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 : **1** semester
Pengambilan pertama kali topik ini pada : Semester **43 - Ganjil 17/18**
Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**
Tipe Laporan : **B** - Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

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

   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
28                                  lower left-hand frame
29         * package-tree.html      Lists class hierarchy for this package
30           package-use            Lists where this package is used
31             doc-files            Directory holding image and example files
32             class-use            Directory holding pages API is used
```

```
33                  Applet.html          Page for uses of Applet class
34                  AppletContext.html   Page for uses of AppletContext interface
35                  AppletStub.html      Page for uses of AppletStub interface
36                  AudioClip.html       Page for uses of AudioClip interface
37      src-html                         Source code directory
38          java                         Package directory
39              applet                   Subpackage directory
40                  Applet.html          Page for Applet source code
41                  AppletContext.html   Page for AppletContext source code
42                  AppletStub.html      Page for AppletStub source code
43                  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 fondasi untuk membuat struktur LaTeXyang 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}

{\it 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}
\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*.

Struktur kode *java* yang digunakan memiliki format sebagai berikut.

```
1  package javadoc;
2
3  /**
4   * Kelas ini merupakan Kelas Pertambahan
5   *
6   */
7  public class Pertambahan {
8
9      /**
10      * Atribut A
11      */
12     private int a;
13     /**
14      * Atribut B
15      */
16     private int b;
17
18     /**
19      * Method Pertambahan
20      *
21      * @param a Bilangan Pertama
22      * @param b Bilangan Kedua
23      *
24      * @return hasil penjumlahan 2 buah bilangan
25      */
26     public int pertambahan(int a, int b) {
27         int hasil = 0;
28         hasil = a + b;
29         return hasil;
30     }
31 }
```

Listing 6: Potongan kode *java*

Potongan kode yang terdapat pada listing 6 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.

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*.

Package yang berisikan sekumpulan *file java* tersebut akan menjadi sebuah masukan perangkat lunak dan akan menghasilkan sebuah dokumen dalam format LaTeX.

```
\begin{enumerate}
  \item \texttt{Pertambahan}\\
  Kelas ini merupakan Kelas Pertambahan.

  Atribut yang dimiliki kelas ini adalah sebagai berikut.
  \begin{itemize}
    \item \texttt{int a} -
    Atribut A.
    \item \texttt{int b} -
    Atribut B.
  \end{itemize}

  {\it Method} yang terdapat pada kelas Pertambahan adalah sebagai berikut.
  \begin{itemize}
    \item \texttt{public int pertambahan(int a, int b)}\\
    Method Pertambahan.

    \textbf{Parameter:}
    \begin{itemize}
      \item \texttt{int a} -
      Bilangan Pertama.
      \item \texttt{int b} -
      Bilangan Kedua.
    \end{itemize}

    \textbf{Return Value:} hasil penjumlahan 2 buah bilangan.\\
    \textbf{Exception:} tidak memiliki \textit{exception}.
  \end{itemize}
\end{enumerate}
```

Listing 7: Hasil konversi *Javadoc* ke LaTeX

Hasil konversi 7 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 :

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

6. Melakukan pengujian perangkat lunak
status : dihapuskan/tidak dikerjakan
hasil :

7. Menulis dokumen skripsi
status : Ada sejak rencana kerja skripsi.
hasil : Dokumen skripsi telah ditulis sampai bab 3. 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.

3 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		0
5	20		20		0
6	20		20		0
7	20	10	10	Bab 1 hingga Bab3 di S1	10
Total	100	40	60		40

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 sidah diselesaikan sampai saat ini

4 Kendala yang dihadapi

Kendala - kendala yang dihadapi selama mengerjakan skripsi :

- Tidak ada.

Bandung, 19/11/2017

Adli Fariz Bonaputra

Menyetujui,

Nama: Pascal Alfadian
Pembimbing Tunggal