Static Source Code Analysis Tools and their Application to the Detection of Plagiarism in Java Programs

James Hamilton

Presentation Outline

Program Transformation and Static Analysis

Evaluation of Static Analysis Tools

Plagiarism Detection

Conclusion and Further Work

Program Transformation

Program transformation is the act of changing one program into another.

```
class Rephrasing {
        public static void main(String[] args) {
                for(int i = 0; i < 10; i++) {
                        System.out.println("Hello World");
                }
        }
}
class Rephrasing {
        public static void main(String[] args) {
                int i = 0;
                while(i < 10) {
                        System.out.println("Hello World");
                        i++;
                }
        }
}
```

Static Analysis

Static analysis involves analysing program code without executing it

For example counting how many variables are declared in a program

```
public class HowManyVariables {

        public static void main(String[] args) {

                int x = 5;
                int y = 4;
            if(args[0].equals("hello")) {
                int z = x * y;

                System.out.println(z);
            }
        }
}
```

4 Variables

Tools

ANTLR
JavaCC
javac / Java Compiler API
Eclipse JDT
TXL

Tool: ANTLR

- Parser generator which outputs a parser in Java or other languages
- Takes a grammar defining a language as input
- Definitive ANTLR book is very useful
- Grammar download doesn't include clases like JavaCC
- Two Java 1.5 grammars

Tool: JavaCC

Parser generator which outputs a Java parser
Takes a grammar defining a language as input
Grammar is very untidy
Lacks documentation
Download includes a complete Java Grammar
  with AST classes

Tool: javac / Java Compiler API

Written in Java by Sun Microsystems
In 2006, Bruce Eckel found the Eclipse Compiler was more accurate than javac
Not designed for Static Analysis
Hard to convert for our purpose
Internal classes aren't documented and liable to change

Tool: Eclipse JDT

- Package for parsing, compiling, analysing and transformation Java source.
- Basis of the Eclipse IDE
- Accurate implementation
- No need for grammars
- Lack of documentation especially for tree re-writing
- But easy to understand

Tool: TXL

functional language, very different from the other tools – doesn't produce a parser

A lot of documentation is available from the website

The Java 1.5 grammar had an error

Harder for to use

Tool Conclusion

Eclipse is chosen as the best tool for our purpose.
Eclipse IDE is also very good.
TXL is very different from the other tools.
javac, JavaCC, Eclipse visitors all very similar
For parsing languages other than Java ANTLR would be the best choice.

A plagiarism detector will served as a test to try Eclipse with a larger static analysis task

Plagiarism Detection

Detecting the similarity between Java source code pairs in sets of student's Java assignments

Involves static analysis of the Java programs

Assignment of a similarity value between program pairs

Plagiarism Techniques

Level 1 comments e.g. add, remove or change comments

Level 2 identifiers e.g. rename identifiers

Level 3 code positions e.g. move field variable declarations from the top of the source to the bottom

Level 4 procedure combination e.g. in-lining procedures

Level 5 program statements e.g. rearranging program statements

Level 6 control logic e.g. changing for-loops to while-loops

Faidhi and Robinson defined 6 levels of source code plagiarism

Plagiarism Techniques

changing identifier names

```java
public class VarChangeA {

        public static void main(String[] args) {
                for(int i = 1; i <= 10; i++)
                        System.out.println(i + ": " + factorial(i));
        }

        public static int factorial(int n) {
                if(n <= 1)
                        return 1;
                else
                        return n * factorial(n - 1);
        }
}
```

```java
public class VarChangeB {

        public static void main(String[] n) {
                for(int b = 1; b <= 10; b++)
                        System.out.println(b + ": " + f(b));
        }

        public static int f(int a) {
                if(a <= 1)
                        return 1;
                else
                        return a * f(a - 1);
        }
}
```

Plagiarism Techniques

adding comments

```
/*  Program: A program to output numbers 1 - 10 and their factorials
        Author: James
        Date:   12/01/08
*/

public class VarChangeAWithComments {

        /* main method */
        public static void main(String[] args) {
                /* iterate through numbers 1 - 10 */
                for(int i = 1; i <= 10; i++)
                /* print out the number i and it's factorial */
                        System.out.println(i + ": " + factorial(i));
        }

        /* method to calculate factorial for a number */
        public static int factorial(int n) {
                /* computing factorial */
                if(n <= 1)
                        return 1;
                else
                        return n * factorial(n - 1);
        }
}
```

Plagiarism Techniques

restructuring

```java
public class RestructureA {

        private int x = 5;
        private int y = 3;
        private int z = 2;

        public static void main(String[] args) {
                new RestructureA();
        }

        public RestructureA() {
                doAnotherThing();
        }

        public void doSomething() {
                System.out.println(x);
        }

        public void doAnotherThing() {
                System.out.println(z);
                doSomething();
        }

        public String toString() {
                System.out.println("x: " + x + ", y: " + y + ", z: " + z);
        }
}
```

```java
public class RestructureB {

        public void doAnotherThing() {
                System.out.println(z);
                doSomething();
        }

        public RestructureB() {
                doAnotherThing();
        }

        public void doSomething() {
                System.out.println(x);
        }

        public String toString() {
                System.out.println("x: " + x + ", y: " + y + ", z: " + z)
        }

        private int x = 5;
        private int y = 3;
        private int z = 2;

        public static void main(String[] args) {
                new RestructureB();
        }
}
```

Plagiarism Techniques

negating *if* constructs

```
if(true) {
        //do this
}else{
        //do that
}
```

```
if(!true) {
        //do that
}else{
        //do this

}
```

Plagiarism as Code Obfuscation

Code obfuscation is the transformation of source code in such a way that makes it unintelligible to human readers of the code and reverse engineering tools, such as program slicing tools

Plagiarism can be view as a form of code obfuscation – students obfuscate code in simple ways to avoid copied code being detected as plagiarised.

Collberg *et al* used code obfuscation tools as tools for plagiarising with several experiments to test MOSS with submissions plagiarised automatically with the SANDMARK framework.

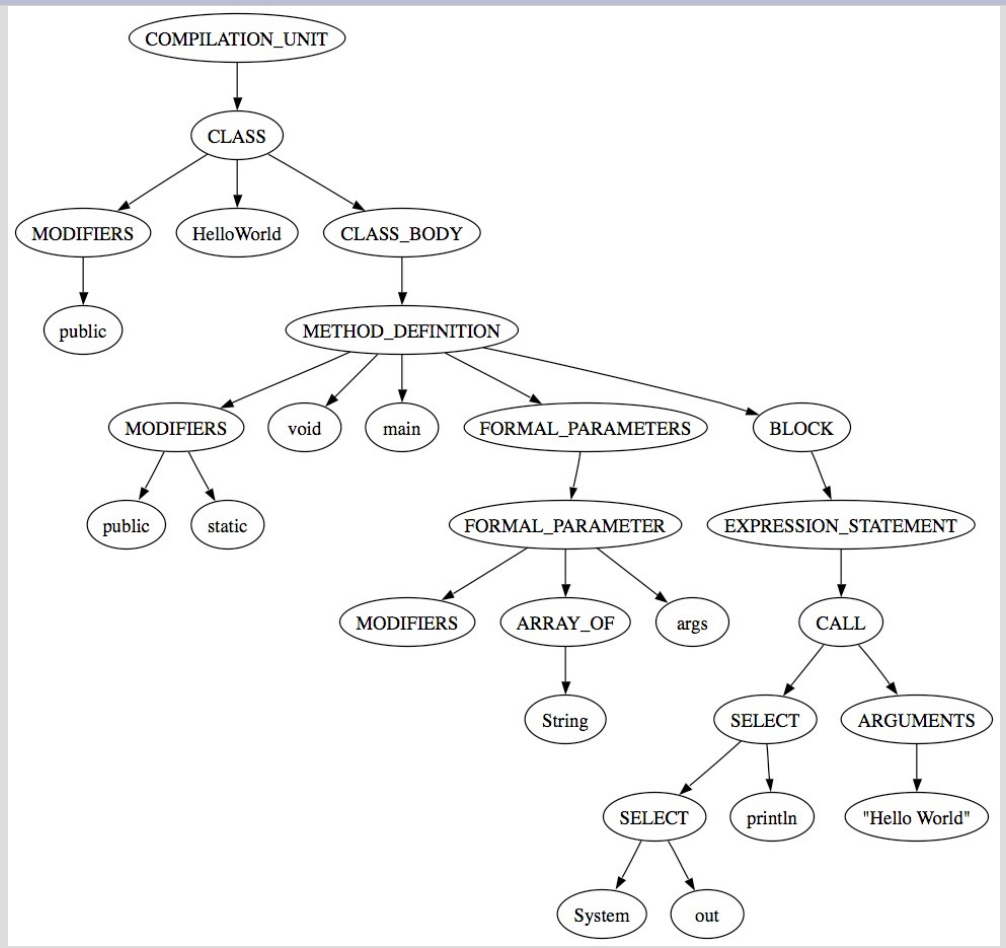
Example of Code Obfuscation

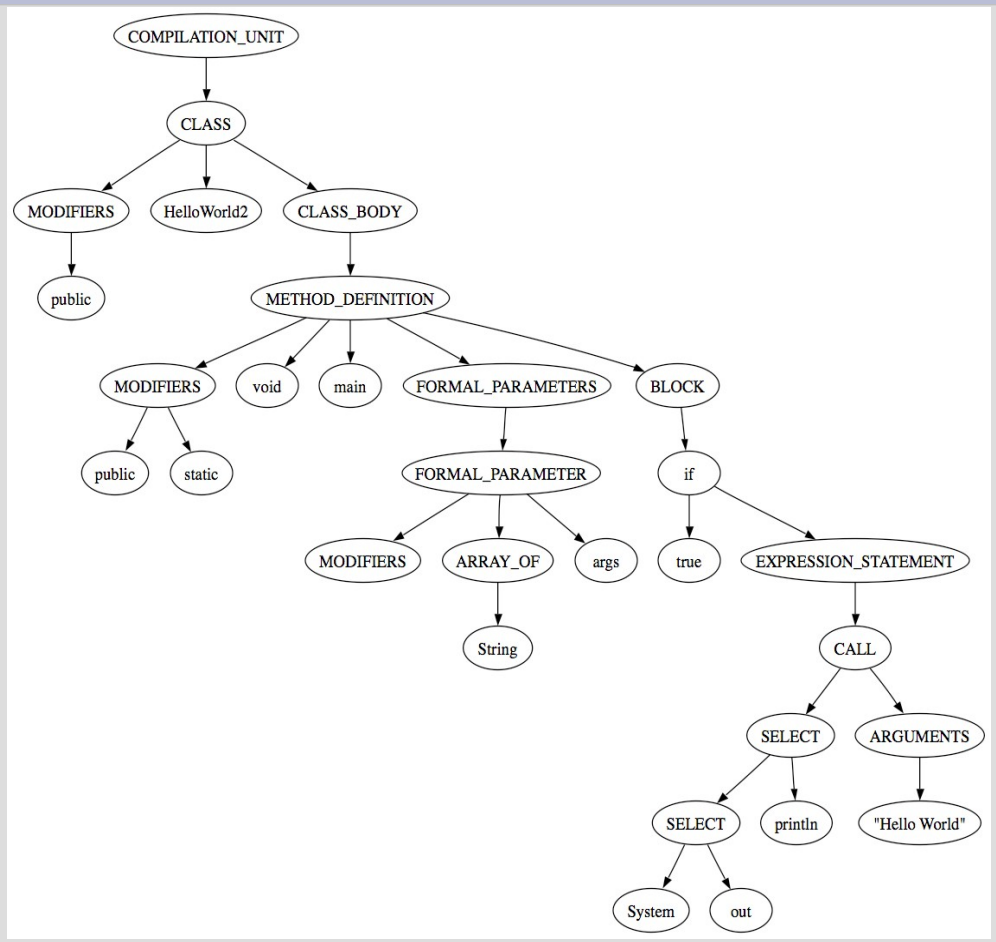
```
public static final double TAX_RATE = 15;

public int calculateTax(Product product) {
        double price = product.getValue();
        return price + ( price * TAX_RATE );
}
```

```
public static final double asfkfiasdy8213jhg23 = 15;

public int jk2h138f7s7d8f(sjdh862312 aksjdshagd2lkjLKAJMNCX) {
        double msd99227djdg = aksjdshagd2lkjLKAJMNCX.kv00238123bfsdf();
        return msd99227djdg + ( msd99227djdg * asfkfiasdy8213jhg23 );
}
```

A Plagiarism Detection Technique

AST Comparison



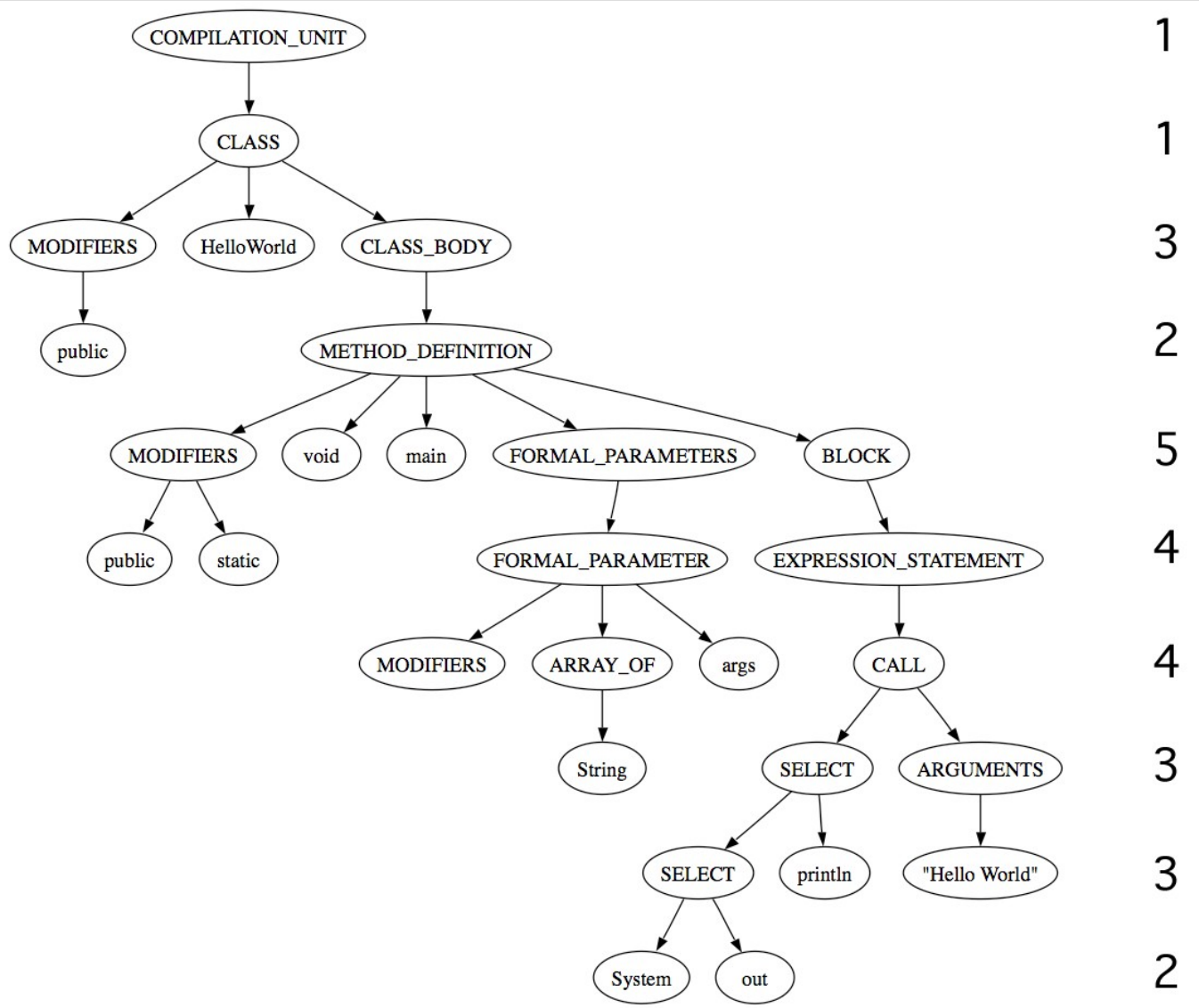



```
public class HelloWorld {

        public static void main(String[] args) {
                System.out.println("Hello World");
        }

}
```

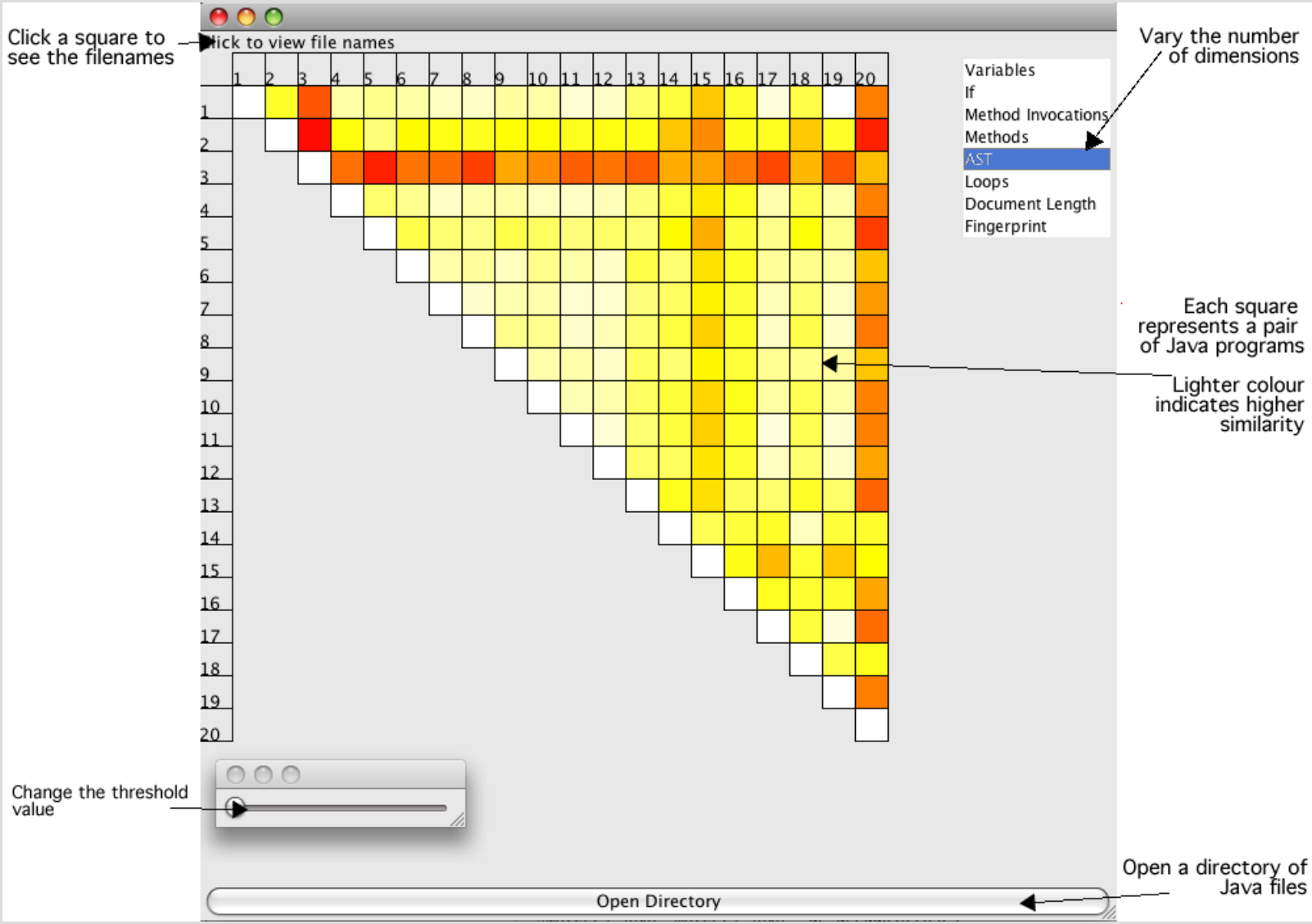
```
public class HelloWorld2 {

        public static void main(String[] args) {
                if(true)
                        System.out.println("Hello World");
        }

}
```

A Plagiarism Detection Technique

AST Node counting



The Plagiarism Detector

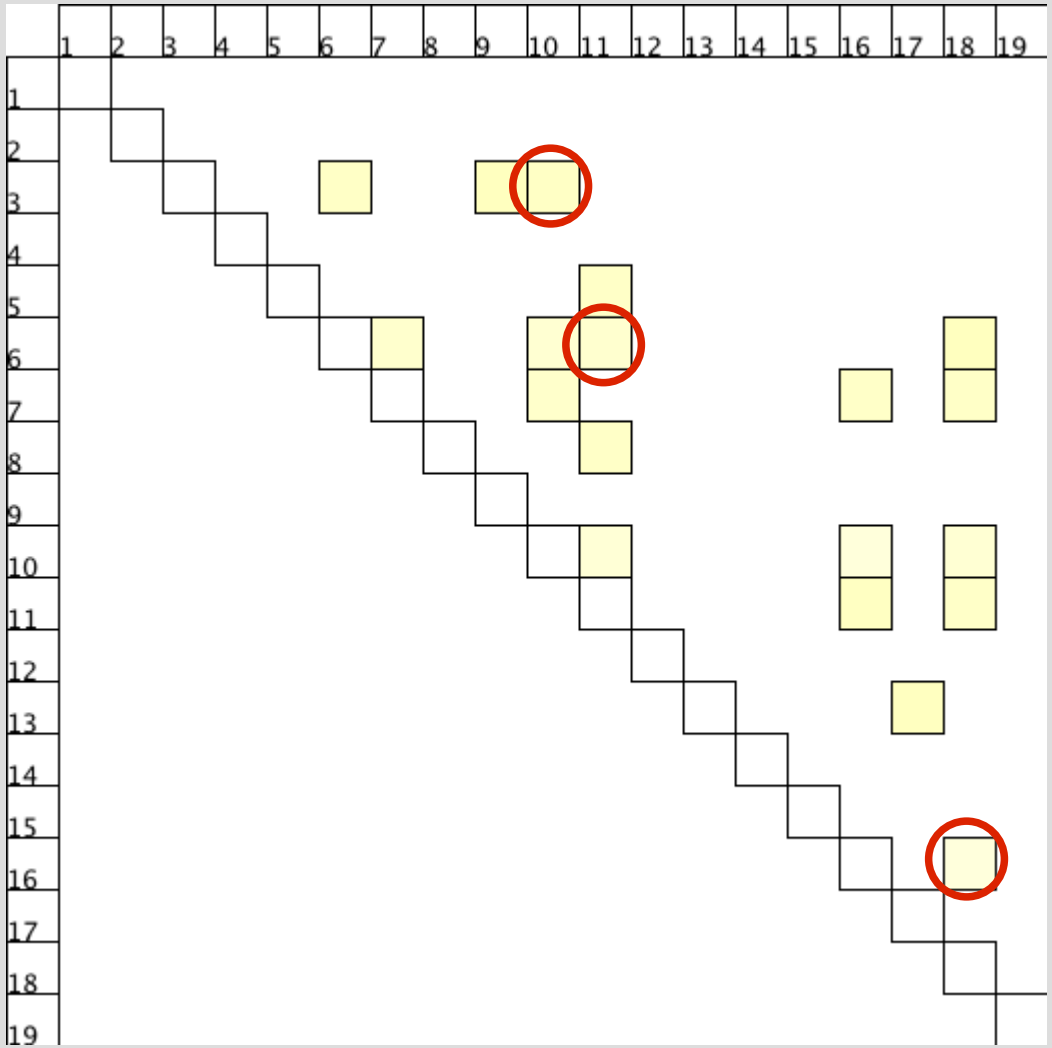


Empirical Study

8 first year Java assignments with between 13 and 30 submissions for each

'maze' assignment chosen randomly for manual checking finding 16 plagiarised pairs

Maze Set Results



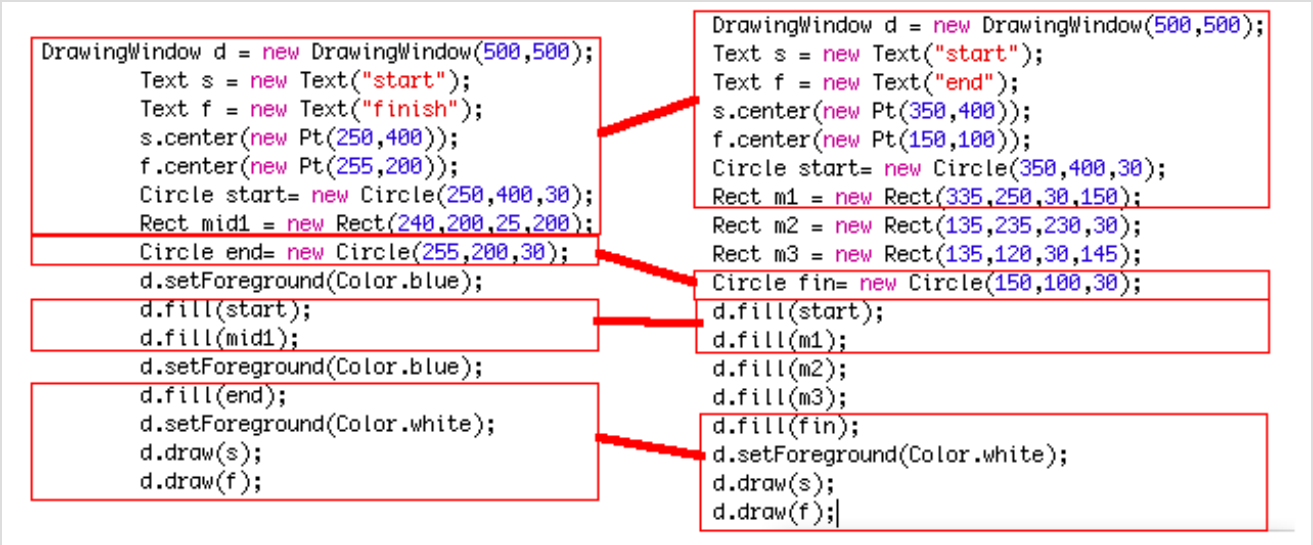
19 pairs detected as plagiarised which includes 8 of the 16 known plagiarised pairs

After analysing the results and re-checking several more plagiarised pairs were found

Some false-positives

Maze Set Results

Extract from P_6 and P_{11}



```
DrawingWindow d = new DrawingWindow(500,500);
        Text s = new Text("start");
        Text f = new Text("finish");
        s.center(new Pt(250,400));
        f.center(new Pt(255,200));
        Circle start= new Circle(250,400,30);
        Rect mid1 = new Rect(240,200,25,200);
        Circle end= new Circle(255,200,30);
        d.setForeground(Color.blue);
        d.fill(start);
        d.fill(mid1);
        d.setForeground(Color.blue);
        d.fill(end);
        d.setForeground(Color.white);
        d.draw(s);
        d.draw(f);
```

```
DrawingWindow d = new DrawingWindow(500,500);
Text s = new Text("start");
Text f = new Text("end");
s.center(new Pt(350,400));
f.center(new Pt(150,100));
Circle start= new Circle(350,400,30);
Rect m1 = new Rect(335,250,30,150);
Rect m2 = new Rect(135,235,230,30);
Rect m3 = new Rect(135,120,30,145);
Circle fin= new Circle(150,100,30);
d.fill(start);
d.fill(m1);
d.fill(m2);
d.fill(m3);
d.fill(fin);
d.setForeground(Color.white);
d.draw(s);
d.draw(f);
```

Maze Set Results

Extract from P_3 and P_{10}

```
DrawingWindow d = new DrawingWindow(500,500);

Text s = new Text("start");

Text f = new Text("end");

s.center(new Pt(180,410));

f.center(new Pt(400,200));

Circle start = new Circle(180,410,30);

Rect midl = new Rect(175,200,13,200);

Rect midl2 = new Rect(175,200,200,13);

Circle finish = new Circle(400,200,30);

d.fill(start);

d.fill(midl);

d.fill(midl2);

d.fill(finish);

d.setForeground(Color.white);

d.draw(s);

d.draw(f);
```

```
DrawingWindow maze = new DrawingWindow(700,700, "Maze");
long startTime = 0, stopTime, finishTime = 0;
            Text s = new Text("Start");
            Text f = new Text("Finish");
            s.center (new Pt(60,55));
            f.center (new Pt(590,55));
            Circle start = new Circle (60,60,50);
            Rect mid1 = new Rect (108,50,450,15);
            Circle finish = new Circle (600,55,50);
            maze.fill(start);
            maze.fill(finish);
            maze.fill(mid1);
            maze.setForeground(Color.blue);
            maze.draw(s);
            maze.draw(f);
```

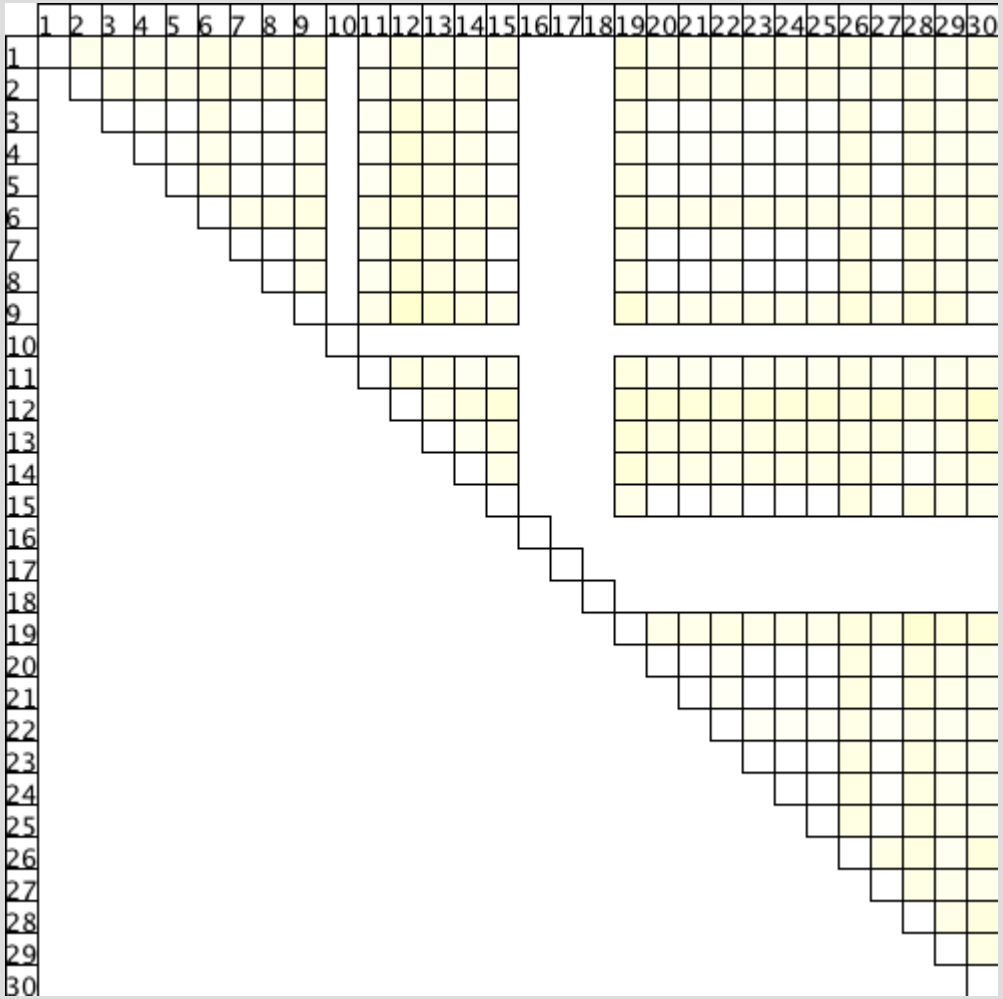
Maze Set Results

Extract from P_{16} and P_{18}

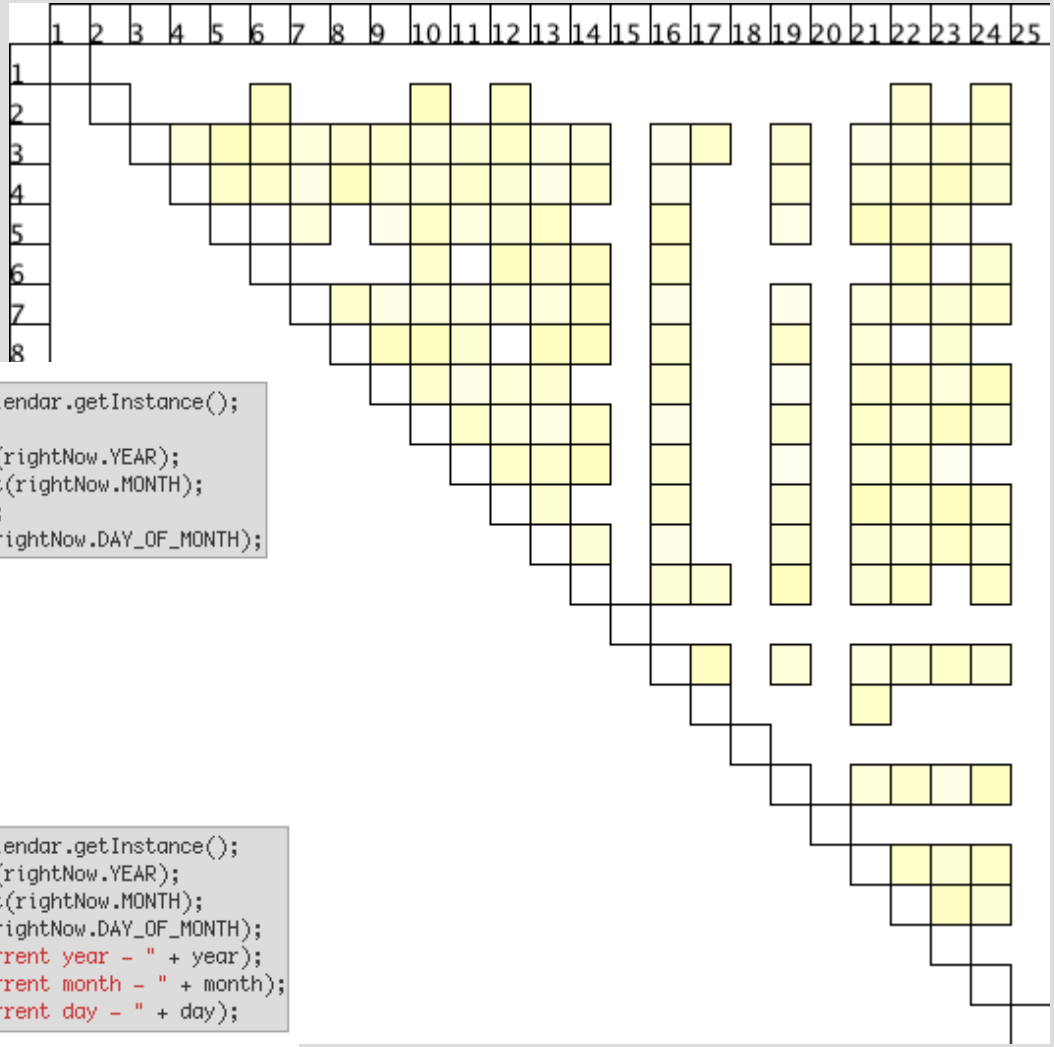
```
while(true)//loop function
    {
        Pt p1=d.getMouse();//the normal mouse function
        while(!start.contains(p1))//when the mouse does not contain in the start
        {
        /*  p1=d.getMouse();
            while(!start.contains(p1))*/
            {
                p1=d.getMouse();//then the mouse should stay normal
        }
    }
    long a=System.currentTimeMillis();//this is used to get the time in the function "a"
    while(start.contains(p1) || mid1.contains(p1)) //|| mid2.contains(p1))/*|| mid3.contains(p1)|| mid4.contains(p1)|| mid5.contains(p1)|| mid6.contains(p1)
    //|| mid7.contains(p1)|| mid8.contains(p1)|| mid9.contains(p1)|| mid10.contains(p1)|| mid11.contains(p1)|| mid12.contains(p1))
//while the mouse is on any mid points the time should continue running*/
    {
        p1=d.getMouse();//this is used to get the mouse function back again
}
    if (finish.contains(p1))//when the mouse reaches to the finish spot
    {
            long b=System.currentTimeMillis();// this is used to get the time in the function "b"
            System.out.println("Well done " + (b-a)/1000.0 + " seconds");
            //the time is then subtracted to get know how long it took. its then divided to 1000 so it turns to millisecs
        }
            else System.out.println("try again");//if the mouse doesnt stay in the mids then it should say the message try again
        }
```

```
do
{
     Pt pt;

     for(pt = drawingwindow.getMouse(); !circle.contains(pt); pt = drawingwindow.getMouse());

     long t = System.currentTimeMillis();

     for(; circle.contains(pt) || rect.contains(pt); pt = drawingwindow.getMouse());

     if(circle1.contains(pt))
     {
         long t1 = System.currentTimeMillis();
         System.out.println((new StringBuilder()).append("You completed the maze in ").append((double)(t1 - t) / 1000D).append(" seconds").toString());
     }
     else
     {
         System.out.println("You have failed the maze game!");
     }
} while(true);
```

Die Set Results



Age Set Results



Try out this program:

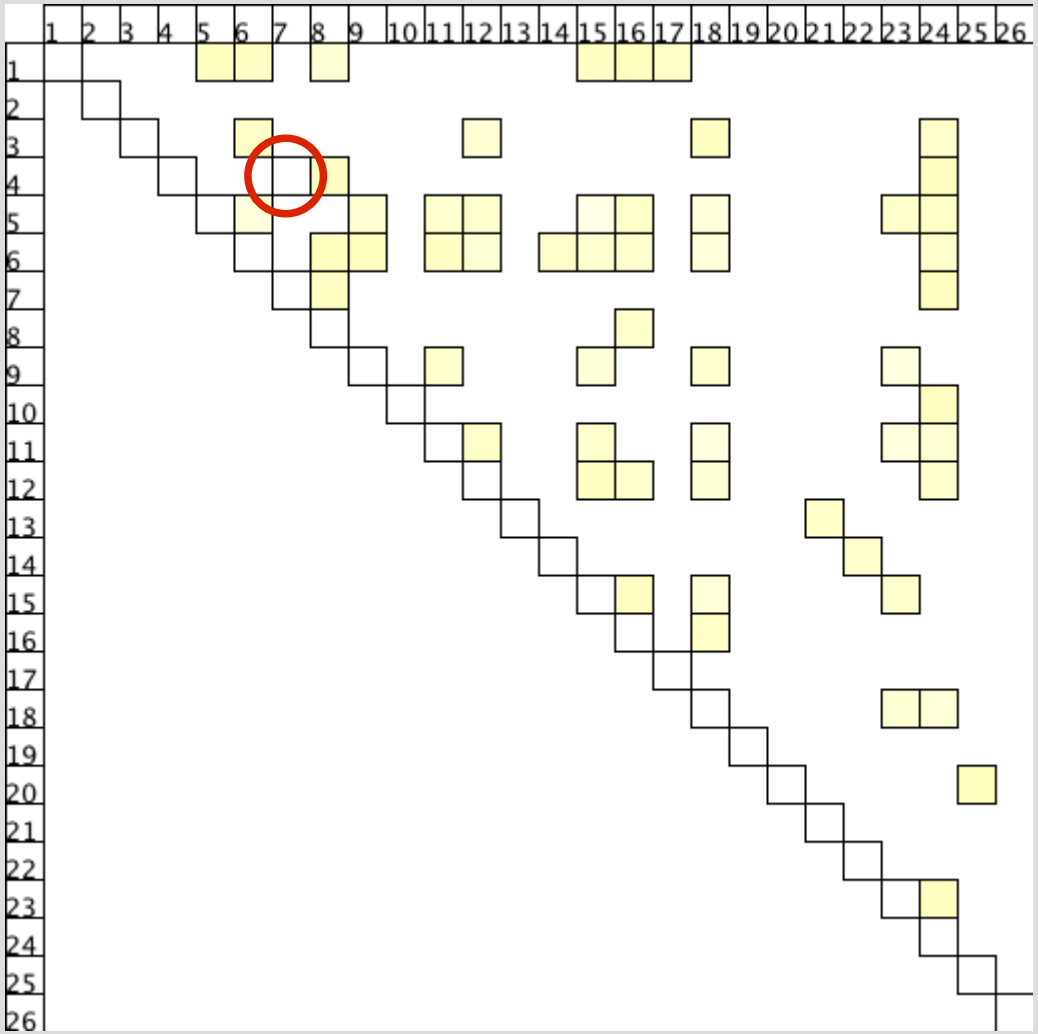
```
import java.util.Calendar;
class age
{

 public static void main( String []  args)
 {
   Calendar rightNow = Calendar.getInstance();
   int year =rightNow.get(rightNow.YEAR);
   int month =rightNow.get(rightNow.MONTH);
   int day =rightNow.get(rightNow.DAY_OF_MONTH);
   System.out.println(year);
   System.out.println(month);
   System.out.println(day);
 }

}
```

```
Calendar rightNow = Calendar.getInstance();

int year =rightNow.get(rightNow.YEAR);
int month =rightNow.get(rightNow.MONTH);
int monthnow = month+1;
int day =rightNow.get(rightNow.DAY_OF_MONTH);
```

```
Calendar rightNow = Calendar.getInstance();
int year =rightNow.get(rightNow.YEAR);
int month =rightNow.get(rightNow.MONTH);
int day =rightNow.get(rightNow.DAY_OF_MONTH);
System.out.println("Current year - " + year);
System.out.println("Current month - " + month);
System.out.println("Current day - " + day);
```

Write a program which asks the user for their date of birth and then tells them how old they are.

Guess Set Results



Guess Set Results

Extract from P_4 and P_7

```java
import java.util.Scanner;
class Guess
{
    public static void main(String []args)
    {

        System.out.println("Think of a number between 1 and 1000.");
        System.out.println("If I guess too low, enter '1', or too high, enter '2'.");
        System.out.println("If I guess right, enter any other number.");

        int min = 0;
        int max = 1001;

        int counter = 0;

        boolean stillguessing = true;

        while(stillguessing)
        {
            int guess = (min + max)/2;

            counter = counter + 1;

            System.out.println();
            System.out.println("Is your number "+ guess +"?");

            Scanner in = new Scanner(System.in);
            int userin = in.nextInt();
            if (userin == 1)
            {...}

            else if (userin == 2)
            {...}

            else if (min == guess)
            {...}


            else
            {...}

        }

    }
}
```

```java
import java.util.Scanner;
class Guess
{
    public static void main(String []args)
    {

        System.out.println("Think of a number between 1 and 1000.");
        System.out.println("If I guess too low, enter '1', or too high, enter '2'.");
        System.out.println("If I guess right, enter any other number.");

        int min = 0;
        int max = 1000;

        int counter = 0;

        boolean stillguessing = true;

        while(stillguessing)
        {
            int guess = (min + max)/2;

            counter = counter + 1;

            System.out.println();
            System.out.println("Is your number "+ guess +"?");

            Scanner in = new Scanner(System.in);
            int userin = in.nextInt();
            if (userin == 1)
            {...}

            else if (userin ==2)
            {...}

            else if (min == guess)
            {...}


            else
            {...}

        }

    }
}
```

Conclusion

Eclipse was found to be a good tool for static analysis.

Implementation of a plagiarism detector was a good test of the abilities of Eclipse for this.

The plagiarism detector was not the best but it did find several plagiarised pairs.

Further Work

Further work with Eclipse for static analysis and also program transformations.

Evaluate extra tools such as Stratego.

Code obfuscator which outputs plagiarised programs to be used for testing a plagiarism detector (a good test for Eclipse's program transformation abilities).

Implementation of better program similarity algorithms for the plagiarism detector.

Questions?