TESIS – KI142502

KOMENTAR SEMI OTOMATIS UNTUK MEMUDAHKAN PEMAHAMAN PADA BAHASA PEMROGRAMAN JAVA

ANDY RACHMAN
5111201031

DOSEN PEMBIMBING
Dr.Ir. Siti Rochimah, MT
Dwi Sunaryono, S.Kom, M.Kom

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA
2017

[HALAMAN INI SENGAJA DIKOSONGKAN]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Andy Rachman
Nrp. 5111201031

Dengan judul :
KOMENTAR SEMI OTOMATIS UNTUK MEMUDAHKAN PEMAHAMAN PADA BAHASA PEMROGRAMAN JAVA

Tanggal Ujian : 26-7-2016
Periode Wisuda : Maret 2017

Disetujui oleh:

Dr. Ir. Siti Rochimah, M.T
NIP. 196810021994032001
(Pembimbing 1)

Dwi Sunaryono, S.Kom., M.Kom
NIP. 197205281997021001
(Pembimbing 2)

Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng
NIP. 197411232006041001
(Penguji 1)

Sarwosri, S.Kom, M.T
NIP. 197608092001122001
(Penguji 2)

Umi Laili Yuhana, S.Kom, M.Sc
NIP. 197906262005012002
(Penguji 3)

Direktur Program Pasca Sarjana,

an. Direktur Program Pascasarjana
Asisten Direktur

Prof. Dr. Ir. Tri Widjaja, M.Eng.
NIP. 19611021 198603 1 001

Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D.
NIP. 196012021987011001

KOMENTAR SEMI OTOMATIS UNTUK MEMUDAHKAN PEMAHAMAN PADA BAHASA PEMROGRAMAN JAVA

Nama Mahasiswa : Andy Rachman
NRP : 5111201031
Dosen Pembimbing : Dr. Ir. Siti Rochimah, M.T
Dwi Sunaryono, S.Kom.,M.Kom

ABSTRAK

Studi memperkirakan bahwa biaya perawatan perangkat lunak meningkat dengan sangat cepat hingga mencapai 90% dari biaya keseluruhan dalam daur hidup pengembangan perangkat lunak. Perawatan perangkat lunak menjadi sangat sulit dikarenakan tidak adanya dokumentasi pada program yang ada. Salah satu bentuk dokumentasi program adalah pemberian komentar program. Komentar sangat berguna dalam pemahaman program dan pemeliharaan program. Komentar memungkinkan pengembang dalam memahami kode lebih cepat dalam pembacaan kode program. Pemberian komentar program biasanya dilakukan manual oleh programmer. Pada tesis ini, dilakukan dua kegiatan pemberian komentar pada bahasa pemrograman java, yaitu pemberian komentar secara otomatis dan pemberian komentar secara semi otomatis. Komentar semi otomatis yang diberikan pada program secara langsung dapat mempermudah proses pemahaman pada program khususnya bahasa pemrograman java. Penggunaan ekspresi regular sangat membantu dalam mempolakan baris kode sumber. Aplikasi yang dibangun oleh peneliti telah mampu memberikan kontribusi pemahaman terhadap program sebesar 14.29% sampai dengan 42.86%

Kata Kunci : Komentar program, perawatan perangkat lunak, regular ekspresi.

[HALAMAN INI SENGAJA DIKOSONGKAN]

SEMI AUTOMATIC COMMENTS TO FACILITATE UNDERSTANDING ON JAVA PROGRAMMING LANGUAGE

Student Name : Andy Rachman
NRP : 5111201031
Supervisor : Dr. Ir. Siti Rochimah, M.T
Dwi Sunaryono, S.Kom.,M.Kom

ABSTRACT

The study estimates that the cost of software maintenance is increasing very rapidly until it reaches 90% of the overall cost of the software development life cycle. Software maintenance becomes very difficult due to the lack of documentation on the existing program. One form of program documentation was commenting program. Comments are very useful in understanding the program and maintenance program. The comment allows developers to understand the code faster in reading the program code. Commenting program is usually done manually by the programmer. In this thesis, carried out two activities commenting on the Java programming language, the automatic comment and semi-automatic comment. Semi-automatic comments given on direct program is expected to simplify the process of understanding the particular program java programming language. The use of regular expressions is helpful in patterns of lines of source code. Applications built by the researchers have been able to contribute to the understanding of the program at 14.29% to 42.86%.

Keywords: *code comments, software maintenance, regular expressions*.

[HALAMAN INI SENGAJA DIKOSONGKAN]

KATA PENGANTAR

Segala puji syukur kepada Allah SWT yang telah melimpahkan rahmat dan hidayah Nya sehingga penulis dapat menyelesaikan Tesis yang berjudul "Komentar Semi Otomatis Untuk Memudahkan Pemahaman Pada Bahasa Pemrograman Java" sesuai dengan target dan waktu yang diharapkan.

Proses pembuatan dan pengerjaan Tesis ini merupakan pengalaman yang sangat berharga bagi penulis untuk memperdalam ilmu pengetahuannya khususnya di bidang rekayasa perangkat lunak dan teknologi informasi. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada :

1. Allah SWT atas limpahan rahmat Nya sehingga penulis dapat menyelesaikan Tesis ini dengan baik.
2. Bapak dan Mama yang selalu memberikan motivasi dan dorongan untuk proses penyelesain tesis ini.
3. Istriku yang tersayang Sulistyowati, anak-anakku mas Naufal Rafif, kakak Dary Ananda dan mbak Aulia, kalian semua semangat ayah menempuh dan menyelesaikan jenjang Strata 2.
4. Ibu Dr. Ir Siti Rochimah, M.T dan Bapak Dwi Sunaryono, S.Kom.,M.Kom selaku Dosen Pembimbing yang selalu membantu dan membimbing penulis sampai dengan penyelesaian tesis ini.
5. Bapak Daniel Oranova Siahaan, S.Kom., M.Sc., PD.Eng, Ibu Sarwosri, S.Kom.,M.T, Ibu Umi Laili Yuhana, S.Kom.,M.Sc selaku dosen penguji telah memberikan bimbingan, arahan, nasehat dan koreksi dalam pengerjaan Tesis ini.
6. Bapak Waskitho Wibisono, S.Kom., M.Eng., PhD selaku ketua program Pascasarjana Teknik Informatika ITS yang selalu memberikan arahan dan dukungan untuk selalu menyelesaikan tesis ini.
7. Ibu Dr. Chastine Fatichah, S.Kom.,M.Kom yang selalu memberikan arahan dan dukungan untuk selalu menyelesaikan tesis ini.
8. Bapak, Ibu Dosen Pascasarjana Teknik Informatika ITS lainnya yang tidak sempat disebutkan, yang telah memberikan ilmunya kepada penulis

9. Mbak Lina, Mas Kunto dan segenap staf Tata Usaha yang telah
10. memberikan segala bantuan dan kemudahan kepada penulis selama
11. menjalani kuliah di Teknik Informatika ITS.
12. Teman-teman seperjuangan 2011, tim AGA, dan the last fighter of 2011 (mbak Ratih, mas Billy, bang Hengki)
13. Tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu disini yang telah membantu terselesaikannya Tesis ini

Penulis menyadari bahwa Tesis ini masih jauh dari kesempurnaan dan banyak kekurangan. Untuk itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari para pembaca.

Surabaya, Januari 2017

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[HALAMAN INI SENGAJA DIKOSONGKAN]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam beberapa tahun terakhir perkembangan kinerja perangkat keras komputer semakin meningkat dan canggih. Perangkat lunak menjadi kunci utama dalam kesuksesan sistem berbasikan komputer. Harga perangkat lunak menjadi sangat mahal bila dibandingkan dengan harga perangkat keras saat ini (Zhizhong Jiang and Peter Naude, 2007). Perangkat lunak sendiri dibagi menjadi dua bagian besar, yaitu perangkat lunak aplikasi dan perangkat lunak sistem (Andrew S. Tanenbaum, 2008). Perangkat lunak aplikasi dan perangkat lunak sistem telah dikenal dengan baik dan digunakan setiap hari oleh pengguna komputer. Dibutuhkan pendidikan khusus dan lulusan yang memadai dalam rekayasa perangkat lunak, dimana dengan keilmuannya ini mampu membangun aplikasi baru dan sistem baru dengan menggunakan pendekatan rekayasa perangkat lunak (Skhoukani and Abu Lail, 2012).

Evolusi perangkat lunak merupakan subyek dari berbagai penelitian, baik dibidang akademik ataupun bidang industri dan bagian utama pada pengembangan perangkat lunak saat ini adalah perawatan perangkat lunak (Palacios et.al., 2011). Perawatan perangkat lunak merupakan aktifitas yang sangat penting dalam rekayasa perangkat lunak(Chakraverti et.at.,2012).

Perawatan perangkat lunak sendiri saat ini menjadi fokus utama dalam pengembangan teknologi informasi. Saat ini perusahaan lebih berfokus pada pengembangan aplikasi yang ada daripada mengimplementasikan aplikasi yang baru (Chua and Verner, 2010). Studi memperkirakan bahwa biaya perawatan perangkat lunak meningkat dengan sangat cepat hingga mencapai 90% dari biaya keseluruhan dalam daur hidup pengembangan perangkat lunak (Al-Badareen et.al.,2011). Pengembangan perangkat lunak membutuhkan pekerjaan sangat besar yang didalamnya terdapat beberapa kekhawatiran atau pertimbangan bagi para pengembang dalam mengimplementasikan sebuah sistem perangkat lunak. Dengan mudularitas yang baik akan mengurangi kekhawatiran dalam

pengembangan perangkat lunak serta mengurangi kompleksitas perangkat lunak(Arora et.al.,2011).

Komentar sangat berguna dalam pemahaman program dan pemeliharaan program. Penggunaan komentar dalam sebuah program sering kali diabaikan, meskipun para pemrograman mengetahui fungsi dari komentar bagi pemahaman program. Membaca kode merupakan dasar selama pembuatan perangkat lunak dan sebuah program lebih sering dibaca dari pada ditulis. Komentar memungkinkan seseorang dalam memahami kode lebih cepat dalam pembacaan kode (Fluri et.al., 2007). Pada saat melakukan perubahan pada kode sumber, pengembang melakukan dua kegiatan yang berhubungan dengan komentar, yaitu pembaruan komentar yang berhubungan dengan kode sumber (pembaruan yang konsisten) dan pembaruan komentar yang tidak dibarengi dengan pembaruan kode sumber (pembaruan yang tidak konsisten). Komentar yang tidak terbarui akan menyebabkan pengembang mengalami kesalahan pelacakan dan kesalahan perangkat lunak (Ibrahim et al.,2012).

Saat ini, ketersediaan kode sumber bagi pengguna komputer menjadi sangat penting terutama sebagai pendukung dalam karya ilmiah. Dalam perjalannya, kode sumber yang ada saat ini tidak selalu terdapat dokumentasinya sehingga sangat susah dimengerti. Data yang tidak terdokumentasi akan menjadi sangat tidak berguna (Sajnani, 2012). Sebagian besar pengembang perangkat lunak menghabiskan banyak waktu dalam memeriksa kode sumber yang ada. Beberapa memperkirakan bahwa memahami kode sumber membutuhkan lebih dari separuh biaya keseluruhan perawatan perangkat lunak (Ishio et al.,2012). Kebutuhan mengidentifikasi kode sumber yang dapat menunjukkan karakteristik tertentu dari sebuah program sangatlah penting dalam memahami program (Noguera et al.,2012). Pemahaman program merupakan sebuah proses kognitif internal dimana hasilnya tidak dapat dilihat secara langsung, namun dengan memahami program secara umum dapat meningkatkan validasi percobaan dan membantu menafsirkan hasil yang ada (Feigenspan et al.,2012).

Pembuatan kode program membutuhkan pengalaman dan pemahaman dari para pengembang. Pemahaman pengembang terhadap kode program yang ada menjadi hal yang sangat utama (Scott Blinman and Andy Cockburn, 2005).

Beberapa informasi dapat digali pada sebuah kode program. Komentar program tidak hanya digunakan untuk menyatakan tujuan dari bagian program, tetapi juga menjelaskan cara kerja sebuah program (Annie Chen et.al.,2001). Pemberian komentar kode program sudah dilakukan oleh pengembang sejak lama. Komentar yang diberikan bersifat langsung, deskripsi, dan mudah untuk dipahami. Komentar menjadi praktek standar pada pengembangan perangkat lunak untuk meningkatkan pembacaan pada sebuah program (Lin Tan et.al.,2007).

1.2. Perumusan Masalah

Berdasarkan latar belakang diatas, maka penulis merumuskan beberapa permasalahan sebagai berikut.

1. Bagaimana memisahkan antara komentar dan yang bukan komentar bahasa pemrograan java menggunakan regular expression?
2. Bagaimana melakukan studi empiris pada pemahaman program berdasarkan komentar pada program ?
3. Bagaimana memberikan komentar baru pada sebuah program menggunakan regular expression?

1.3. Tujuan dan Manfaat Penelitian

Tujuan dari tesis ini adalah membangun sebuah perangkat lunak pemahaman program berdasarkan komentar pada program-program java dimana aplikasi yang dibangun mampu memetakan isi dari program yang termasuk komentar dan yang bukan komentar serta memberikan komentar otomatis pada porgram java. Program yang telah terpisah dari komentar akan dilakukan proses pemberian komentar tambahan. Manfaat dari tesis ini adalah dihasilkannya perangkat lunak pemahaman program yang mampu melakukan hal-hal sebagai berikut.

1. Memisahkan komentar dan yang bukan komentar.
2. Memberikan komentar tambahan secara otomatis pada program java.

1.4. Batasan Masalah

Adapun batasan masalah pada tesis ini adalah sebagai berikut.

1. Aplikasi yang digunakan pada tesis adalah aplikasi yang dibangun dari bahasa pemrograman Java.
2. Program yang digunakan didapatkan pada situs internet dan program-program praktikum.

1.5. Kontribusi

Kontribusi yang dilakukan pada tesis ini adalah sebagai berikut.

1. Memberikan komentar tambahan secara semi otomatis pada program yang ada menggunakan regular expression dan Komentar yang dihasilkan dalam bahasa Indonesia.

BAB II
KAJIAN PUSTAKA

2. 1. Pemahaman Program

Pemahaman program komputer merupakan salah satu bagian aktifitas rekayasa perangkat lunak. Pemahaman perangkat lunak dibutuhkan ketika seorang pemrogram melakukan kegiatan perawatan, penggunaan kembali, migrasi, perekayasaan, atau meningkatkan sistem perangkat lunak. Sejumlah besar penelitian telah dilakukan dalam upaya sebagai panduan dan dukungan perekayasa perangkat lunak dalam melakukan proses-proses yang berhubungan pengembangan perangkat lunak (O'Brien, 2003)

Semenjak perkembangan program komputer pertama kalinya, beberapa pendekatan tentang pengembangan perangkat lunak telah dikenalkan mulai dari bahasa asembly, bahasa pemrograman prosedural seperti ADA dan Fortran serta bahasa pemrograman berorientasikan obyek. Saat ini, pemrograman berorientasi obyek merupakan paradigma pemrograman baru. Beberapa penelitian menunjukkan bahwa efek positif dari pemrograman berorientasi obyek adalah pemahaman program (Feigenspan, 2011).

Pemahaman program merupakan faktor manusia yang sangat penting dalam ranah ilmu komputer. Memahami sebuah program merupakan kegiatan yang sangat penting bagi perawatan perangkat lunak dan dalam pengembangan perangkat lunak kegiatannya berkisar antara 50% sampai dengan 60%. Program yang mudah dipamahi dapat mengurangi waktu pengembangan perangkat lunak. Pada saat perawatan perangkat lunak, program yang mudah dipahamai dapat mengurangi pembiayaan sampai dengan 70% (Feigenspan et al.,2011).

Pemahaman program merupakan sebuah proses kognitif internal dimana tidak dapat dihasilnya tidak dapat dilihat secara langsung. Pemahaman program tidak hanya tergantung pada properti kode tetapi juga pengguna yang bekerja dengan kode sumber (Feigenspan and Apel et al., 2011).

Suksesnya perawatan program bergantung pada pemahaman program tersebut. Seorang pemrogram harus memiliki kemampuan dalam memahami

program yang ada dan langkah-langkahnya sehubungan dengan proses modifikasi yang dikerjakan serta akhiran program tanpa melakukan kesalahan. Dibutuhkan pengetahuan yang lebih tentang strategi selama proses perawatan sampai dengan pemahaman program, penggunaan informasi sumber yang selalu berubah, serta dampak paradigma pemrograman sehubungan dengan pemahaman program selama kegiatan perawatan dilaksanakan. Isu-isu penting ini sangat perlu diperhatikan semenjak pemahaman program menjadi kunci utama dalam melakukan proses perawatan perangkat lunak (Corritore and Wiedenbeck, 2001).

Pemahaman program merupakan pekerjaan yang sangat kompleks. Para perekayasa perangkat lunak harus memeriksa dua hal yaitu aspek terstruktur dari kode sumber (misalnya sintak bahasa pemrograman) dan masalah utama yang sering terjadi (misalnya komentar, dokumentasi dan penamaan variabel) sehubungan dengan ekstraksi kembali informasi yang dibutuhkan dalam pemahaman seluruh bagian dari sistem perangkat lunak (Maletic and Marcus, 2001).

Pemahaman program merupakan pekerjaan yang kritis dalam sebuah organisasi untuk dua alasan utama, yaitu terjadinya perubahan pemrograman secara berkala dan orang-orang yang secara teratur bergabung dalam sebuah proyek, dimana orang-orang baru ini harus dapat memahami bagian-bagian program yang telah dikerjakan oleh pemrogram senior dan yang kedua adalah sebagian besar program yang ada dalam prosesnya orang-orang yang baru tidak dilibatkan dalam membuat program sehingga diharuskan melakukan pembuatan program mulai dari dasar. Kegiatan pembuatan program ini salah satunya terdapat pada pemahaman program (Ramalingam and Wiedenbeck, 1997).

Perusahaan teknologi informasi membutuhkan dana yang sangat besar untuk perawatan perangkat lunak. Hal ini dikarenakan kode sumber lebih sering dibaca daripada ditulis, elemen penting dalam perawatan adalah pemahaman kode sumber (Frey et al.,2011). Pemahaman program terjadi sebelum kode berubah, karena pengembang harus mencari kode sumber dan artifak lainnya dalam menemukan dan memahami bagian dari kode yang sesuai dengan perubahan yang dimaksud (Roehm et al., 2012). Sehubungan dengan pemahaman program, alat bantu merupakan kebutuhan yang sangat penting dan dapat membantu dalam

proses-proses pemahaman program. Alat bantu yang baik dan berguna adalah alat bantu yang dapat mendukung kegiatan pengguna (Knight and Munro, 2002).

Pemahaman program dimulai dengan kode sumber dan diakhiri dengan model mental yang menjelaskan tujuan kode yang ada, operasi dan abstraksi. Untuk membangun model ini, pemrogram harus memeriksa teks kode sumber. Terdapat dua pendekatan yang biasa digunakan, yaitu *top-down* dan *bottom-up* (Fisher et al.,2006).

2.1.1. Pendekatan top-down

Pemahaman dengan menggunakan pendekatan *top-down* dimana pemrogram mengkaji kegunaan domain dari program dan memetakannya kedalam kode sebenarnya (O'Brien and Buckley, 2001). Pendekatan ini meningkatkan dekomposisi masalah dalam bagian masalah dan menyelesaikan bagian masalah. Pendekatan *top-down* memberikan solusi yang dimulai dari masalah yang ada, lalu melakukan penguraian tujuan dari program yang ada kedalam bagian-bagian tujuan yang diinginkan dan mengimplementasikan bagian-bagian tujuan tersebut. Pendekatan ini terdiri dari hipotesis program lalu validasi hipotesis yang ada menggunakan komponen-komponen dari program (Sharma et al.,2012).

Pendekatan *top-down* digunakan jika pemrogram telah mengetahui terlebih dahulu domain program yang ada (Siegmund, 2012). Misalnya akan dibuat program kalkulator bagi anak Sekolah Dasar. Di sini, pemrogram telah mengetahui permasalahan yang ada, setelah itu dilanjutkan dengan mendesain logika yang ada dan diakhiri dengan membuat program yang sesuai dengan domain permasalahan. Pendekatan *top-down* akan dilaksanakan jika memang keadaannya memungkinkan (Feigenspan et al., 2011).

2.1.2. Pendekatan bottom-up

Pendekatan *bottom-up* adalah suatu proses pemahaman yang dimulai dari kode sumber dan berurutan, artifak perangkat lunak dikelompokkan ke dalam level abstraksi yang lebih tinggi (Denis Pinheiro et.al, 2008). Pada penelitian lainnya, pemahaman program dengan menggunakan pendekatan *bottom-up* dapat diartikan sebagai

deskripsi kode sumber perbaris kode sampai mengarah ke pemahaman maksud dari program yang ada, disini ditandai dengan pembangunan pengetahuan dan pemahaman pada program yang ada menjadi sebuah kesimpulan (Michael P. O'Brien, Teresa M et.al.,2001).

Pelacakan program menghubungkan antara kode dan informasi sumber lain yang berguna dalam membantu proses analisis dari berbagai informasi yang ada dan pada akhirnya membangun model perangkat lunak yang ada. Sehubungan dengan pelacakan program, pemahaman *bottom-up* membantu pemrogram dalam memahami kode sumber yang ada sampai dengan menjadi sebuah konsep (G. Antoniol et.al., 2000).

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworldp;

/**
 *
 * @author andy
 */
public class HelloWorldP {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int i,batas;

        i=1;batas=10;
        for (i=1;i<=10;i++)
        {
            System.out.print(i+"=");
            System.out.println("SELAMAT BELAJAR");
        }
    }
}
```

Gambar 2.1. Program HelloWorldP.

Pada gambar diatas menunjukkan tentang program HelloWorldP. Pemahaman program dengan menggunakan pendekatan *bottom-up*, maka telah tersedia program seperti pada gambar diatas, lalu pengembang akan dihadapkan pada pemahaman tentang program tersebut dan pengembang harus memahami maksud dari program tersebut serta cara kerja dan keluaran dari program tersebut.

2. 2. EKSPRESI REGULAR

Ekspresi Regular untuk selanjutnya disebut dengan REGEX, merupakan cara yang digunakan untuk pencocokan string pada sebuah teks, seperti karakter, kata ataupun model dari sebuah katakter. Sebuah REGEX ditulis dalam bahasa formal, beberapa contoh penggunaan REGEX antara lain : validasi email, pencarian file dan penggantian kata (Rashmi Sinha and Ashish Dewangan, 2012). REGEX mampu memberikan kemudahan fleksibilitas dan sangat efisien dalam pemrosesan teks. Pustaka pada REGEX mendukung proses pencarian dalam basis data REGEX. Penandaan/ekspresin REGEX ditandai dan ditutup dengan tanda *slash* (‘ / ‘) (Saurabh Jain et.al, 2012), misalnya /[A-Za-z0-9]/.

REGEX diperkenalkan pertama kali oleh Klenee pada tahun 1956. Sejak saat itu REGEX menjadi pusat ilmu komputer baik secara teoritis ataupun terapannya. Saat ini hampir setiap bahasa pemrograman modern memberikan kemampuan REGEX dalam bentuk pustaka masing-masing (Dominik D.F, 2011).

Dominik D.F and Timo Kötzing pada penelitiannya menggunakan REGEX pada dokumen web yang berbasiskan XML dimana REGEX dikenakan pada XML Schema Documents (XSDs) dan digunakan sebagai Document Type Definitions (DTD) (Dominik D.F and Timo Kötzing, 2012). Pada penelitiannya Dominik D.F dan Timo Kötzing menambahkan kemampuan REGEX menjadi Single Occurrence Regular Expressions (SORES) dan Chain Regular Expressions (CHARES) menjadi lebih optimal.

Sindu dan Prasanna pada penelitiannya menggunakan menggunakan REGEX untuk proses pencocokkan pada arsitektur Regular Expression NFA (RE-NFA) dalam FCPGA. Pada penelitian yang lain dengan topik yang sama Yang dan Prasanna meningkatkan arsitektur yag telah dilakukan oleh Sindu dan

Prasanna dimana pada penelitiannya Yang dan Prasanna melakukan prosesnya dengan cara 2 karakter per clock cycle dan menghasilkan troughput 14.4 Gbps untuk 760 pencocokkan REGEX pada FPGA (Cui Linhai, 2014).

REGEX sendiri merupakan sebuah implementasi dari sebuah pencocokkan string berdasarkan suatu pola tertentu. Pada REGEX proses pencarian teks yang panjang akan lebih sederhana dengan menerapkan sebuah pola yang ditetapkan. REGEX terdiri dari string yang membentuk kombinasi karakter normal, *metacharacter* tertentu dan karakter *metasequence* (Tony Stubblebine,2007) Beberapa bahasa pemrograman saat ini yang mendukung fasilitas REGEX ini antara lain .NET, java, javascript, XRexExp, PCRE, Perl, Python, dan Ruby. Fungsi REGEX pada masing-masing bahasa pemrograman terdapat pada paket ataupun utilitas dari masing-masing bahasa pemrograman. Untuk .NET agar dapat menjalankan REGEX diperlukan paket *System.Text.RegularExpression* (Jan Goyvaerts and Steven Levithan, 2012), bahasa pemrograman Java agar dapat menjalankan REGEX diperlukan paket *java.util.REGEX*. Pada javascript, REGEX mengikuti aturan standar ECMA-262. ECMA singkatan dari European Computer Manufactures Association. ECMA merupakan sebuah asosiasi industri internasional yang didirikan tahun 1961 yang berkontribusi untuk standarisasi sistem informasi dan komunikasi dunia. ECMA-262 merupakan standarisasi skrip untuk implementasi JavaScript dan JScript di semua web browser (ECMA, 2011).

2.3.1. Pencocokkan Pola

Pola merupakan suatu bentuk tertentu yang mempunyai aturan tertentu pula (Jie Liu et.al.,2006) Pembentukan pola untuk menyelesaikan masalah tertentu sangatlah membantu proses penyelesaian masalah. Pola kalimat (string) dapat digunakan sebagai representasi ringkas dari sebuah himpunan untuk menyamarkan tugas dari formula yang ada (Guilermo et.al.,2015).

Pencocokkan pola pada REGEX terdapat empat cara yaitu *string literal*, *digits*, *letters* dan sembarang karakter. String literal merupakan kumpulan dari karakter yang membentuk sebuah kalimat tetapi tipe data yang digunakan untuk menyimpan bukan bertipe string tetapi class. Jika sebuah data bertipe string akan disimpan pada variabel bertipe String, misalnya jika pada sebuah program

dituliskan X^2 akan bermakna berapapun nilai X akan dipangkatkan 2 yaitu jika X=10, maka X^2 adalah 10*10 sama dengan 100, tetapi jika kita pada sebuah program kita akan mencari sebuah kalimat yang berisi X^2 tetapi X^2 bukan rumus matematika, maka pada REGEX kita menggunakan string literal dimana untuk membedakan tanda ^ dengan "^" adalah dengan menambahkan tanda "\" didepan "^"., sehingga dalam REGEX penulisannya menjadi X\^2 (Yunyao Li et.al., 2008).

2.3.2. Wildcard

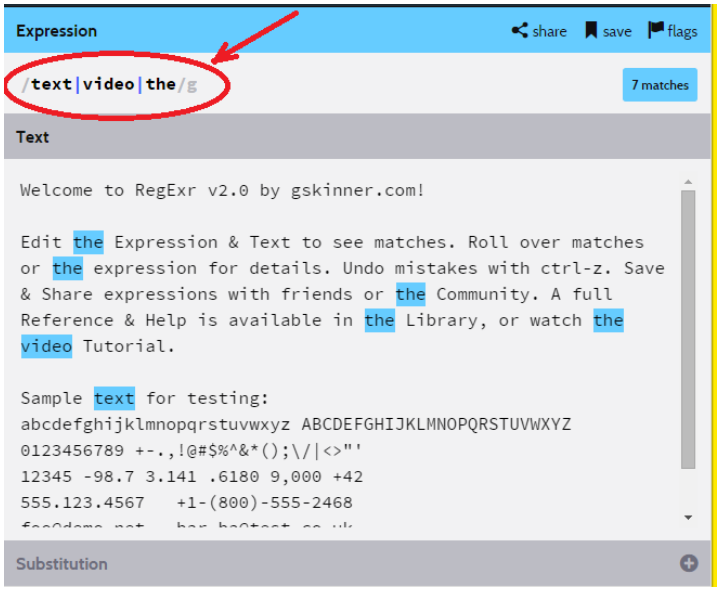
Pencocokan pola ganda dengan menggunakan wildcard sangatlah membantu dan penting. Penelitian-penelitian seperti sistem temu kembali informasi, bioinformatics, dan pengindeksan teks sangatlah membutuhkan keberadaan wildcard ini (Qiang et.al, 2013). *Wildcard* merupakan sebuah bentuk primitif dari REGEX dan banyak digunakan pada DOS dan Linux shell. Sedangkan pada lingkungan Unix, *wildcard* lebih dikenal dengan istilah *globbing*. *Wildcard* adalah sebuah string pola yang digunakan untuk mencocokkan sekumpulan file atau direktori. Dengan menggunakan *wildcard* maka proses pencocokkan akan dapat dilakukan dengan lebih sederhana (Koteswara et.al., 2011)(Sandeep Kumar S et.al., 2010). Misalnya untuk mencari sembarang kata dengan kombinasi kata depan e atau s atau es dengan sembarang kata belakang s, maka dapat dituliskan hanya dengan satu perintah tunggal pada REGEX : es*s.

Gambar 2.2. Pencarian kata dengan REGEX untuk kombinasi es*s.

2.3.3. Karakter Meta

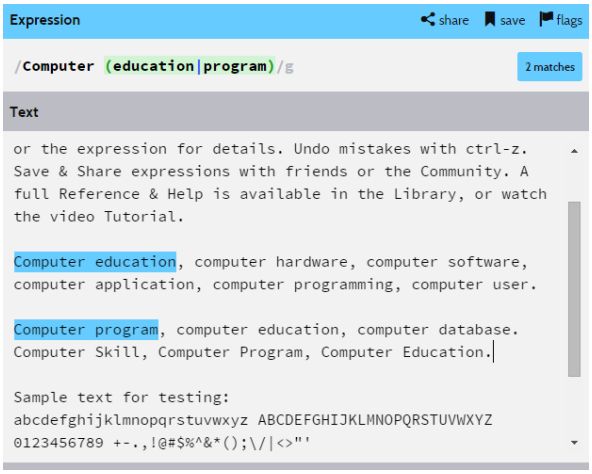
Metacharacter untuk selanjutnya disebut dengan karakter meta, merupakan kombinasi khusus dari alfanumerik dan atau karakter simbol yang mempunyai maksud-maksud tertentu dalam REGEX (Mark Tabladillo, 2012). REGEX memiliki karakter khusus, seperti *+?[{\.(^$ masing-masing karakter tersebut mempunyai fungsi-fungsi khusus.

Karakter meta untuk pemilihan pada REGEX adalah " | ". Karakter meta ini digunakan untuk menemukan beberapa data secara langsung. Jika didalam pembuatan program adalah if ., misal akan dicari data "text, atau video, atau the", maka penulisan REGEXnya dapat dilakukan seperti berikut : text|video|the. Dari penulisan REGEX ini akan secara otomatis mencari data "text", "video" atau "the" pada sebuah dokumen. Hasil dari REGEX tersebut seperti pada gambar 2.3.



Gambar 2.3. REGEX untuk mencari data text, video atau dokumen.

Karakter meta pada REGEX yang digunakan untuk melakukan proses pengelompokkan adalah diawali dengan tanda "(" dan diakhiri dengan tanda ")". Pada saat karakter meta ini digunakan maka komputer secara otomatis akan mencari kelompok kata yang dibutuhkan, misalnya akan dicara data tentang "Computer education" atau "Computer program", maka pada REGEX akan dituliskan "Computer (education|program)", dengan REGEX ini hasil dapat dilihat pada gambar 2.4.



Gambar 2.4. Pencarian data dengan menggunakan pengelompokkan REGEX.

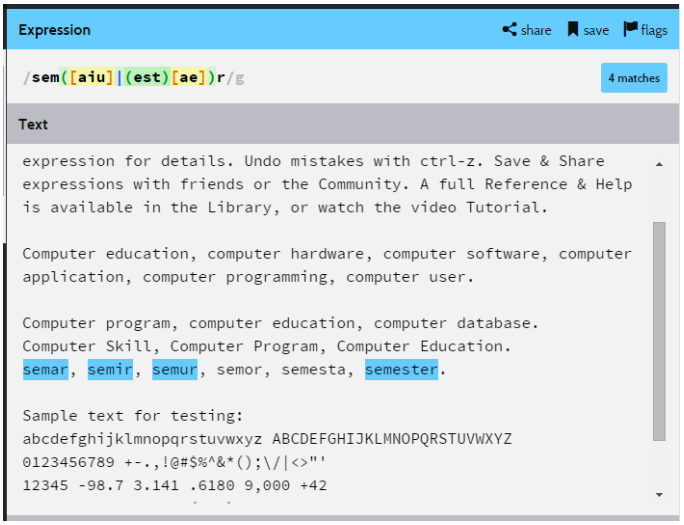
Pada REGEX juga terdapat cara untuk membentuk karakter set. Karakter set ini diawali dengan tanda "[" dan diakhiri dengan tanda "]". Karakter set ini fungsinya mirip dengan karakter meta "|". Perbedaan karakter ini adalah pada karakter meta "|" hanya menampilkan sesuai dengan yang dicari sedangkan karakter set "[" "]" dapat berisi rentang nilai dan negasi (Brent Welch, 1999). Misalnya akan mencari nilai dari a-z, maka karakter set yang dapat dituliskan adalah [a-z].

Gambar 2.5. REGEX pencarian data huruf kecil a-z

Untuk menampilkan huruf kapital dari A-Z menggunakan REGEX dapat dilakukan dengan cara yang pertama [A-Z] seperti terlihat pada gambar 2.6.

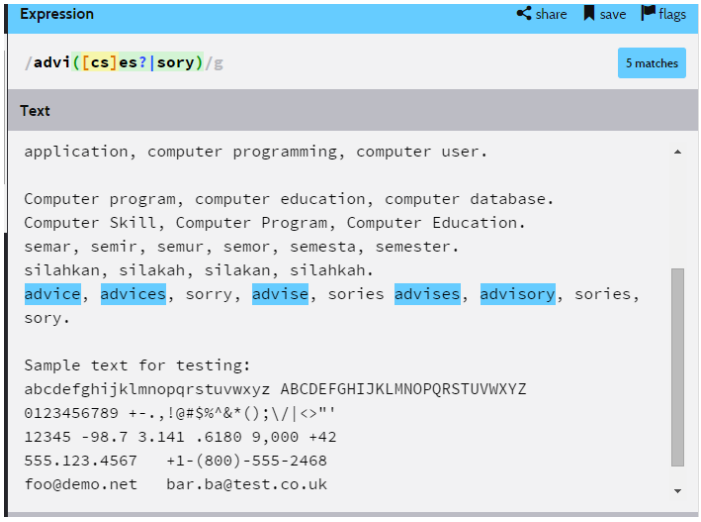
Gambar 2.6. REGEX pencarian data kapital dari A-Z

Sedangkan untuk mencari data semar, semir, ataupun semur, semestar atau semester pada sebuah dokumen dengan menggunakan REGEX dapat dilakukan dengan kombinasi pengelompokkan dan karakterset, yaitu sem([aiu])r seperti tampak pada gambar 2.7.



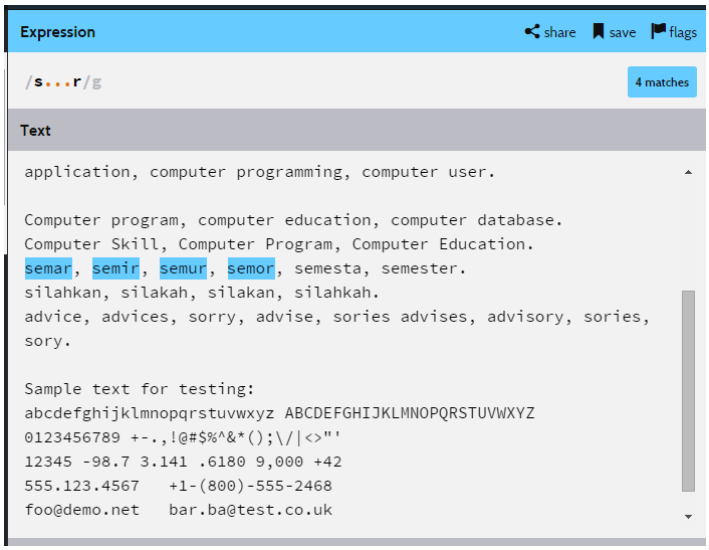
Gambar 2.7. REGEX pencarian data tertentu pada dokumen.

Karakter meta pada REGEX untuk pemilihan bileh ada atau tidak (optional) diwakili oleh simbol ? (tanda tanya). Karakter meta ini mempunyai arti segala sesuatu yang berada pada sisi kiri tanda ? boleh ada ataupun tidak. Misalkan terdapat REGEX dengan model advi([cs]es?|sory), maka hasilnya akan tampak seperti gambar 2.8.



Gambar 2.8. Penggunaan REGEX karakter meta optional pada dokumen

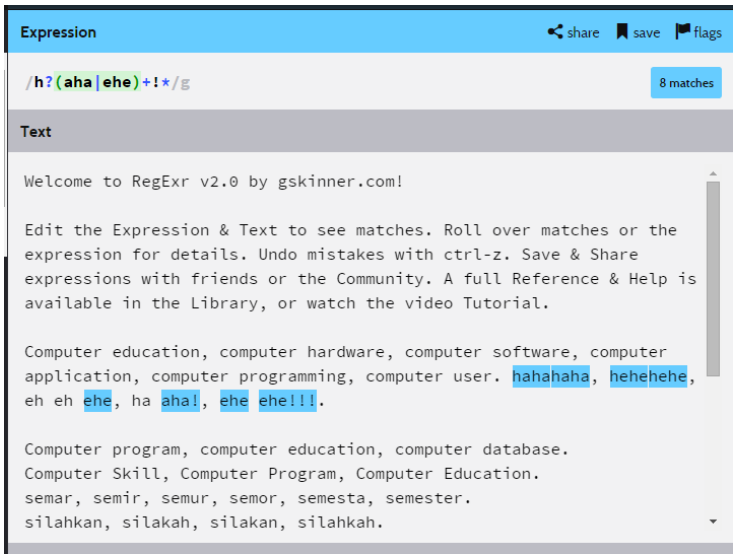
Karakter meta dot atau titik pada REGEX digunakan untuk semua karakter tunggal. Proses pencarian kata semar, semir, semur, semor akan dapat dilakukan dengan mudah. REGEX untuk pencarian kata tersebut diatas adalah s...r. Hasil dari proses REGEX tersebut tampak seperti pada gambar 2.9.



Gambar 2.9. Penggunaan karakter meta dot untuk pencarian karakter tunggal.

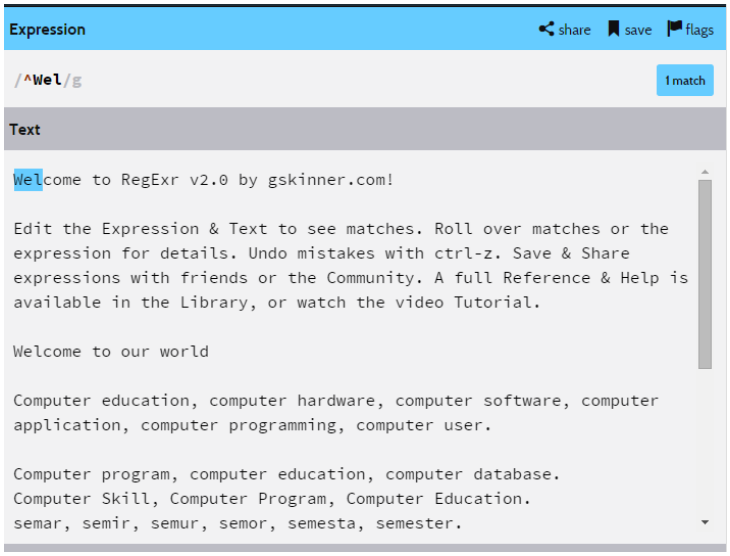
Karakter meta pada REGEX untuk proses perulangan diwakili oleh * atau +. Karakter meta ini digunakan pada saat diinginkan mencari data berulang misalnya hahaha, hehehe dan lainnya. Misalkan akan didapatkan kombinasi data untuk kata aha ataupun ehe meskipun kata tersebut terdapat tanda ! maka REGEX

dapat dituliskan h?(aha|ehe)+!*. Hasil dari REGEX tersebut dapat dilihat pada gambar 2.10.



Gambar 2.10. Pencarian data berulang pada REGEX

Karakter meta jangkar diwakili oleh simbol ^ dan $. Karakter meta ini tidak seperti karakter meta lainnya yang mempunyai fungsi khusus. Karakter meta ini digunakan untuk menandai/mencocokkan posisi sebelum, setelah atau diantara karakter yang diinginkan. Karakter meta ^ dinamakan caret sedangkan karakter meta $ dinamakan dolar. Karakter meta caret digunakan untuk mendapatkan awal dari sebuah paragraf kalimat sesuai kebutuhan yang diinginkan. Pada saat digunakan karakter meta ini maka pada sebuah dokumen akan dicek apakah baris pertama pada dokumen mempunyai kesamaan dengan kondisi yang dimasukkan pada REGEX. Jika ada maka akan ditemukan. Hal ini dapat dilihat pada gambar 2.11. Pada gambar tersebut, meskipun terdapat ada dua kata Welcome tetapi yang diblok hanya kata pertama pada kalimat awal, sedangkan kalimat berikutnya tidak ditemukan. Tidak seperti penulisan langsung pada REGEX misalnya Wel maka semua yang mengandung kata Wel akan ditemukan.



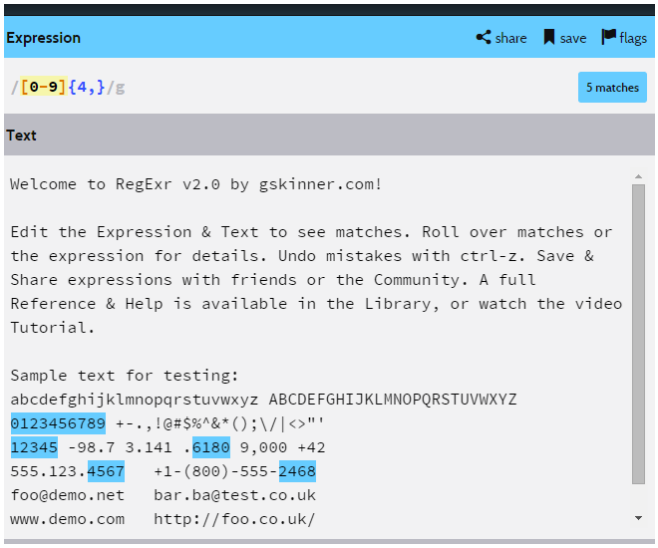
Gambar 2.11. Pencarian data menggunakan caret pada REGEX

REGEX mendukung pencarian data dengan rentang tertentu. Karakter meta yang digunakan adalah karakter meta quantifier. Karakter meta ini digunakan untuk menyatakan rentang atau jumlah karakter yang diperbolehkan dari suatu pola. Pola pada quantifier memiliki fungsi yang berbeda sehubungan dengan penulisannya. Adapun aturan penulisan dan fungsi dari karakter meta quantifier ini adalah sebagai berikut :

1. X{m} artinya set aturan X harus ada sebanyak m kali. Misalnya X=[0-9], m=4, sehingga REGEXnya adalah [0-9]{4}. Hal ini berarti dapatkan data numerik dari 0-9 yang terdiri dari 4 digit atau dapatkan data numerik yang ada dan blok per empat digit. Hal ini dapat dilihat pada gambar 2.12.

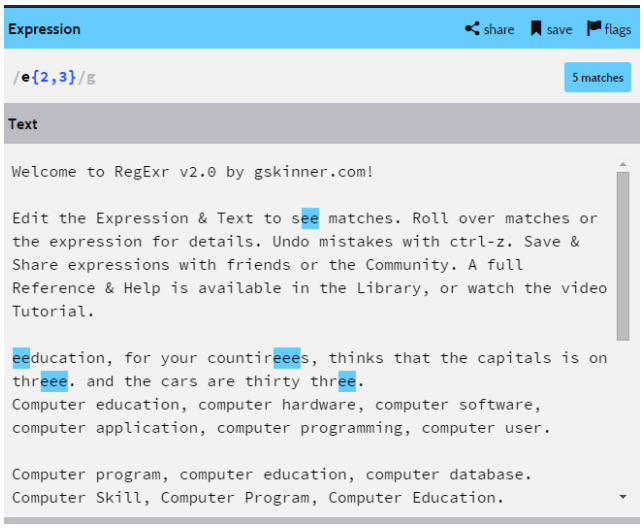
Gambar 2.12. REGEX quantifier dengan kondisi X{m}

2. X{m,} artinya set aturan X harus ada minimal sebanyak m kali. Pada model ini data yang dicari sama dengan kondisi no.1. tetapi juga jika ditemukan data maka data yang keberapapun dibelakan data kemempat akan diblok atau ditemukan tidak seperti kondisi no.1 dimana pada kondisi no.1. jika dibelakang data keempat terdapat nilai lain, maka akan dipotong per empat digit. Hasil REGEX untuk kondisi X{m,} seperti pada gambar 2.13.



Gambar 2.13. REGEX quantifier untuk kondisi X{m,}

3. X{m,n} artinya set aturan X boleh ada dari minimal m buah hingga terulang sebanyak maksimal n buah. Pada model ini, data yang dicari mempunyai nilai minimal sejumlah m dan maksimal n.



Gambar 2.14. REGEX quantifier untuk X{m.n}

Dari gambar 2.14. dapat diketahui bahwa REGEX e{2,3} dimaksudkan untuk mencari data yang mengandung nilai e dimana nilai e nya minimal 2 runtutan dan maksimal 3 runtutan, yaitu ee atau eee.

2.3.4. Karakter Escape.

Karakter escape merupakan karakter khusus yang digunakan untuk merepresentasikan alternatif dari karakter yang ada. Karakter escape merupakan kasus tertentu pada karakter meta. Karakter escape pada REGEX ditandai dengan slash (‘\’) (Ezekiel O and Maduka A, 2010).

Gambar 2.15. Penggunaan Karakter escape pada REGEX

Tabel 1. Karakter escape

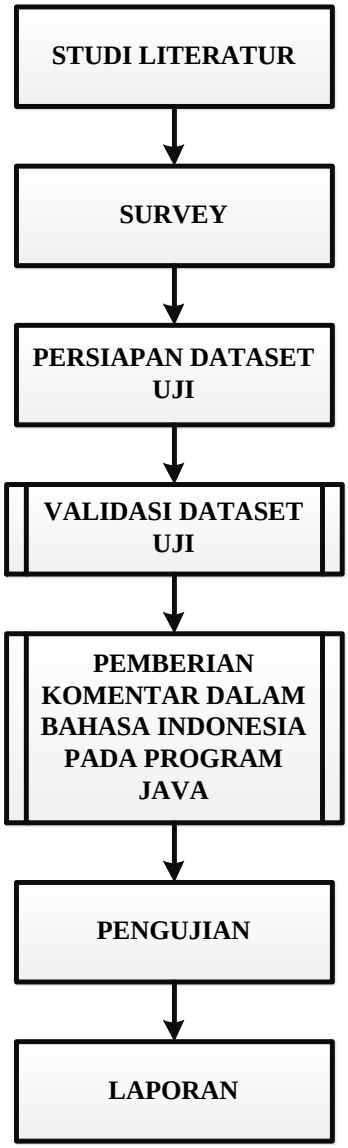
KODE	KETERANGAN
\w	Karakter kata
\W	Non Karakter kata
\d	Karakter digit
\D	Non Karakter digit
\s	Karakter Whitespace
\’	Karakter petik tunggal
\\	Karakter blackslash
\”	Karakter petik ganda

[HALAMAN INI SENGAJA DIKOSONGKAN]

BAB III
METODE PENELITIAN

Metode penelitan yang dilakukan pada thesis ini terbagi menjadi 4 bagian besar, yaitu Input, data, validasi data, pemberian komentar bahasa Indonesia pada program java dan

Pada bagian ini, penulis dalam menyelesaikan thesis ini melakukan kegiatan tahapan penelitian mulai dari analisis, desain, pembuatan program, pengujian, implementasi dan penyusunan laporan akhir.



Gambar 3.1. Metode Penelitian Penyelesaian thesis

3.1. Studi Literatur

Pada tahapan analisis, penulis melakukan kegiatan studi literatur terhadap penelitian-penelitian yang telah dilakukan sebeumnya dan survey pada programmer java dan programmer web.

3.1.1. Studi Literatur

Penelitian terdahulu yang dilakukan oleh Oliver Arafat dan Dirk Riehle dengan judul "*The Comment Density of Open Source Software Code*" tahun 2009 meneliti tentang kerapatan komentar pada peragngkat lunak berbasiskan sumber terbuka. Peneliti melakukan uji terhadap 5.229 proyek berbasiskan sumber terbuka. Peneliti melakukan pengecekkan terhadap *Source Line Of Code* (SLOC), *Comment Line* (CL), *Line of Code* (LoC), *Commit Size*, dan *Commit Density*. SLOC merupakan baris pada program yang berisi kode sumber. CL merupakan baris program yang berisi komentar. LoC merupakan baris dari program selain kode sumber ataupun komentar. *Commit Size* merupakan perubahan pada kode sumber, mulai dari penjumlahan, pengurangan ataupun perubahan. *Comment Density* merupakan file atau kelompok file atau keseluruhan kode sumber dari sebuah proyek didapat dari jumlah keseluruhan komentar dibagi dengan jumlah baris kode program. Dari hasil penelitian didapatkan bahwa:

1. Comment Density yang didapat rata-rata adalah 19%, artiya dalam 5 baris kode program berisi 1 komentar.
2. Hubungan antara *Project Size* dan *Comment Density*, adalah sebesar -7.9%, yang artinya *Project Size* tidak berhubungan dengan jumlah komentar yang ada (*Comment Density*).
3. Hubungan antara *Team Size* dan *Comment Density*, adalah sebesar 2.55% , yang artinya *team size* dan *comment density* tidak saling berhubungan.
4. Hubungan antara *Project Age* dan *Comment Density*, adalah -90.54%, yang berarti *comment density* selalu berkurang sejalan dengan *project age* (umur perangkat lunak).

Pada penelitiannya, Ninus Khamis et.al., dengan judul "*Automatic Quality Assessment of Source Code Comments: The JavadocMiner*" tahun 2010 meneliti tentang penilaian secara otomatis dari Komentar Kode Sumber pada Javadoc. Pada penelitiannya, peneliti menemukan *identifier* pada ArgoUML sebanyak

43.025 dan komentar sebanyak 20.941, sedangkan pada ECLIPSE ditemukan identifier sebanyak 487.192 dan komentar sebanyak 100.451 buah. Dari hasil penelitiannya didapatkan bahwa daerah-daerah yang sering menjadi masalah dapat diminimalisasi dengan menjaga hubungan antara komentar kode sumber dengan perubahan yang ada pada kode sumber.

Paper ketiga adalah dari peneliti Yahya Tashtoush et.al.,2013, dengan judul "*Impact of programming features on Code Readability*". Pada penelitiannya, peneliti menggunakan metode IPCFR (*Impact of Programming Features on Code Readability*). IPCFR digunakan untuk menguji berbagai fitur program dan efeknya pada pembacaan kembali kode program. Peneliti menggunakan SPSS untuk mengolah data yang didapat. Dari uji program, hal yang dapat dibantu pada pembacaan kembali kode program antara lain: maksud dari penamaan, konsistensi dan komentar program.

3.1.2. Survey

Survey awal dilakukan oleh penulis kepada para programmer yang tergabung dalam Asosiasi Programmer Indonesia (Aprogsi) dengan alamat facebook yaitu https://www.facebook.com/groups/aprogsi/ dan dua programmer diluar Aprogsi dengan total responden 12 programmer. Adapun tabel hasil survey seperti pada tabel 2.

Tabel 2. Hasil Survey Lokasi Komentar pada suatu Program

RESPONDEN KE	RINGKASAN SURVEY
1	Tanggal Aplikasi dibuat
	Output dari Fungsi
	Pengambilan data dari variabel mana
	Keterkaitan Function / procedure dg function / procedure lain.
2	Buat Pseudo-code dalam bentuk komentar
	// class: auth service
	// method: authenticate user
	// get user from db by calling user repository
	// hash given password with stored salt
	// if hashed pwd == stored pwd,
	// valid. Create session
	// else invalid. Return error
	//
	// method: bla.. bla..

3	Sebelum membuat Function dikasih komentar dulu. Isi Komentar
	1. Tgl dibuat
	2. Pembuat
	3. Cara Penggunaan
4	Sebelum membuat Function dikasih komentar dulu. Isi Komentar
	1. Tgl dibuat
	2. Pembuat
	3. Cara Penggunaan
5	Komentar ditambahkan setelah pembuatan Function
	Komentar berdasarkan PHPDoc yang mirip dengan JAvADoc
6	Komentar sebagai Pemerjelas
	diletakkan di function
	diletakkan di baris penting
7	diletakkan diatas atau sebelum suatu script dijalankan
	pemberian komentar disesuaikan dengan doxygen
8	Format dibuat berdasarkan model JavaDoc
	Pemberian Komentar di Class, Method atu Function dari program
9	Komentar ditulis diatas pendefinisian Class, Method/Function dan diatas Propoerty / atribut class
	Jika Perlu ditambahkan Inline Comment sebelum Sintaks tertentu
10	Tidak perlu disetiap baris karena hal ini sama dengan belajar bahasa tingkat pemula. Dari pernyataan ini dapat disimpulkan bahwa komentar setiap baris bukannya tidak perlu tetapi jika diberikan setiap baris maka sama dengan Pemberian Komentar Untuk Programmer Tingkat Pemula
	Komentar diberikan pada dibeberapa baris terutama untuk fungsi tertentu.
	Sebelum nama Class/Method/Nama File, terutama nama class/method yang masih belum mencerminkan fungsi dari class/method
11	Komentar ditulis diatas setiap fungsi/prosedur program
	Komentar jika editor bisa otomatis, maka otomatis, jika manual di buat sendiri secara manual
12	Komentar diletakkan dibagian atas jika komentar panjang
	Komentar diletakkan disamping jika digunakan untuk penandaan saja.

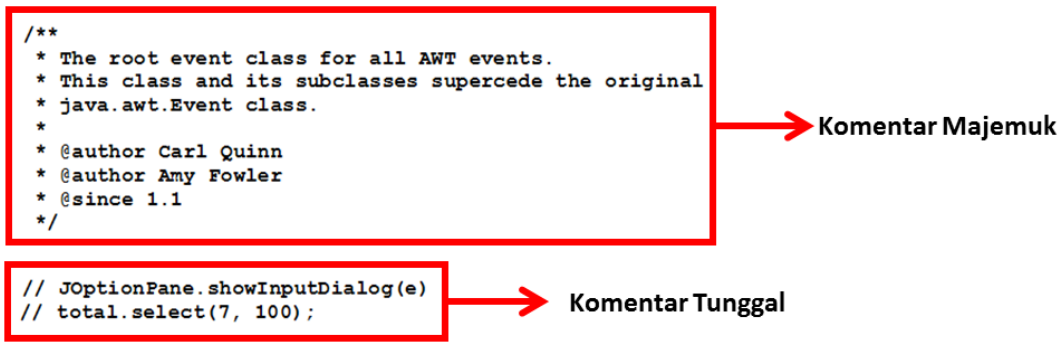
Dari tabel 2 diatas dapat diketahui bahwa pemberian komentar pada sebuah program menurut para programmer adalah sebagai berikut :

1. Komentar diletakkan pada bagian Judul Program dan berisi nama pembuat, tanggal dibuat serta fungsi utama dari program yang ada.
2. Komentar bisa ditulis di setiap method atau class atau function dari sebuah program.
3. Komentar kadang ditulis pada baris-baris tertentu yang memang mempunyai arti.

4. Untuk programmer tingkat mahir, komentar tidak perlu ditulis disetiap baris program tetapi untuk tingkat pemula komentar dapat dituliskan disetiap baris karena dapat membantu programmer tingkat pemula untuk memahami program yang ada.

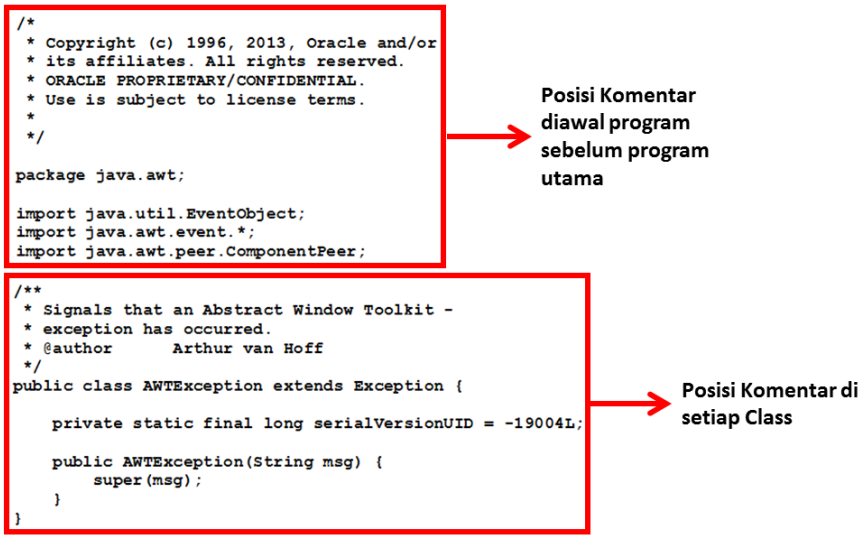
3.1.3. Komentar Program

Komentar kode program pada bahasa pemrograman java ditandai dengan '//' untuk komentar tunggal atau perbaris, dan untuk komentar majemuk atau beberapa baris diawali dengan tanda '**/***' dan diakhiri dengan tanda '***/**'.



Gambar 3.2. Dua jenis komentar pada bahasa pemrograman java

Dari proses studi literatur yang dilakukan penulis, posisi komentar pada program terbagi menjadi lima posisi, yaitu diawal program sebelum program utama, disetiap class, disetiap method, ditiap baris yang membutuhkan kode dan dibelakang baris pada setiap program yang ada (penting). Hal ini dapat dilihat seperti pada gambar 3.3.



Gambar 3.3. Posisi komentar pada kode program

Setelah mempolakan masing-masing proses pada bahasa pemrograman java, proses elanjutnya adalah memberikan informasi tentang pola tersebut.

Tabel 3. Pemberian pola pada program java untuk komentar tambahan

JENIS Program Java	Arti / Maksud
Import <>.???	Penggunaan pustaka <nama pustaka>
System.out.println();	Cetak Informasi di Layar Monitor <Informasi>
Public class <nama class>	Terdapat Klas <nama class> dengan fungsi <nama fungsi>
Public static int/float/string <nama> <tipe method>	Terdapat method dengan nama <nama> dengan tipe data <tipe method> denga modifier <nama public>
For (inisialisasi,kondisi, modifier)	Ulang <inisialisasi> sebanyak <kondisi>
While (kondisi) { statement; }	Selama <kondisi> kerjakan <statement>
Do { statement; } while (kondisi)	Kerjakan <statement> selama <kondisi>
If <kondisi> {	Jika <kondisi> maka
Else	Jika tidak maka

3.2. Regular Expressions (REGEX)

Regular Expression pada thesis ini digunakan untuk memisahkan program dan komentar yang ada pada sebuah kode sumber dan mempolakan program java. Pada thesis ini, Regex digunakan untuk mempolakan program java. Regex pada penelitian ini dapat menangani:

1. Pengenalan Komentar

   Untuk mengenali komentar yang ada pada kode sumber java, kode Regex digunakan untuk mendapatkan komentar tunggal "//" dan komentar majemuk "/* … …… … ... */". Untuk mengenali komentar yang ada pada kode sumber Regexnya seperti pada gambar 3.4.

   ```
   (/\\*([^\\*]|(\\*(?!/))+)*+\\*+/)|(//.*)
   ```

   Gambar 3.4. Pola Regex deteksi komentar

   Dengan Regex seperti gambar 3.4, program akan menghapus komentar yang ada jika ditemukan.

2. Pengenalan PACKAGE

Package pada kode sumber java, diawali dengan kata PACKAGE misalnya: **`package animals;`**. Untuk mengenali Package, peneliti dalam mengenali Package menggunakan Regex terlihat seperti gambar 3.5.

```
package\\s+.+
```

Gambar 3.5. Pola Regex deteksi package

Dengan regex seperti gambar 3.5., akan mengenali keberadaan package pada kode sumber java, setelah ditemukan maka akan diberikan informasi komentar dalam basah Indonesia “terdapat package dengan nama”.

3. Pengenalan IMPORT

Import pada bahasa pemrograman java diawali dengan import, misalnya : **`import java.util.*`**. Untuk mengenali import dengan pada kode sumber java, peneliti mengenalinya dengan menggunakan Regex, seperti gambar 3.6.

```
import\\s+.+
```

Gambar 3.6. Kode Regex untuk mengenali import.

4. Pengenalan CLASS/INTERFACE/ENUM

Class, Interface dan Enum pada bahasa pemrograman java diawali dengan class, interface dan enum. Dalam bahasa pemrograman java, class dapat dikenali dengan sintaks:

```
class <nama_class> { //statement }
```

Untuk interface dalam bahasa pemrograman java dapat dikenali dengan sintaks :

```
interface <nama_interface> { //statement }
```

Untuk enum pada bahasa pemrograman java dapat dikenali dengan sintaks:

```
enum <nama_enum> { //statement }
```

Pada regex untuk mengenali keberadaan class, interface dan enum dapat dilihat pada gambar 3.7.

```
class\s.+|interface\s.+|enum\s.+
```

Gambar 3.7. Kode regex mengenali class, interface dan enum.

5. Pengenalan METHODS/CONSTRUCTOR

Method didalam bahasa pemrograman java memiliki sintaks seperti gambar 3.8:

```
public static int methodName(int a, int b) {
     // body
}
```

Gambar 3.8. Sintaks penulisan method pada bahasa pemrograman java

Konstruktor merupakan method berjenis khusus yang digunakan untuk menginisialisasi obyek. Bentuk konstruktur sama dengan method hanya saja konstruktor tidak memiliki tipe data.

```
public static methodName {
  // body
}
```

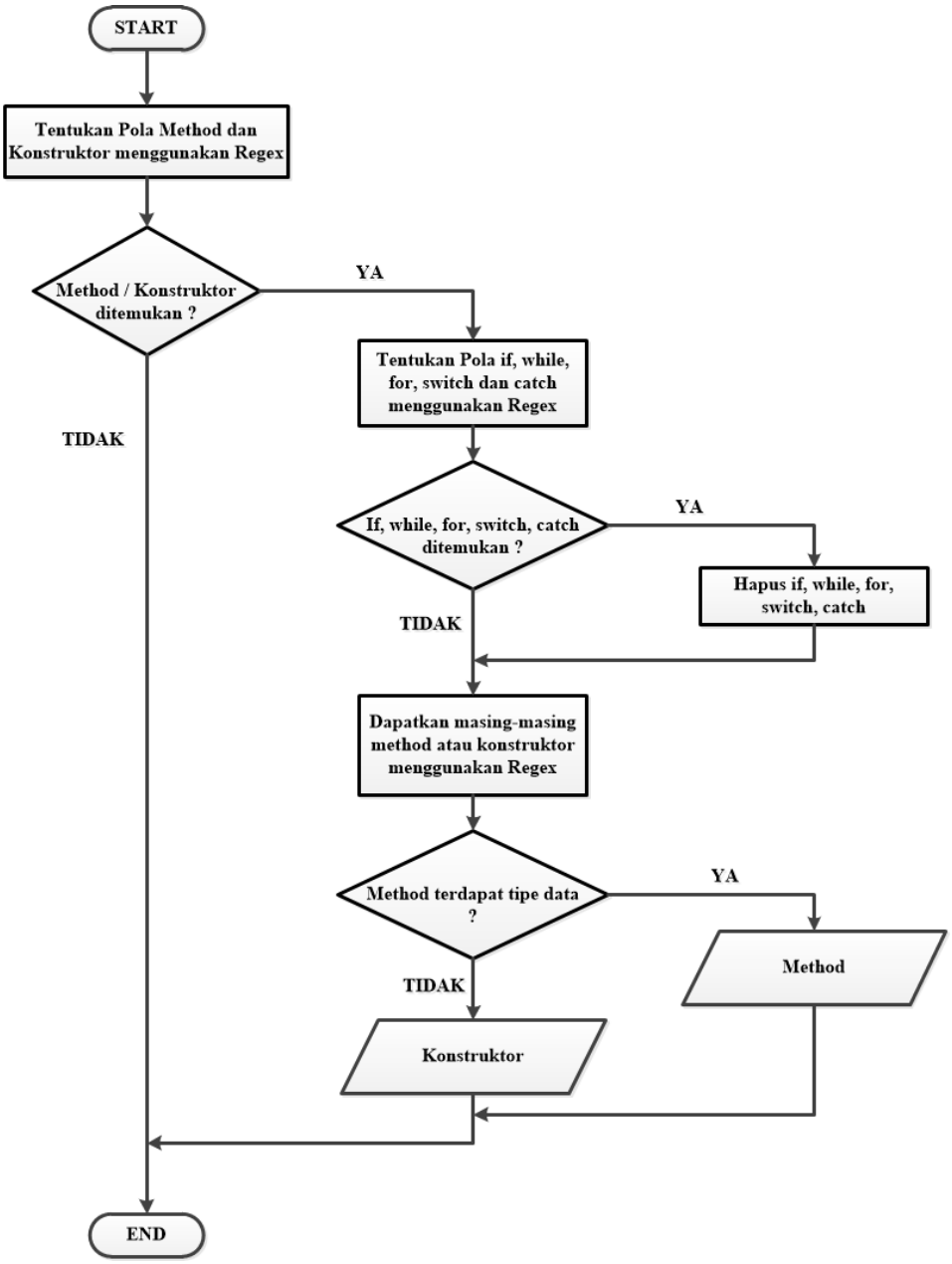
Gambar 3.9. Sintaks konstruktor pada bahasa pemrograman java

Untuk mengenali method dan konstruktor pada bahasa pemrograman java dengan menggunakan regex seperti pada gambar 3.10.

```
^\\s*([A-Za-z0-9_$]+\\s+)*([A-Za-z0-9_$]+
\\s*)\\(([\\w$,\\[\\]))\\s*\\{
```

Gambar 3.10. Regex untuk mengenali method pada kode sumber java

Algoritma pengenalan method dan konstruktor adalah seperti pada gambar 3.11.



Gambar3.11. Alur penentuan method atau konstruktor

Gambar 3.11, menjelaskan tentang proses pengenalan method atau konstruktor pada bahasa pemrograman java. Proses pertama yang dilakukan adalah menentukan pola method dan konstruktor secara umum menggunakan regex dan terlihat seperti pada gambar 3.10. Proses selanjutnya adalah mengecek apakah ada if, while, for, switch, atau catch

pada program java. Proses ini dilakukan karena bentuk penulisan programnya mirip dengan konstruktor, oleh karena itu dicek keberadaannya dan jika ditemukan maka akan dihapus (tidak dicek). Pada langkah ini sudah didapatkan method dan konstruktor tetapi belum didapatkan secara pasti mana yang method dan konstruktor. Proses selanjutnya mengecek apakah method terdapat tipe data atau tidak, jika method tidak terdapat tipe data maka termasuk konstruktor dan sebaliknya maka termasuk method.

6. Pengenalan MODIFIER

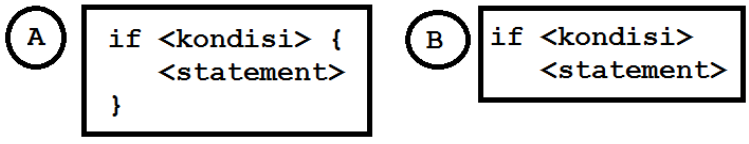
Modifier di java digunakan untuk mengetahui sifat yang dimiliki oleh atribut atau method. Secara umum modifier terbagi menjadi dua jenis, yaitu access modifier dan non-access modifier. Di java access modifier antara lain: public, protected, default dan private, sedangkan non-access modifier antara lain static, final, abstract, synchronized, native, dan transient. Untuk mengenali modifier di java dengan menggunakan regex adalah seperti pada gambar 3.12.

```
                    (public|private|protected)
(static)|(final)|(abstract)|(native)|(synchronized)|(transient)
```

Gambar 3.12. Regex pengenalan modifier di java.

7. Pengenalan Proses PERCABANGAN

Percabangan digunakan untuk melakukan pemilihan dari beberapa tindakan yang sesuai dengan tujuan. Percabangan pada java digunakan dengan menggunakan fungsi if. Terdapat dua cara penulisan if yang biasa dilakukan oleh para pengembang seperti pada gambar 3.13.



Gambar 3.13. Dua cara penulisan percabangan di java

Untuk mengenali proses percabangan if dengan menggunakan regex dapat dilihat pada gambar 3.14.

```
if\\s*\\(.*\\)
```

Gambar 3.14. Regex pengenalan if pada bahasa pemrograman java

Pada gambar 3.14. merupakan regex untuk mengenali keberadaan fungsi percabangan dengan menggunakan if. Regex ini akan mengenali baris-baris program java yang mempunyai percabangan if dan jika ditemukan maka akan diberikan keterangan komentar "Jika"

8. Pengenalan Proses PERULANGAN

Proses perulangan pada java menggunakan tiga bentuk, yaitu : for, while, dan do..while. Sintaks penulisan perulangan for menggunakan sintaks :

```
for (inisialisasi; kondisi; iterasi)
```

Untuk perulangan while pada bahasa pemrograman java menggunakan sintaks:

```
while (kondisi) { //statement perulangan }
```

Untuk perulangan do..while pada bahasa pemrograman java menggunakan sintaks :

```
do { //statement perulangan } while (kondisi);
```

Untuk mengenali proses perulangan dengan menggunakan regex seperti pada gambar 3.15.

```
(for\s.*)|(while\s.*)|(do\s.*|while\s.*;)
```

Gambar 3.15. Regex untuk pengenalan perulangan for, while dan do.while

9. Pengenalan Variabel

Variable merupakan suatu tempat yang digunakan untuk menyimpan data. Pada bahasa pemrograman java penulisan variabel dengan sintaks :

```
TipeData NamaVariabel;
```

Untuk mengenali variabel pada bahasa pemrograman java dengan menggunakan regex seprti pada gambar 3.16.

```
(\w+\s\w+;)|(\w+\s\w+=\d.+)|(\w+\s\w+,.+)|(\w+\s\w+=\'.+)
```

Gambar 3.16. Regex deteksi variabel pada bahasa pemrograman java

Pada gambar 3.16 terdapat empat model regex untuk mendeteksi keberadaan variabel pada bahasa pemrograman java, yaitu

1. Kondisi pertama : **`(\w+\s\w+;)`**. Regex jenis ini akan menangani kondisi variabel seperti: **int a;, int ciciak;, double luas;**.
2. Kondisi kedua : **`(\w+\s\w+=\d.+)`**. Regex jenis ini akan menangani kondisi variabel seperti : **int b=2.534;, double luas_segitiga=14;**.
3. Kondisi ketiga : **`(\w+\s\w+,.+)`**. Regex jenis ini akan menangani kondisi variabel seperti : **int sepeda,mobil,roka=10.45;, char alib,kira;, string buku, gelas;**.
4. Kondisi keempat : **`(\w+\s\w+=\'.+)`**. Regex jenis ini akan menangani kondisi variabel seperti : **char kali=’S’;**.

3.3. Dataset Uji dan Validasi

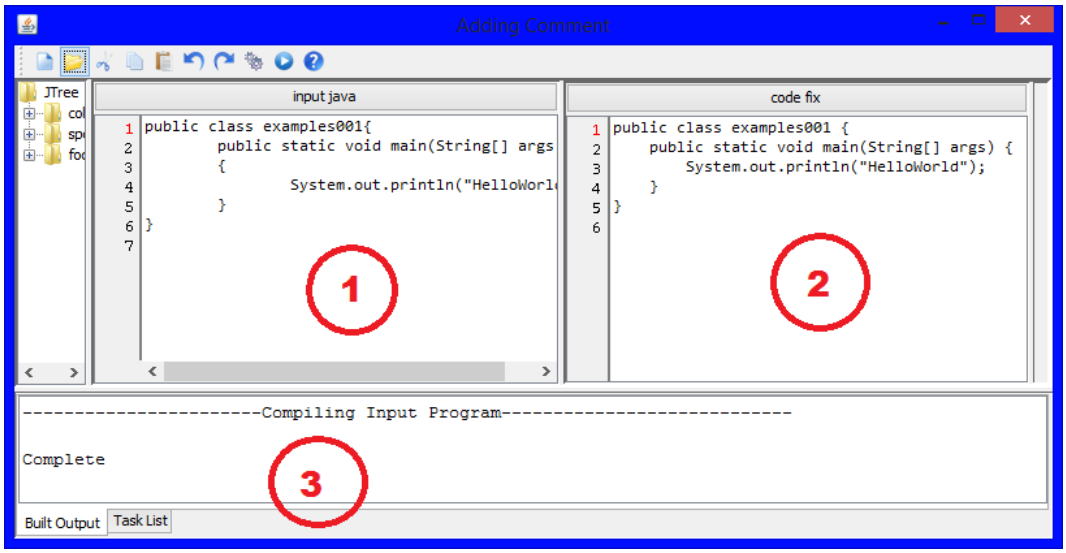
Dataset uji didapatkan dari internet dan soal praktikum pemrograman berorientasi obyek – laboratorium bahasa pemrograman – ITATS. Dari internet, didapatkan banyak contoh program, tetapi dipilih beberapa program yang digunakan untuk uji. Dataset uji ini digunakan untuk proses survey kepada responden dan untuk uji aplikasi yang dibangun.

Adapun data uji untuk proses thesis yang diambil dari internet pada web site antara lain:

1. http://www.java2novice.com/java-sorting-algorithms/bubble-sort/
2. introcs.cs.princeton.edu
3. www.programmingsimplified.com

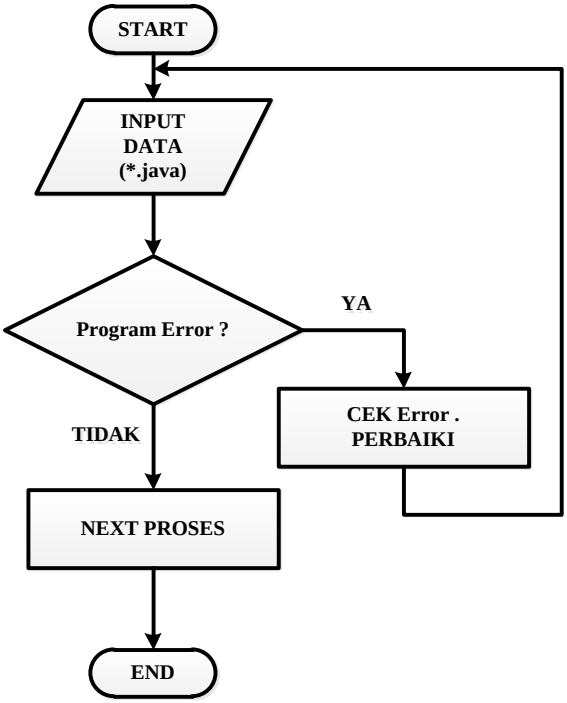
Proses selanjutnya adalah validasi program java yang didownload dan akan digunakan untuk proses uji. Proses validasi dilakukan dengan cara memastikan program java dapat dijalankan. Untuk dapat mengetahui program itu dapat dijalankan ataupun tidak digunakan Aplikasi yang dibangun. Dengan Aplikasi ini,

program yang ada akan dicompile dan dipastikan bahwa program yang ada dapat dijalankan.



Gambar 3.17. Proses Validasi Program Java untuk proses Uji

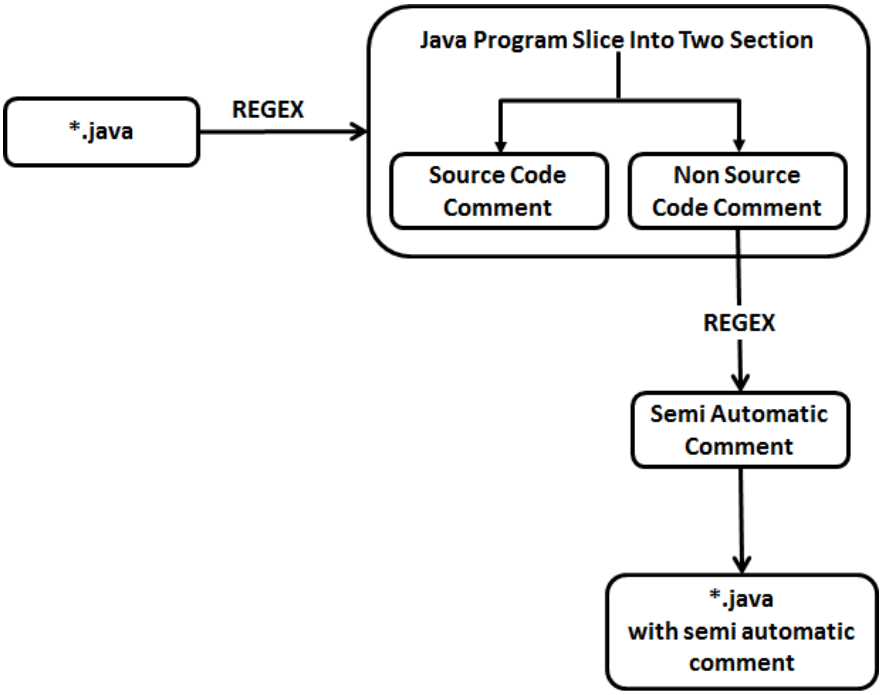
Pada gambar 3.17. dengan menggunakan aplikasi yang dibangun, dapat melakukan pembacaan pada program java, dimana pada gambar 3.17-1, merupakan program java sumber yang dipanggil, sedangkan pada gambar 3.17-2 merupakan program *.java yang dipastikan (divalidasi) oleh aplikasi dan gambar 3.17-3 merupakan informasi dari aplikasi setelah dilakukan pengecekkan, terjadi kesalahan ataukah tidak.



Gambar 3.18. Flowchart Proses Validasi Program

3.4. Desain

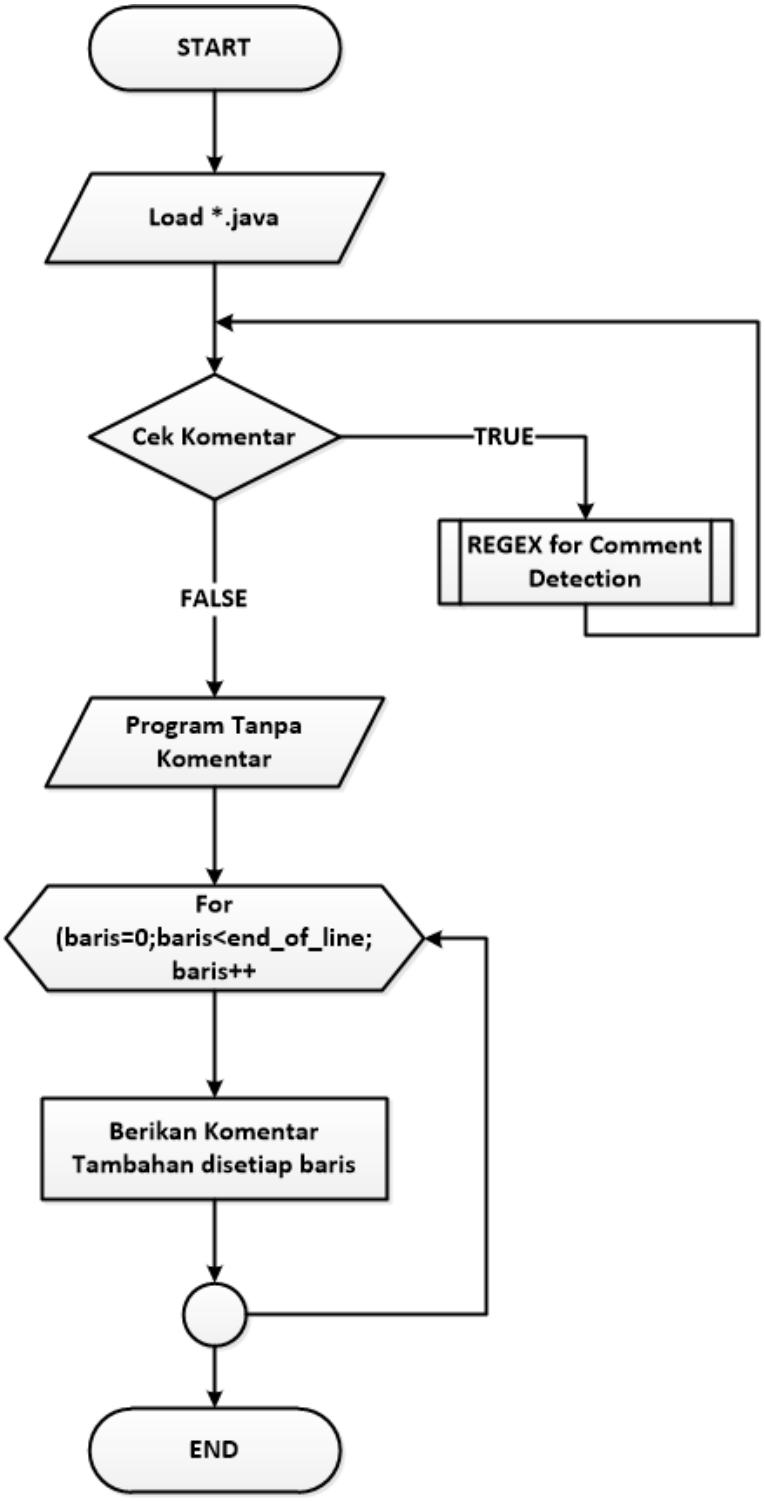
Proses desain dilakukan oleh penulis setelah melakukan tahapan analisis. Proses desain yang dilakukan penulis terlihat pada gambar 3.18.



Gambar 3.19. Blok diagram pengembangan aplikasi

Pada gambar 3.19. dapat dilihat bahwa pada thesis ini penulis menggunakan REGEX untuk proses pemisahan program dengan komentar serta pemberian komentar tambahan. Secara keseluruhan terdapat dua kegiatan utama, yaitu

1. Memisahkan komentar dengan kode sumber
2. Memberikan komentar tambahan otomatis pada kode sumber yang telah dipisahkan dari komentarnya aslinya .
3. Memberikan kemampuan pada komentar yang dibuat secara otomatis untuk menjadi semi otomatis dimana komentar semi otomatis ini dapat dilakukan secara manual oleh pengguna aplikasi jika komentar yang dibuat secara otomatis dirasa kurang baik.



Gambar 3.20. Flowchart pemisahan Komentar dan Program serta pemberian komentar otomatis perbaris kode sumber.

Pada gambar 3.20. merupakan flowchart desain aplikasi untuk membagi kode sumber menjadi komentar dan program serta pemberian komentar otomatis setiap baris kode sumber. Proses pertama adalah mendeteksi keberadaan komentar yang ada disetiap kode sumber. Proses pendeteksian komentar telah dijelaskan pada gambar 3.19. disini komentar yang telah dideteksi akan dihapus, setelah itu program hasil proses pertama akan diberikan komentar baru. Pemberian komentar baru ini dicocokan dengan pola yang ada dimana proses pencocokannya penulis menggunakan REGEX.

3.5. Pengkodean

Pengkodean merupakan proses pembuatan aplikasi yang diinginkan. Pada saat program dijalankan, maka langkah selanjutnya proses yang dilakukan adalah membuka program java. Proses untuk membuka dan membaca file java seperti pada gambar 3.20.

```
JFileChooser fc = new JFileChooser("D:\\java");
FileFilter filter1 = new ExtensionFileFilter(
    "Java File", new String[] { "java" });
fc.setFileFilter (filter1);
```

Gambar 3.21. Program untuk membaca file *.java.

Pada gambar 3.21. dapat dilihat bahwa penulis menggunakan `JFileChooser`. `JFileChooser` digunakan untuk menentukan letak dari java yang akan dibuka secara default ada di drive D dan direktori `Java (D:\\Java)`. Proses selanjutnya adalah menentukan file apa yang akan diperlukan atau dibuka. Disini ditentukan kebutuhan file yang diperlukan yaitu `*.java`, oleh karena itu dibagian selanjutnya digunakan fungsi `FileFilter`. `FileFilter filter1= new ExtensionFileFilter ("Java File", new String[] {"java"});`. Untuk menyeleksi file java saja yang akan dibuka atau ditampilkan maka `new String[] {"java"}` diaktifkan.

Setelah proses mengambil dan mengenali jenis file yang diinginkan, maka proses selanjutnya adalah tahap kompilasi program yang diletakkan pada file `comple.java`. Tahapan ini digunakan untuk mendeteksi apakah program yang dipilih benar keberadaannya ataukah masih ada kesalahan. Untuk mendeteksi

program yang dipilih terdapat kesalahan atau tidak dapat dilihat pada gambar 3.22.

```
JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
DiagnosticCollector<JavaFileObject> diagnostics = new
DiagnosticCollector<JavaFileObject>();
StandardJavaFileManager fileManager =
compiler.getStandardFileManager(diagnostics, null, null);
Iterable<? extends JavaFileObject> compilationUnits =
fileManager.getJavaFileObjectsFromStrings(Arrays.asList(file+""
));
String[] option=new String[]{"-g","-parameters"};
Iterable<String> options = Arrays.asList(option);

JavaCompiler.CompilationTask task=null;
task = compiler.getTask(null, fileManager, diagnostics,
options,null, compilationUnits);
          success = task.call();
          fileManager.close();
          System.out.println("Success: " + file + " = " +
success);
          if(success==true)
          {
                errortab.setSelectedIndex(0);
                errortext.append("\nComplete");
                if(!cekfile==false)
                {
                      try
                      {

                      }
                      catch(Exception ex)
                      {
                            try
                            {
//                                File err = new
File("error.txt");
//                                java.io.PrintStream ps = new
java.io.PrintStream(err);
                                ex.printStackTrace();
//                                ps.close();
                            }
                            catch(Exception exx){}
                      }
                }
          }
```

Gambar 3.22. Program untuk Kompilasi file *.java.

Pada gamabr 3.22. proses diawali dengan memastikan program benar atau salah dengan menjalankan perangkat bantu `DiagnosticCollector<JavaFileObject>`. dengan ini maka program akan dilakukan pengecekkan hasilnya diletakkan pada `fileManager` dan jika memenuhi syarat `JavaCompiler`, yaitu pada

`JavaCompiler.CompilationTask task=null;`, maka proses berarti sukses dan dapat dikompilasi dan akan muncul pesan informasi `Complete`. Jika pada saat proses kompilasi terdapat kesalahan, maka kesalahan tersebut diletakkan pada sebuah tabel. Proses pengecekan kesalahan terdapat pada bagian else pada gambar 3.22 dan pada bagian tersebut langsung dilempar ke program komenadd.java untuk membantuk informasi di tabelnya dimana pada tabel hanya muncul tiga kolom, yaitu **NO**, **DESKRIPSI** dan **LINE** degnan lebih kolom masing-masing adalah **50**, **1000**, dan **500**. Pada gambar 3.22. merupakan proses detail informasi kesalahan yang dideteksi.

```
errortab.add("Built Output",new JScrollPane(errortext));
errortab.add("Task List",new JScrollPane(errortable));
errortab.setTabPlacement(JTabbedPane.BOTTOM);
tablemodel.addColumn("No");
tablemodel.addColumn("Deskripsi");
tablemodel.addColumn("Line");

TableColumn kolom=null;
kolom=errortable.getColumnModel().getColumn(0);
kolom.setPreferredWidth(50);
        kolom=errortable.getColumnModel().getColumn(1);
        kolom.setPreferredWidth(1000);
        kolom=errortable.getColumnModel().getColumn(2);
        kolom.setPreferredWidth(500);
```

Gambar 3.23. Pembuatan tabel penganan kesalahan program saat proses kompilasi

Setelah proses kompilasi program, maka proses yang selanjutnya terjadi adalah menghilangkan semua komentar yang ada didalam program, pembuatan program penghilangan komentar yang ada di program dapat dilihat pada gamabr 3.24.

```
sourcefix.setText(Pattern.compile("(/\\*(
[^\\*]|(\\*(?!/))+)*+\\*+/)|(//.*)").

 matcher(source.getText()).replaceAll("")
);

 String nostring;
 hasilcode=split;

nostring=split.replaceAll("(\".*[^\\\\]\"
)|(\"[^\"]*\")", "");

hasilcode+=new
process(hasilcode).getCode();
```

Gambar 3.24. Proses deteksi dan penghapusan komentar pada program java

Pada baris 2 sampai dengan 10 merupakan proses deteksi keberadaan komentar tunggal, jika ditemukan komentar tunggal maka komentar tersebut dihilangkan. Untuk proses penghilangan komentar terlihat pada baris 6, yaitu `return "";`. sedangkan untuk komentar yang lainnya dilakukan proses pengenalan seperti terlihat pada baris 12 , yaitu dipolakan dulu dengan regex dan dilanjutkan dengan proses menghapusnya komentar pada baris 13. Hasil dari program yang sudah hilang komentarnya disimpan pada `hasilcode+` dan diletakkan pada `JTextArea` kedua.

Proses berikutnya adalah pemberian komentar otomatis pada program yang sedang aktif. Untuk memberikan komentar pada program yang mengandung package dengan informasi **"terdapat package dengan nama" <nama package>**. Untuk memberikan komentar otomatis tersebut programmnya dapat dilihat pada gambar 3.25.

```
Matcher paket=Pattern.compile(getPattern
("package\\s+.+;")). matcher(hasilcode);

if(paket.find()){
    String paketfix=paket.group().replaceAll
      ("^\\s*", "");
    String getpackagename=paketfix.replaceAll
      ("package\\s+|;", "");

   hasilcode+=setComment(hasilcode)+" terdapat
   package dengan nama "+getpackagename;
 }
```

Gambar 3.25. Proses pemberian komentar Package pada Pembuatan Aplikasi

Proses selanjutnya adalah pengecekkan keberadaan class, interface dan enum. Pada baris 152 sampai 153 merupakan pemberian pola pengenalan `class, interface,` dan `enum` dengan menggunakan regex. Pada baris 154 sampai 182, dilakukan proses pengecekkan pada setiap baris program apakah terdapat `class, interface` ataupun `enum`, jika ditemukan maka (baris 154) hapus semua spasi dan cek yang terdeksi (157-161) pada baris tersebut class atau interface atau enum, beri keterangan (baris 162-167) yaitu "`class | interface | enum dengan nama <nama_class | nama_interface | nama_enum>`". Pada gambar 3.24., juga dilakukan pengecekkan keberadaan `extends` dan `implements` pada sebuah program

java. Untuk mendeteksi keberadaan `extends`, mula-mula mempolakan bentuk extend seperti pada baris 168-169 lalu proses selanjutnya adalah dilakukan pengecekkan keberadaan `extend` apakah ditemukan, jika ya maka muncul informasi " `extends dengan pewarisan class <nama extend>`". Untuk mendeteksi keberadaan `implements`, yang dilakukan pertama kali adalah mempolakan `implements` dengan regex seperti terlihat pada baris 175-176 dan dilanjutkan dengan mengecek keberadaan `implements`, jika ada maka akan memunculkan informasi (baris 177-181) "`implements dengan pewarisan interface <nama implements>`".

Proses selanjutnya adalah mendeteksi keberadaan `method` dan `constructor`. Untuk mendeteksi method dan constructor yang dilakukan pertama kali adalah mempolakan model sintaks method dan constructor menggunakan regex. Pada program `ProcessComment.java` dibaris 202-203 adalah proses memolakan atau mengenali pola method dengan menggunakan regex, lalu dilanjutkan dengan mendeteksinya pada program java yaitu pada baris 204-208, jika method ditemukan (`if (methods.find)`) maka cek apakah ditemukan `if` atau `while` atau `for` jika iya maka hapus. Proses selanjutnya adalah dapatkan nama dari masing-masing `method` atau `konstruktor` (baris 213-264), jika tidak mengandung tipe data maka dapat dipastikan adalah konstruktor dan cetak dengan informasi "`terdapat constructor dengan nama <nama_constructor>`" (baris 229). Pada saat mendeteksi method dan konstruktor jika terdapat void dan tanpa nilai balik berarti method dan mencetak informasi "`terdapat method dengan nama <nama_method> dengan type data <type_data>`", tetapi jika terdapat nilai balik maka akan mencetak "`terdapat function dengan nama <nama_function> dengan type data <type_data>`".

Proses selanjutnya pada program `ProcessComment.java` adalah deteksi keberadaan proses percabangan. Proses percabangan pada java terdiri dari fungsi `IF` dan `IF..ELSE`. Pada saat pengenalan IF biasanya diikuti dengan proses aritmatika, yaitu sama dengan (==), tidak sama dengan (!=), lebih kecil atau sama dengan (<=), lebih besar atau sama dengan (>=), lebih kecil dari (<), lebih besar dari (>), or (||) dan and (&&).

Proses selanjutnya adalah pendeteksian keberadaan perulangan dimana di java terdapat tiga cara, yaitu menggunakan `while`, `do..while` dan `for`. Pendeteksian perulangan `while`, `do..while` dan `for`, pada langkah awal adalah menentukan pola regexnya. Untuk proses perulangan dengan `while` setelah ditentukan pola regexnya pada baris 276-293, disini proses pertama diawali dengan mempolakan perulangan while dengan menggunakan regex (baris 276). Langkah selanjutnya adalah mengecek keberadaan perulangan while pada program, jika pada program java terdapat proses perulangan while maka tampilkan informasi " `selama <proses kondisi> maka` " (baris 290-291).

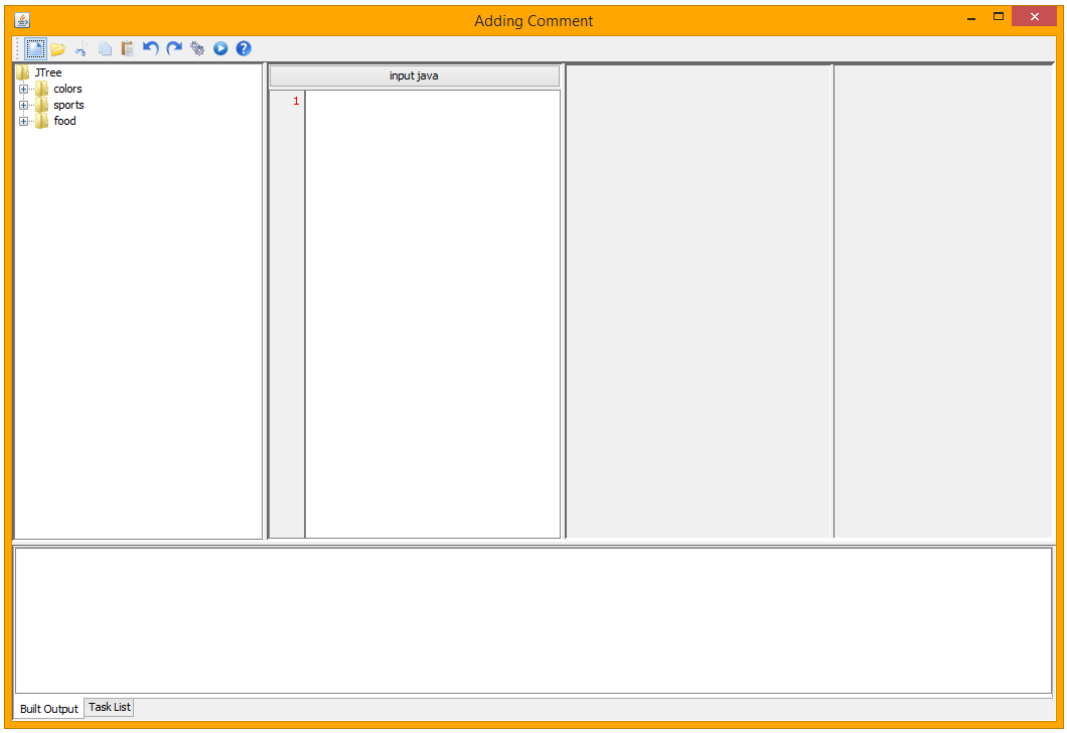
Proses deteksi selanjutnya adalah perulangan `do..while`. Hal yang dilakukan pertama kali adalah proses mempolakan perulangan `do..while` dimana pola regex untuk `do..while` terlihat pada baris 297 lalu dilanjutkan dengan pengecekkan apakah didalam program java terdapat perulangan `do`, jika ditemukan maka berikan informasi " `kerjakan` " lalu dilanjutkan dengan isi dari program looping `while` seperti pada baris 303-320.

Proses deteksi perulangan `for` yang dilakukan pertama kali adalah menentukan pola perulangan `for` (baris 529 dan 538), dimana pola perulangan `for` ditandai dengan kata `for <kondisi> {`. Jika ditemukan maka beri penjelasan informasi " `ulangi` " (baris 547) `dengan inisial <inisialisasi>` (baris 557) `sebanyak kondisi <kondisi>` (baris 562) `dengan incremental <incremental>` (baris 567).

BAB IV
HASIL dan PEMBAHASAN

4.1. Hasil

Pada bagian ini merupakan tahapan implementasi dari aplikasi yang sudah dibangun. Pada saat aplikasi pertama kali dijalankan akan menampilkan menu utama.

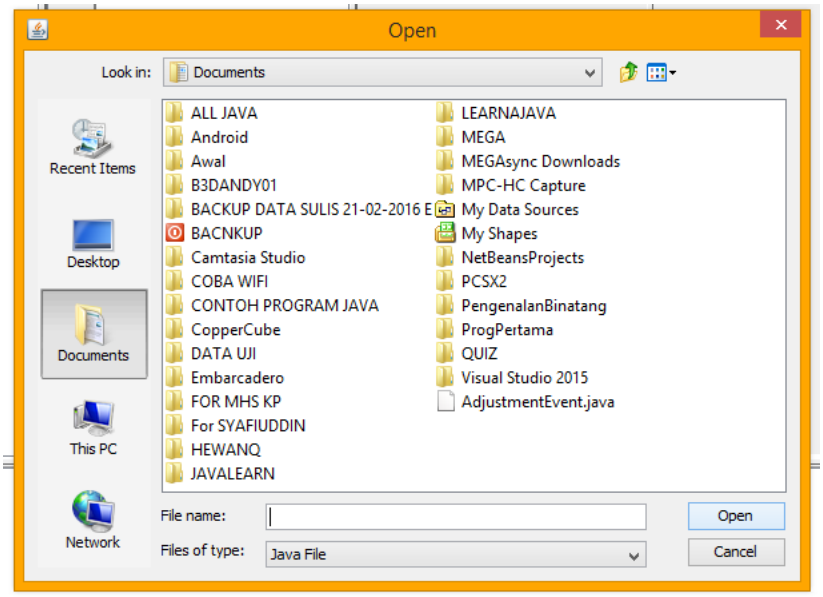


Gambar 4.1. Menu utama aplikasi

Dari gambar 4.1. terdapat 4 kolom dan 1 baris dibagian bawah. Kolom pertama disebelah kiri sebagai informasi daftar program dalam bentuk pohon. Kolom kedua merupakan tempat program yang dipanggil. Kolom ketiga merupakan kolom dimana tempat program setelah proses kompilasi dilakukan. Kolom keempat merupakan tempat program setelah dijalankan dan hasil pemberian komentar (**code with comment**). Baris dibagian bawah merupakan tempat informasi berhasil atau tidaknya program yang dipanggil dikompilasi.

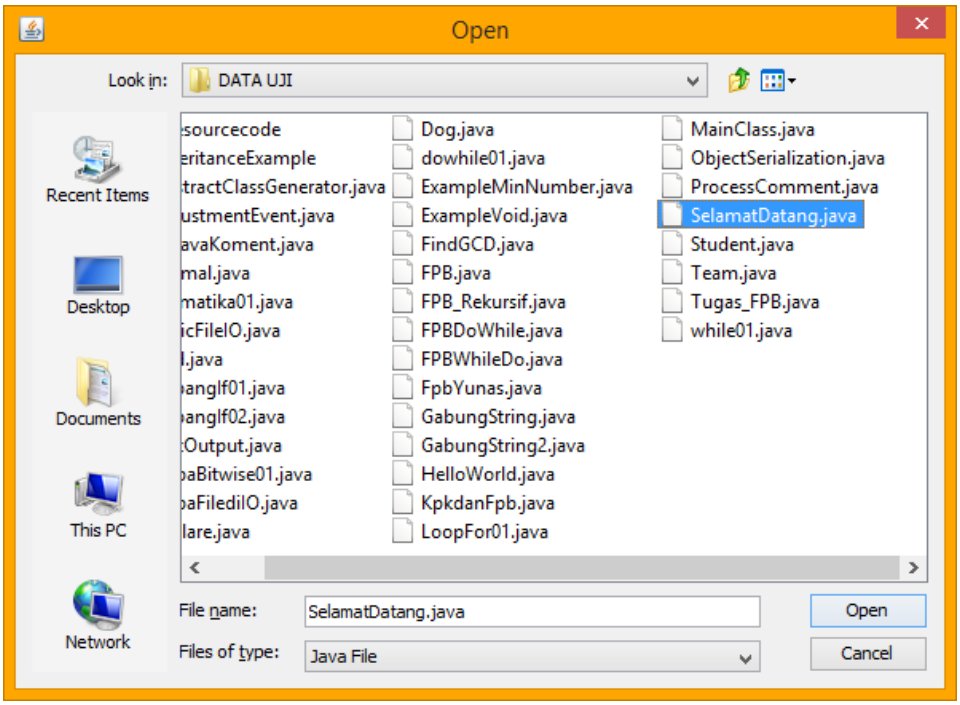
Tombol `open` digunakan untuk membuka program `*.java` yang akan dicek untuk diberikan komentar program. Secara `default`, pada aplikasi lokasi

diset di drive `d:\java`, tetapi jika pada saat klik tidak ditemukan direktori java di drive d, maka secara default akan membuka direktori `c:\documents`.



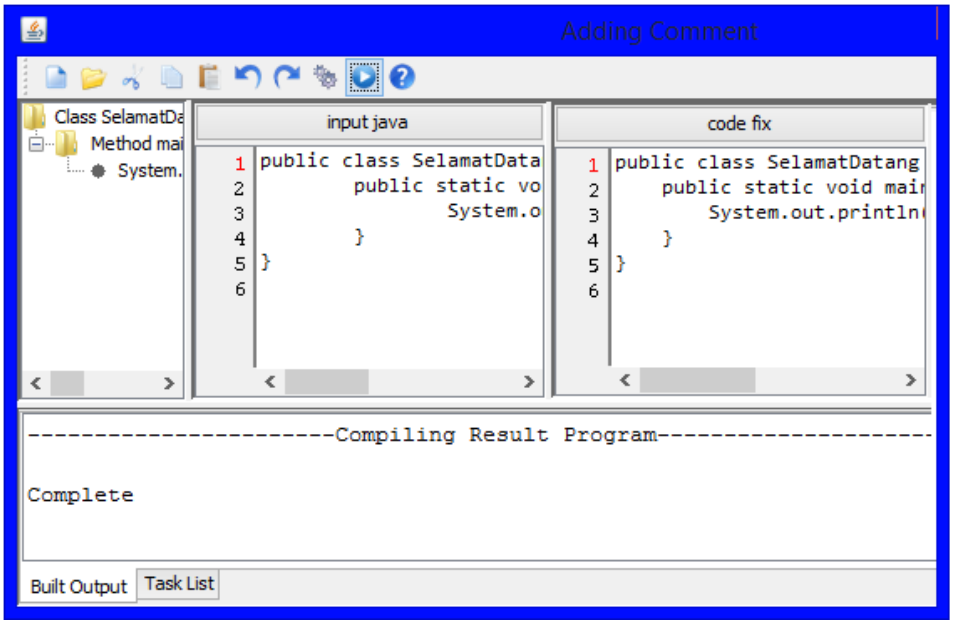
Gambar 4.2. Pencarian file java yang gagal di drive d:\java.

Setelah itu tentukan direktori tempat penyimpanan file java, misalnya `DATA UJI`. Langkah selanjutnya adalah pemilihan file, misalnya `SelamatDatang.java` lalu diakhiri dengan menekan tombol `open` Seperti pada gambar 4.3.



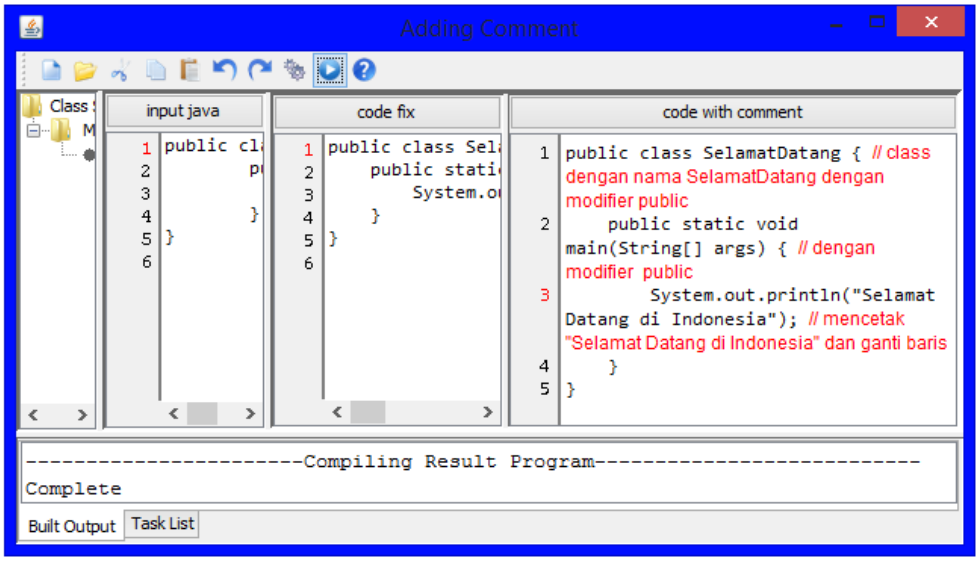
Gambar 4.3. Pemilihan file java pada aplikasi

Saat program telah dipilih maka aplikasi secara otomatis akan menampilkan data file tersebut. Dilanjutkan dengan aplikasi melakukan proses kompilasi secara otomatis. Proses kompilasi ini untuk melakukan proses pengecekkan telah terjadi kesalahan ataupun tidak. Setelah tidak ada kesalahan maka akan muncul kode sumber yang telah dikompilasi.



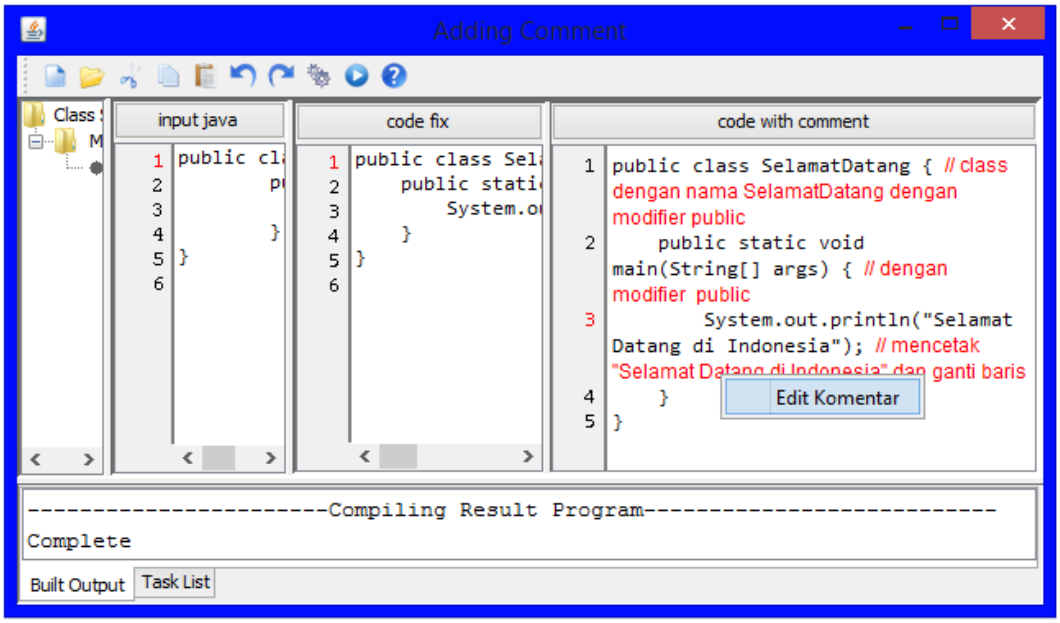
Gambar 4.4. Proses Kompilasi kode program pada aplikasi

Untuk mendapatkan pemahaman tentang sebuah program, maka klik tombol RUN untuk menjalankan aplikasi dan mendapatkan komentar yang dibuat secara otomatis.



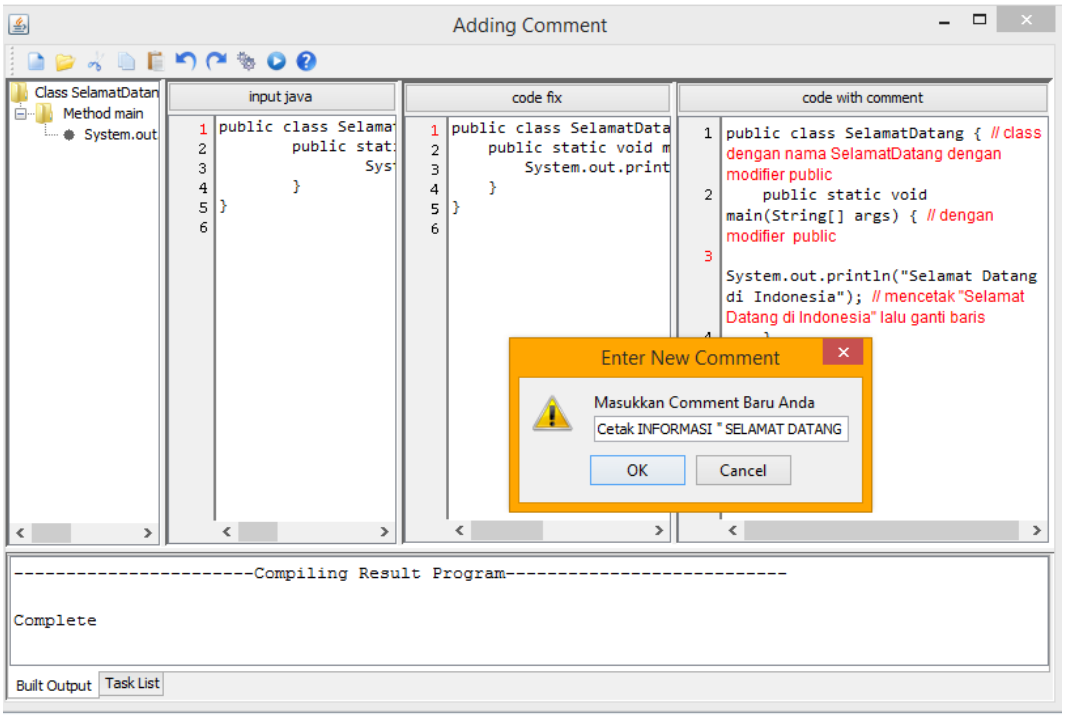
Gambar 4.5. Proses pemberian komentar otomatis pada aplikasi

Pada gambar 4.5. merupakan proses pemberian komentar pada kode program secara otomatis, sedangkan untuk komentar semi otomatis merupakan komentar dimana pemberiannya dilakukan secara manual oleh pengguna dimana dilakukan pada kode program yang sudah diberikan komentar secara otomatis dan dirasa bahwa komentar otomatis kurang lengkap atau kurang tepat. Untuk membuat komentar semi otomatisnya dapa dilakukan dengan cara klik kanan pada komentar otomatis, lalu akan muncul menu `Edit Komentar`.

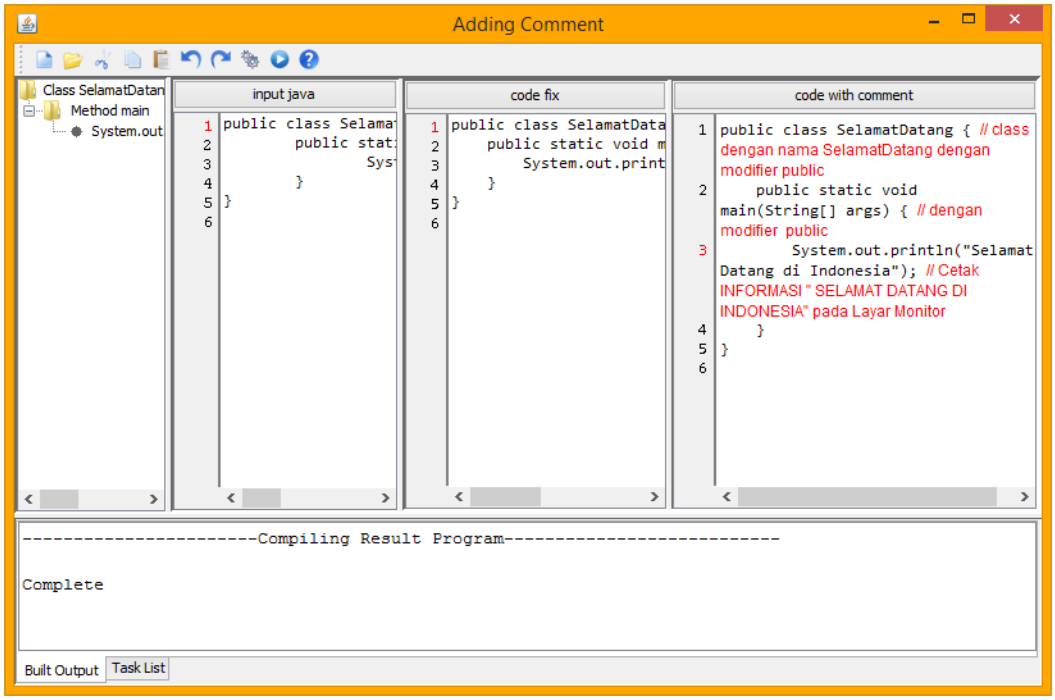


Gambar 4.6. Proses Edit Komentar pada aplikasi

Setelah klik `EDIT KOMENTAR` maka akan muncul window `ENTER NEW COMMENT` (gambar 4.7). Pada window ini, pengguna bisa menginputkan komentar baru yang sesuai dengan keinginan, setelah dirasa sudah cocok, maka akhiri dengan menekan tombol `OK`. Hasil dari perubahan komentar dapat dilihat pada gambar 4.8. Pada Gambar 4.8. dapat dilihat bahwa posisi komentar sesuai dengan tempat yang diinginkan dan isinyapun sesuai dengan proses perubahan yang dilakukan.



Gambar 4.7. Proses pemberian komentar baru pada kode program



Gambar 4.8. Proses Edit Komentar telah berhasil dilakukan pada kode program

4.2. Pembahasan

4.2.1. Uji Aplikasi

Aplikasi yang telah dibangun oleh peneliti, telah diuji coba untuk mengeneralisasi komentar pada file-file java yang ada. Terdapat 31 file java yang telah diuji coba oleh peneliti. Dari uji coba tersebut rekap data seperti pada gambar 4.9.

NO	NAMA FILE	JUMLAH BARIS	LOAD	KOMPILASI	PEMBERIAN KOMENTAR		%	KEBERHASILAN PEMBERIAN KOMENTAR PROGRAM	TIDAK BERHASIL MEMBERIKAN KOMENTAR PROGRAM	RATA-RATA %
					DETEKSI (baris)	TIDAK TERDEKSI (Baris)				
1	LoopFor01	10	1	1	10	0	100%	BERHASIL	BERHASIL	100%
2	Animal	14	1	1	8	0	100%	BERHASIL	BERHASIL	100%
3	InputFile03	43	1	1	28	2	93%	BERHASIL	BERHASIL	93%
4	AbstractClassGenerator	245	1	0	0	0	0%	TIDAK BERHASIL	TIDAK BERHASIL	0%
5	AdjustmentEvent	140	1	1	32	108	23%	BERHASIL	BERHASIL	23%
6	Aritmatika01	21	1	1	17	0	100%	BERHASIL	BERHASIL	100%
7	BasicFileIO	18	1	1	11	0	100%	BERHASIL	BERHASIL	100%
8	Bird	17	1	0	0	0	0%	TIDAK BERHASIL	TIDAK BERHASIL	0%
9	CabangIf01	10	1	1	6	0	100%	BERHASIL	BERHASIL	100%
10	CabangIf02	14	1	1	8	0	100%	BERHASIL	BERHASIL	100%
11	CabangIf03	12	1	1	8	0	100%	BERHASIL	BERHASIL	100%
12	CekOutput	8	1	1	3	0	100%	BERHASIL	BERHASIL	100%
13	CobaBitwise01	12	1	1	6	1	86%	BERHASIL	BERHASIL	86%
14	CobaFilediIO	18	1	1	11	0	100%	BERHASIL	BERHASIL	100%
15	do_while01	11	1	1	6	1	86%	BERHASIL	BERHASIL	86%
16	dowhile01	15	1	1	8	1	89%	BERHASIL	BERHASIL	89%
17	ExampleMinNumber	22	1	1	13	0	100%	BERHASIL	BERHASIL	100%
18	ExampleVoid	16	1	1	10	0	100%	BERHASIL	BERHASIL	100%
19	FindGCD	77	1	1	45	1	98%	BERHASIL	BERHASIL	98%
20	FPB	29	1	1	13	2	87%	BERHASIL	BERHASIL	87%
21	FPB_Rekursif	29	1	1	12	2	86%	BERHASIL	BERHASIL	86%
22	FBP_Do_While	29	1	1	17	2	89%	BERHASIL	BERHASIL	89%
23	FBP_While_Do	29	1	1	17	2	89%	BERHASIL	BERHASIL	89%
24	Calculator	21	1	1	9	3	75%	BERHASIL	BERHASIL	75%
25	FpbYunas	27	1	1	15	2	88%	BERHASIL	BERHASIL	88%
26	GabungString	17	1	1	10	0	100%	BERHASIL	BERHASIL	100%
27	GabungString2	12	1	1	8	0	100%	BERHASIL	BERHASIL	100%
28	HelloWorld	6	1	1	3	0	100%	BERHASIL	BERHASIL	100%
29	loopfor02	10	1	1	5	0	100%	BERHASIL	BERHASIL	100%
30	SelamatDatang	6	1	1	3	0	100%	BERHASIL	BERHASIL	100%
31	while01	16	1	1	7	1	88%	BERHASIL	BERHASIL	88%

Gambar 4.9. Data uji aplikasi pemberian komentar otomatis pada program java

Dari gambar 4.9. dapat diketahui bahwa terdapat beberapa kolom pada saat uji coba aplikasi. Kolom **`NAMA FILE`**, merupakan tempat yang digunakan untuk menyimpan data file **`*.java`** yang telah diuji. Kolom **`LOAD (uji)`**, merupakan tempat untuk merekap data file **`*.java`** bisa di load atau tidak, pada program aplikasi bagian ini ada pada Kolom Ke-2. Kolom **`LOAD`** akan bernilai 1 jika file **`*.java`** berhasil ditampilkan dan bernilai 0 jika tidak bisa ditampilkan. Dari ke 31 file java yang di **`LOAD`**, semuanya berhasil ditampilkan.

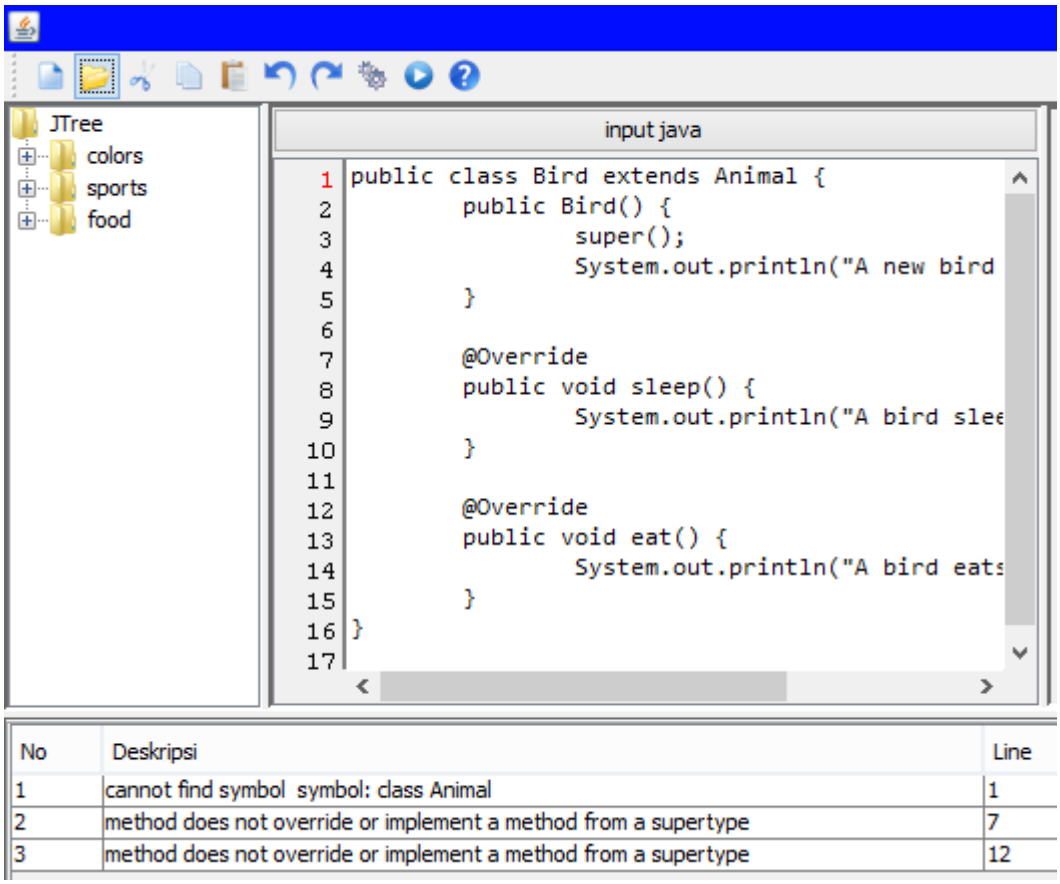
Kolom **`KOMPILASI`** merupakan tempat dimana untuk merekap data file **`*.java`** yang di **`LOAD`** berhasil dikompilasi ataukah tidak. Kolom **`KOMPILASI`** bernilai 1 jika file **`*.java`** telah berhasil dikompilasi, sedangkan jika bernilai 0 maka file tersebut tidak bisa dikompilasi.

File yang gagal dikompilasi dikarenakan beberapa sebab, antara lain:

1. Tidak terdapatnya file pendukung pada kode program.
2. Terjadi salah penulisan kode program.

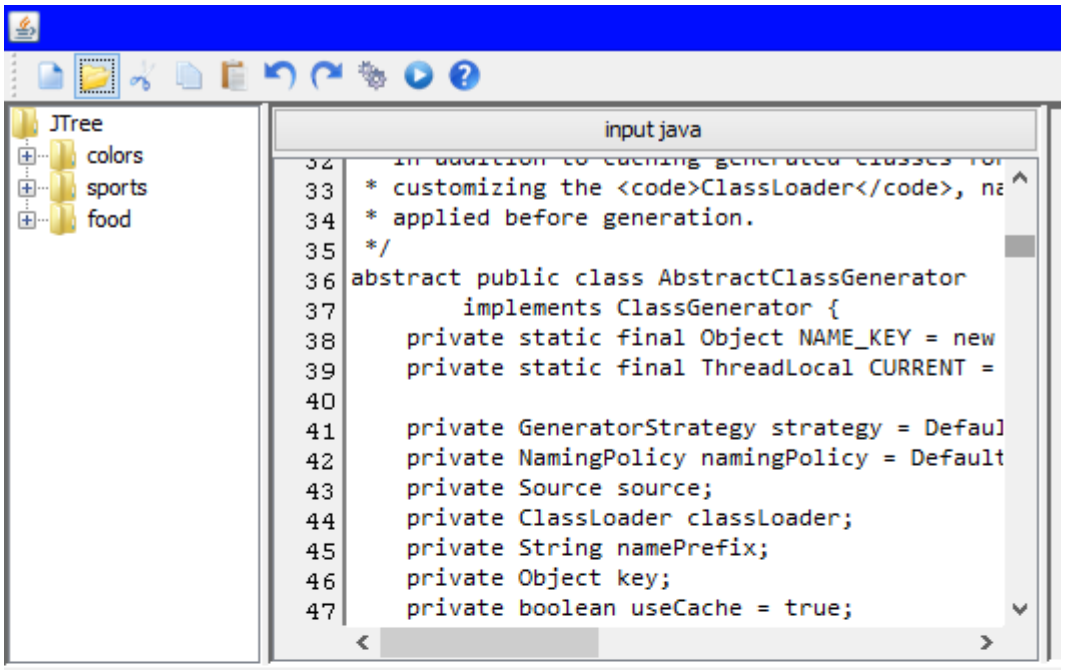
Dari 31 file yang di **LOAD** terdapat 2 file yang tidak dapat di **KOMPILASI** yaitu pada file ke-4 yang bernama **AbstractClassGenerator** dan file ke-8 yang bernama **bird.**

Pada file **bird.java** terdetteksi 3 jenis kesalahan yang terletak pada baris 1, 7 dan 12. Pesan kesalahan yang dapat terdeteksi oleh aplikasi yang dibangun oleh peneliti dijelaskan pada kolom Deskripsi, dengan penjelesana ini, maka pengguna aplikasi akan dapat mengetahui segala jenis kesalahan yang ada pada suatu program. Pada saat diklik 2x maka aplikasi akan membawa ke baris yang ada pada program sehingga dengan langkah seperti ini akan membantu pengguna untuk lebih cepat mengakses baris yang ada kesalahannya.



Gambar 4.10. Kesalahan yang terdeteksi pada file bird.java

Pada File **`AbstractClassGenerator`** terdiri dari 297 baris kode program dan terdapat 20 jenis kesalahan yang terdeteksi. Kesalahan-kesalahan yang terdeteksi pada file AbstractClassGenerator.java dapat dilihat pada gambar



No	Deskripsi	Line
1	package org.objectweb.asm does not exist	25
2	package org.objectweb.asm does not exist	26
3	package org.objectweb.asm does not exist	27
4	package org.objectweb.asm does not exist	28
5	cannot find symbol symbol: class ClassGenerator	37
6	cannot find symbol symbol: class GeneratorStrategy location: class net.sf.cglib.core.A...	41
7	cannot find symbol symbol: class NamingPolicy location: class net.sf.cglib.core.Abstra...	42
8	cannot find symbol symbol: class NamingPolicy location: class net.sf.cglib.core.Abstra...	110
9	cannot find symbol symbol: class NamingPolicy location: class net.sf.cglib.core.Abstra...	120
10	cannot find symbol symbol: class GeneratorStrategy location: class net.sf.cglib.core.A...	156
11	cannot find symbol symbol: class GeneratorStrategy location: class net.sf.cglib.core.A...	166
12	cannot find symbol symbol: variable DefaultGeneratorStrategy location: class net.sf.c...	41
13	cannot find symbol symbol: variable DefaultNamingPolicy location: class net.sf.cglib.co...	42
14	cannot find symbol symbol: class Predicate location: class net.sf.cglib.core.AbstractCl...	78
15	cannot find symbol symbol: variable DefaultNamingPolicy location: class net.sf.cglib.co...	112
16	cannot find symbol symbol: variable DefaultGeneratorStrategy location: class net.sf.c...	158
17	cannot find symbol symbol: class ClassReader location: class net.sf.cglib.core.Abstrac...	270
18	cannot find symbol symbol: variable ClassNameReader location: class net.sf.cglib.core...	270
19	cannot find symbol symbol: variable ReflectUtils location: class net.sf.cglib.core.Abstr...	272
20	cannot find symbol symbol: class CodeGenerationException location: class net.sf.cglib...	290

Gambar 4.11. Kesalahan yang terdeteksi pada file AbstractClassGenerator.java.

Pada gambar 4.9, Kolom **`PEMBERIAN KOMENTAR (DETEKSI)`** merupakan tempat untuk menyimpan data jumlah baris yang telah berhasil diberikan

komentar secara otomatis, sedangkan PEMBERIAN KOMENTAR (TIDAK TERDETEKSI) merupakan tempat untuk menyimpan data jumlah baris yang tidak berhasil diberikan komentar oleh aplikasi. Jika kolom PEMBERIAN KOMENTAR (TIDAK TERDETEKSI) berisi 0 dan kolom KOMPILASI bernilai 1 maka hal itu berarti bahwa PEMBERIAN KOMENTAR (DETEKSI) setiap baris program berhasil dilakukan pemberian komentar secara otomatis, sedangkan jika kolom KOMPILASI bernilai 1 dan kolom PEMBERIAN KOMENTAR (TIDAK TERDETEKSI) bernilai selain 0, maka pemberian komentar secara otomatis tetap berhasil dilakukan tetapi jumlahnya tidak seluruhnya diberikan komentar, hal ini dikarenakan belum termasuk pada proses REGEX yang dilakukan peneliti. Keberhasilan memberikan komentar secara otomatis adalah sebesar **94%**.

Kolom % merupakan tempat untuk menyimpan data nilai persentase keberhasilan pemberian komentar secara otomatis pada file java. Untuk mendapatkan nilai persentasi didapat dari :

$$\frac{\text{DETEKSI}}{\text{TIDAK TERDETEKSI}} \text{ x } 100$$

Gambar 4.12. Perhitungan nilai persentase pada data uji.

Dari keseluruhan data yang diuji total persentasi program yang dapat diberikan komentar secara otomatis adalah RATA-RATA (TOTAL TERDETEKSI)x100 dengan jumlah total **92%**.

4.2.2. Responden Mahasiswa

Aplikasi juga telah diuji pada pengguna. Pengguna aplikasi yang diuji adalah Mahasiswa Jurusan Teknik Informatika dengan persyaratan sebagai berikut:

1. Mahasiswa berada pada semester 2 sampai dengan semester 8.
2. Mahasiswa yang masih semester 1, tetapi dengan Sekolah Menengah jurusan rekayasa perangkat lunak atau teknik multimedia dan jaringan.

Proses uji aplikasi dilakukan oleh peneliti dengan dua cara yaitu melalui Kuesioner program dan melalui Aplikasi yang dibangun. Kuesioner program

terdiri dari 7 soal dimana soal program yang diuji adalah program dengan menggunakan bahasa pemrograman java (`*.java`), soal terlampir. Untuk mendapatkan hasil dari kuesioner, dilakukan dua kegiatan yaitu mengerjakan soal dan menghitung waktu mengerjakan soal.

Tabel 4. Waktu penyelesaian soal sebelum menggunakan aplikasi.

NAMA	**SEBELUM MENGGUNAKAN APLIKASI (Menit)**						
	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6	Prog7
Wahyu Dwi F	0.2	0.53	6.53	0.9	0.62	23	0.27
Ismoyo Ilham P	0.17	0.45	3.48	1.2	0.63	32	0.32
Ghanis Albashits	0.17	0.67	4.2	2.267	1.52	27.2	1.38
Kusuma Andriyanto	0.42	1.03	0.52	0.45	0.82	10.3	0.27
M. Iqbal Ismail	0.27	0.47	2.3	1.717	1.52	17.4	0.33
Vivi Nur Hidayati	0.4	0.92	2.22	1.367	1.08	21.2	0.42
Dwi Puji Lestari	0.17	1.4	3.27	0.6	1.57	20.2	0.25
Reka Darlian	0.2	0.67	1.82	0.9	1.87	19.4	0.38
Jaka Prasetya Sugianto	0.48	0.98	1.51	1.8	2.28	21.4	3.2

Tabel 5. Uji penyelesaian program sebelum menggunakan aplikasi.

NAMA	**SEBELUM MENGGUNAKAN APLIKASI**						
	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6	Prog7
Wahyu Dwi F	✓	✓	✓	✓	**x**	**x**	✓
Ismoyo Ilham P	✓	✓	✓	✓	✓	**x**	✓
Ghanis Albashits	✓	**x**	✓	✓	**x**	**x**	**x**
Kusuma Andriyanto	✓	**x**	**x**	**x**	**x**	**x**	✓
M. Iqbal Ismail	✓	✓	**x**	**x**	**x**	**x**	✓

Vivi Nur Hidayati	✓	✓	x	x	x	✓	✓
Dwi Puji Lestari	✓	✓	✓	✓	✓	✓	✓
Reka Darlian	✓	✓	x	x	x	✓	x
Jaka Prasetya Sugianto	✓	x	x	✓	x	x	x

Tabel 4. merupakan data waktu yang dibutuhkan oleh responden untuk menyelesaikan soal pemrograman yang ada. Jumlah responden adalah 9 orang mahasiswa. Program 1 keberhasilan mahasiswa dalam mengerjakan soal tersebut adalah 100% yang berarti dari 9 orang responden untuk soal pertama keseluruhan mahasiswa mampu mengerjakan dengan benar semuanya dengan waktu rata-rata yang dibutuhkan adalah 0.27 menit.

$$ProgNRata2 = \frac{ProgYR1 + ProgYR2 + \cdots + ProgYRn}{n}(menit)$$

$$Prog1Rata2 = \frac{0.2 + 0.17 + 0.17 + 0.42 + 0.27 + 0.4 + 0.17 + 0.2 + 0.48}{9}(menit)$$

$$Prog1Rata2 = \frac{2.47}{9}x100\% = 0.27\ menit$$

Untuk program 2, keberhasilan mahasiswa dalam mengerjakan soal adalah 67% dimana dari 9 responden terdapat 6 orang yang benar menjawab soal dan 3 yang salah dengan waktu rata-rata yang dibutuhkan adalah sebesar 0.79 menit.

$$Prog2Rata2 = \frac{0.53 + 0.45 + 0.67 + 1.03 + 0.47 + 0.92 + 1.4 + 0.67 + 0.98}{9}(menit)$$

$$Prog2Rata2 = \frac{7.12}{9}(menit) = 0.79\ menit$$

Untuk program 3, keberhasilan mahasiswa dalam mengerjakan soal adalah 44% dimana dari 9 responden terdapat 4 orang yang benar menjawab soal dan 5 orang yang salah dengan waktu rata-rata yang dibutuhkan adalah sebesar 2.87 menit.

$$Prog3Rata2 = \frac{6.53 + 3.48 + 4.2 + 0.52 + 2.3 + 2.2. + 3.3 + 1.8 + 1.5}{9}(menit)$$

$$Prog3Rata2 = \frac{25.8}{9}(menit) = 2.87\ menit$$

Untuk program 4, keberhasilan mahasiswa dalam mengerjakan soal adalah 56% dimana dari 9 responden terdapat 5 orang yang benar menjawab soal dan 4 orang

yang menjawab salah dengan waktu rata-rata yang dibutuhkan adalah sebesar 1.24 menit.

$$Prog4Rata2 = \frac{0.9 + 1.2 + 2.27 + 0.45 + 1.72 + 1.37 + 0.6 + 0.9 + 1.8}{9}(menit)$$

$$Prog4Rata2 = \frac{11.2}{9}(menit) = 1.24\ menit$$

Untuk program 5, keberhasilan mahasiswa dalam mengerjakan soal adalah 22% dimana dari 9 responden terdapat 2 orang yang menjawab benar dan 7 menjawab salah dengan waktu rata-rata yang dibutuhkan adalah sebesar 1.32 menit.

$$Prog5Rata2 = \frac{0.62 + 0.63 + 1.52 + 0.82 + 1.52 + 1.08 + 1.57 + 1.87 + 2.28}{9}(menit)$$

$$Prog5Rata2 = \frac{11.9}{9}(menit) = 1.32\ menit$$

Untuk program 6, keberhasilan mahasiswa dalam mengerjakan soal adalah 22% dimana dari 9 responden terdapat 2 orang yang menjawab benar dan 7 menjawab salah dengan waktu rata-rata yang dibutuhkan adalah sebesar 21.3 menit.

$$Prog6Rata2 = \frac{23 + 32 + 27.2 + 10.3 + 17.4 + 21.2 + 20.2 + 19.4 + 21.4}{9}(menit)$$

$$Prog6Rata2 = \frac{192}{9}(menit) = 21.3\ menit$$

Untuk program 7, keberhasilan mahasiswa dalam mengerjakan soal adalah 22% dimana dari 9 responden terdapat 2 orang yang menjawab benar dan 7 menjawab salah dengan waktu rata-rata yang dibutuhkan adalah sebesar 21.3 menit.

$$Prog7Rata2 = \frac{0.27 + 0.32 + 1.38 + 0.27 + 0.33 + 0.42 + 0.25 + 0.38 + 3.2}{9}(menit)$$

$$Prog7Rata2 = \frac{6.82}{9}(menit) = 0.76\ menit$$

Setelah uji tulis, masing-masing responden diperlihatkan dan mencoba secara langsung aplikasi yang dibangun oleh penulis. Setelah mencoba aplikasi lalu responden mengerjakan kembali soal yang sama dengan menggunakan pembantu aplikasi yang dibangun penulis. Dari hasil uji coba kedua dengan menggunakan aplikasi yang dibangun penulis, masing-masing responden terdapat peningkatan kemampuan dalam menjawab soal.

Tabel 6. Data kemampuan responden menyelesaikan soal program java setelah menggunakan bantuan aplikasi komentar semi otomatis

SEBELUM MENGGUNAKAN APLIKASI							
RESPONDEN	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6	Prog7
1	✓	✓	✓	✓	x	x	✓
2	✓	✓	✓	✓	✓	x	✓
3	✓	x	✓	✓	x	x	x
4	✓	x	x	x	x	x	✓
5	✓	✓	x	x	x	x	✓
6	✓	✓	x	x	x	✓	✓
7	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	x	x	x	✓	x
9	✓	x	x	✓	x	x	x

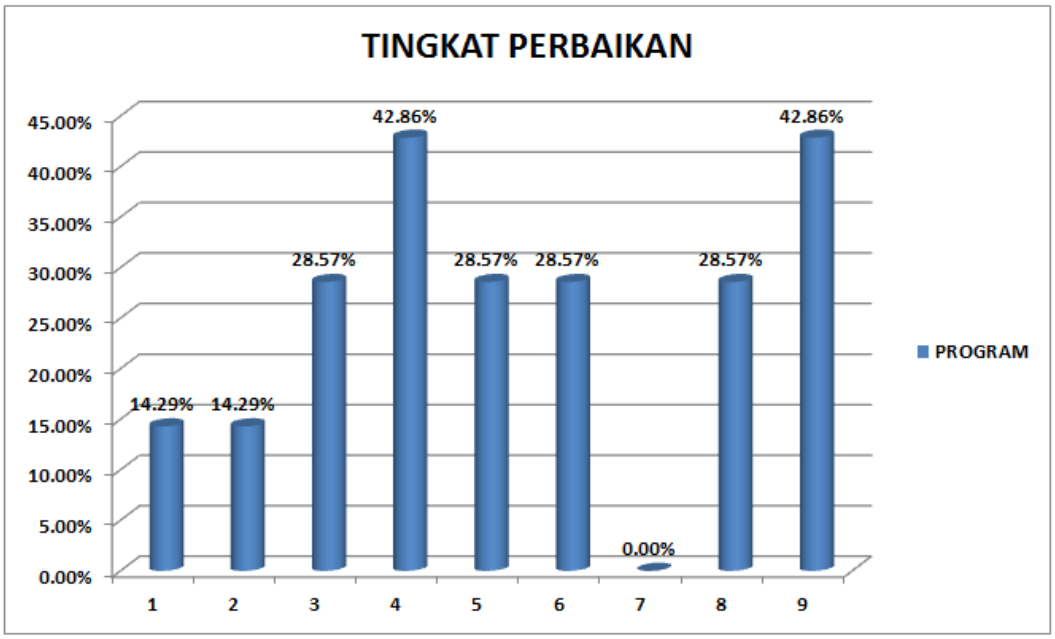
Tabel 7. Data kemampuan responden menyelesaikan soal program java setelah menggunakan bantuan aplikasi komentar semi otomatis

SETELAH MENGGUNAKAN APLIKASI							
RESPONDEN	Prog1	Prog2	Prog3	Prog4	Prog5	Prog6	Prog7
1	✓	✓	✓	✓	✓	x	✓
2	✓	✓	✓	✓	✓	✓	✓
3	✓	x	✓	✓	✓	x	✓
4	✓	x	✓	✓	✓	x	✓
5	✓	✓	x	✓	✓	x	✓
6	✓	✓	x	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	x	✓	x	✓	✓
9	✓	x	✓	✓	✓	x	✓

Pada tabel 6 dan tabel 7 dapat dilihat bahwa pada tabel 7. responden saat mengerjakan soal program java dengan dibantu aplikasi komentar semi otomatis terdapat peningkatan kemampuan dalam menjawab soal, dimana responden 1 mampu memperbaiki satu jawaban soal program, yaitu pada program 5. Hal ini berarti responden 1 kemampuan menjawab soal meningkat 14.29% dari sebelumnya. Responden 2 mendapatkan hasil yang sama dengan responden 1 dimana mampu memperbaiki jawaban dari soal 6 dan hal ini berarti responden 2 telah mampu meningkatkan kemampuan dalam menjawab soal program java sebesar 14.29%.

Responden 3 pada pada awalnya hanya mampu menyelesaikan soal program java sejumlah 3 soal dan setelah menggunakan aplikasi yang ada responden 3 mampu memperbaiki 2 jawaban yaitu pada program 5 dan program 7. Hal ini berarti responden 3 dengan menggunakan aplikasi pembantu yang dibangun mampu meningkat sebesar 28.57%. Responden 4 awalnya hanya mampu menyelesaikan 2 soal program java, setelah dibantu dengan menggunakan aplikasi komentar semi otomatis, responden 4 mampu memperbaiki jawaban soal pada program 3, 4, dan 5 yang berarti responden 4 kemampuannya dalam menyelesaikan soal program java meningkat sebesar 42.86%.

Responden 5 dan 6 mampu meningkatkan kemampuannya dalam menyelesaikan soal program java sebesar 2 soal atau sebesar 28.57% , yaitu pada soal program 4 dan program 5. Untuk responden 7 tidak ada peningkatkan karena pada awalnya responden 7 telah mampu menyelesaikan soal program java keseluruhan tetapi responden 7 secara rata-rata waktu yang dibutuhkan untuk memahami keseluruhan soal lebih cepat 25%. Untuk responden 8 yang awalnya hanya mampu menyelesaikan soal program java sebesar 3 soal, setelah menggunakan aplikasi komentar semi otomatis, responden mampu menambahkan jawaban yang benar sebesar 2 soal lagi, sehingga kemampuan responden 2 setelah menggunakan aplikasi komentar semi otomatis meningkat sebesar 28.57%. Responden 9 yang sebelumnya hanya mampu menyelesaikan soal progra java sebanyak 2 soal, setelah menggunakan aplikasi komentar semi otomatis kemampuannya meningkat sebesar 42.86% atau mampu menambah jawaban soal program java yang benar sebanyak 3 soal, yaitu diprogram 3, 5 dan 7. Data dapat dilihat pada gambar 4.12.



Gambar 4.13. Peningkatkan kemampuan responden menyelesaiakan soal

4.2.3. Responden Programmer dan Dosen

Untuk mendapatkan hasil lebih dari aplikasi yang dibangun sehubungan dengan kegunaannya, maka dilakukan uji aplikasi oleh dosen di jurusan teknik informatika yang telah memahami pemrograman dan atau telah mengajar pemrograman minimal 1 tahun. Selain dosen jurusan teknik informatika, aplikasi juga diuji oleh programmer yang minimal telah mempunyai kemampuan memprogram minimal 1 tahun pada bidang yang sama.

Uji aplikasi oleh dosen, dilakukan oleh 1 orang dosen dengan cara mencoba aplikasi dan fitur-fitur yang ada, mulai dari proses kompilasi, proses penghasilan komentar berbahasa Indonesia secara otomatis, fitur edit komentar yang ada serta fitur penampilan struktur program. Terdapat 4 pertanyaan yang harus dijawab oleh dosen dan programmer dengan cara memberikan nilai antara 1 sampai 5, dimana :

1. Nilai 1 menyatakan program sangat tidak setuju (STS)
2. Nilai 2 menyatakan program tidak setuju (TS)
3. Nilai 3 menyatakan ragu-ragu (R)
4. Nilai 4 menyatakan setuju (S)
5. Nilai 5 menyatakan sangat setuju (SS)

Bobot untuk masing-masing nilai adalah:

```
STS = 1
TS = 2
R = 3
S =4
SS = 5
```

Tabel 8. Pertanyaan untuk uji aplikasi bagi programmer dan dosen

PERTANYAAN
Menurut anda, apakah aplikasi yang ada dapat membantu proses pemahaman program.
Menurut anda, apakah program sudah menjalankan fungsi utamanya.
Menurut anda, komentar yang dihasilkan oleh aplikasi bisa membantu proses pemahaman program.
Menurut anda, apakah komentar yang dihasilkan tempatnya sudah sesuai.

Tabel 9. Presentasi Penilaian Indeks

PRESENTASE NILAI	**KETERANGAN**
0 – 19.99%	Sangat Tidak Setuju
20 – 39.99%	Tidak Sejutu
40 – 59.99%	Ragu-Ragu
60 – 79.99%	Setuju
80 – 100%	Sangat Setuju

Dari hasil kuesioner untuk pertanyaan pertama didapatkan data bawah responden 1sampai 3 memilih nilai 5, sehingga didapatkan:

```
Total   = (Q1*B1)+(Q2*B2)+(Q3*B3)+(Q4*B4)+(Q5*B5)
Total   = (0*1)+(0*2)+(0*3)+(0*4)+(3*5)
Total   = 15
```

Dimana :

```
Q   = Jumlah (Quantity)
B   = Bobot Nilai
```

$$\text{Skor Tertinggi} = B_{(tertinggi)} * \text{Total Responden}$$
$$= 5 * 3$$
$$\text{Skor Tertinggi} = 15.$$
$$\text{Skor Terendah} = B_{(terendah)} * \text{Total Responden}$$
$$= 1 * 3$$
$$\text{Skor Terendah} = 3$$

Untuk mengetahui "**apakah aplikasi komentar semi otomatis dapat membantu proses pemahaman program atau tidak**" maka harus didapatkan harga indeks dari pertanyaan tersebut dimana indeks didapatkan dari `Total` dibagi dengan `Skor Tertinggi` dikalikan dengan `100%`, maka didapatkan :

```
Indeks  = (15 / 15) * 100%
Indeks  = 100%
```

Dari hasil perhitungan didapatkan nilai Indeks adalah **100%**, maka dapat dikatakan bahwa Aplikasi sangat dapat membantu proses pemahaman program.

Pertanyaan kedua tentang "**program telah menjalankan fungsi utamanya**", responden pertama memberikan nilai 4, responden kedua memberikan nilai 5 dan responden ketiga memberikan nilai 4, sehingga total nilai dan indeks yang didapat dari pertanyaan kedua adalah:

```
Total  = (Q1*B1)+(Q2*B2)+(Q3*B3)+(Q4*B4)+(Q5*B5)
Total  = (0*1)+(0*2)+(0*3)+(2*4)+(1*5)
Total  = 8 + 5 = 13

Skor Tertinggi  = B(tertinggi) * Total Responden
                = 5 * 3
Skor Tertinggi  = 15.
Skor Terendah   = B(terendah) * Total Responden
                = 1 * 3
Skor Terendah   = 3

Indeks  = (13 / 15) * 100%

Indeks  = 87%
```

Dari hasil perhitungan diatas didapatkan bahwa nilai Indeks adalah **87%.** Jika dilihat sesuai tabel 4.6. dapat dikatakan bahwa responden dosen dan programmer "**Sangat Setuju**" jika aplikasi telah menjalankan fungsi utamanya.

Pada pertanyaan ketiga, yaitu tentang "**Komentar dapat yang dihasilkan oleh aplikasi dapat membantu proses pemahaman program**", para responden memberikan nilai 5, sehingga total nilai dan indeks

```
Total  = (Q1*B1)+(Q2*B2)+(Q3*B3)+(Q4*B4)+(Q5*B5)
Total  = (0*1)+(0*2)+(0*3)+(0*4)+(3*5)
Total  = 15

Skor Tertinggi  = B(tertinggi) * Total Responden
                = 5 * 3
Skor Tertinggi  = 15.
Skor Terendah   = B(terendah) * Total Responden
                = 1 * 3
Skor Terendah   = 3

Indeks  = (15 / 15) * 100%

Indeks  = 100s%
```

Dari hasil perhitungan diatas didapatkan bahwa nilai Indeks adalah **100%.** Jika dilihat sesuai tabel 4.6. dapat dikatakan bahwa responden dosen dan programmer "**Sangat Setuju**" jika komentar yang dihasilkan aplikasi bisa membantu proses pemahaman program.

Pertanyaan keempat tentang "**komentar yang dihasilkan tempatnya sudah sesuai**", responden pertama memberikan nilai 4, responden kedua memberikan nilai 5 dan responden ketiga memberikan nilai 4, sehingga total nilai dan indeks :

```
Total  = (Q1*B1)+(Q2*B2)+(Q3*B3)+(Q4*B4)+(Q5*B5)
Total  = (0*1)+(0*2)+(0*3)+(2*4)+(1*5)
Total  = 8 + 5 = 13

Skor Tertinggi  = B(tertinggi) * Total Responden
                = 5 * 3
Skor Tertinggi  = 15.
Skor Terendah   = B(terendah) * Total Responden
                = 1 * 3
Skor Terendah   = 3

Indeks  = (13 / 15) * 100%

Indeks  = 87%
```

Dari hasil perhitungan diatas didapatkan bahwa nilai Indeks adalah **87%.** Jika dilihat sesuai tabel 4.6. dapat dikatakan bahwa responden dosen dan programmer "**Sangat Setuju**" jika komentar yang dihasilkan tempatnya sudah sesuai.

BAB V
KESIMPULAN

5.1. Kesimpulan

Dari tesis yang telah dilakukan oleh peneliti, dapat diambil kesimpulan antara lain:

1. Peneliti telah mampu melakukan pemisahan antara program dan yang bukan program pada bahasa pemrograman java.
2. Proses pemberian komentar semi otomatis pada bahasa pemrograman java telah mampu dilaksanakan oleh aplikasi yang dibangun peneliti dengan tingkat keberhasilan sebesar 94% dan dengan rata-rata keberhasilan sebesar 92%.
3. Aplikasi Komentar Semi Otomatis dapat memberikan kemudahan pemahaman terhadap bahasa pemrograman java antara 14.29% sampai dengan 42.86%.

5.2. Saran

Dari tesis yang telah dilakukan oleh peneliti, saran yang dapat diberikan oleh peneliti sehubungan dengan aplikasi yang dibangun antara lain:

1. Aplikasi yang dibangun masih belum dapat menangani seluruh isi dari program java, karena proses pemolaan.
2. Aplikasi masih belum dapat memberikan hubungan antara method, class dan isi dari method dan class yang ada.

[HALAMAN INI SENGAJA DIKOSONGKAN]

DAFTAR PUSTAKA

Aakanksha Pandey, Nilay Khare and Akhtar Rasool, 2012, Efficient Design and Implementation of DFA Based Pattern Matching on Hardware, International Journal of Computer Science Issues, Vol.9, Issue 2, No.1,pp. 286-290, ISSN: 1694-0814.

Alina Misrah, Subhrakantha Panda and Dishant Munjal, 2013, Dynamic Slicing of Aspect-Oriented UML Communication Diagram, International Journal of Computer Science and Informatics, Volume 3, Issue 2, ISSN: 2231-5292.

Anas Bassam Al-Badareen, Moh Hasan Selamat, Marzanah A. Jabar, Jamilah Din, Sherzod Turaev, 2011, The Impact of Software Quality on Maintenance Process, International Journal of Computers, Issue 2. Volume 5, 2011.

Andrew S. Tanenbaum, 2008, Modern Operating Systems Third Edition, Prentice Hall.

Ari Faradisa, 2015, Identifikasi Emosi dari Status Pengguna Blackberry Messenger Menggunakan Metode Naïve BayesClassifier, Jurusan Teknik Informatika, ITATS.

B. Suganya and P. Kirthika, 2014, Analysis on Clustering Techniques Based on Similarity of Text Documents, International Journal of Advance Research in Computer Science and Management Studies, Volume 2, Issue 2.

Barry M and Browne M, 2005, Understanding Search Engines: Mathematical Modelling and Text Retrieval, SIAM.

Beat Fluri, Michael Wursch and Harald C. Gall, 2007, Do Code and Comments Co-Evolve? On the Relation Between Source Code and Comment Changes, Working Conference on Reverse Engineering.

Bee Bee Chua and June Verver, 2010, Examining Requirements Change Rework Effort: A Study, International Journal of Software Engineering & Applications, Vol.1. No.3, July 2010

Carlos Noguera, Coen De Roover, Andy Kellens, Viviane Jonckers, 2012, Code Querying By UML, IEEE

Chukma Indahyani, 2015, Efisiensi Source Code pada bahasa pemrograman java dengan menggunakan metode static slicing, Jurusan Teknik Informatika,ITATS.

Claire Knight and Malcolm Munro, 2002, Program Comprehension Experiences With GXL: Comprehension for Comprehension, IEEE.

Cui Linhai, 2014, An Innovative Approach for Regular Expression Matching Based on NoC Architecture, International Journal of Smart Home, Vol.8 No.1 , PP. 45-52

Cynthia L. Corritore and Susan Wiedenbeck, 2001, An Exploratory study of program comprehension strategies of procedural and object -oriented programmers,International Journal Human-Computer Studies.

D. Bala Krishna Kamesh, A.K. Vasishtha, JKR Sastry, and V. Chandra Prakash, 2013, Estimating the Failure Ratesof Embedded Software Through Program Slices, International Journal of Systems and Technologies, ISSN: 0974-2107

Denis Pinheiro, Marcelo Maia, Raquel Prates, and Roberto Bigonha, 2008, Assessing Program Comprehension Tools With The Communicability Evaluation Method, Workshop de Manutencao de Software Moderna.

Dennis Giffhorn and Christian Hammer, 2007, An Evaluation of Slicing Algorithms for Concurrent Programs, http://www.ieee-scam.org/2007/papers/04.pdf.

Dominik D. Freydenberger and Timo Kotzing, 2012, Fast Descriptive Generalization of Restricted Regular Expressions and DTDs, 22nd Theorietag Automata and Formal Languages, Prague Proceedings.

Dominik D. Freydenberger, 2011, Extended Regular Expressions: Succinctness and Decidability, 28th Symposium on Theoretical Aspects of Computer Science, Germany.

Drexel University, 2015, Dependable Software System, Drexel University, https://www.cs.drexel.edu/~spiros/teaching/CS576/slides/5.slicing.ppt.

Durin Savina, Istrat Visnja, and Terek Edit, 2012, Achieving Competence of Domestic Business Organisations Thourgh International Standard

Implementation, Journal of Engineering Management and Competitiveness, Vol.2, No.1,pp. 33-37, ISSN 2217-8147.

Ezekiel Okike and Maduka Attamah, 2010, Evaluation of Jflex Scanner Generator Using Form Fields Validity Checking, International Journal of Computer Science Issues, Vol.7, Issue 3, No.4, ISSN: 1694-0784.

Fatiha Boubekeur and Wassila Azzoug, 2013, Concept-Based Indexing in Text Information Retrieval, International Journal of Computer Science & Information Technology, Vol.5 No.1, DOI : 10.5121/ijcsit.2013.5110

Fotis Lazarinis, 1998, Combining Information Retrieval with Information Extraction for Efficient Retrieval of Calls for Papers, IRSG.

Fumiaki Umemori, Kenji Konda, Reishi Yokomori and Katsuro Inoue, 2003, Design and Implementation of Bytecode-based Java Slicing System,http://barbie.uta.edu/~jli/Resources/Resource%20Provisoning%26Performance%20Evaluation/68.pdf.

G. Antoniol, G. Canfora, G Casazza, A. De Lucia and E. Merlo, 2000, Tracing Object-Oriented Code Into Functional Requirements, IEEE.

Guillermo De Ita Luna, J. Raymundo Marcial-Romero, Pilar Pozos-Parra, and Jose A. Hernandez, 2015, Using Binary Patterns for Counting Falsifying Assignments of Conjunctive Forms, Electronic Notes in Theoritical Computer Science315, pp. 17-30, 1571-0661.

Hamideh Sabouri and Marjan Sirjani, 2010, Actor-Based Slicing Techniques for Efficient Reduction of Rebeca Models, Science of Computer Programming, Elsevier, 2010.

Hitesh Sajnani, 2012, Automatic Software Architecture Recovery: A Machine Learning Approach, IEEE

Jan Goyvaerts and Steven Levithan, 2012, Regular Expressions Cookbook, Second Edition, O'Rreilly.

Janet Feigenspan, 2011, Program Comprehensionof Feature-Oriented Software Development, University of Magdeburg, Germany.

Janet Feigenspan, Christian Kastner, Jorg Leibig and Sven Apel, Stefan Hanenberg, 2012, Measuring Program Experience, IEEE

Janet Feigenspan, Nobert Siegmund, Andreas Hesselberg, Markus Koppen, 2011, PROPHET: Tool Infrastructure To Support Program Comprehension Experiments.

Janet Feigenspan, Sven Apel, Jorg Liebig, and Christian Kastner, 2011, Exploring Software Measure to Assess Program Comprehension.

Janet Feignespan, Sven Apel, Jorg Liebig and Christian Kastner, 2011, Exploring Software Measures to Assess Program Comprehension, IEEE.

Jie Liu, Jigui Sun, and Shengsheng Wang, 2006, Pattern Recognition: An Overview, International Journal of Computer Science and Network Security, Vol. 6, No. 6.

Jipeng Qiang, Dan Guo, Yuan Fang, Weidong Tian and Xuegang Hu, 2013, Multiple Pattern Matching with Wildcards on one-off Condition, Journal of Computational Information Systems, ISSN: 1553-9105.

Jonathan L. Maletic and Andrian Marcus, 2001, Supporting program comprehension Using Semantic and Structural Information, IEEE.

Jyoti Arora and Randhika Thapar, 2014, Debugging Using Static Program Slices, International Journal of Computer inforamtics & Technological Engineering, Volume 1, Issue 8, ISSN : 2348-8557.

K. Koteswara Rao, Srinivasan Nagaraj and Dr. GSVP Raju, 2011, The Efficient Way to Identify the Regular Expession in Text Database, International Journal of Advanced Science Technology, Vol.35, pp. 11-28.

Kshitij Sharma, Patrick Jermann, Marc-Antonie Nussli, and Pierre Dillenbourg, 2012, Gaze Evidence for Different Activities in Program Understanding.

Kumar Sambit Patra, 2013, optimizing AST Node for JavaScript Compiler A Lightweight Interpreter for Embedded Device, Journal of Computers, Vol. 8, No.2, doi: 10.4304/jcp.8.2.349-355.

Madhulika Arora, S.S. Sarangdevot, Vikram Singh Rathore, Jitendra Deegwal and Sonia Arora, 2011, Refactoring, Way for Software Maintenance, International Journal of Computer Science, Vol. 8, Issue 2, March 2011

Mark Tabladillo, 2012, Regular Expressions in SAS® Enterprise Guide®,SAS Global Forum 2012, pp.1-10.

Maryanne Fisher, Anthony Cox and Lin Zhao, 2006, Using Sex Differences to Link Spatial Cognition and Program Comprehension, IEEE.

Michael Fitzgerald, 2012, Introduction to Regular Expression, O'Reilly, pp. 13

Michael P. O'Brien , Teresa M. Shaft, and Jim Buckley, 2001, An Open-Source Analysis Schema for Identifying Software Comprehension Process, 13th Workshop of the Psychology of Programming Interest Group.

Michael P. O'Brien and Jim Buckley, 2001, Inference-based and Expectation-based Processing in Program Comprehension, IEEE.

Mohammad Skhoukani and Rawan Abu Lail, 2012, The Importance of Restructuring Software Engineering Education Startegies in order to Minimize the Gap Between Academic supply Chain and Industry Demand in Software Engineering Field, International Journal of Reviews in Computing, Vol.11, 30 September 2012

N. Sasirekha, A. Edwin Robert, and M. Hemalatha, 2011, Program Slicing Techniques and Its Applications, International Journal of Software Engineering & Applications, Vol. 2, No. 3, DOI: 10.5121.

R. Bhuvaneswari and K. Kalai Selvi, 2012, Software Support for XML Schema Design Patterns and Pattern Matching of XML Schemas, International Journal of Scientific and Research Publications, Volumen 2, Issue 2, ISSN : 2250-3153.

Rashmi Sinha and Ashish Dewangan, 2012, Transmutation of Reguler Expression to Source Code Using Code Generators, International Journal of Computer Trends and Technology, Volume 3, Issue 6, ISSN: 2231-2803.

Ricardo Colomo-Palacios, Eduardo Fernandes, Pedro Soto-Acosta and Marc Sabbagh, 2011, Software Product Evolution for Intellectual Capital Management: The Case of Meta4 PeopleNet, International Journal of Information Management.

Rifqie Shahab, 2012, Pengertian, Macam-macam dan Contoh Part of Speech, http://bahasainggrisonlines.blogspot.com/2012/12/parts-of-speech.html, diakses 24 April 2014.

Rupinder Singh and Vinay Arora, 2013, Literature Analysis on Model based Slicing, International Journal of Computer Applications, Volume 70, No. 16.

Sandeep Kumar Satapathy, Shruti Mishra and Debahuti Mishra, 2010, Search Technique Using Wildcards or Truncation: A Tolerance Rough Set Clustering Approach, International Journal of Advanced Computer Science and Applications, Vol. 1, No.4, pp. 73-77.

Saurabn Jain, Deepak Signh Tomar, and Divya Rishi Sahu, 2012, Detection of JavaScript Vulnerability at Client Agen, International Journal of Scientific & Technology Research, Volume 1, Issue 7, ISSN : 2277-8616.

Sergio Cozzetti B. De Souza, Nicolas Anquetil, Kathai M. De Oliveira, 2005, A Study of the Documentation Essential to Software Maintenance, ACM.

Shedd, M., 2008, More Letter-Sound Knowledge, Vocabulary and Morphology, East Lansinb, Michigan.

Shivakumar Swamy N and Sanjeev C. Lingareddy, 2015, A Semantic Based Search for Test Case Repository, International Journal of Research Science & Management, ISSN: 2349-5197, pp. 11-20.

Siegmund Janet, 2012, Framework for Measuring Program Comprehension, Dissertation Universitat Magdeburg, Jerman.

Sitalakshmi Venkatraman and Shadana J. Kamatkar, 2013, Intelligent Information Retrieval and Recomender System Framework, International Journal of Future Computer and Communication, Vol.2, No. 2.

Sonam Jain and Sandeep Poonia, 2013, A New Approach of Program Slicing: Mixed S-D(Static & Dynamic) Slicing, International Journal of Advaced Reseearch in Computer and Communication Engineering, Vol.2, Issue 5, ISSN: 2278-1021.

Sugandha Chakraverti, Sheo Kumar, S.C. Agarwal, Ashish Kumar Cakraverti, 2012, Modified Cocomo Model for Maintenance Cost Estimation of Real Time System Software, International Journal of Computer Science and Network, Vol.1, Issue 1, February 2012

Takashi Ishio, Shogo Etsuda and Katsuro Inoue, 2012, A Lightweight Visualizationof Interprocedural Data-Flow Paths for Source Code Reading, IEEE

Tim Frey, Marius Gelhausen and Gunter Saake, 2011, Categorization of Concern: A Categorical Program Comprehension Model,ACM

Tobias Roehm, Rebecca Tiarks, Rainer Koschke and Walid Maalej, 2012, How do Professional Developers Comprehend Software ?,IEEE

Tony Stubblebine, 2007, Regular Expression Pocket Reference, Second Edition, O'Reilly.

Vennila Ramalingam and Susan Wiedenbeck, 1997, An Empirical Study of Novice Program Comprehension in the Imperative and Object-Oriented Styles.

Vimala Balakrishnan and Kian Ahmadi, 2013, An Architecture to Enchance Post Retrieval Document Relevancy Using Integrated Techniques, International Journal of Machine Learning and Computing, Vol.3 No.2, DOI: 10.7763/IJMLC.2013.V3.309.

Walid M. Ibrahim, Nicolas Betternburg, Bram Adams, Ahmed E. Hassan, 2012, On the Relationship Between Comment Update Practices and Software Bugs, Elsevier.

Yahya Tashtoush, Zeinab Odat, Izzat Alsmadi and Maryan Yatim, 2013, Impact of Programming Features on Code Readability, International Journal of Software Engineering and Its Applications, Vol.7, No. 6, ISSN: 1738-9984 IJSEIA.

Youssef Bassil and Paul Semaan, 2012, Semantic-Sensitive Web Information Retrieval Model for HTML Documents, European Journal of Scientific Research, Vol. 69 No. 4, ISSN: 1450-216X.

Zhizhong Jiang and Peter Naude, 2007, An Eximination of the Factors Influencing Software Development Effort, International Journal of Computer and Information Engineering.

Zohreh Sharafi, Zephryin Soh, Yann-Gael Gueheneuc and Giuliano Antonio, 2012, Women and Man - Different but Equal: On the Impact of Identifier Style on Source Code Reading, IEEE

[HALAMAN INI SENGAJA DIKOSONGKAN]

LAMPIRAN 1 Kuesioner Bagi Mahasiswa

Dari soal dibawah ini, dapatkan hasil dari Program Java dengan menuliskan OUTPUT (keluaran) dari Program 1 sampai dengan Program 7 pada lembar yang telah disediakan.

PROGRAM 1

```
public class examples001{
      public static void main(String[] args)
      {
            System.out.println("HelloWorld");
      }
}
```

PROGRAM 2

```
import java.io.*;

public class examples002{
      public static void main(String[] args){
            File file=new File("Test.txt");

            try{
            PrintWriter output=new PrintWriter(file);
            output.println("Mike Dark Cascos");
            output.println(42);
            output.close();
            } catch(IOException ex){
                  System.out.printf("ERROR: %s\n", ex);
            }

      }
}
```

PROGRAM 3

```
import java.util.Scanner;

public class examples003 {

public static void main(String[] args) {

int n, m, r;

Scanner dataInput = new Scanner(System.in);

System.out.print("Masukkan nilai n : ");
n = dataInput.nextInt();
System.out.print("Masukkan nilai m : ");
m = dataInput.nextInt();
```

```
System.out.println();

r = m % n;

do{
n = m;
m = r;
r = n % m;
} while (r != 0);

System.out.println("Hasi Eksekusi = "+m);
}
}
```

PROGRAM 4

```
public class examples004 {

public static void main(String[] args) {

int num1 = 324;
int num2 = 234;

if(num1 > num2){
      System.out.println(num1 + " is greater than " + num2);
}
else if(num1 < num2){
       System.out.println(num1 + " is less than " + num2);
}
else{
      System.out.println(num1 + " is equal to " + num2);
}
  }
}
```

PROGRAM 5

```
import java.io.*;

public class examples005 {

  public static void main(String[] args) {

    File file1 = new File("C://FileIO//demo1.txt");

    File file2 =  new File("C://FileIO//demo1.txt");

     if(file1.compareTo(file2) == 0)
     {
       System.out.println("Both paths are same!");
     }
     else
```

```
     {
       System.out.println("Paths are not same!");
     }
  }
}
```

PROGRAM 6

```
public class examples006 {
    static void Sort(int[] arr) {
        int n = arr.length;
        int temp = 0;
         for(int i=0; i < n; i++){
                 for(int j=1; j < (n-i); j++){
                          if(arr[j-1] > arr[j]){
                                  temp = arr[j-1];
                                  arr[j-1] = arr[j];
                                  arr[j] = temp;
                          }

                 }
         }

    }
    public static void main(String[] args) {
                int arr[] ={3,60,35,2,45,320,5};

                System.out.println("Array Before Sort");
                for(int i=0; i < arr.length; i++){
                        System.out.print(arr[i] + " ");
                }
                System.out.println();

                Sort(arr);

                System.out.println("Array After Sort");
                for(int i=0; i < arr.length; i++){
                        System.out.print(arr[i] + " ");
                }

        }
}
```

PROGRAM 7

```
class Percabangan {
  public static void main(String[] args) {
    boolean prosesBelajar = true;

    if (prosesBelajar) {
      System.out.println("Programmer");
    }
```

```
    else {
      System.out.println("Bukan Programmer");
    }
  }
}
```

LAMPIRAN 2 Kuesioner Bagi Dosen dan Programmer

Centanglah salah satu kolom dari nilai yang ada ditabel. Nilai yang dapat dipilih adalah antara 1 sampai dengan 5. Semakin besar nilainya maka semakin baik.

PERTANYAAN	**1**	**2**	**3**	**4**	**5**
Menurut anda, apakah aplikasi yang ada dapat membantu proses pemahaman program.					
Menurut anda, apakah program sudah menjalankan fungsi utamanya.					
Menurut anda, komentar yang dihasilkan oleh aplikasi bisa membantu proses pemahaman program.					
Menurut anda, apakah komentar yang dihasilkan tempatnya sudah sesuai.					

[HALAMAN INI SENGAJA DIKOSONGKAN]

LAMPIRAN 3 DAFTAR FUNGSI REGEX

REGEX	FUNGSI	CONTOH
(".*[^\\]")|("[^"]*")	Mendeteksi String	"" "Buku" "Microsoft Office"
package\s+[\w.]+\s*;	Mendeteksi keberadaan Package	Package org; Package org.sample; Package org.sample_page;
PatternPackage.replaceAll("package\s+|;", "");	Mendeteksi nama Package	~~Package~~ org~~;~~ ~~Package~~ org.sample~~;~~ ~~Package~~ org.saple_package~~;~~
(?:^|\n)\s*import.+;	Mengambil Pustaka	import org.MyClass; import org.sample_package.MyClass2; import javax.swing.*;
Pattern.compile("import\s+").matcher(PatternImport).replaceAll("");	Mengambil nama Pustaka	~~import~~ org.MyClass; ~~import~~ org.sample_package.MyClass2; ~~import~~ javax.swing.*;
[\w$*]+\s*;$	Mengambil Class pada pustaka saja	MyClass; MyClass2;
PatternPustakaOnly.replaceAll("[\w$*]+\s*;$", "")	Mengambil package pada pustaka	org~~.MyClass;~~ org.sample_package~~.MyClass2;~~ javax.swing~~.*;~~

(?:^|\n)\s*.*((class)|(interface)|(enum)).+\\{	Mengambil class, interface dan enum	public class Sample_Class {
		class SampleClassWithExtends extends JFrame{
		abstract class AbstractClass{
		Interface InterfaceClass extends ExtendsClass{
		enum Enumeration{
		public enum PublicEnum{

(class)|(interface)|(enum)	Mengambil tipe dari class, interface dan enum	Class Interface enum
((class)|(interface)|(enum))\\s+[A-Za-z0-9_$]+	Mengambil Type dan nama class	class Sample_Class SampleClass class AbstractClass interfaceClass interface

		PublicInterface enum Enumeration enum PublicEnum
extends\\s+[A-Za-z0-9_$.]+	Mengambil extend claass jika terdapat pewarisan class	extends Jframe extends org.ExtendsClass extends ExtendsClass
implements\\s+([a-zA-Z0-9$_.]+,\\s*)*[a-zA-Z0-9$_]+	Mengambil implements jika class terdapat pewarisan interface	

^\\s*super\\s*\\([\\s\\S]*\\)\\s*;	Mendeteksi super	super();
		super(this);
		super(a.getText());
		super("text");

\\([\\s\\S]*\\)	Mengambil sesuatu yang ada dalam kurung	() (this) (a.getText()) ("text")
((try)|(finally))\\s*\\{	Mengambil try dan finally	try { finally {

(?:^|\n)\s*^\\s*([\\w$]+\\s+)*([\\w$]+\\s*)\\(([\\w$,\\[\\]<>\\(\\).])*\\)\\s*;	Mengambil method bila class adalah interface	public static void method(param param1, param param2);
		String method();
		int method(param param1);
		private void Method(param[] array, param<element> param2);

([\\w$]+\\s*)\\(([\\w$,\\[\\].])*\\)	Ambil nama method dan paramnya	method(param param1, param param2) method() method(param param1) Method(param[] array, param<element> param2)
MethodPattern.replaceAll("\\(([\\w$,\\[\\].])*\\)","");	Mengambil nama method	method~~(param param1, param param2)~~ method~~()~~ method~~(param param1)~~ Method~~(param[] array, param<element> param2)~~
(public)|(private)|(protected)|(static)|(final)|(abstract)|(native)|(synchronized)|(transient)|(throws\\s+[\\w$\\{\\}\\s*(,\\s*\\w$\\{\\})*]+)|(\\s*)").	Mengambil semua komponen didalam penulisan method	public static void method(param param1, param param2); String method(); int method(param param1); private void Method(param[] array,

matcher(getType);		param<element> param2);
\\(.*\\)	Mengambil parameter	(param param1, param param2) () (param param1) (param[] array, param<element> param2)
(?:^|\n)\s*while\\s*\\(.+\\)\\s*\\{	Mendeteksi keberadaan while loop	While-do loop
(?:^|\n)\s*do\s*\{ (?:^|\n)\s*while\s*\(.+\)\s*;	Mendeteksi keberadaan do-while loop	do-while loop
(?:^|\n)\s*if\\s*\\(.*\\)	Mendeteksi keberadaan percabangan if	If <condition>
(?:^|\n)\s*return\\s+.+\\s*;	Mendeteksi keberadaan return	return

(?:^|\n)\s*new\\s+[\\w$]+\\s*\\([\\s\\S]*\\)(\\s*\\.[\\w$]+\\(.*\\))*;	Mengambil new object	new Object();
		new Object().Method();
		new Object().Method(param param1, param param2);
		new Object(param1 param1, param1 param2).Method(param2 param1, param2 param2)

(?:^|\n)\s*(public|private|protected)|(static)|(final)|(abstract)|(native)|(synchronized)|(transient)	Mendeteksi keberadaan modifier	Modifier
(?:^|\n)\s*System\\.((out)|(err))\\.print(ln|f)?.+;	Mencetak lewat Command Prompt atau Command Line	Print Console
(?:^|\n)\s*(([\\w$()]+\\s*[+-]{2}\\s*)\|([+-]{2}\\s*)[\\w$()]+\\s*);	Mendeteksi proses incremental	i++, i--, var++, var--

^\\s*(this\\s*\\.)?[\\w$\\[\\]]+\\s*(\\+)?\\=[^=]+;	Mendeteksi keberadaan Inisialisasi, proses dan preparation	a = 0;
		a = "";
		this.a = 0; this.a = "";
		a = new Object(param param).method(param1 param, param1 param2);
		a = Method(param param,

		param1 param1); a = a + 1; a = '';
initialPattern.replaceAll("=[^=]+;","").replaceAll("(\\s)|(\\+)|(this\\s*\\.)",""");	Mengeambil variabel inisialisasi	a ~~= new Object().Method();~~
InitialPattern.replaceAll("^\\s*(this\\s*\\.)?[\\w$\\[\\]]+\\s*\\=",""");	Mengambil nilai dari inisialisasi	~~a =~~ new Object().Method();

BIODATA PENULIS

Penulis dilahirkan di Surabaya tanggal 22 Februari 1977. Penulis menempuh jenjang Strata 1 di ITATS – Jurusan Teknik Informatika pada tahun 1995 dan menyelesaikan jenjang Strata 1 pada tahun 2000 dengan tema skripsi "Virtual Reality". Penulis menjadi dosen di Institut Teknologi Adhi Tama Surabaya (ITATS) tahun 2001 dan mendapatkan Jabatan Fungsional Asisten Ahli tahun 2007. Di ITATS, penulis mendapatkan amanah untuk mengajar mata kuliah Pengantar Teknologi Informasi, Sistem Operasi, Rekayasa Perangkat Lunak dan Pengembangan Game dan Realitas Virtual. Selain menjadi dosen tetap ITATS, penulis tahun 2015 juga menjadi ASESOR TIK – BNSP. Ditahun yang sama pula Penulis juga menjadi Reviewer jurnal Virtual Reality – SPRINGER, Science Publication, dan Internatioal Journal of Software Engineering & Application. Di bulan Januari tahun 2017 ini penulis mendapatkan kehormatan untuk mengisi Acara di ANTV dalam acara BINCANG TOKOH dengan tema "Game Edukasi dan Gamers" karena penulis dipandang telah menjadi pelopor bagi mahasiswa di Kampus ITATS untuk mengikuti lomba dan *event-event* regional, nasional sampai dengan Internasional.

Surabaya, Januari 2017