Konversi Javadoc ke LaTeX

Adli Fariz Bonaputra–2012730082

1 Data Skripsi

Pembimbing utama/tunggal: **Pascal Alfadian**
Pembimbing pendamping: **-**
Kode Topik : **PAN4302**
Topik ini sudah dikerjakan selama : **2** semester
Pengambilan pertama kali topik ini pada : Semester **43 - Ganjil 17/18**
Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**
Tipe Laporan : **C** - Dokumen pendukung untuk **pengambilan ke-2 di Skripsi 2**

2 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencan kerja/laporan perkembangan terkahir :

1. Melakukan studi literatur mengenai *syntax* LaTeX dan Javadoc Doclet API.
   status : Ada sejak rencana kerja skripsi.
   hasil :

   (a) **Javadoc**

   Javadoc adalah sebuah *tools* yang dimiliki oleh *Java* yang berguna untuk mengekstrak informasi dari sekumpulan *source file java* menjadi sebuah dokumentasi. Umumnya *Javadoc* menghasilkan sekumpulan *file* HTML yang mendeskripsikan sebuah kelas, *interface*, *method* dan *custom tag*. *Javadoc* dapat mengekstraksi informasi tersebut dari sebuah *package java*, sebuah *file java* atau keduanya.

   Processing of source files

   Javadoc akan memproses *file* yang memiliki akhiran *".java"* dan keseluruhan *file* yang terdapat di dalam folder yang sama. *Javadoc* dapat mengambil informasi dari 1 atau lebih *file java* dan sebuah *package*.

   Javadoc dapat memproses sebuah *link* secara otomatis yang mengarah kepada sebuah *package*, kelas dan sebuah nama yang akan didokumentasikan pada saat *Javadoc* memprosesnya. *Link-link* tersebut berada pada beberapa posisi seperti:

   i. *Declaration* (*return types*, *argument types*, *field types*).
   ii. Bagian *"See Also"* yang dihasilkan oleh *tag @see*.
   iii. *In-line text* yang dihasilkan oleh *tag @link*.
   iv. *Exeption* yang dihasilkan oleh *tag @throws*.
   v. *Link "Specified by"* untuk *member* dari sebuah *interface*.
   vi. *Link "Override"* untuk *member* dari sebuah kelas.
   vii. Ringkasan daftar tabel *package*, kelas dan seluruh anggota dari kelas.
   viii. Turunan dari setiap *package* dan kelas.
   ix. Indeks

   Dalam mengekstrak informasi yang terdapat dalam sebuah *package java* atau beberapa *file java* umumnya menghasilkan sebuah dokumentasi standar yang berbentuk *file* HTML dan format

penulisan yang mengikuti standar *Javadoc*, akan tetapi untuk menghasilkan sebuah format dokumentasi yang diingin, dapat menggunakan sebuah *doclet* yang disediakan oleh *Javadoc*.

Terminologi

Terdapat beberapa istilah yang memiliki arti spesifik dalam konteks *Javadoc* sebagai berikut:

- *Generated Document*
  Dokumen yang dihasilkan oleh *Javadoc tools* adalah sebuah *file* HTML dan dibuat oleh *standard doclet*
- *Name*
  Nama dari sebuah perangkat lunak dituliskan dalam bahasa *Java*. Nama-nama tersebut yaitu nama *package*, kelas, *interface*, *field*, *constructor* atau *method*. Nama tersebut dapat berupa informasi lengkapnya seperti *java.lang.String.equals(java.lang.Object)* atau informasi pendeknya seperti *equals(Object)*
- *Documented Classes*
  Detail dari sebuah kelas dan *interface* akan didokumentasikan pada saat *Javadoc* berjalan. Untuk dapat didokumentasikan, *source file* harus tersedia, kemudian nama dari *source file* atau nama dari *package* tersebut harus diletakkan pada *Javadoc command-line*
- *Included Classes*
  kelas dan *Interface* akan didokumentasikan pada saat *Javadoc* berjalan, hal ini sama seperti *Documented Classes*
- *Excluded Classes*
  kelas dan *Interface* tidak akan didokumenasikan pada saat *Javadoc* berjalan.
- *Referenced Classes*
  kelas dan *Interface* yang secara eksplisit disebut oleh kelas dan *interface* lainnya, seperti *return type*, *parameter type*, *cast type*, *extended class*, *implemented interface*, *imported class*, kelas yang digunakan pada *method body*, *@see*, *@link*, *@linkplain* dan *@inheritDoc tag*
- *External Referenced Classes*
  kelas yang tidak dihasilkan saat *Javadoc* berjalan. Dengan kata lain, kelas tersebut tidak diletakkan pada *Javadoc command-line*. *Links* akan dihasilkan jika sebuah kelas mengatakan memiliki *external references* atau *external link*.

Source Files

Javadoc akan menghasilkan *output* yang berasal dari beberapa tipe *file*, yaitu sebagai berikut:

- *Class Source Code Files*
  Setiap kelas atau *interface* dapat memiliki dokumentasinya masing-masing yang terdapat pada *file java*
- *Package Comment Files*
  Setiap *package* dapat memiliki dokumentasinya masing-masing yang terdapat pada *root* folder kemudian *Javadoc* akan menggabungkan *file-file* yang terdapat pada *root* menjadi sebuah ringkasan. Untuk membuat dokumentasi tersebut, terdapat 2 pilihan yaitu sebuah *file* package.html 1 atau sebuah *file* package-info.java 2.

```
1  <html>
2  <body>
3  Provides the classes necessary to create an applet and the classes
4  an applet uses to communicate with its applet context.
5
6  @since 1.0
7  @see java.awt
```

```
8   </body>
9   </html>
```

Listing 1: *File* package.html

```
1   /**
2    * Provides the classes necessary to create an applet
3    * and the classes an applet uses to communicate
4    * with its applet context.
5    *
6    * @since 1.0
7    * @see java.awt
8    */
9   package java.lang.applet;
```

Listing 2: *File* package-info.java

Ketika *Javadoc* memproses *package* tersebut, *Javadoc* akan melakukan beberapa langkah yaitu sebagai berikut:

i. Menyalin informasi untuk diproses. Jika *file* berupa HTML maka pada bagian *<body>* hingga *</body>* akan disalin.

ii. Memproses semua *tag* pada *package* yang ada.

iii. Memasukan teks yang sudah diproses tersebut pada bagian bawah halaman dokumentasi yang dihasilkan.

iv. Salin kalimat pertama pada *package* tersebut pada bagian atas halaman dokumentasi

- *Overview Comment Files*
  Setiap aplikasi atau sekumpulan *package* yang akan didokumentasikan akan memiliki dokumentasi *overview*. Dokumentasi tersebut dapat dibuat lebih dari 1, jika pada saat pembuatan perangkat lunak menggunakan sekumpulan *package* yang berbeda. Untuk membuat sebuah dokumentasi ini, perlu membuat sebuah *file* HTML yang umumnya bernama *overview.html*. Kemudian *Javadoc* akan memproses seperti pada *Package Comment Files*
- *Miscellaneous Unprocessed Files*
  File tersebut dapat berubah sebuah *graphic files*, *file java* dan sebuah *file* HTML.

Generated Files

Secara *default*, *Javadoc* akan menggunakan *standard doclet* yang akan menghasilkan sebuah dokumentasi berformat HTML. Doclet tersebu akan menghasilkan *file* HTML secara terpisah. Terdapat 3 grup yang masing-masing grup memiliki kriterianya sendiri, 3 grup tersebut adalah sebagai berikut:

- *Basic Content Pages*
  - sebuah halaman kelas atau *interface* (*classname*.html) untuk masing-masing kelas atau *interface* yang akan didokumentasikan
  - sebuah halaman *package* (*package-summary.html*) untuk masing-masing *package* yang akan didokumentasikan
  - sebuah halaman *overview* (*overview-summary.html*) untuk keseluruhan sekumpulan *package*. Halaman ini adalah halaman utama yang dihasilkan.
- *Cross-Reference Pages*
  - sebuah halaman hirarki dari kelas untuk sekumpulan dari semua *package* (*overview-tree.html*)
  - sehalaman hirarki dari kelas untuk setiap *package* (*package-tree.html*)

  - sehalaman *"use"* (*package-use.html*) yang berisikan *package*, *classes*, *methods*, *constructors* atau *interface*. Jika diberikan sebuah kelas bernama A, makan halaman tersebut akan berisikan *subclasses* dari A, *methods* yang memiliki *return* A dan *methods* atau *constructors* dengan parameter bertipe A.
  - sebuah halaman *deprecated API* (*deprecated-list.html*). Halaman ini adalah halaman dari sekumpulan nama yang tidak direkomendasikan untuk digunakan.
  - sebuah halaman sekumpulan nilai *constant* (*constant-values.html*) untuk sekumpulan nilai *static*.
  - sebuah halaman *serialized form* (*serialized-form.html*)
  - sebuah halaman *index* (*index-*.html*).
- *Support Files*
  - sebuah halaman bantuan (*help-doc.html*).
  - sebuah halaman *index* (*index.html*) yang membuat sebuah HTML *frames*.
  - beberapa *frame file* (**-frame.html*) yang berisi sekumpulan *packages*, kelas dan *interface* dan digunakan pada saat HTML *frames* ditampilkan
  - sebuah *file* teks *package list* (*package-list*).
  - sebuah *style sheet file* (*stylesheet.css*) untuk mengontrol warna, jenis *font*, ukuran *font* dan posisi dari halamanan yang dihasilkan
  - sebuah *doc-files* yang berisikan gambar dan beberapa contoh *file java*

Javadoc akan menghasilkan 2 atau 3 HTML *frame*. *Javadoc* akan membuat minimum *frame* yang dibutuhkan. Jika hanya terdapat 1 *package*, maka *Javadoc* akan membuat 1 *frame* yang berisi dari sekumpulan kelas pada *package* tersebut. Jika terdapat lebih dari 2 *package*, maka *Javadoc* akan membuat 3 *frame* dari sekumpulan *package*. Jika kelas yang digunakan adalah *java.applet.Applet* dan semua dokumentasi yang dihasilkan akan berada pada folder yang bernama *apidocs*, struktur *file* yang dihasilkan adalah sebagai berikut:

```
1  apidocs                          Top directory
2    index.html                     Initial page that sets up HTML frames
3  * overview-summary.html          Lists all packages with first sentences summaries
4    overview-tree.html             Lists class hierarchy for all packages
5    deprecated-list.html           Lists deprecated API for all packages
6      constant-values.html         Lists values of static fields for all packages
7      serialized-form.html         Lists serialized form for all packages
8   * overview-frame.html           Lists all packages, used in upper-left frame
9      allclasses-frame.html        Lists all classes for all packages, used in
10                                  lower-left frame
11     help-doc.html                Lists user help for how these pages are organized
12     index-all.html               Default index created without -splitindex option
13     index-files                  Directory created with -splitindex option
14         index-<number>.html      Index files created with -splitindex option
15     package-list                 Lists package names, used only for
16                                  resolving external refs
17     stylesheet.css               HTML style sheet for defining fonts, colors and
18                                  positions
19     java                         Package directory
20         applet                   Subpackage directory
21             Applet.html          Page for Applet class
22             AppletContext.html   Page for AppletContext interface
23             AppletStub.html      Page for AppletStub interface
24             AudioClip.html       Page for AudioClip interface
25         * package-summary.html   Lists classes with first sentence summaries
26                                  for this package
27         * package-frame.html     Lists classes in this package, used in
```

```
                                    lower left-hand frame
        * package-tree.html         Lists class hierarchy for this package
          package-use               Lists where this package is used
            doc-files               Directory holding image and example files
            class-use               Directory holding pages API is used
                Applet.html         Page for uses of Applet class
                AppletContext.html  Page for uses of AppletContext interface
                AppletStub.html     Page for uses of AppletStub interface
                AudioClip.html      Page for uses of AudioClip interface
    src-html                        Source code directory
        java                        Package directory
            applet                  Subpackage directory
                Applet.html         Page for Applet source code
                AppletContext.html  Page for AppletContext source code
                AppletStub.html     Page for AppletStub source code
                AudioClip.html      Page for AudioClip source code
```

Listing 3: Struktur *file* yang dihasilkan

(b) **Doclet**

Doclet yang terdapat pada *Javadoc* dapat digunakan untuk menghasilkan sebuah *output Javadoc* yang dapat disesuaikan. Standar *doclet* yang dihasilkan oleh *Javadoc* adalah dokumentasi dengan format HTML. Selain menghasilkan *output* yang dapat disesuaikan, Doclet juga dapat mengekstrak informasi secara spesifik [**?**].

***Interface-interface* pada Doclet**

Berikut adalah beberapa *interface* yang terdapat pada *Doclet*:

- `RootDoc` sebuah *interface* yang menyatakan sebuah *root* dari perangkat lunak yang dibuat. Dari *root* tersebut semua informasi dapat diekstrak. *Method-method* yang digunakan adalah sebagai berikut
  - `classes()`
    Method ini akan mengembalikan sejumlah kelas dan *interface* pada *package*
- `ClassDoc` sebuah *interface* yang menyatakan informasi dari sebuah kelas. Informasi tersebut dapat berupa nama kelas, nama *method* dan *tag*. *Method-method* yang digunakan adalah sebagai berikut
  - `name()`
    Method ini akan mengembalikan sebuah nama kelas atau *interface* pada *package*
  - `commentText()`
    Method ini akan mengembalikan sebuah informasi dari deskripsi kelas
  - `methods()`
    Method ini akan mengembalikan sebuah *array of methods*
- `MethodDoc` sebuah *interface* yang menyatakan informasi dari sebuah *method*. *Method-method* yang digunakan adalah sebagai berikut
  - `name()`
    Method ini akan mengembalikan sebuah nama *method*
  - `modifiers()`
    Method ini akan mengembalikan sebuah *access modifier* dari sebuah *method*
  - `returnType()`
    Method ini akan mengembalikan sebuah *return type* dari sebuah *method*
  - `flatSignature()`

Method ini akan mengembalikan *signature* dari sebuah *method*. Jika terdapat *Method* dengan parameter (String x, int y), maka akan mengembalikan (String, int)

- `ParamTag` sebuah *interface* yang menyatakan informasi dari sebuah *Tag* parameter. *Method-method* yang digunakan adalah sebagai berikut
  - `name()`
    Method ini akan mengembalikan sebuah *tag @param*
  - `parameterName()`
    Method ini akan mengembalikan sebuah nama parameter dari sebuah *method*
  - `parameterComment()`
    Method ini akan mengembalikan sebuah deskripsi dari parameter yang terdapat pada *method*

Penggunaan Doclet

Doclet dapat menghasilkan sebuah *output Javadoc* yang dapat disesuaikan. Penggunaan *Doclet* API dapat mengekstrak bermacam-macam informasi seperti nama kelas, nama *method*, deskripsi singkat untuk sebuah parameter dari sebuah *method* hingga *return type* dari *method*.

Berikut adalah langkah-langkah untuk menggunakan *doclet*:

i. Membuat sebuah kelas pada *java* sebagai *doclet*. *class java* tersebut harus meng-*import* `com.sun.javadoc.*` untuk menggunakan *doclet* API.

ii. *Doclet* tersebut diawali dengan sebuah *method* `public static boolean start` yang memiliki parameter `RootDoc`.

iii. *Compile doclet* tersebut dengan menggunakan *compiler* Java 2 SDK yaitu *javac* pada *command prompt(Windows)/terminal(Linux)*.

iv. Jalankan *Javadoc* menggunakan `-doclet` ***startingclass*** *option* untuk menghasilkan *output* yang telah disesuaikan, dimana ***startingclass*** adalah sebuah kelas yang sudah dibuat pada langkah 1.

File doclet API terdapat pada direktori *folder jdk* yang ter-*install* pada komputer pada *subfolder* `lib\tools.jar`.*doclet* yang sudah dibuat harus di-*compile* menggunakan *file* `tools.jar` dan menambahkan *option* `-classpath` setelah *command* `javac`. Jika tidak menggunakan *option* `-doclet` , *Javadoc* akan menghasilkan *output* standar yaitu berupa *file* HTML.

Package `com.sun.javadoc` terdiri *interface* yang mendefinisikan *doclet* API dan sedangkan *file* `tools.jar` berisikan *interface-interface* tersebut dan juga berisikan *private package* dengan *class-class* yang mengimplementasi *interface* tersebut serta *file* tools.jar berisikan pula *class-class* yang mengimplementasi sebuah standar *doclet*.

```
1  import com.sun.javadoc.*;
2
3  public class ListClass {
4    public static boolean start(RootDoc doc) {
5      ClassDoc[] classes = doc.classes();
6      for(int i=0, i < classes.length; i++) {
7        System.out.println(classes[i]);
8      }
9      return true;
10   }
11 }
```

Listing 4: kelas ListClass.java

Potongan *program* ini 4 adalah sebuah *doclet* sederhana untuk menampilkan nama-nama kelas pada *file java*. Hal pertama yang harus dilakukan adalah meng-*import package* `com.sun.javadoc.*`,

kemudian membuat sebuah *method* `public static boolean start` dengan parameter sebuah `RootDoc` doc yang akan menampung sekumpulan *file java* yang akan diproses. `ClassDoc` pada *method* tersebut akan menampung nama-nama kelas yang terdapat pada variabel `doc` dengan menggunakan *method* `classes()`.

2. Melakukan survei mengenai format penulisan pada dokumen LaTeX
   status : Ada sejak rencana kerja skripsi.
   hasil : Survei yang dilakukan adalah mengamati bab 4 dari Skripsi mahasiswa Teknik Informatika angkatan 2012 bernama Herfan Heryandi. Penulisan pada bab 4 tersebut menjadikan contoh untuk membuat struktur LaTeX yang dibuat.

3. Menganalisis kebutuhan perangkat lunak
   status : Ada sejak rencana kerja skripsi.
   hasil :

   (a) **Analisis Kebutuhan Perangkat Lunak**

   Struktur LaTeX yang digunakan memiliki format sebagai berikut.

```
\begin{enumerate}
\item \texttt{namaKelas}\\
{penjelasan kelas}

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
  \item \texttt{atribut} -
  {penjelasan tentang atribut}.
\end{itemize}

\textit{Method} yang terdapat pada kelas Pertambahan adalah sebagai berikut.
\begin{itemize}
  \item \texttt{method}\\
  {penjelasan method}

  \textbf{Parameter:}
  \begin{itemize}
    \item \texttt{parameter} -
    {penjelasan dari parameter}.
  \end{itemize}

  \textbf{Return Value:} {penjelasan return-type method}\\
  \textbf{Exception:} {penjelasan exception jika terdapat exception}
  \textbf{See Also:} {penjelasan tag @see jika terdapat tag tersebut}
  \textbf{Override:} {penjelasan apabila jika terdapat {\it override method} }
\end{itemize}
```

Listing 5: Potongan kode LaTeX

Potongan kode yang terdapat pada listing 5 adalah struktur lengkap LaTeX yang digunakan, akan dijelaskan sebagai berikut.

i. *List level* pertama
   Pada *list level* pertama ini menampilkan sebuah nama kelas dan penjelasan terkait dengan kelas tersebut. *List* yang dibuat menggunakan *ordered list* dengan *command* `\begin{enumerate}`... `\end{enumerate}` dan *command* `\texttt{namaKelas}` akan digunakan untuk menampilkan nama kelas.

ii. *List level* kedua
   Pada *list level* kedua ini terdapat dua *list* yang masing-masing menampilkan atribut dan *method* yang dimiliki oleh kelas tersebut. *List* pertama yang dibuat menggunakan *unordered list*

dengan *command* `\begin{itemize}...\end{itemize}` untuk mengisi atribut-atribut yang terdapat pada kelas ini jika kelas ini tidak memiliki atribut maka menampilkan tulisan tidak memiliki atribut. *Command* `\texttt{atribut}` digunakan untuk menampilkan atribut. Atribut ini menampilkan tipe atribut dan nama atribut.
List kedua menggunakan *unordered list* dengan *command* `\begin{itemize}...\end{itemize}` untuk mengisi *method-method* yang terdapat pada kelas ini dan penjelasan terkait dengan *method* tersebut. *Command* `\texttt{method}` digunakan untuk menampilkan *method*. *Method* ini menampilkan *access modifier* dari *method*, tipe kembalian *method*, nama *method* dan daftar nama parameter.

iii. *List level* ketiga
Pada *list level* ketiga ini menampilkan parameter yang digunakan pada *method* dan penjelasan terkait dengan parameter tersebut. *List* yang dibuat menggunakan *unordered list* dengan *command* `\begin{itemize}...\end{itemize}` jika *method* tidak memiliki parameter maka menampilkan tulisan tidak memiliki parameter dan *command* `\texttt{parameter}` akan digunakan untuk menampilkan parameter. Parameter ini menampilkan tipe parameter dan nama parameter.

iv. *Return Value & Exception*
Return value yang terdapat dalam *method* tersebut akan ditampilkan setelah *list level* ketiga jika tipe *return value* adalah `void` maka akan menampilkan tulisan tidak memiliki *return value*. *Exception* maka ditampilkan setelah *Return value* jika *method* tidak terdapat *exception* maka akan menampilkan tulisan tidak memiliki *exception*.

v. *Optional Tags*
Optional tags akan menampilkan informasi dari sebuah *tag* `@see` atapun *tag* `{@link}`. Jika tidak ada informasi dari *tag* - *tag* tersebut akan menampilkan tulisan tidak ada.

vi. *Override*
Override akan menampilkan informasi apakah *method* dari sebuah *superclass* ditulis kembali di sebuah *subclass*. jika tidak ada informasi tersebut maka bagian penjelasan akan dihilangkan.

Perangkat lunak yang dibuat akan menerima sebuah masukan berupa sekumpulan *file java* yang berada di dalam sebuah *package*. Struktur kode *java* yang digunakan memiliki format sebagai berikut.

```
1  package javadoc;
2
3  /**
4   * Kelas Abstract OperasiMatematika
5   * @author Adli Fariz Bonaputra
6   */
7  public abstract class OperasiMatematikaInterface {
8
9    /**
10    * Method untuk menghasilkan perhitungan 2 buah bilangan
11    * @param a Bilangan pertama
12    * @param b Bilagan kedua
13    * @return hasil perhitungan 2 buah bilangan
14    */
15   public int calculate(int a, int b){return 0;}
16 }
```

Listing 6: Contoh kode *java* dari *superclass* yang diuji

```
1  package javadoc;
2
3  /**
```

```
4  * Kelas ini merupakan Kelas Pertambahan kelas ini memiliki beberapa fungsi
5  * yaitu {@link #calculate(int,int) calculate}
6  *
7  * @author Adli Fariz Bonaputra
8  * @see "Pertambahan"
9  *
10 */
11 public class Pertambahan extends OperasiMatematikaInterface {
12
13   /**
14    * Atribut A
15    */
16   private int a;
17   /**
18    * Atribut B
19    */
20   private int b;
21
22   @Override
23   public int calculate(int a, int b) {
24     int hasil = 0;
25     hasil = a + b + 10;
26     return hasil;
27   }
28 }
```

Listing 7: Contoh kode *java* dari *subclass* yang diuji

Potongan kode yang terdapat pada listing 6 dan 7 adalah struktur *file java* yang digunakan, akan dijelaskan sebagai berikut.

i. Setiap *file java* harus terletak di dalam sebuah *package* yang sama.

ii. Setiap deklarasi kelas harus diawali dengan huruf kapital serta memiliki javadoc untuk penjelasan tentang kelas tersebut dan secara opsional dapat menambahkan *tag* - *tag* javadoc seperti *tag* `@see` sebagai penunjuk ke sebuah referensi dan *tag* `{@link}` sebagai penunjuk ke dokumentasi sebuah *package*, *class* ataupun *method* yang dimiliki oleh kelas lain.

iii. Setiap deklarasi atribut harus memiliki *access modifier*, tipe atribut dan nama atribut serta memiliki javadoc untuk penjelasan tentang atribut tersebut.

iv. Seiap deklarasi *method* harus memiliki *access modifier*, tipe kembalian, nama *method*, tipe dan variabel parameter serta memiliki javadoc untuk penjelasan *method*, parameter yang digunakan dan hasil kembalian sebuah *method*.

Hasil dari sebuah perangkat lunak yang dibuat adalah sebuah *file* berformat LaTeX. Perangkat lunak akan membaca satu persatu *file java* dan informasi yang terdapat pada setiap *file java* tersebut dimasukan ke dalam *file* LaTeX.

```
1  \begin{enumerate}
2    \item \texttt{Pertambahan}\\
3    Kelas ini merupakan Kelas Pertambahan.
4
5    Atribut yang dimiliki kelas ini adalah sebagai berikut.
6    \begin{itemize}
7      \item \texttt{int a} -
8      Atribut A.
9      \item \texttt{int b} -
10     Atribut B.
11   \end{itemize}
12
```

```
13   \textit{Method} yang terdapat pada kelas Pertambahan adalah sebagai berikut.
14   \begin{itemize}
15     \item \texttt{public int pertambahan(int a, int b)}\\
16     Method Pertambahan.
17
18     \textbf{Parameter:}
19     \begin{itemize}
20       \item \texttt{int a} -
21       Bilangan Pertama.
22       \item \texttt{int b} -
23       Bilangan Kedua.
24     \end{itemize}
25
26     \textbf{Return Value:} hasil penjumlahan 2 buah bilangan.\\
27     \textbf{Exception:} tidak memiliki \textit{exception}.
28     \textbf{Override:} \textt{pertambahan} dari kelas \textt{operasiMatematikaInterface}
29   \end{itemize}
30 \end{enumerate}
```

Listing 8: Contoh hasil konversi *Javadoc* ke LaTeX

Hasil konversi 8 akan menampilkan nama kelas serta penjelasan kelas tersebut, atribut yang digunakan serta penjelasan untuk setiap atributnya, *method* yang digunakan serta penjelasan *method*, parameter yang digunakan serta penjelasan setiap parameternya, *return value* dan *exception*.

Analisis Program Sejenis TeXDoclet TeXDoclet merupakan sebuah program yang mengimplementasi *Doclet* yang dimiliki oleh *Java*. Program ini akan mengkonversi sekumpulan *file java* yang terletak di dalam satu *package* yang sama. TeXDoclet dapat menghasilkan dokumen berupa *file* LaTeX atau *file* PDF. Untuk dapat menghasilkan *file* PDF, TeXDoclet mengintegrasikan LuaLaTeX untuk menghasilkan dokumen PDF dari sebuah *file* LaTeX.

TeXDoclet memiliki beberapa *option* yang dapat digunakan, akan dijelaskan sebagai berikut.

i. `-sectionlevel <level>`
   Untuk menentukan *level* teratas dari *section* sebuah dokumen. *Section* tersebut bisa berupa *chapter*, *section* atau *subsection*

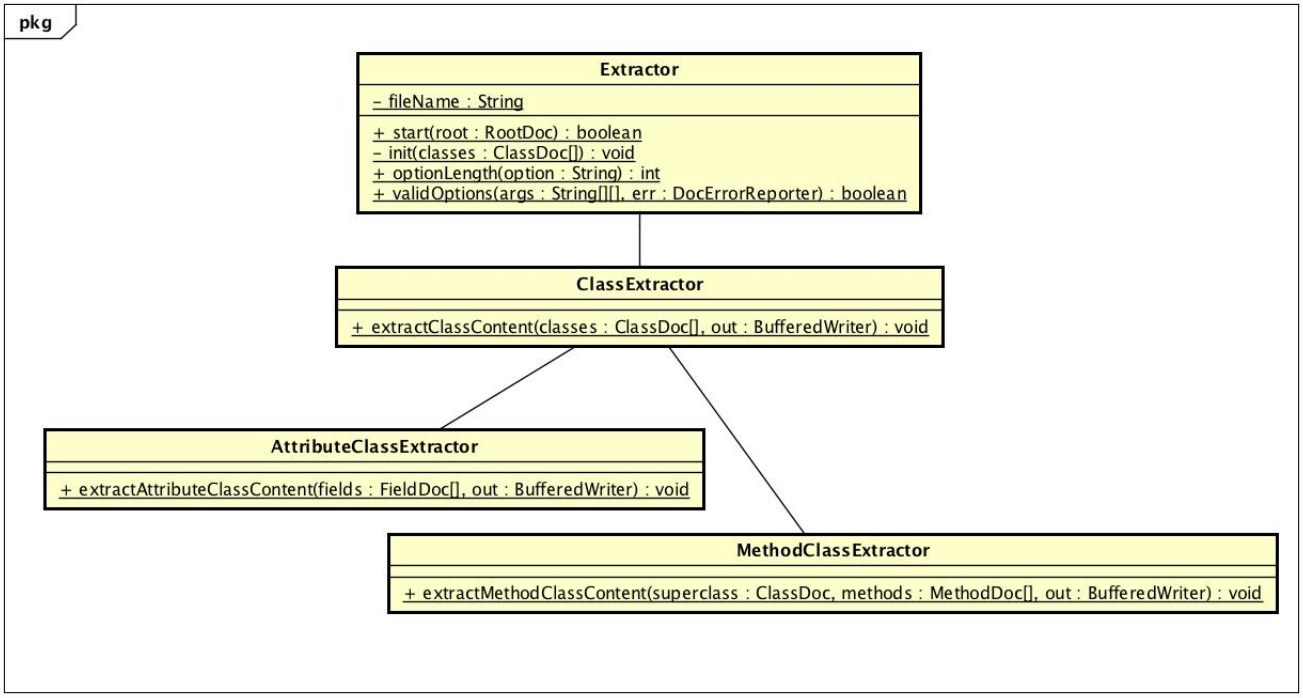
ii. `-createPdf`
   Untuk menghasilkan *file* PDF dari sebuah hasil *file* LaTeX dengan menggunakan LuaLaTeX.

iii. `-twosided`
   Untuk menghasilkan dokumen 2 sisi. Jika dokumen tersebut menggunakan *option* ini maka dokumen tersebut pada saat dicetak akan memiliki 2 sisi yaitu depan dan belakang.

iv. `-texinit <file>`
   Untuk menambahkan *command-command* yang lain sebelum *command* `\begin{document}`.

v. `-docclass <class>`
   Untuk menentukan tipe dokumen yang akan dibuat. *Default* untuk *option* adalah tipe dokumen *report*.

4. Merancang perangkat lunak
   status : Ada sejak rencana kerja skripsi.
   hasil : Rancangan kelas dibawah ini akan menampilkan keseluruhan kelas yang akan digunakan. Deskripsi kelas berserta fungsi dari diagram kelas tersebut adalah sebagai berikut:



Gambar 1: Kelas Diagram

(a) `AttributeClassExtractor`

Kelas ini merupakan kelas untuk mengambil informasi sebuah atribut yang terdapat pada kelas.

Kelas ini tidak memiliki atribut. *Method-method* yang dimiliki kelas ini adalah sebagai berikut.

- `public static void extractAttributeClassContent(com.sun.javadoc.FieldDoc[] fields, BufferedWriter out)`

  Method ini akan menampilkan atribut-atribut yang dimiliki oleh sebuah kelas

  Parameter:

  - `com.sun.javadoc.FieldDoc[] fields` - sebuah array berisikan sejumlah atribut dari kelas
  - `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text

  Kembalian: Tidak memiliki *return value*

  Exception: Tidak memiliki *exception*

(b) `ClassExtractor`

Kelas ini merupakan kelas untuk mengambil informasi dari sebuah kelas.

Kelas ini tidak memiliki atribut. *Method-method* yang dimiliki kelas ini adalah sebagai berikut.

- `public static void extractClassContent(com.sun.javadoc.ClassDoc[] classes, BufferedWriter out)`

  Method ini akan menampilkan nama kelas berserta penjelasan dari sebuah kelas

  Parameter:

  - `com.sun.javadoc.ClassDoc[] classes` - sebuah array berisikan sejumlah kelas
  - `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text

  Kembalian: Tidak memiliki *return value*

  Exception: Tidak memiliki *exception*

(c) `Extractor`

Kelas ini merupakan kelas untuk menjalan *custom doclet*.

Atribut yang dimiliki kelas ini adalah sebagai berikut.

- `String fileName` - atribut untuk nama *file*

Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public static boolean start(RootDoc root)`
  Method ini berperan sebagai *method* untuk menjalankan *custom doclet*
  Parameter:
  - `RootDoc root` - berperan sebagai mengambil seluruh informasi spesifik dari *option* yang terdapat pada *command-line* sebuah *terminal*. Selain itu berperan juga untuk mengambil informasi dari sekumpulan *file java* yang akan di proses.

  Kembalian: kondisi true
  Exception: Tidak memiliki *exception*
- `private static void init(com.sun.javadoc.ClassDoc[] classes)`
  Method ini berperan untuk menulis kedalam sebuah *file* saat *javadoc* berjalan.
  Parameter:
  - `com.sun.javadoc.ClassDoc[] classes` - sebuah array yang berisikan sekumpulan *file java* yang akan di proses.

  Kembalian: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public static int optionLength(String option)`
  Method untuk menghitung banyak option yang digunakan pada *command-line*
  Parameter:
  - `String option` - sebuah option

  Kembalian: panjang setiap option
  Exception: Tidak memiliki *exception*
- `public static boolean validOptions(java.lang.String[][] args, DocErrorReporter err)`
  Pengecekan option valid
  Parameter:
  - `java.lang.String[][] args` - String array 2 dimensi dari option
  - `DocErrorReporter err` - sebuah error jika tidak terdapat option tersebut.

  Kembalian: bernilai true jika option tersebut dikenali, false jika option tersebut tidak dikenali
  Exception: Tidak memiliki *exception*

(d) `MethodClassExtractor`

Kelas ini merupakan kelas untuk mengambil informasi sebuah *method* terdapat pada kelas.

Kelas ini tidak memiliki atribut. *Method-method* yang dimiliki kelas ini adalah sebagai berikut.

- `public static void extractMethodClassContent(ClassDoc superclass, com.sun.javadoc.MethodDoc[] methods, BufferedWriter out)`
  Method ini akan menampilkan *method-method* yang dimiliki oleh sebuah kelas
  Parameter:
  - `ClassDoc superclass` - sebuah objek ClassDoc
  - `com.sun.javadoc.MethodDoc[] methods` - sebuah array berisikan sejumlah *method* dari kelas
  - `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text

  Kembalian: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

3 Rancangan Antarmuka

Rancangan antarmuka perangkat lunak yang dibuat adalah melalui sebuah *terminal* pada *Linux* dan *command prompt* pada *Windows*. Berikut adalah antarmuka jika menggunakan *terminal* pada *Linux*:

```
Last login: Sun Apr 29 17:06:20 on ttys001
abathz:~ abathz$ cd Documents/KULIAH/Skripsi/javadoc-to-latex/
```

Gambar 2: Mengarahkan kedalam folder dari perangkat lunak

Langkah pertama adalah berpindah dari direktori awal ke direktori perangkat lunak yang dibuat. Untuk berpindah direktori perlukan *command* `cd` atau kepanjangan dari *change directory* lalu diikuti dengan lokasi direktori yang diinginkan. Pada gambar 2 direktori perangkat lunak terdapat di dalam folder Document lalu folder KULIAH lalu folder Skripsi dan terakhir folder javadoc-to-latex kemudian tekan tombol *enter* lalu direktori akan langsung berpindah ke direktori yang dituju.

```
abathz:javadoc-to-latex abathz$ javadoc -filename bab4 -classpath dist/javadoc-to-latex.jar -doclet
extractor.Extractor -docletpath dist/javadoc-to-latex.jar ../javadoc/*
```

Gambar 3: Memasukkan *option* yang akan digunakan

Langkah kedua adalah menjalan perangkat lunak yang dibuat. Diawali dengan command `javadoc` lalu dikuti 5 buah argumen. Argumen pertama(hijau) adalah *option* untuk menamai *file* sesuai dengan yang ditentukan. Sebagai contoh pada gambar 3, *file* akan bernama "bab4", jika argumen pertama tidak dimasukkan pada *command-line* maka nama dari *file* tersebut secara otomatis menjadi "doc". Argumen kedua(biru muda) berperan sebagai penunjuk kelas-kelas yang digunakan. Argumen kedua ini bersifat *optional*, jika kode program yang akan didokumentasikan menggunakan *external library* maka argumen ini digunakan. Argumen ketiga(jingga) adalah sebuah kelas untuk menjalankan *custom doclet* dari perangkat lunak yang dibuat. Argumen ketiga tersebut akan menjalankan kelas bernama `Extractor` yang terdapat didalam *package* `extractor`. Kemudian argumen keempat(kuning) adalah *custom doclet* yang berperan untuk mengambil informasi kelas, atribut, *method* dari sekumpulan *file java*. Argumen kelima(biru) adalah lokasi sekumpulan *file java* yang akan diproses. Pada gambar 3, lokasi *file-file* tersebut terdapat pada folder javadoc. Folder javadoc tersebut berada direktori folder Skripsi.

```
abathz:javadoc-to-latex abathz$ javadoc -filename bab4 -classpath dist/javadoc-to-latex.jar -doclet
extractor.Extractor -docletpath dist/javadoc-to-latex.jar ../javadoc/*
Loading source file ../javadoc/Pembagian.java...
Loading source file ../javadoc/Pengurangan.java...
Loading source file ../javadoc/Perkalian.java...
Loading source file ../javadoc/Pertambahan.java...
Loading source file ../javadoc/operasiMatematikaInterface.java...
Constructing Javadoc information...
abathz:javadoc-to-latex abathz$
```

Gambar 4: Hasil tampilan jika proses konversi selesai

Perangkat lunak yang dibuat akan membaca seluruh isi folder yang dituju, pada contoh gambar 4, terdapat 5 *file java* yang terdapat didalam folder javadoc. Lalu perangkat lunak akan melakukan ekstrasi informasi terhadap masing-masing *file* tersebut. Jika proses ekstraksi selesai maka proses berhenti.

5. Mengimplementasi Javadoc Doclet API
   status : Ada sejak rencana kerja skripsi.

hasil : Pada bagian ini akan dibahas mengenai implementasi perangkat lunak yang telah dibangun. Sub bab ini terdiri atas tiga bagian, yaitu lingkungan perangkat lunak, hasil implementasi perangkat lunak, dan Pengujian perangkat lunak.

3.1 Lingkungan Perangkat Lunak

Dalam proses membangun perangkat lunak ini digunakan spesifikasi perangkat sebagai berikut.

(a) Processor: Intel Core i7 2.5-3.7GHz

(b) RAM: 16.00 GB DDR3

(c) Harddisk : 512MB SSD

(d) VGA : Intel Iris Pro dan AMD Radeon R9 M370X

(e) Sistem Operasi: macOS High Sierra

(f) Versi Java: 1.8.0_121

(g) Code Editor: Netbeans 8.2

3.2 Hasil Implementasi

Kode program pada perangkat lunak ditulis dengan bahasa pemrograman *java*. Perangkat lunak akan menghasilkan sebuah *file* LaTeX yang berisikan dokumentasi sekumpulan kelas-kelas *java*. Berikut *file* LaTeX yang dihasilkan dari kode program perangkat lunak yang telah dibuat.

```
\begin{enumerate}
\item \texttt{AttributeClassExtractor}\\
Kelas ini merupakan kelas untuk mengambil informasi sebuah atribut yang
 terdapat pada kelas.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai be
\begin{itemize}
\item \texttt{public static void extractAttributeClassContent(com.sun.javadoc.FieldDoc[] fields, B
\textit{Method} ini akan menampilkan atribut-atribut yang dimiliki oleh
 sebuah kelas

\textbf{Parameter:}
\begin{itemize}
\item \texttt{com.sun.javadoc.FieldDoc[] fields} -
sebuah array berisikan sejumlah atribut dari kelas
\item \texttt{BufferedWriter out} -
turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
 file text
\end{itemize}
\textbf{Kembalian}: Tidak memiliki \textit{return value}

\textbf{Exception}: Tidak memiliki \textit{exception}

\end{itemize}
\item \texttt{ClassExtractor}\\
Kelas ini merupakan kelas untuk mengambil informasi dari sebuah kelas.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai be
\begin{itemize}
\item \texttt{public static void extractClassContent(com.sun.javadoc.ClassDoc[] classes, BufferedW
\textit{Method} ini akan menampilkan nama kelas berserta penjelasan dari sebuah kelas

```

```
\textbf{Parameter:}
\begin{itemize}
\item \texttt{com.sun.javadoc.ClassDoc[] classes} -
sebuah array berisikan sejumlah kelas
\item \texttt{BufferedWriter out} -
turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
\end{itemize}
\textbf{Kembalian}: Tidak memiliki \textit{return value}

\textbf{Exception}: Tidak memiliki \textit{exception}

\end{itemize}
\item \texttt{Extractor}\\
Kelas ini merupakan kelas untuk menjalan \textit{custom doclet}.

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{String fileName} - atribut untuk nama \textit{file}
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public static boolean start(RootDoc root)}\\
\textit{Method} ini berperan sebagai \textit{method} untuk menjalankan
 \textit{custom doclet}

\textbf{Parameter:}
\begin{itemize}
\item \texttt{RootDoc root} -
berperan sebagai mengambil seluruh informasi spesifik dari
 \textit{option} yang terdapat pada \textit{command-line} sebuah
 \textit{terminal}. Selain itu berperan juga untuk mengambil informasi dari
 sekumpulan \textit{file java} yang akan di proses.
\end{itemize}
\textbf{Kembalian}: kondisi true

\textbf{Exception}: Tidak memiliki \textit{exception}

\item \texttt{private static void init(com.sun.javadoc.ClassDoc[] classes)}\\
\textit{Method} ini berperan untuk menulis kedalam sebuah \textit{file}
 saat \textit{javadoc} berjalan.

\textbf{Parameter:}
\begin{itemize}
\item \texttt{com.sun.javadoc.ClassDoc[] classes} -
sebuah array yang berisikan sekumpulan \textit{file java}
 yang akan di proses.
\end{itemize}
\textbf{Kembalian}: Tidak memiliki \textit{return value}

\textbf{Exception}: Tidak memiliki \textit{exception}

\item \texttt{public static int optionLength(String option)}\\
Method untuk menghitung banyak option yang digunakan pada
 \textit{command-line}

\textbf{Parameter:}
\begin{itemize}
\item \texttt{String option} -
sebuah option
\end{itemize}
```

```
93 \textbf{Kembalian}: panjang setiap option
94
95 \textbf{Exception}: Tidak memiliki \textit{exception}
96
97 \item \texttt{public static boolean validOptions(java.lang.String[][] args, DocErrorReporter err)}\
98 Pengecekan option valid
99
100 \textbf{Parameter:}
101 \begin{itemize}
102 \item \texttt{java.lang.String[][] args} -
103 String array 2 dimensi dari option
104 \item \texttt{DocErrorReporter err} -
105 sebuah error jika tidak terdapat option tersebut.
106 \end{itemize}
107 \textbf{Kembalian}: bernilai true jika option tersebut dikenali, false jika option
108  tersebut tidak dikenali
109
110 \textbf{Exception}: Tidak memiliki \textit{exception}
111
112 \end{itemize}
113 \item \texttt{MethodClassExtractor}\\
114 Kelas ini merupakan kelas untuk mengambil informasi sebuah \textit{method}
115  terdapat pada kelas.
116
117 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai be
118 \begin{itemize}
119 \item \texttt{public static void extractMethodContent(ClassDoc superclass, com.sun.javadoc.MethodD
120 \textit{Method} ini akan menampilkan \textit{method-method} yang dimiliki
121  oleh sebuah kelas
122
123 \textbf{Parameter:}
124 \begin{itemize}
125 \item \texttt{ClassDoc superclass} -
126 sebuah objek ClassDoc
127 \item \texttt{com.sun.javadoc.MethodDoc[] methods} -
128 sebuah array berisikan sejumlah \textit{method} dari kelas
129 \item \texttt{BufferedWriter out} -
130 turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
131  file text
132 \end{itemize}
133 \textbf{Kembalian}: Tidak memiliki \textit{return value}
134
135 \textbf{Exception}: Tidak memiliki \textit{exception}
136
137 \end{itemize}
138 \end{enumerate}
```

Listing 9: Hasil Implementasi

6. Melakukan pengujian perangkat lunak
   status : Pengujian Fungsional sudah dilakukan, Pengujian Eksperimental masih terdapat kendala
   hasil :

3.3 Pengujian Fungsional

Pada pengujian fungsional dilakukan untuk mengetahui fungsi-fungsi yang terdapat pada perangkat lunak berjalan sesuai dengan yang diharapkan. Status pengujian dibagi menjadi 2 yaitu "OK" dan "GAGAL". Berikut ini adalah hasil pengujian fungsional yang telah dilakukan.

(a) Langkah Pengujian: Memanggil fungsi `extractClassContent`
Hal yang diharapkan: pada saat fungsi `extractClassContent` dipanggil maka informasi berkaitan dengan kelas tersebut terekstraksi.
Hasil Pengujian: Informasi yang berkaitan dengan kelas telah terekstraksi.
Status: OK

(b) Langkah Pengujian: Memanggil fungsi `extractAttributeClassContent`
Hal yang diharapkan: pada saat fungsi `extractAttributeClassContent` dipanggil maka informasi berkaitan dengan atribut-atribut yang terdapat pada kelas tersebut terekstraksi.
Hasil Pengujian: Informasi yang berkaitan dengan atribut-atribut yang terdapat pada kelas telah terekstraksi.
Status: OK

(c) Langkah Pengujian: Memanggil fungsi `extractMethodClassContent`
Hal yang diharapkan: pada saat fungsi `extractMethodClassContent` dipanggil maka informasi berkaitan dengan fungsi-fungsi yang terdapat pada kelas tersebut terekstraksi.
Hasil Pengujian: Informasi yang berkaitan dengan fungsi-fungsi yang terdapat pada kelas telah terekstraksi.
Status: OK

3.4 Pengujian Eksperimental

Pengujian eksperimental dilakukan terhadap 3 pengujian yaitu pengujian terhadap kode program sederhana, pengujian terhadap kode program perangkat lunak dan pengujian terhadap kode program yang memiliki jumlah *file* yang banyak.

3.4.1 Pengujian Terhadap Kode Program Sederhana

Pengujian pertama ini melibatkan kode program sederhana yaitu Operasi Matematika. Kode program ini memiliki 5 buah kelas yaitu `OperasiMatematikaInterface`, `Pembagian`, `Perkalian`, `Pertambahan` dan `Pengurangan`. Berikut hasil *file* LaTeX yang dihasilkan.

```
1 \begin{enumerate}
2 \item \texttt{OperasiMatematikaInterface}\\
3 Kelas Abstract OperasiMatematika.
4
5 Kelas ini tidak memiliki atribut. \textit{Method–method} yang dimiliki kelas ini adalah sebagai berik
6 \begin{itemize}
7 \item \texttt{public int calculate(int a, int b)}\\
8 Method untuk menghasilkan perhitungan 2 buah bilangan
9
10 \textbf{Parameter:}
11 \begin{itemize}
12 \item \texttt{int a} –
13 Bilangan pertama
14 \item \texttt{int b} –
15 Bilagan kedua
16 \end{itemize}
17 \textbf{Kembalian}: hasil perhitungan 2 buah bilangan
18
19 \textbf{Exception}: Tidak memiliki \textit{exception}
20
21 \end{itemize}
22 \item \texttt{Pembagian}\\
23 Kelas ini merupakan Kelas Pembagian.
```

```
24
25 Atribut yang dimiliki kelas ini adalah sebagai berikut.
26 \begin{itemize}
27 \item \texttt{int a} - Atribut A
28 \item \texttt{int b} - Atribut B
29 \end{itemize}
30 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
31 \begin{itemize}
32 \item \texttt{public int calculate(int a, int b)}\\
33 Method untuk menghasilkan perhitungan 2 buah bilangan
34
35 \textbf{Parameter:}
36 \begin{itemize}
37 \item \texttt{int a} -
38 Bilangan pertama
39 \item \texttt{int b} -
40 Bilagan kedua
41 \end{itemize}
42 \textbf{Kembalian}: hasil perhitungan 2 buah bilangan
43
44 \textbf{Exception}: Tidak memiliki \textit{exception}
45
46 \end{itemize}
47 \item \texttt{Pengurangan}\\
48 Kelas ini merupakan Kelas Pengurangan.
49
50 Atribut yang dimiliki kelas ini adalah sebagai berikut.
51 \begin{itemize}
52 \item \texttt{int a} - Atribut A
53 \item \texttt{int b} - Atribut B
54 \end{itemize}
55 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
56 \begin{itemize}
57 \item \texttt{public int calculate(int a, int b)}\\
58 Method untuk menghasilkan perhitungan 2 buah bilangan
59
60 \textbf{Parameter:}
61 \begin{itemize}
62 \item \texttt{int a} -
63 Bilangan pertama
64 \item \texttt{int b} -
65 Bilagan kedua
66 \end{itemize}
67 \textbf{Kembalian}: hasil perhitungan 2 buah bilangan
68
69 \textbf{Exception}: Tidak memiliki \textit{exception}
70
71 \end{itemize}
72 \item \texttt{Perkalian}\\
73 Kelas ini merupakan Kelas Perkalian.
74
75 Atribut yang dimiliki kelas ini adalah sebagai berikut.
76 \begin{itemize}
77 \item \texttt{int a} - Atribut A
78 \item \texttt{int b} - Atribut B
79 \end{itemize}
80 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
81 \begin{itemize}
82 \item \texttt{public int calculate(int a, int b)}\\
83 Method untuk menghasilkan perhitungan 2 buah bilangan
```

```
84
85 \textbf{Parameter:}
86 \begin{itemize}
87 \item \texttt{int a} -
88 Bilangan pertama
89 \item \texttt{int b} -
90 Bilagan kedua
91 \end{itemize}
92 \textbf{Kembalian}: hasil perhitungan 2 buah bilangan
93
94 \textbf{Exception}: Tidak memiliki \textit{exception}
95
96 \end{itemize}
97 \item \texttt{Pertambahan}\\
98 Kelas ini merupakan Kelas Pertambahan.
99
100 Atribut yang dimiliki kelas ini adalah sebagai berikut.
101 \begin{itemize}
102 \item \texttt{int a} - Atribut A
103 \item \texttt{int b} - Atribut B
104 \end{itemize}
105 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
106 \begin{itemize}
107 \item \texttt{public int calculate(int a, int b)}\\
108 Method untuk menghasilkan perhitungan 2 buah bilangan
109
110 \textbf{Parameter:}
111 \begin{itemize}
112 \item \texttt{int a} -
113 Bilangan pertama
114 \item \texttt{int b} -
115 Bilagan kedua
116 \end{itemize}
117 \textbf{Kembalian}: hasil perhitungan 2 buah bilangan
118
119 \textbf{Exception}: Tidak memiliki \textit{exception}
120
121 \end{itemize}
122 \end{enumerate}
```

Listing 10: Hasil Pengujian Pertama

3.4.2 Pengujian Terhadap Kode Program Perangkat Lunak

Pengujian kedua ini melibatkan kode program perangkat lunak. Kode program ini memiliki 4 buah kelas yaitu `Extractor`, `ClassExtractor`, `AttributeClassExtractor` dan `MethodClassExtractor`. Berikut hasil *file* LaTeX yang dihasilkan.

```
1 \begin{enumerate}
2 \item \texttt{AttributeClassExtractor}\\
3 Kelas ini merupakan kelas untuk mengambil informasi sebuah atribut yang
4  terdapat pada kelas.
5
6 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berik
7 \begin{itemize}
8 \item \texttt{public static void extractAttributeClassContent(com.sun.javadoc.FieldDoc[] fields, Buff
9 \textit{Method} ini akan menampilkan atribut-atribut yang dimiliki oleh
10  sebuah kelas
11
```

```
12 \textbf{Parameter:}
13 \begin{itemize}
14 \item \texttt{com.sun.javadoc.FieldDoc[] fields} -
15 sebuah array berisikan sejumlah atribut dari kelas
16 \item \texttt{BufferedWriter out} -
17 turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
18  file text
19 \end{itemize}
20 \textbf{Kembalian}: Tidak memiliki \textit{return value}
21
22 \textbf{Exception}: Tidak memiliki \textit{exception}
23
24 \end{itemize}
25 \item \texttt{ClassExtractor}\\
26 Kelas ini merupakan kelas untuk mengambil informasi dari sebuah kelas.
27
28 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai be
29 \begin{itemize}
30 \item \texttt{public static void extractClassContent(com.sun.javadoc.ClassDoc[] classes, BufferedW
31 \textit{Method} ini akan menampilkan nama kelas berserta penjelasan dari sebuah kelas
32
33 \textbf{Parameter:}
34 \begin{itemize}
35 \item \texttt{com.sun.javadoc.ClassDoc[] classes} -
36 sebuah array berisikan sejumlah kelas
37 \item \texttt{BufferedWriter out} -
38 turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
39 \end{itemize}
40 \textbf{Kembalian}: Tidak memiliki \textit{return value}
41
42 \textbf{Exception}: Tidak memiliki \textit{exception}
43
44 \end{itemize}
45 \item \texttt{Extractor}\\
46 Kelas ini merupakan kelas untuk menjalan \textit{custom doclet}.
47
48 Atribut yang dimiliki kelas ini adalah sebagai berikut.
49 \begin{itemize}
50 \item \texttt{String fileName} - atribut untuk nama \textit{file}
51 \end{itemize}
52 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
53 \begin{itemize}
54 \item \texttt{public static boolean start(RootDoc root)}\\
55 \textit{Method} ini berperan sebagai \textit{method} untuk menjalankan
56  \textit{custom doclet}
57
58 \textbf{Parameter:}
59 \begin{itemize}
60 \item \texttt{RootDoc root} -
61 berperan sebagai mengambil seluruh informasi spesifik dari
62  \textit{option} yang terdapat pada \textit{command-line} sebuah
63  \textit{terminal}. Selain itu berperan juga untuk mengambil informasi dari
64  sekumpulan \textit{file java} yang akan di proses.
65 \end{itemize}
66 \textbf{Kembalian}: kondisi true
67
68 \textbf{Exception}: Tidak memiliki \textit{exception}
69
70 \item \texttt{private static void init(com.sun.javadoc.ClassDoc[] classes)}\\
71 \textit{Method} ini berperan untuk menulis kedalam sebuah \textit{file}
```

```
 saat \textit{javadoc} berjalan.

\textbf{Parameter:}
\begin{itemize}
\item \texttt{com.sun.javadoc.ClassDoc[] classes} -
sebuah array yang berisikan sekumpulan \textit{file java}
 yang akan di proses.
\end{itemize}
\textbf{Kembalian}: Tidak memiliki \textit{return value}

\textbf{Exception}: Tidak memiliki \textit{exception}

\item \texttt{public static int optionLength(String option)}\\
Method untuk menghitung banyak option yang digunakan pada
 \textit{command-line}

\textbf{Parameter:}
\begin{itemize}
\item \texttt{String option} -
sebuah option
\end{itemize}
\textbf{Kembalian}: panjang setiap option

\textbf{Exception}: Tidak memiliki \textit{exception}

\item \texttt{public static boolean validOptions(java.lang.String[][] args, DocErrorReporter err)}\\
Pengecekan option valid

\textbf{Parameter:}
\begin{itemize}
\item \texttt{java.lang.String[][] args} -
String array 2 dimensi dari option
\item \texttt{DocErrorReporter err} -
sebuah error jika tidak terdapat option tersebut.
\end{itemize}
\textbf{Kembalian}: bernilai true jika option tersebut dikenali, false jika option
 tersebut tidak dikenali

\textbf{Exception}: Tidak memiliki \textit{exception}

\end{itemize}
\item \texttt{MethodClassExtractor}\\
Kelas ini merupakan kelas untuk mengambil informasi sebuah \textit{method}
 terdapat pada kelas.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berik
\begin{itemize}
\item \texttt{public static void extractMethodContent(ClassDoc superclass, com.sun.javadoc.MethodDoc[
\textit{Method} ini akan menampilkan \textit{method-method} yang dimiliki
 oleh sebuah kelas

\textbf{Parameter:}
\begin{itemize}
\item \texttt{ClassDoc superclass} -
sebuah objek ClassDoc
\item \texttt{com.sun.javadoc.MethodDoc[] methods} -
sebuah array berisikan sejumlah \textit{method} dari kelas
\item \texttt{BufferedWriter out} -
turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
 file text
```

```
\end{itemize}
\textbf{Kembalian}: Tidak memiliki \textit{return value}

\textbf{Exception}: Tidak memiliki \textit{exception}

\end{itemize}
\end{enumerate}
```

Listing 11: Hasil Pengujian Kedua

7. Menulis dokumen skripsi
   status : Ada sejak rencana kerja skripsi.
   hasil : Dokumen skripsi telah ditulis sampai bab 6. Bab 1 membahas mengenai latar belakang, rumusan masalah, tujuan, batas masalah, metodologi penelitian dan sistematika penulisan. Bab 2 membahas mengenai pengertian *Javadoc*, Doclet dan LaTeX. Bab 3 membahas mengenai analisis struktur LaTeX dan analisis program sejenis TeXDoclet. Bab 4 membahas mengenai perancangan perangkat lunak. Bab 5 membahas mengenai pengujian terhadap perangkat lunak. Bab 6 membahas mengenai kesimpulan dan saran terhadap perangkat lunak yang dibuat.

4 Pencapaian Rencana Kerja

Persentase penyelesaian skripsi sampai dengan dokumen ini dibuat dapat dilihat pada tabel berikut :

1*	2*(%)	3*(%)	4*(%)	5*	6*(%)
1	10	10			10
2	10	10			10
3	10	10			10
4	10		10		10
5	20		20		20
6	20		20		10
7	20	10	10	Bab 1 hingga Bab 3 di S1, Bab 4 hingga Bab 6 di S2	20
Total	100	40	60		90

Keterangan (*)
1 : Bagian pengerjaan Skripsi (nomor disesuaikan dengan detail pengerjaan di bagian 5)
2 : Persentase total
3 : Persentase yang akan diselesaikan di Skripsi 1
4 : Persentase yang akan diselesaikan di Skripsi 2
5 : Penjelasan singkat apa yang dilakukan di S1 (Skripsi 1) atau S2 (skripsi 2)
6 : Persentase yang sudah diselesaikan sampai saat ini

5 Kendala yang dihadapi

Kendala - kendala yang dihadapi selama mengerjakan skripsi :

- Saat melakukan pengujian eksperimental masih terdapat kendala dalam mendokumentasikan SIAModels

Bandung, 08/05/2018

Adli Fariz Bonaputra

Menyetujui,

Nama: Pascal Alfadian
Pembimbing Tunggal