BAB 1

PENDAHULUAN

1.1 Latar Belakang

Dalam sebuah penelitian, membuat dokumentasi perlu dilakukan. Dokumentasi yang dibuat bisa dalam bentuk *hardcopy* atau *softcopy*, tergantung kebutuhannya. Dokumentasi adalah kegiatan untuk mencatat suatu peristiwa atau aktifitas yang dianggap berharga atau penting. Dokumentasi yang sudah dibuat dapat menjadi referensi untuk memandu dalam melakukan sebuah aktifitas.

Dalam bidang Teknologi Informasi, dokumentasi kode program java umumnya ditulis dalam format *Javadoc*. *Javadoc* adalah sebuah *tools* yang dimiliki oleh Java yang berguna untuk mengekstrak informasi dari sebuah *file* java menjadi sebuah dokumentasi. Umumnya digunakan untuk mendokumentasikan sebuah nama kelas, *interface*, *method* dan *custom tag*. Oleh karena itu, *Javadoc* sangatlah penting karena dapat memuat berbagai informasi dari sebuah *file* java. Informasi tersebut dapat menjelaskan sebuah kelas yang dibuat dalam sebuah dokumentasi perangkat lunak.

Skripsi mahasiswa Program Studi Teknik Informatika Fakultas Teknologi Informasi dan Sains (FTIS) Universitas Katolik Parahyangan (Unpar) adalah membuat perangkat lunak. Perangkat lunak yang dibuat umumnya menggunakan bahasa pemrograman *java*. Seperti yang sudah dijelaskan, bahasa pemrograman *java* memiliki *Javadoc* sebagai informasi dari kelas, *interface*, *method* dan juga *custom tag* yang dibuat, sehingga informasi tersebut dapat digunakan sebagai penjelasan perangkat lunak pada dokumentasi perangkat lunak. Untuk mendokumentasikan perangkat lunak yang dibuat, seluruh mahasiswa diwajibkan untuk menggunakan LaTeX dalam pembuatan sebuah dokumentasi Skripsi. LaTeX merupakan bahasa *markup* untuk menyusun sebuah dokumentasi. LaTeX membuat apa yang ditampilkan sama seperti apa yang yang ditulis. Umumnya bentuk akhir dari dokumen yang dibuat oleh LaTeX biasanya berupa sebuah *file* PDF

Pada salah satu bab dokumentasi Skripsi, terdapat penjelasan dari setiap kelas pada perangkat lunak yang dibuat. Penjelasan tersebut sebenarnya dapat diambil dari *Javadoc* yang telah dibuat pada kelas *java*, namun saat ini berdasarkan pengamatan tersebut masih diketik secara manual dari *Javadoc* ke dalam format LaTeX, sehingga membutuhkan lebih banyak waktu untuk mendokumentasikan setiap kelas pada perangkat lunak yang dibuat.

Oleh karena itu, perlu dikembangkan sebuah perangkat lunak yang dapat mengekstraksi informasi pada *Javadoc* ke format LaTeX secara otomatis. Perangkat lunak ini mengimplementasikan sebuah *Application Programming Interface* (API) yang digunakan untuk mengambil informasi berupa nama kelas, *interface*, *method* dan juga *custom tag* yang terdapat pada sebuah *file java*

1.2 Rumusan Masalah

Bedasarkan latar belakang yang telah disebutkan di atas, maka dihasilkan beberapa poin yang menjadi rumusan masalah dari masalah ini. Rumusan masalah yang akan dibangun antara lain sebagai berikut:

1. Bagaimana membuat perangkat lunak yang dapat mengonversikan format *Javadoc* ke dalam format LaTeX secara otomatis?

2. Bagaimana antarmuka yang baik untuk perangkat lunak yang akan dibuat?

1.3 Tujuan

Adapun tujuan yang ingin dicapai dari penelitian ini adalah menjawab rumusan masalah di atas, yaitu:

1. Membuat perangkat lunak yang dapat mengonversikan format *Javadoc* ke format LaTeX secara otomatis.
2. Mempelajari antarmuka yang baik untuk perangkat lunak yang akan dibuat.

1.4 Batasan Masalah

Agar pembahasan masalah tidak terlalu luas, masalah yang akan dikaji di dalam penelitian ini memiliki batasan, yaitu:

1. Perangkat lunak yang dikembangkan menggunakan bahasa pemrograman *Java*.
2. Perangkat lunak hanya dapat menerima masukan data berupa sekumpulan *file java*.
3. Perangkat lunak hanya menghasilkan *output* berupa format LaTeX yang selanjutnya akan dimasukkan ke dalam file LaTeX.

1.5 Metodologi

Untuk menyelesaikan penelitian ini disusunlah tahap-tahap tugas yang perlu dilakukan. Tahap-tahap yang dimaksud adalah sebagai berikut:

1. Melakukan studi literatur untuk mengetahui *syntax* yang terdapat pada LaTeX dan mengetahui apa saja isi dari dokumentasi Javadoc Doclet API.
2. Melakukan survei terhadap format penulisan pada suatu bab pada skripsi yang berisi tentang dokumentasi perangkat lunak yang dibuat. Membutuhkan minimal 3 dokumen skripsi sebagai panduan format penulisan.
3. Mengimplementasikan langkah-langkah untuk mengkonversi *Javadoc* ke format LaTeX.
4. Melakukan pengujian terhadap perangkat lunak yang telah diimplementasi.
5. Menarik kesimpulan berdasarkan hasil pengujian.

1.6 Sistematika Pembahasan

1. Bab 1 Pendahuluan
   Bab ini akan membahas mengenai latar belakang, rumusan masalah, tujuan, batas masalah, metodologi penelitian dan sistematika penulisan.
2. Bab 2 Dasar Teori Bab ini akan membahas mengenai pengertian *Javadoc*, Doclet dan LaTeX.
3. Bab 3 Analisis Bab ini akan membahas mengenai analisis struktur LaTeX dan analisis program sejenis TeXDoclet.
4. Bab 4 Perancangan Bab ini akan membahas mengenai tahap-tahap perancangan dan penjelasan perangkat lunak.

5. Bab 5 Implementasi dan Pengujian Bab ini akan membahas mengenai implementasi kode program dan pengujian perangkat lunak.

6. Bab 6 Kesimpulan dan Saran
   Bab ini akan membahas mengenai kesimpulan dari penelitian yang telah dilakukan dan saran-saran untuk pengembangan lebih lanjut dari penelitian ini.

BAB 2

LANDASAN TEORI

Bab ini akan membahas teori-teori yang akan menjadi dasar dari penelitian ini. Teori yang dibahas yaitu mengenai *Javadoc*, *Doclet* dan LaTeX.

2.1 Javadoc

Javadoc adalah sebuah *tools* yang dimiliki oleh *Java* yang berguna untuk mengambil 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 [1].

2.1.1 *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:

1. *Declaration* (*return types*, *argument types*, *field types*)
2. Bagian *"See Also"* yang dihasilkan oleh *tag @see*
3. *In-line text* yang dihasilkan oleh *tag @link*
4. *Exeption* yang dihasilkan oleh *tag @throws*
5. *Link "Specified by"* untuk *member* dari sebuah *interface*
6. *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*.

2.1.2 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*.

2.1.3 *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 2.1 atau sebuah *file* package-info.java 2.2.

```
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 2.1: *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 2.2: *File* package-info.java

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

1. Menyalin informasi untuk diproses. Jika *file* berupa HTML maka pada bagian *<body>* hingga *</body>* akan disalin.
2. Memproses semua *tag* pada *package* yang ada.
3. Memasukan teks yang sudah diproses tersebut pada bagian bawah halaman dokumentasi yang dihasilkan.
4. 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.

2.1.4 *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 2.3: Struktur *file* yang dihasilkan

2.2 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 [2].

2.2.1 *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*

2.2.2 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*:

1. Membuat sebuah kelas pada *java* sebagai *doclet*. *class java* tersebut harus meng-*import* `com.sun.javadoc.*` untuk menggunakan *doclet* API.
2. *Doclet* tersebut diawali dengan sebuah *method* `public static boolean start` yang memiliki parameter `RootDoc`.
3. *Compile doclet* tersebut dengan menggunakan *compiler* Java 2 SDK yaitu *javac* pada *command prompt(Windows)/terminal(Linux)*.
4. 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 2.4: kelas ListClass.java

Potongan *program* ini 2.4 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()`.

2.3 LaTeX

LaTeX adalah sebuah bahasa *markup* untuk sistem penulisan dokumen yang dikembangkan oleh Leslie B. Lamport dan dirilis pada tahun 1985 [3]. LaTeX Memiliki filosofi WYMIWYG (*What you Mean Is What You Get*) yang berarti sesuatu yang ditulis akan berdasarkan arti dari hal tersebut. Oleh karena itu, untuk menambahkan suatu perintah pada dokumen yang sedang ditulis perlu menambahkan suatu *command*. *Command* adalah kata spesial yang menentukan suatu sifat pada LaTeX. Hampir semua *command* pada LaTeX selalu diawali dengan tanda '\' dan beberapa *command* memiliki *parameter*. *Parameter* diawali dengan tanda kurung kurawal buka dan diakhiri dengan kurung kurawal tutup ({...}). File LaTeX memiliki ekstensi .tex. Pada saat membuat sebuah *project* LaTeX hanya perlu menuliskan *command* `\documentclass[option]{class}` 1 kali.

Untuk menulis dokumen pada LaTeX dibutuhkan beberapa *command* yang wajib ada dalam sebuah dokumen, yaitu:

1. `\documentclass[option]{class}`
   Digunakan untuk menentukan jenis dokumen yang *layout* dokumen. Bagian *option* dapat dikosongkan atau dapat digunakan untuk menyimpan pilihan pengaturan *layouting*. Pada Bagian kelas digunakan untuk menentukan tipe dokumen yang akan dibuat. *Command* ini hanya perlu ditulis 1 kali dalam sebuah dokumen.

2. `\maketitle`
   Digunakan untuk menampilkan halaman judul. Biasanya halaman judul akan memuat judul dokumen, nama pengarang dan tanggal pembuatan dokumen. Judul dokumen, nama pengarang dan tanggal pembuatan dapat ditampilkan dengan menambahkan perintah `\title{judul}`, `\author{nama}` dan `\date{tanggal}`.

3. `\begin{document}`...`\end{document}`
   Digunakan untuk mengawali dan mengakhiri sebuah dokumen.

4. `\section{section}`
   Digunakan untuk menampilkan subbab sebuah dokumen.

5. `\texttt{text}`
   Digunakan untuk menampilkan tulisan *monospaced*.

6. `\begin{enumerate}`...`\end{enumerate}`
   Digunakan untuk menampilkan *ordered list*. *List* ini akan menampilkan angka yang terurut. Di dalam *list* ini terdapat *command* `\item` untuk menambahkan isi dari *list* tersebut.

7. `\begin{itemize}`...`\end{itemize}`
   Digunakan untuk menampilkan *unordered list*. *List* ini akan menampilkan simbol spesial. Di dalam *list* ini terdapat *command* `\item` untuk menambahkan isi dari *list* tersebut.

BAB 3

ANALISIS

Bab ini membahas mengenai analisis kebutuhan perangkat lunak dan analisis program sejenis.

3.1 Analisis Kebutuhan Perangkat Lunak

Struktur LaTeX yang digunakan memiliki format sebagai berikut.

```
1  \begin{enumerate}
2  \item \texttt{namaKelas}\\
3  {penjelasan kelas}
4
5  Atribut yang dimiliki kelas ini adalah sebagai berikut.
6  \begin{itemize}
7    \item \texttt{atribut} -
8    {penjelasan tentang atribut}.
9  \end{itemize}
10
11 \textit{Method} yang terdapat pada kelas Pertambahan adalah sebagai berikut.
12 \begin{itemize}
13   \item \texttt{method}\\
14   {penjelasan method}
15
16   \textbf{Parameter:}
17   \begin{itemize}
18     \item \texttt{parameter} -
19     {penjelasan dari parameter}.
20   \end{itemize}
21
22   \textbf{Return Value:} {penjelasan return-type method}\\
23   \textbf{Exception:} {penjelasan exception jika terdapat exception}
24   \textbf{See Also:} {penjelasan tag @see jika terdapat tag tersebut}
25 \end{itemize}
```

Listing 3.1: Potongan kode LaTeX

Potongan kode yang terdapat pada listing 3.1 adalah struktur lengkap LaTeX yang digunakan, akan dijelaskan sebagai berikut.

1. *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.

2. *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.

3. *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.

4. *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*.

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

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 ini merupakan Kelas Pertambahan
5   * kelas ini memiliki beberapa fungsi yaitu {@link #pertambahan(int,int) pertambahan}
6   *
7   * @author Adli Fariz Bonaputra
8   * @see "Pertambahan"
9   *
10  */
11 public class Pertambahan {
12
13     /**
14      * Atribut A
15      */
16     private int a;
17     /**
18      * Atribut B
19      */
20     private int b;
21
22     /**
23      * Method Pertambahan
24      *
25      * @param a Bilangan Pertama
26      * @param b Bilangan Kedua
27      *
```

```
28      * @return hasil penjumlahan 2 buah bilangan
29      */
30     public int pertambahan(int a, int b) {
31         int hasil = 0;
32         hasil = a + b;
33         return hasil;
34     }
35 }
```

Listing 3.2: Contoh kode *java* yang diuji

Potongan kode yang terdapat pada listing 3.2 adalah struktur *file java* yang digunakan, akan dijelaskan sebagai berikut.

1. Setiap *file java* harus terletak di dalam sebuah *package* yang sama.
2. 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.
3. Setiap deklarasi atribut harus memiliki *access modifier*, tipe atribut dan nama atribut serta memiliki javadoc untuk penjelasan tentang atribut tersebut.
4. 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  \end{itemize}
29 \end{enumerate}
```

Listing 3.3: Contoh hasil konversi *Javadoc* ke LaTeX

Hasil konversi 3.3 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*.

3.2 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.

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

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

3. `-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.

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

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

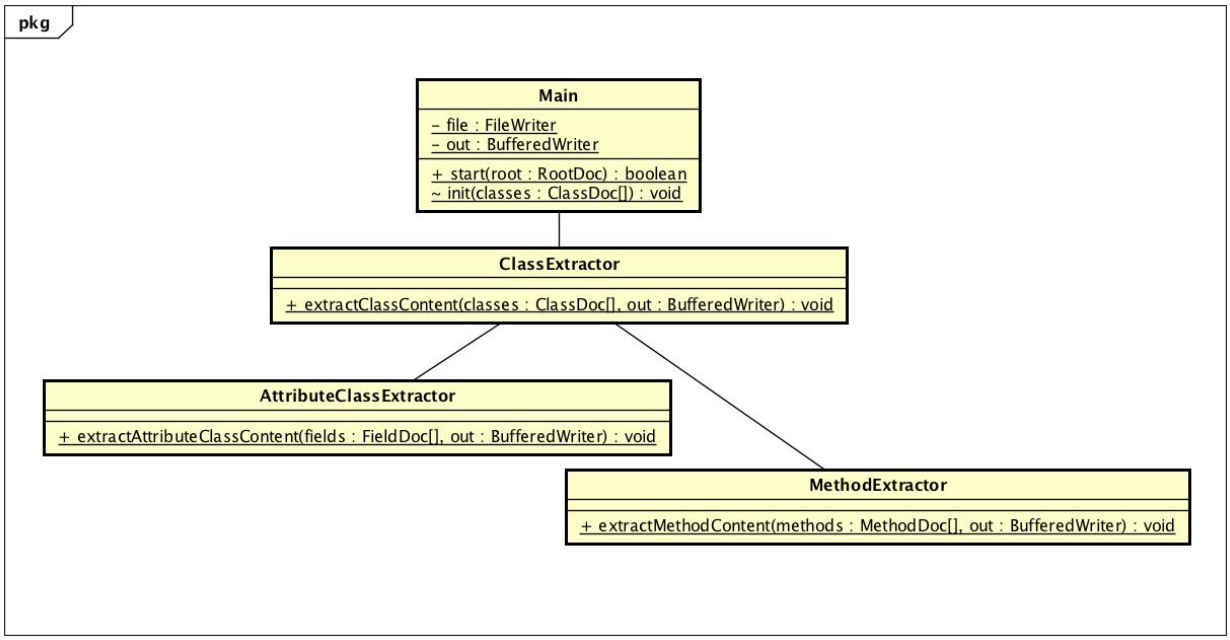
BAB 4

PERANCANGAN

Bab ini membahas mengenai perancangan aplikasi yang akan dibangun meliputi diagram kelas rinci beserta deskripsi dan fungsinya.

4.1 Rancangan Kelas Lengkap

Rancangan kelas dibawah ini akan menampilkan keseluruhan kelas yang akan digunakan. Deskripsi kelas berserta fungsi dari diagram kelas tersebut adalah sebagai berikut:



Gambar 4.1: Kelas Diagram

1. `ClassExtractor`
   Kelas ini merupakan kelas untuk mengambil informasi sebuah kelas. Atribut yang dimiliki oleh kelas ini adalah sebagai berikut:
   -
2. `AttributeClassExtractor`
   Kelas ini merupakan kelas untuk mengambil informasi sebuah atribut yang terdapat pada kelas. Atribut yang dimiliki oleh kelas ini adalah sebagai berikut:
   -

3. `MethodExtractor`
   Kelas ini merupakan kelas untuk mengambil informasi sebuah *method* terdapat pada kelas. Atribut yang dimiliki oleh kelas ini adalah sebagai berikut:

   -

4.2 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*:

DAFTAR REFERENSI

[1] Oracle (1993) javadoc - the java api documentation generator. http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/javadoc.html#public. 27 September 2017.

[2] Oracle (1993) Javadoc doclet api. http://docs.oracle.com/javase/7/docs/jdk/api/javadoc/doclet/index.html. 5 oktober 2017.

[3] Lamport, L. (1994) *LaTeX: A Document Preparation System*, 2 edition.