove the line displayed
Examples of the commands mentioned	
5d4	Delete line 5 from the file 1 and starting from line4 file 2 will match file 1
21a22	Add new line after 21 of the old file
7c8	Line 7 of new file has been change

Figure 15: Help window for the output.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

Plagiarism is a crime that is taken lightly by most students because it is not likely to be reported that a particular student is going for a hearing due to plagiarism. As a result they tend to copy other student's work not taking into consideration the penalties they might face. Having regulations against plagiarism and implementing this system will serve as a threshold to the counteraction against plagiarism in UTP.

The application can detect plagiarism that obvious and easy to spot as well as the complicated ones. Students who are oblivious about their work will be exposed beyond doubts. Since the students have the same lecture and have the same privileges to the UTP resources, some plagiarism cases can be debated and /or justified as why they are little bit similar. The intention of this application is not make life hard for the student or increase work for the lectures but to shape the minds of the student while allowing them to unleash their creativity. Innovation is hindered when creativity is duplicated.

5.2 RECOMMENDATIONS

- The Application can be modified to cater for any type of files.
- Implement the function application to compare the submitted files with internet resources
- Improve the system to capture every plagiarism cases based on student id, for future references

REFERENCES

- [1] George town university Honor Council (1999)
- [2] J. Evan Noynaert , department of computers science , *Plagiarism Detection Software mathematics and physics*
- [3] B.Belkouch, Anastacia Nix , Johnette hasfell EECS department Tulane University , *Plagiarism Detection in Software Designs.*
- [4] Maxim Mozgovoy, Kimmo Fredriksson
- [5] Lutz Prechelt Guido malpoh, *Finding Plagiarisms among a Set of Programs with Jplag*, 30 march 2000.
- [6] Christian Arwin, *Plagiarism Detection across Programming Languages*
- [7] Xin Chen , brent , Mingli , Brian McKinnon Amit Seker , *Shared Information and Program Plagiarism Detection.* December 13, 2003.
- [8] Kevin w. Browser and Lawrence O.Hall , *Experience Using "MOSS" to Detect Cheating On Programming Assignments*, University of south florida
- [9] Young Chul Kim –Yong –Yooncho, Jong Bae Moon , *Computing technology , A Plagiarism Detection System Using A, Syntax-Tree* December 2004.
- [10] www.mariosalexandrou.com
- [11] <http://gervaseprograms.georgetown.edu/hc/plagiarism.html>
- [12] J. Faidhi and S. Robinson. *An empirical approach for detecting program similarity and plagiarism within a program*
- [13] Jeong-Hoon Ji, Gyun Woo, Hwan-Gue Cho, *A Source Code Linearization Technique for Detecting Plagiarized Programs*
- [14] Paul Clough, July 2000, *Plagiarism in natural and programming languages: an overview of current tools and technologies.*
- [15] <http://genome.math.uwaterloo.ca/SID/>
- [16] Michael J. Wise, *Improved detection of similarities in computer program and other texts*, Department of Computer Science, University of Sydney, Australia
- [17] Shauna D. Stephens , *Using Metrics to Detect Plagiarism*
Department of Computer and Information Sciences

APPENDICES

Appendix A

Plagiarism Detection Application Timeline (First Half)

ID	Task Name	Start	Finish	Duration	Jan 28 2007			Feb 4 2007			Feb 11 2007			Feb 18 2007			Feb 25 2007			Mar 4 2007																		
					29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3
1	Research	1/29/2007	3/9/2007	30d	[Task 1: Research, 30 days duration, from 1/29/2007 to 3/9/2007]																																	
2	Feasibility Research	1/29/2007	2/2/2007	5d	[Task 2: Feasibility Research, 5 days duration, from 1/29/2007 to 2/2/2007]																																	
3	Literature review	2/1/2007	2/5/2007	2d 4h	[Task 3: Literature review, 2 days 4 hours duration, from 2/1/2007 to 2/5/2007]																																	
4	Analysis	2/5/2007	2/23/2007	15d	[Task 4: Analysis, 15 days duration, from 2/5/2007 to 2/23/2007]																																	
5	User Requirements	2/5/2007	2/9/2007	5d	[Task 5: User Requirements, 5 days duration, from 2/5/2007 to 2/9/2007]																																	
6	System Requirements	2/12/2007	2/16/2007	5d	[Task 6: System Requirements, 5 days duration, from 2/12/2007 to 2/16/2007]																																	
7	Algorithm Analysis	2/19/2007	2/23/2007	5d	[Task 7: Algorithm Analysis, 5 days duration, from 2/19/2007 to 2/23/2007]																																	
8	Tool Gathering	2/26/2007	3/8/2007	9d	[Task 8: Tool Gathering, 9 days duration, from 2/26/2007 to 3/8/2007]																																	
9	Software Downloading	2/26/2007	3/2/2007	5d	[Task 9: Software Downloading, 5 days duration, from 2/26/2007 to 3/2/2007]																																	
10	Setting the environment	3/2/2007	3/8/2007	5d	[Task 10: Setting the environment, 5 days duration, from 3/2/2007 to 3/8/2007]																																	

Figure 7: plagiarism detection Progress timeline

Appendix B

Plagiarism Detection Application (Second Half)

ID	Task Name	Start	Finish	Duration	Mar 11 2007							Mar 18 2007							Mar 25 2007							Apr 1 2007							Apr 8 2007							Apr 15 2007							Apr 22 2007				
					12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
1	Research	3/12/2007	4/26/2007	34d	[Task 1: Research - Active from 3/12 to 4/26]																																														
2	Literature Review	3/12/2007	3/16/2007	5d	[Task 2: Literature Review - Active from 3/12 to 3/16]																																														
3	Feasibility Research	3/16/2007	3/21/2007	4d	[Task 3: Feasibility Research - Active from 3/16 to 3/21]																																														
4	Analysis	3/21/2007	3/23/2007	3d	[Task 4: Analysis - Active from 3/21 to 3/23]																																														
5	System requirements	3/26/2007	3/30/2007	5d	[Task 5: System requirements - Active from 3/26 to 3/30]																																														
6	Package gathering	4/2/2007	4/6/2007	5d	[Task 6: Package gathering - Active from 4/2 to 4/6]																																														
7	Package installation	4/6/2007	4/13/2007	6d	[Task 7: Package installation - Active from 4/6 to 4/13]																																														
8	Code Analysis	4/13/2007	4/18/2007	4d	[Task 8: Code Analysis - Active from 4/13 to 4/18]																																														
9	Interface Design	4/17/2007	4/24/2007	6d	[Task 9: Interface Design - Active from 4/17 to 4/24]																																														
10	Creating Functions	4/24/2007	4/26/2007	3d	[Task 10: Creating Functions - Active from 4/24 to 4/26]																																														

Figure 11: Plagiarism detection Timeline Second Half.

Appendix D

Interface Code

```
/*
 * DetectionApplicationUI.java
 *
 * Created on May 2, 2007, 3:58 PM
 */
package DetectionApplication;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import javax.swing.JFileChooser;
import java.io.InputStream;
import java.io.InputStreamReader;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.text.BadLocationException;

class StreamGobbler extends Thread
{
    InputStream is;
    String type;
    Object o;

    private String txtAreaOpenFileValue;

    private String txtAreaExecValue;

    StreamGobbler(InputStream is, String type)
    {
        this.is = is;
        this.type = type;
        o = null;
    }

    StreamGobbler(InputStream is, String type, Object o)
```



```

    {
        this.is = is;
        this.type = type;
        this.o = o;
    }

    public void run()
    {
        try
        {
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            String line=null;
            while ( (line = br.readLine()) != null) {
                if (o != null) {
                    javax.swing.JTextArea printArea = (javax.swing.JTextArea) o;
                    printArea.append(type + ">" + line+ "\n");
                }
                else
                    System.out.println(type + ">" + line);
            }
        } catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
    }

    /**
     *
     * @author User
     */
    public class DetectionApplicationUI extends javax.swing.JFrame {

        String txtAreaOpenFileValue;
        String txtAreaExecValue;

        /** Creates new form DetectionApplicationUI */
        public DetectionApplicationUI() {
            txtAreaExecValue = "";
            initComponents();
        }
    }

```

```

//private static FileChooser filechoose = new FileChooser();

}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
private void initComponents() {
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    browsebutton = new javax.swing.JButton();
    directorypath = new javax.swing.JTextField();
    sourcebutton = new javax.swing.JButton();
    filepath = new javax.swing.JTextField();
    jScrollPane1 = new javax.swing.JScrollPane();
    file1area = new javax.swing.JTextArea();
    stopbutton = new javax.swing.JButton();
    jLabel4 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    helpbutton = new javax.swing.JButton();
    jScrollPane2 = new javax.swing.JScrollPane();
    directoryfiles = new javax.swing.JTextArea();
    Runbutton = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Plagiarism Detection Application");
    jLabel1.setIcon(new javax.swing.ImageIcon("D:\\ACADEMICS\\FINAL YEAR PROJECT\\aplname.gif"));
    jLabel1.setText("Applicationname");

    jLabel2.setIcon(new javax.swing.ImageIcon("D:\\ACADEMICS\\FINAL YEAR PROJECT\\searchiconA.gif"));
    jLabel2.setText("Searchicon");

    jLabel3.setIcon(new javax.swing.ImageIcon("D:\\ACADEMICS\\FINAL YEAR PROJECT\\Logonew.gif"));
    jLabel3.setName("utplogo");

    browsebutton.setText("Browse");
    browsebutton.addMouseListener(new java.awt.event.MouseAdapter() {

```

```

    public void mouseClicked(java.awt.event.MouseEvent evt) {
        browsebuttonMouseClicked(evt);
    }
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        browsebuttonMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        browsebuttonMouseExited(evt);
    }
});
browsebutton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        browsebuttonActionPerformed(evt);
    }
});
browsebutton.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyTyped(java.awt.event.KeyEvent evt) {
        browsebuttonKeyTyped(evt);
    }
});

sourcebutton.setText("Source");
sourcebutton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        sourcebuttonMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        sourcebuttonMouseExited(evt);
    }
});
sourcebutton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        sourcebuttonActionPerformed(evt);
    }
});

file1area.setColumns(20);
file1area.setRows(5);
jScrollPane1.setViewportView(file1area);

stopbutton.setText("Stop");
stopbutton.addMouseListener(new java.awt.event.MouseAdapter() {

```

```

    public void mouseEntered(java.awt.event.MouseEvent evt) {
        stopbuttonMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        stopbuttonMouseExited(evt);
    }
});

jLabel4.setText("status bar");

jLabel7.setText("Choose Folder with all Submitted files ");

jLabel8.setText("Click browse to search the submitted file ");

helpbutton.setText("Help");
helpbutton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        helpbuttonMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        helpbuttonMouseExited(evt);
    }
});
helpbutton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        helpbuttonActionPerformed(evt);
    }
});

directoryfiles.setColumns(20);
directoryfiles.setRows(5);
jScrollPane2.setViewportView(directoryfiles);

Runbutton.setText("Run");
Runbutton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        RunbuttonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);

```

```

layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 239,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(29, 29, 29)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel7)
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                            .addComponent(runbutton)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 103,
Short.MAX_VALUE)
                            .addComponent(stopbutton)
                            .addGap(122, 122, 122)
                            .addComponent(helpbutton, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(622, 622, 622))
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jScrollPane2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 362, Short.MAX_VALUE)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(sourcebutton)
                                    .addGap(19, 19, 19)
                                    .addComponent(directorypath, javax.swing.GroupLayout.DEFAULT_SIZE, 278,
Short.MAX_VALUE))
                                .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 362, Short.MAX_VALUE)
                            .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
                                .addComponent(browsebutton)
                                .addGap(17, 17, 17)
                                .addComponent(filepath, javax.swing.GroupLayout.DEFAULT_SIZE, 278,
Short.MAX_VALUE))
                            .addComponent(jLabel8, javax.swing.GroupLayout.Alignment.LEADING))
                    .addGap(94, 94, 94)
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 151,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(411, 411, 411))))

```

```

        .addComponent(jLabel1,          javax.swing.GroupLayout.PREFERRED_SIZE,      838,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel4,          javax.swing.GroupLayout.PREFERRED_SIZE,      630,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap())
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel1,          javax.swing.GroupLayout.PREFERRED_SIZE,      61,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(0, 0, Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jLabel3,          javax.swing.GroupLayout.PREFERRED_SIZE,      490,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel2,          javax.swing.GroupLayout.PREFERRED_SIZE,      167,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jLabel8)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(browsebutton)
                                .addComponent(filepath,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jScrollPane1,          javax.swing.GroupLayout.PREFERRED_SIZE,      175,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(24, 24, 24)
                            .addComponent(jLabel7)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(sourcebutton)
                                .addComponent(directorypath,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(22, 22, 22)

```

```

        .addComponent(jScrollPane2,          javax.swing.GroupLayout.PREFERRED_SIZE,      170,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(58, 58, 58)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(stopbutton)
            .addComponent(helpbutton)
            .addComponent(Runbutton))))))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel4,          javax.swing.GroupLayout.PREFERRED_SIZE,      22,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(25, 25, 25))
    };
    pack();
} // </editor-fold>

```

```

private void RunbuttonActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    txtAreaOpenFile.setText("");
    txtAreaOpenFileValue = "";
    txtAreaExecValue = "";
    // txtAreaExec.setText("");

    try {
        int lineStartOffset, lineEndOffset, lineCount = directoryfiles.getLineCount();
        String name1, name2;
        long d1, d2;
        // txtAreaExec.setText("");
        frmDiffJ myDiffJWindow = new frmDiffJ();
        for (int i=0; i<lineCount-1; i++) {
            lineStartOffset = directoryfiles.getLineStartOffset(i);
            lineEndOffset = directoryfiles.getLineEndOffset(i);
            name1 = directoryfiles.getText(lineStartOffset, lineEndOffset-lineStartOffset-1);
            for (int j=i+1; j<lineCount-1; j++) {
                lineStartOffset = directoryfiles.getLineStartOffset(j);
                lineEndOffset = directoryfiles.getLineEndOffset(j);
                name2 = directoryfiles.getText(lineStartOffset, lineEndOffset-lineStartOffset-1);
                if (diffj (name1, name2, myDiffJWindow.getTxtAreaDiffJ()) == 0) {
                    /*
                     * Get the timestamp from file 1
                    */
                    String f1 = "Square.java";
                    long d1 = new File(f1).lastModified();
                }
            }
        }
    }
}

```

```

if (d1 == d2)
relation = "the same age as";
else if (d1 < d2)
relation = "Created before";
else
relation = "created after";
System.out.println(f1 + " was " + relation + "' + f2);
    */
    d1 = new File (name1).lastModified();
    d2 = new File (name2).lastModified();
    if (d1 == d2) {
        //txtAreaOpenFile.append("PLAGIARISM: " + name1 + " and " + name2 + " also has the same
timestamp.\n");
        txtAreaOpenFileValue = txtAreaOpenFileValue + "PLAGIARISM: " + name1 + " and " + name2 +
" also has the same timestamp.\n";
    } else if (d1 < d2) {
        //txtAreaOpenFile.append("PLAGIARISM: " + name1 + " is created before " + name2 + ".\n");
        txtAreaOpenFileValue = txtAreaOpenFileValue + "PLAGIARISM: " + name1 + " is created before "
+ name2 + ".\n";
    } else
        //txtAreaOpenFile.append("PLAGIARISM: " + name1 + " is created after " + name2 + ".\n");
        txtAreaOpenFileValue = txtAreaOpenFileValue + "PLAGIARISM: " + name1 + " is created after "
+ name2 + ".\n";
    } else {
        //txtAreaOpenFile.append(name1 + " - " + name2 + " are different\n");
        txtAreaOpenFileValue = txtAreaOpenFileValue + name1 + " - " + name2 + " are different\n";
    }
}
}
}
System.out.println("Debug txtAreaExecValue:\n"+txtAreaExecValue);

//myDiffJWindow.setTxtAreaDiffJValue(txtAreaOpenFileValue);
//myDiffJWindow.setTxtAreaDiffJValue(txtAreaExecValue);
myDiffJWindow.setVisible(true);
} catch (BadLocationException ex) {
    ex.printStackTrace();
}
// --- end of execute diffj ---

}

private void btnExecActionPerformed(java.awt.event.ActionEvent evt) {

```



```

diffj ("C:/diffj/Hello.java", "C:/diffj/Hello2.java", null);
}

private int diffj(String name1, String name2, Object o){
// TODO add your handling code here:
try
{
String osName = System.getProperty("os.name" );
String cmd;
cmd = "java -cp C:/diffj/classes org.incava.diffj.DiffJ " + name1 + " " + name2;
Runtime rt = Runtime.getRuntime();
System.out.println("Execing " + cmd);
Process proc = rt.exec(cmd);

// any error message?
//this line will display the output un the main window so not ncessary
// StreamGobbler errorGobbler = new StreamGobbler(proc.getErrorStream(), "ERROR", txtAreaExec);

// any output?
//StreamGobbler outputGobbler = new StreamGobbler(proc.getInputStream(), "OUTPUT", txtAreaExec);
StreamGobbler outputGobbler = new StreamGobbler(proc.getInputStream(), "OUTPUT", o);

// kick them off
// errorGobbler.start();
outputGobbler.start();

// any error???
int exitVal = proc.waitFor();
System.out.println("ExitValue: " + exitVal);
return exitVal;
} catch (Throwable t)
{
t.printStackTrace();
}
return -1;
}

private void sourcebuttonActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
JFileChooser chooser = new JFileChooser();
chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

```

```

        int returnVal = chooser.showDialog(this,"Choose Source Directory");
// int returnVal = chooser.showOpenDialog(this);
if(returnVal == JFileChooser.APPROVE_OPTION) {
    directorypath.setText(chooser.getSelectedFile().getAbsolutePath());
    File[] allFiles = chooser.getSelectedFile().listFiles();
    directoryfiles.setText("");
    for (int i=0; i<allFiles.length; i++) {
        if (allFiles[i].isDirectory()) {
            /*--- temporary disabled ---*/
            //txtAreaSource.append("Dir: " + allFiles[i].getAbsolutePath() + "\n");
        } else {
            if
                (allFiles[i].getAbsolutePath().substring(allFiles[i].getAbsolutePath().length()-
4).compareToIgnoreCase("java")==0)
                directoryfiles.append(allFiles[i].getAbsolutePath() + "\n");
        }
    }
}

private void helpbuttonActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    new Helpfile().setVisible(true);
}

private void helpbuttonMouseExited(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
    jLabel4.setText("Status Bar");
}

private void helpbuttonMouseEntered(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
    jLabel4.setText("View the help File");
}
/*

private void sourcebuttonActionPerformed(java.awt.event.ActionEvent evt) {

}
*/

private void browsebuttonActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:

```

```

JFileChooser chooser = new JFileChooser();
// Note: source for ExampleFileFilter can be found in FileChooserDemo,
// under the demo/jfc directory in the Java 2 SDK, Standard Edition.
ExampleFileFilter filter = new ExampleFileFilter();
filter.addExtension("txt");
filter.addExtension("java");
filter.setDescription("Text & Java files");
chooser.setFileFilter(filter);
int returnVal = chooser.showOpenDialog(this);
if(returnVal == JFileChooser.APPROVE_OPTION) {
    //System.out.println("You chose to open this file: " + chooser.getSelectedFile().getName());
    try {
        filepath.setText(chooser.getSelectedFile().getAbsolutePath());
        BufferedReader in = new BufferedReader(new FileReader(chooser.getSelectedFile().getAbsolutePath()));
        file1area.setText("");
        String str;
        while ((str = in.readLine()) != null) {
            file1area.append(str+"\n");
        }
    } catch (FileNotFoundException fe) {

    } catch (IOException ie) {

    }
}

private void stopbuttonMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
    jLabel4.setText("Status Bar");
}

private void stopbuttonMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
    jLabel4.setText("Abort the application's operation");
}

private void sourcebuttonMouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
    jLabel4.setText("Status Bar");
}

```

```

    }

    private void browsebuttonMouseExited(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
        jLabel4.setText("Status Bar");
    }

    private void sourcebuttonMouseEntered(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
        jLabel4.setText("Searching for the destination folder");
    }

    private void browsebuttonMouseEntered(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:

        jLabel4.setText("Browsing for submitted file" );
    }

    private void browsebuttonKeyTyped(java.awt.event.KeyEvent evt) {
// TODO add your handling code here:
    }

    private void browsebuttonMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:
// new FileChooser().setVisible(true);
        new Multi().setVisible(true);

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new DetectionApplicationUI().setVisible(true);
            }
        });
    }
}

```

```

// Variables declaration - do not modify
private javax.swing.JButton Runbutton;
private javax.swing.JButton browsebutton;
private javax.swing.JTextArea directoryfiles;
private javax.swing.JTextField directorypath;
private javax.swing.JTextArea filelarea;
private javax.swing.JTextField filepath;
private javax.swing.JButton helpbutton;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JButton sourcebutton;
private javax.swing.JButton stopbutton;
// End of variables declaration

}

```

OutPut Code

```

/*
 * frmDiffJ.java
 * Created on September 24, 2007, 8:25 PM
 */

/**
 *
 * @author User
 */
public class frmDiffJ extends javax.swing.JFrame {

    /** Creates new form frmDiffJ */
    public frmDiffJ() {
        initComponents();
    }
}

```

```

public void setTxtAreaDiffJValue (String val) {
    txtAreaDiffJ.setText(val);
}

public javax.swing.JTextArea getTxtAreaDiffJ () {
    return this.txtAreaDiffJ;
}

public void setTxtAreaExecValue (String val) {
    txtAreaExec.setText(val);
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc="Generated Code ">
private void initComponents() {
    btnClose = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    txtAreaDiffJ = new javax.swing.JTextArea();
    jScrollPane2 = new javax.swing.JScrollPane();
    txtAreaExec = new javax.swing.JTextArea();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    btnClose.setText("Close");
    btnClose.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnCloseActionPerformed(evt);
        }
    });

    txtAreaDiffJ.setColumns(20);
    txtAreaDiffJ.setRows(5);
    jScrollPane1.setViewportView(txtAreaDiffJ);

    txtAreaExec.setColumns(20);
    txtAreaExec.setRows(5);
    jScrollPane2.setViewportView(txtAreaExec);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

```

```

        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(411, Short.MAX_VALUE)
            .addComponent(btnClose)
            .addContainerGap())
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                .addComponent(jScrollPane2, javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScrollPane1,
                    javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 343, Short.MAX_VALUE))
            .addContainerGap(127, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                184,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(26, 26, 26)
            .addComponent(jScrollPane2,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                127,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 23, Short.MAX_VALUE)
            .addComponent(btnClose)
            .addContainerGap())
    );
    pack();
} // </editor-fold>

private void btnCloseActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    this.setVisible(false);
    this.dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new frmDiffJ().setVisible(true);
        }
    });
}

```

```
    }  
  });  
}
```

```
// Variables declaration - do not modify  
private javax.swing.JButton btnClose;  
private javax.swing.JScrollPane jScrollPane1;  
private javax.swing.JScrollPane jScrollPane2;  
private javax.swing.JTextArea txtAreaDiff;  
private javax.swing.JTextArea txtAreaExec;  
// End of variables declaration
```

```
}
```