

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 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

Berdasarkan 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.

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 Perangkat Lunak
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 L^AT_EX.

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.
7. Ringkasan daftar tabel *package*, kelas dan seluruh anggota dari kelas.
8. Turunan dari setiap *package* dan kelas.
9. 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 diinginkan, 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 tersebut 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, maka 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 halaman 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 lower-left frame
10		
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 resolving external refs
16		
17	stylesheet.css	HTML style sheet for defining fonts, colors and positions
18		
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 for this package
26		
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 L^AT_EX

L^AT_EX adalah sebuah bahasa *markup* untuk sistem penulisan dokumen yang dikembangkan oleh Leslie B. Lamport dan dirilis pada tahun 1985 [3]. L^AT_EX 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 L^AT_EX. Hampir semua *command* pada L^AT_EX selalu diawali dengan tanda '`' dan beberapa *command* memiliki *parameter*. *Parameter* diawali dengan tanda kurung kurawal buka dan diakhiri dengan kurung kurawal tutup (`{...}`). File L^AT_EX memiliki ekstensi `.tex`. Pada saat membuat sebuah *project* L^AT_EX hanya perlu menuliskan *command* `\documentclass[option]{class}` 1 kali.

Untuk menulis dokumen pada L^AT_EX 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 \textbf{Override:} {penjelasan apabila jika terdapat {\it override method} }
26 \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* ataupun *tag {@link}*. Jika tidak ada informasi dari *tag - tag* tersebut akan menampilkan tulisan tidak ada.

6. *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 dapat dilihat pada lampiran A. Struktur kode 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. Setiap 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 *L^AT_EX*. Perangkat lunak akan membaca satu persatu *file java* dan informasi yang terdapat pada setiap *file java* tersebut dimasukkan ke dalam *file L^AT_EX*.

```

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:} \texttt{pertambahan} dari kelas \texttt{operasiMatematikaInterface}
29   \end{itemize}
30 \end{enumerate}

```

Listing 3.2: Contoh hasil konversi *Javadoc* ke \LaTeX

Hasil konversi 3.2 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 Lua \LaTeX untuk menghasilkan dokumen PDF dari sebuah *file \LaTeX*. Hasil PDF yang dihasilkan oleh TeXDoclet dapat dilihat pada lampiran ??

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 Lua \LaTeX .

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*.

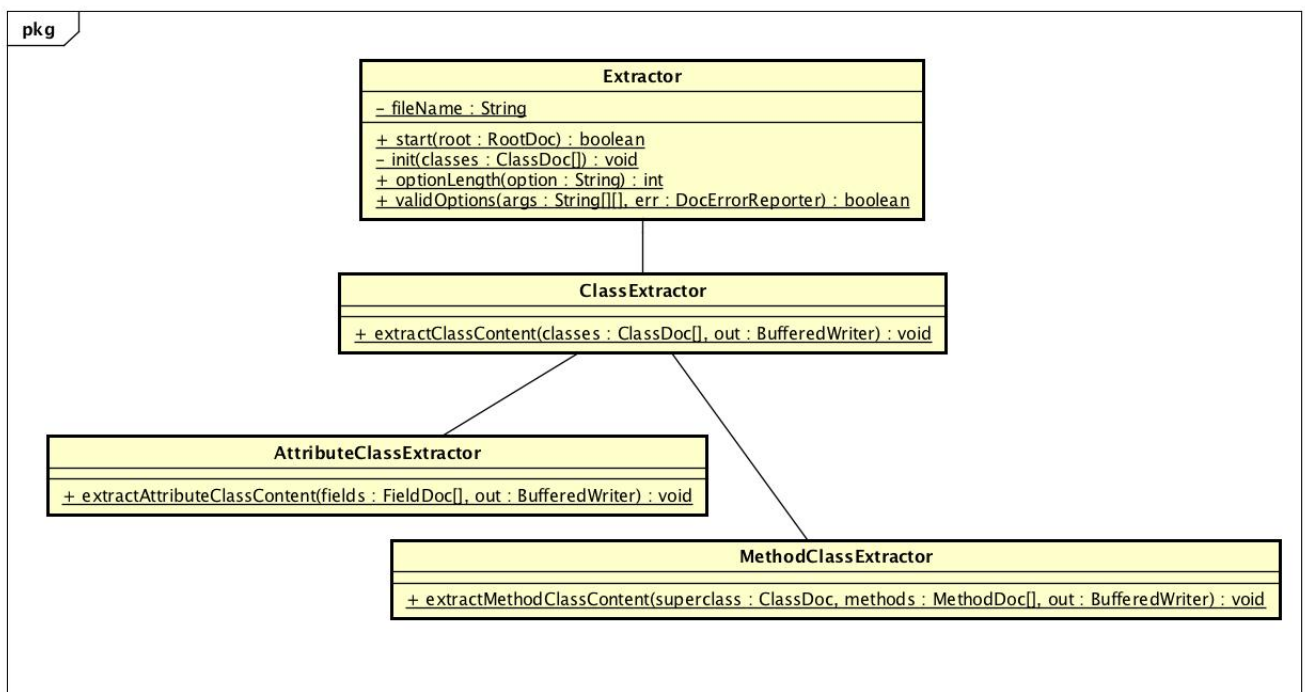
BAB 4

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 beserta fungsi dari diagram kelas tersebut adalah sebagai berikut:



Gambar 4.1: Kelas Diagram

1. 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, java.io.BufferedWriter out)`

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

Parameter:

- `FieldDoc fields` - sebuah array berisikan sejumlah atribut dari kelas

- `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

2. 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, java.io.BufferedWriter out)`

Method ini akan menampilkan nama kelas berserta penjelasan dari sebuah kelas

Parameter:

- `ClassDoc classes` - sebuah array berisikan sejumlah kelas
- `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

3. Extractor

Kelas ini merupakan kelas untuk menjalankan *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(com.sun.javadoc.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.

Return Value: 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:

- `ClassDoc classes` - sebuah array yang berisikan sekumpulan *file java* yang akan di proses.

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

- `public static int optionLength(java.lang.String option)`

Method untuk menghitung banyak *option* yang digunakan pada *command-line*

Parameter:

- `String option` - sebuah *option*

Return Value: panjang setiap *option*

Exception: Tidak memiliki *exception*

- `public static boolean validOptions(java.lang.String[] [] args, com.sun.javadoc.DocErrorReporter err)`

Pengecekan option valid

Parameter:

- `String args` - String array 2 dimensi dari option
- `DocErrorReporter err` - sebuah error jika tidak terdapat option tersebut.

Return Value: bernilai true jika option tersebut dikenali, false jika option tersebut tidak dikenali

Exception: Tidak memiliki *exception*

4. 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(com.sun.javadoc.ClassDoc superclass, com.sun.javadoc.MethodDoc[] methods, java.io.BufferedWriter out)`

Method ini akan menampilkan *method-method* yang dimiliki oleh sebuah kelas

Parameter:

- `ClassDoc superclass` - sebuah objek `ClassDoc`
- `MethodDoc methods` - sebuah array berisikan sejumlah *method* dari kelas
- `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

- `private static void ParameterMethod(java.io.BufferedWriter out, com.sun.javadoc.ParameterDoc paramMethod, com.sun.javadoc.ParamTag[] paramTags)`

Method ini akan menampilkan parameter *method-method* yang dimiliki oleh sebuah kelas

Parameter:

- `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text
- `Parameter paramMethod` - sebuah array berisikan sejumlah *method* dari kelas
- `ParamTag paramTags` - sebuah array berisikan sejumlah parameter *method* dari kelas

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

- `private static void ReturnTypeMethod(java.io.BufferedWriter out, com.sun.javadoc.TypeDoc type, com.sun.javadoc.Tag[] returnTags)`

Method ini akan menampilkan *return type* dari *method-method* yang dimiliki oleh sebuah kelas

Parameter:

- `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text
- `Type type` - sebuah objek `Type`
- `Tag returnTags` - sebuah array berisikan sejumlah *return type* dari *method* dari kelas

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

- `private static void ExceptionMethod(java.io.BufferedWriter out, com.sun.javadoc.Tag[] throwTags)`

Method ini akan menampilkan *return type* dari *method-method* yang dimiliki oleh sebuah kelas

Parameter:

- `BufferedWriter out` - turunan dari kelas `Writer` yang digunakan untuk menulis file text
- `Tag throwTags` - sebuah array berisikan sejumlah *exception* dari *method* dari kelas

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

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*:

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

Gambar 4.2: 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 4.2 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 4.3: Memasukkan *option* yang akan digunakan

Langkah kedua adalah menjalankan perangkat lunak yang dibuat. Diawali dengan *command* `javadoc` lalu diikuti 5 buah argumen. Argumen pertama(hijau) adalah *option* untuk menamai *file* sesuai dengan yang ditentukan. Sebagai contoh pada gambar 4.3, *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 4.3, 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 4.4: Hasil tampilan jika proses konversi selesai

Perangkat lunak yang dibuat akan membaca seluruh isi folder yang dituju, pada contoh gambar 4.4, 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.

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Bab ini terdiri atas dua bagian, yaitu Implementasi Perangkat Lunak dan Pengujian Perangkat Lunak. Bagian implementasi berisi penjelasan bagaimana perangkat lunak dibuat dan langkah-langkah dalam penggunaan perangkat lunak. Sedangkan bagian pengujian berisi hasil pengujian fungsional terhadap perangkat lunak yang telah dibuat.

5.1 Implementasi Perangkat Lunak

Perangkat lunak dibuat menggunakan bahasa *java* dan dimasukkan ke dalam sebuah *jar* sehingga dapat digunakan dengan cara mengeksekusi perintah. Penggunaan *file jar* tersebut dapat dilakukan melalui *Terminal* di Linux/Mac atau *Command Prompt* di Windows. Berikut perintah yang digunakan untuk menjalankan perangkat lunak.

```
javadoc [options] {sourcefiles|packagenames}
```

Pada potongan perintah diatas memiliki 2 parameter yaitu *option* dan *packagenames*. Parameter *packagename* adalah parameter untuk *package* yang akan sebagai masukan dari perangkat lunak. Parameter *option* adalah beberapa perintah pendukung. Berikut beberapa perintah *option* yang digunakan untuk mendukung berjalannya perangkat lunak.

- `-filename <file-name>` - Menghasilkan *output file* dengan nama `file-name.tex`. Jika *option* ini tidak digunakan maka nama *file* yang dihasilkan akan bernama `doc.tex`
- `-doclet <class>` - Kelas yang dibuat untuk menghasilkan *output*. Pada kasus ini adalah `extractor.Extractor`
- `-docletpath <path>` - Letak *doclet* yang sudah di-*package* menjadi *file jar*
- `-sourcepath <pathlist>` - Letak *source file* sebagai masukan.
- `-subpackages <subpkglist>` - Letak *subpackage* yang akan dimuat secara rekursif.

Untuk penggunaan perintah diatas, Langkah pertama membuka aplikasi *Terminal* atau *Command Prompt*. Langkah kedua mengetik perintah `javadoc` lalu diikuti dengan perintah pendukungnya seperti yang sudah dijelaskan diatas. Berikut contoh perintah lengkap yang digunakan.

```
javadoc -filename <file-name>
      -doclet extractor.Extractor
      -docletpath GenerateJavadocToLatex.jar
      -sourcepath <path/to/directory>
      -subpackages <packagenames> <sourcefiles|packagenames>
```

5.2 Pengujian Perangkat Lunak

Pada sub bab ini akan menjelaskan Lingkungan Pengujian dan Pengujian Fungsional. Pengujian Fungsional akan menguji perangkat lunak terhadap kode program sederhana serta menguji kode program perangkat lunak yang dibuat.

5.2.1 Lingkungan Pengujian

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

1. Processor: Intel Core i7 2.5-3.7GHz
2. RAM: 16.00 GB DDR3
3. Harddisk : 512MB SSD
4. VGA : Intel Iris Pro dan AMD Radeon R9 M370X
5. Sistem Operasi: macOS High Sierra
6. Versi Java: 1.8.0_121
7. Code Editor: Netbeans 8.2

5.2.2 Pengujian Fungsional

Pada pengujian fungsional dilakukan terhadap 2 kasus yaitu pengujian kode program sederhana dan kode program perangkat lunak yang dibuat.

Pada kasus pertama dijalankan perintah sebagai berikut.

```
javadoc -filename operasimatematika
        -doclet extractor.Extractor
        -docletpath GenerateJavadocToLatex.jar
        ../OperasiMatematika/*
```

Perintah tersebut mengekstrak kode program yang dapat dilihat pada lampiran [A.1](#). Setelah perintah tersebut dieksekusi akan menghasilkan *file* \LaTeX yang dapat dilihat pada lampiran [A.2](#). Jika *file* \LaTeX tersebut dieksekusi menggunakan perintah `pdflatex` akan menghasilkan sebuah *file* PDF yang dapat dilihat pada lampiran [A.3](#).

Dapat dilihat dari hasil [A.3](#) terdapat 5 kelas yang diekstrak menjadi 5 buah *item list enumerate* yang masing berisikan nama kelas dari 5 file tersebut. *Method-method* yang terdapat kelas tersebut akan dibuat dengan menggunakan *list itemize* sebanyak *n* buah *method* dan parameter dari *method* tersebut dibuat menggunakan *list itemize* sebanyak *n* buah parameter.

Pada kasus kedua dijalankan perintah sebagai berikut.

```
javadoc -filename javadoctolatex
        -doclet extractor.Extractor
        -docletpath GenerateJavadocToLatex.jar
        src/extractor/*
```

Perintah tersebut mengekstrak kode program yang dapat dilihat pada lampiran [??](#). Setelah perintah tersebut dieksekusi akan menghasilkan *file* \LaTeX yang dapat dilihat pada lampiran [??](#). Jika *file* \LaTeX tersebut dieksekusi menggunakan perintah `pdflatex` akan menghasilkan sebuah *file* PDF yang dapat dilihat pada lampiran [??](#).

Dapat dilihat dari hasil [??](#) terdapat 4 kelas yang diekstrak menjadi 4 buah *item list enumerate* yang masing berisikan nama kelas dari 4 file tersebut. Atribut kelas akan dibuat dengan menggunakan

list itemize sebanyak n buah atribut. *Method-method* yang terdapat kelas tersebut akan dibuat dengan menggunakan *list itemize* sebanyak n buah *method* dan parameter dari *method* tersebut dibuat menggunakan *list itemize* sebanyak n buah parameter.

5.2.3 Pengujian Eksperimental

Pengujian eksperimental dilakukan terhadap kode program SIAModels. SIAModels adalah sekumpulan kelas *java* yang mewakili objek-objek Sistem Informasi Akademik Universitas Katolik Parahyangan. SIAModels ini berisikan objek matakuliah pada Fakultas Teknologi Informasi dan Sains.

Pada pengujian menggunakan SIAModels dilakukan perintah sebagai berikut.

```
javadoc -filename siamodels
        -doclet extractor.Extractor
        -docletpath GenerateJavadocToLatex.jar
        -sourcepath SIAModels/src/main/java
        -subpackages id.ac.unpar.siamodels
        id.ac.unpar.siamodels
```

Perintah tersebut mengekstrak kode program yang dapat dilihat pada lampiran ???. Setelah perintah tersebut dieksekusi akan menghasilkan *file* L^AT_EX yang dapat dilihat pada lampiran ???. Jika *file* L^AT_EX tersebut dieksekusi menggunakan perintah *pdflatex* akan menghasilkan sebuah *file* PDF yang dapat dilihat pada lampiran ??.

Dapat dilihat dari hasil ??? terdapat n buah kelas dan n buah *innerclass* yang diekstrak menjadi n buah *item list enumerate* yang masing berisikan nama kelas dan *innerclass* dari n *file* tersebut. Atribut kelas akan dibuat dengan menggunakan *list itemize* sebanyak n buah atribut. *Method-method* yang terdapat kelas tersebut akan dibuat dengan menggunakan *list itemize* sebanyak n buah *method* dan parameter dari *method* tersebut dibuat menggunakan *list itemize* sebanyak n buah parameter. Terdapat beberapa keanehan yang ditemukan setelah mengeksekusi perintah *pdflatex*.

- Terdapat karakter aneh yaitu tanda "!" terbalik yang merepresentasikan tanda lebih kecil "<" dan tanda "?" terbalik yang merepresentasikan tanda lebih besar ">". Karakter aneh ini muncul pada kelas:
 1. TahunSemester - pada atribut kelas
 2. Mahasiswa - pada *method* *calculateIPKLulus()*, *calculateIPLulus()*, *calculateIPTempuh()*, *calculateIPKumulatif()*, *calculateIPKTempuh()* dan *calculateIPS()*

Karakter aneh tersebut muncul karena perangkat lunak tidak menangani hal tersebut.

- Beberapa deskripsi tidak muncul, contohnya seperti pada kelas:
 1. Semester - pada deskripsi kelas, atribut kelas dan *method*
 2. TahunSemester - pada deskripsi *method* seperti pada *method* *getTahun()*
 3. Mahasiswa - pada deskripsi kelas, atribut kelas dan beberapa *method*
 4. Mahasiswa.Nilai - pada beberapa deskripsi *method*
 5. JadwalKuliah - pada deskripsi kelas, atribut kelas dan *method*
 6. MataKuliah - pada deskripsi kelas, atribut kelas, dan *method*
 7. Dosen - pada deskripsi kelas, atribut kelas dan *method*

Setelah dilakukan pemeriksaan terhadap deskripsi yang tidak muncul pada hasil PDF ternyata kesalahan bukan dari perangkat lunak melainkan kode program yang menjadi masukan perangkat lunak memiliki dokumentasi yang kurang lengkap.

- Parameter setiap *method* memiliki tipe variabel yang panjang. Setelah dilakukan pemeriksaan ternyata kesalahan terdapat pada perangkat lunak yang tidak menangani agar tipe variabel menjadi lebih singkat.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil pembangunan perangkat lunak Konversi Javadoc ke L^AT_EX , didapatkanlah kesimpulan-kesimpulan sebagai berikut:

1. Telah berhasil mengimplementasikan *library* Javadoc Doclet API untuk mengekstraksi informasi dari sekumpulan *file java*.

6.2 Saran

Dari hasil penelitian termasuk kesimpulan yang didapat, berikut adalah beberapa saran untuk pengembangan.

1. Pada saat ini, perangkat lunak tidak mengatasi *tag-tag html*. Pada pengujian menggunakan SIAModels terdapat beberapa kelas yang memiliki *tag-tag html* pada *javadoc* seperti contoh pada kelas **Mahasiswa** pada fungsi `calculateIPKLulus()`. Terdapat kode aneh yaitu tanda "!" yang merepresentasikan tanda lebih kecil "<" dan tanda "?" terbalik yang merepresentasikan tanda lebih besar ">". Sebaiknya perangkat lunak dapat mengatasi *tag-tag html* jika javadoc memiliki *tag-tag* tersebut.
2. Parameter setiap *method* memiliki tipe variabel yang panjang. Sebaiknya perangkat lunak dapat mengatasi tipe variabel yang panjang tersebut menjadi lebih singkat agar *method* yang didokumentasikan akan sama seperti pada kode program yang dibuat.
3. Pada saat ini, perangkat lunak menampilkan tulisan "tidak memiliki *exception*" pada bagian **Exception**. Sebaiknya mengikuti standar dari *javadoc* bahwa jika *method* tidak memiliki *exception* maka bagian **Exception** tidak akan muncul pada dokumentasi.

DAFTAR REFERENSI

- [1] Oracle (1993) javadoc - the java api documentation generator. <https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html>. 27 September 2017.
- [2] Oracle (1993) Javadoc doclet api. <https://docs.oracle.com/javase/8/docs/jdk/api/javadoc/doclet/index.html>. 5 oktober 2017.
- [3] Lamport, L. (1994) *LaTeX: A Document Preparation System*, 2 edition.

LAMPIRAN A

PENGUJIAN TERHADAP KODE PROGRAM SEDERHANA

A.1 Kode Program

Listing A.1: OperasiMatematikaInterface.java

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package javadoc;
7
8  /**
9   * Kelas Abstract OperasiMatematika. Kelas ini memiliki method {@link #calculate(int, int)}
10   *
11   * @author abathz
12   */
13  public abstract class OperasiMatematikaInterface {
14
15      /**
16       * Method untuk menghasilkan perhitungan 2 buah bilangan
17       * @param a Bilangan pertama
18       * @param b Bilangan kedua
19       * @return hasil perhitungan 2 buah bilangan {@link {@link Integer#NaN}}
20       */
21      public int calculate(int a, int b){return 0;}
22  }
```

Listing A.2: Pembagian.java

```
1  package javadoc;
2
3  /**
4   * Kelas ini merupakan Kelas Pembagian
5   *
6   * @author Adli Fariz Bonaputra
7   * @see "Pembagian"
8   */
9  public class Pembagian extends OperasiMatematikaInterface {
10
11      /**
12       * Atribut A
13       */
14      private int a;
15      /**
16       * Atribut B
17       */
18      private int b;
19
20      @Override
21      public int calculate(int a, int b) {
22          int hasil = a / b;
23          return hasil;
24      }
25
26  }
```

Listing A.3: Pengurangan.java

```
1  package javadoc;
2
3  /**
4   * Kelas ini merupakan Kelas Pengurangan
5   *
6   * @author Adli Fariz Bonaputra
7   * @see "Pengurangan"
8   */
9  public class Pengurangan extends OperasiMatematikaInterface {
10
11      /**
12       * Atribut A
13       */
```

```

14 private int a;
15 /**
16  * Atribut B
17  */
18 private int b;
19
20 @Override
21 public int calculate(int a, int b) {
22     int hasil = a - b;
23     return hasil;
24 }
25 }

```

Listing A.4: Perkalian.java

```

1 package javadoc;
2
3 /**
4  * Kelas ini merupakan Kelas Perkalian
5  *
6  * @author Adli Fariz Bonaputra
7  * @see "Perkalian"
8  *
9  */
10 public class Perkalian extends OperasiMatematikaInterface {
11
12     /**
13     * Atribut A
14     */
15     private int a;
16     /**
17     * Atribut B
18     */
19     private int b;
20
21     @Override
22     public int calculate(int a, int b) {
23         int hasil = a * b;
24         return hasil;
25     }
26 }

```

Listing A.5: Pertambahan.java

```

1 package javadoc;
2
3 /**
4  * Kelas ini merupakan Kelas Pertambahan
5  *
6  * @author Adli Fariz Bonaputra
7  * @see "Pertambahan"
8  *
9  */
10 public class Pertambahan extends OperasiMatematikaInterface {
11
12     /**
13     * Atribut A
14     */
15     private int a;
16     /**
17     * Atribut B
18     */
19     private int b;
20
21     @Override
22     public int calculate(int a, int b) {
23         int hasil = a + b;
24         return hasil;
25     }
26 }

```

A.2 Hasil Latex

Listing A.6: operasimatematika.tex

```

1 \documentclass{article}
2 \begin{document}
3 \begin{enumerate}
4 \item \texttt{OperasiMatematikaInterface}
5
6 Kelas Abstract OperasiMatematika. Kelas ini memiliki method \texttt{calculate(int, int)}
7
8 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
9 \begin{itemize}
10 \item \texttt{public int calculate(int a, int b)}
11
12 Method untuk menghasilkan perhitungan 2 buah bilangan
13
14 \textbf{Parameter:}
15 \begin{itemize}
16 \item \texttt{int a} -
17 Bilangan pertama

```



```

18 \item \texttt{int b} -
19 Bilangan kedua
20 \end{itemize}
21 \textbf{Return Value}: hasil perhitungan 2 buah bilangan DoubleNaN
22
23 \textbf{Exception}: Tidak memiliki \textit{exception}
24
25 \end{itemize}
26 \item \texttt{Pembagian}
27
28 Kelas ini merupakan Kelas Pembagian
29
30 Atribut yang dimiliki kelas ini adalah sebagai berikut.
31 \begin{itemize}
32 \item \texttt{int a} - Atribut A
33 \item \texttt{int b} - Atribut B
34 \end{itemize}
35 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
36 \begin{itemize}
37 \item \texttt{public int calculate(int a, int b)}
38
39 Method untuk menghasilkan perhitungan 2 buah bilangan
40
41 \textbf{Parameter}:
42 \begin{itemize}
43 \item \texttt{int a} -
44 Bilangan pertama
45 \item \texttt{int b} -
46 Bilangan kedua
47 \end{itemize}
48 \textbf{Return Value}: hasil perhitungan 2 buah bilangan DoubleNaN
49
50 \textbf{Exception}: Tidak memiliki \textit{exception}
51
52 \end{itemize}
53 \item \texttt{Pengurangan}
54
55 Kelas ini merupakan Kelas Pengurangan
56
57 Atribut yang dimiliki kelas ini adalah sebagai berikut.
58 \begin{itemize}
59 \item \texttt{int a} - Atribut A
60 \item \texttt{int b} - Atribut B
61 \end{itemize}
62 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
63 \begin{itemize}
64 \item \texttt{public int calculate(int a, int b)}
65
66 Method untuk menghasilkan perhitungan 2 buah bilangan
67
68 \textbf{Parameter}:
69 \begin{itemize}
70 \item \texttt{int a} -
71 Bilangan pertama
72 \item \texttt{int b} -
73 Bilangan kedua
74 \end{itemize}
75 \textbf{Return Value}: hasil perhitungan 2 buah bilangan DoubleNaN
76
77 \textbf{Exception}: Tidak memiliki \textit{exception}
78
79 \end{itemize}
80 \item \texttt{Perkalian}
81
82 Kelas ini merupakan Kelas Perkalian
83
84 Atribut yang dimiliki kelas ini adalah sebagai berikut.
85 \begin{itemize}
86 \item \texttt{int a} - Atribut A
87 \item \texttt{int b} - Atribut B
88 \end{itemize}
89 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
90 \begin{itemize}
91 \item \texttt{public int calculate(int a, int b)}
92
93 Method untuk menghasilkan perhitungan 2 buah bilangan
94
95 \textbf{Parameter}:
96 \begin{itemize}
97 \item \texttt{int a} -
98 Bilangan pertama
99 \item \texttt{int b} -
100 Bilangan kedua
101 \end{itemize}
102 \textbf{Return Value}: hasil perhitungan 2 buah bilangan DoubleNaN
103
104 \textbf{Exception}: Tidak memiliki \textit{exception}
105
106 \end{itemize}
107 \item \texttt{Pertambahan}
108
109 Kelas ini merupakan Kelas Pertambahan
110
111 Atribut yang dimiliki kelas ini adalah sebagai berikut.
112 \begin{itemize}
113 \item \texttt{int a} - Atribut A
114 \item \texttt{int b} - Atribut B
115 \end{itemize}
116 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.

```

```
117 \begin{itemize}
118 \item \texttt{public int calculate(int a, int b)}
119
120 Method untuk menghasilkan perhitungan 2 buah bilangan
121
122 \textbf{Parameter:}
123 \begin{itemize}
124 \item \texttt{int a} -
125 Bilangan pertama
126 \item \texttt{int b} -
127 Bilangan kedua
128 \end{itemize}
129 \textbf{Return Value:} hasil perhitungan 2 buah bilangan DoubleNaN
130
131 \textbf{Exception:} Tidak memiliki \texttt{exception}
132
133 \end{itemize}
134 \end{enumerate}
135 \end{document}
```

A.3 Hasil PDF

1. OperasiMatematikaInterface

Kelas Abstract OperasiMatematika. Kelas ini memiliki method `calculate(int, int)`

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

- `public int calculate(int a, int b)`
Method untuk menghasilkan perhitungan 2 buah bilangan
Parameter:
 - `int a` - Bilangan pertama
 - `int b` - Bilangan kedua**Return Value:** hasil perhitungan 2 buah bilangan DoubleNaN
Exception: Tidak memiliki *exception*

2. Pembagian

Kelas ini merupakan Kelas Pembagian

Atribut yang dimiliki kelas ini adalah sebagai berikut.

- `int a` - Atribut A
- `int b` - Atribut B

Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public int calculate(int a, int b)`
Method untuk menghasilkan perhitungan 2 buah bilangan
Parameter:
 - `int a` - Bilangan pertama
 - `int b` - Bilangan kedua**Return Value:** hasil perhitungan 2 buah bilangan DoubleNaN
Exception: Tidak memiliki *exception*

3. Pengurangan

Kelas ini merupakan Kelas Pengurangan

Atribut yang dimiliki kelas ini adalah sebagai berikut.

- `int a` - Atribut A
- `int b` - Atribut B

Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public int calculate(int a, int b)`
Method untuk menghasilkan perhitungan 2 buah bilangan
Parameter:

- `int a` - Bilangan pertama
- `int b` - Bilangan kedua

Return Value: hasil perhitungan 2 buah bilangan DoubleNaN

Exception: Tidak memiliki *exception*

4. Perkalian

Kelas ini merupakan Kelas Perkalian

Atribut yang dimiliki kelas ini adalah sebagai berikut.

- `int a` - Atribut A
- `int b` - Atribut B

Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public int calculate(int a, int b)`

Method untuk menghasilkan perhitungan 2 buah bilangan

Parameter:

- `int a` - Bilangan pertama
- `int b` - Bilangan kedua

Return Value: hasil perhitungan 2 buah bilangan DoubleNaN

Exception: Tidak memiliki *exception*

5. Pertambahan

Kelas ini merupakan Kelas Pertambahan

Atribut yang dimiliki kelas ini adalah sebagai berikut.

- `int a` - Atribut A
- `int b` - Atribut B

Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public int calculate(int a, int b)`

Method untuk menghasilkan perhitungan 2 buah bilangan

Parameter:

- `int a` - Bilangan pertama
- `int b` - Bilangan kedua

Return Value: hasil perhitungan 2 buah bilangan DoubleNaN

Exception: Tidak memiliki *exception*