SKRIPSI

PERANGKAT LUNAK KONVERSI KOMENTAR JAVA KE LaTeX

Adli Fariz Bonaputra

NPM: 2012730082

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2018

UNDERGRADUATE THESIS

JAVA COMMENT TO LATEX SOFTWARE CONVERSION

Adli Fariz Bonaputra

NPM: 2012730082

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2018

LEMBAR PENGESAHAN

PERANGKAT LUNAK KONVERSI KOMENTAR JAVA KE LaTeX

Adli Fariz Bonaputra

NPM: 2012730082

Bandung, 18 Desember 2018

Menyetujui,

Pembimbing

Dr. Veronica Sri Moertini

Ketua Tim Penguji	Anggota Tim Penguji
Dott. Thomas Anung Basuki	**Elisati Hulu, M.T.**

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PERANGKAT LUNAK KONVERSI KOMENTAR JAVA KE LATEX

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 18 Desember 2018

Meterai
Rp. 6000

Adli Fariz Bonaputra
NPM: 2012730082

ABSTRAK

Tujuan 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*. Bahasa pemrograman *Java* memiliki komentar yang berguna 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. Bahasa *markup* adalah sebuah bahasa yang menganotasikan dokumen dengan cara menambahkan sintaks tertentu agar dapat dibedakan. LaTeX dapat menghasilkan dokumen yang terstruktur sesuai dengan yang ditulis oleh penulis. 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*.

Kata-kata kunci: Skripsi, *Javadoc*, *Doclet*, LaTeX

ABSTRACT

The purpose of the undergraduate thesis is the Information Technology and Science Faculty (FTIS) Informatics Engineering Study Program at Parahyangan Catholic University (Unpar) is to make software. Software made generally uses the programming language *Java*. The programming language it Java has useful comments as information from the class, *interface*, *method* and also *custom tag* created, so that the information can be used as a software explanation in the device documentation soft. To document the software made, all students are required to use LaTeX in making a thesis documentation. LaTeX is the language of *markup* to compile a documentation. Language *markup* is a language that annotates documents by adding certain syntax so that they can be distinguished. LaTeX can produce structured documents as written by the author. Generally the final form of a document created by LaTeX is usually a *file* PDF.

In one of the thesis documentation chapters, there is an explanation of each class in the software created. This explanation can actually be taken from *Javadoc* which was created in the *Java* class, but currently based on these observations it is still typed manually from *Javadoc* into the format LaTeX, so that it takes more time to document each class in the software created.

Therefore, it is necessary to develop a software that can extract information on *Javadoc* to LaTeX format automatically. This software implements a *Application Programming Interface* (API) which is used to retrieve information in the form of a class name, *interface*, *method* and also *custom tag* contained in a *Java file*.

Keywords: Thesis, Javadoc, Doclet, LaTeX

Keluarga, karmila, orang terdekat dan diri sendiri

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas seluruh berkat yang diberikan kepada penulis sehingga dapat menyelesaikan tugas akhir dengan judul Perangkat Lunak Konversi Komentar Java ke LaTeX dengan baik dan tepat waktu. Penulis juga berterima kasih kepada pihak-pihak yang telah memberikan dukungan dan bantuan kepada penulis dalam menyelesaikan tugas akhir ini, yaitu:

1. Orang tua dan keluarga yang selalu memberikan dukungan kepada penulis.
2. Bapak Pascal Alfadian sebagai dosen pembimbing yang telah membimbing penulis hingga dapat menyelesaikan tugas akhir ini.
3. Bapak Thomas Anung Basuki dan Bapak Elisati sebagai dosen penguji yang telah membantu dalam menguji tugas akhir ini.
4. Karmila Puspitasari yang tidak pernah berhenti untuk selalu menemani, memberikan dukungan dan memberikan semangat kepada penulis hingga dapat menyelesaikan tugas akhir ini.
5. Andre dan Momon yang sampai penulis tidak tahu harus berkata-kata apa. Terima kasih.
6. Teman-teman Zero Hour Coffee dan Bahagia Kopi yang selalu memberikan semangat dan dukungan.
7. Teman-teman kontakan 7B yang sudah memberikan tempatnya sebagai tempat penulis mencari inspirasi.
8. Teman-teman Teknik Informatika UNPAR angkatan 2012 yang telah berbagi ilmu dan selalu memberikan semangat kepada penulis.
9. Pihak-pihak lain yang belum disebutkan, yang berperan dalam penyelesaian tugas akhir ini.

Akhir kata, penulis berharap agar tugas akhir ini dapat bermanfaat bagi pembaca yang hendak melakukan penelitian dan pengembangan yang terkait dengan tugas akhir ini.

Bandung, Desember 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

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 aktivitas yang dianggap berharga atau penting. Dokumentasi yang sudah dibuat dapat menjadi referensi untuk memandu dalam melakukan sebuah aktivitas.

Dalam bidang teknologi informasi, dokumentasi kode program *Java* umumnya ditulis dalam komentar disetiap *file java*. Komentar tersebut berisikan informasi dari sebuah kelas, *interface*, *method* dan *custom tag*. *Javadoc* adalah sebuah *tools* yang dimiliki oleh *Java* yang berguna untuk mengambil informasi tersebut dan menjadikannya sebagai dokumentasi. Umumnya mendokumentasikan sebuah 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.

Tujuan 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*. Bahasa pemrograman *Java* memiliki komentar yang berguna 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. Bahasa *markup* adalah sebuah bahasa yang menganotasikan dokumen dengan cara menambahkan sintaks tertentu agar dapat dibedakan. LaTeX dapat menghasilkan dokumen yang terstruktur sesuai dengan yang ditulis oleh penulis. 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 mengambil 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 sebuah poin yang menjadi rumusan masalah dari masalah ini. Rumusan masalah yang akan dibangun antara lain sebagai berikut:

- Bagaimana masukan yang sesuai dengan perangkat lunak?
- Bagaimana keluaran yang dihasilkan oleh perangkat lunak?
- Bagaimana membuat perangkat lunak yang dapat mengambil informasi dari sekumpulan *file Java* yang terdapat pada sebuah *package* dan mengonversikan format *Javadoc* ke dalam format LaTeX secara otomatis?

1.3 Tujuan

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

- Menganalisa struktur kode program *java* yang baik
- Menganalisa struktur LaTeX yang akan sebagai keluaran perangkat lunak
- Membangun perangkat lunak yang dapat mengambil informasi dari sekumpulan *file Java* yang terdapat pada sebuah *package* dan mengonversikan format *Javadoc* ke dalam 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 memanfaatkan *library Doclet* yang sudah disediakan.
2. Perangkat lunak menghasilkan *file* LaTeX tanpa mengonversikan menjadi *file* PDF.

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 mengonversi *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 Landasan 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 perangkat lunak yang dimiliki oleh *Java* yang berguna untuk mengambil informasi dari sebuah komentar yang terdapat pada sekumpulan *source file Java* menjadi sebuah dokumentasi. Umumnya *Javadoc* menghasilkan sekumpulan *file* HTML yang mendeskripsikan sebuah kelas, *interface*, *method* dan *custom tag*. *Javadoc* dapat mengambil 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"*. *Javadoc* dapat mengambil informasi dari lebih dari 1 *file Java* atau sebuah *package*.

Javadoc secara otomatis akan menambahkan sebuah tautan yang mengarahkan ke sebuah *package*, kelas dan anggota dari kelas tersebut yang akan didokumentasikan pada saat *Javadoc* memprosesnya. Tautan tersebut berada pada beberapa posisi seperti:

1. *Declaration* (*return types*, *argument types*, *field types*).
2. *"See Also"* yang dihasilkan oleh *tag @see*.
3. *In-line text* yang dihasilkan oleh *tag @link*.
4. *Exception* 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 mengambil 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 mengimplementasi sebuah API yang disediakan oleh *Javadoc*.

2.1.2 Terminologi

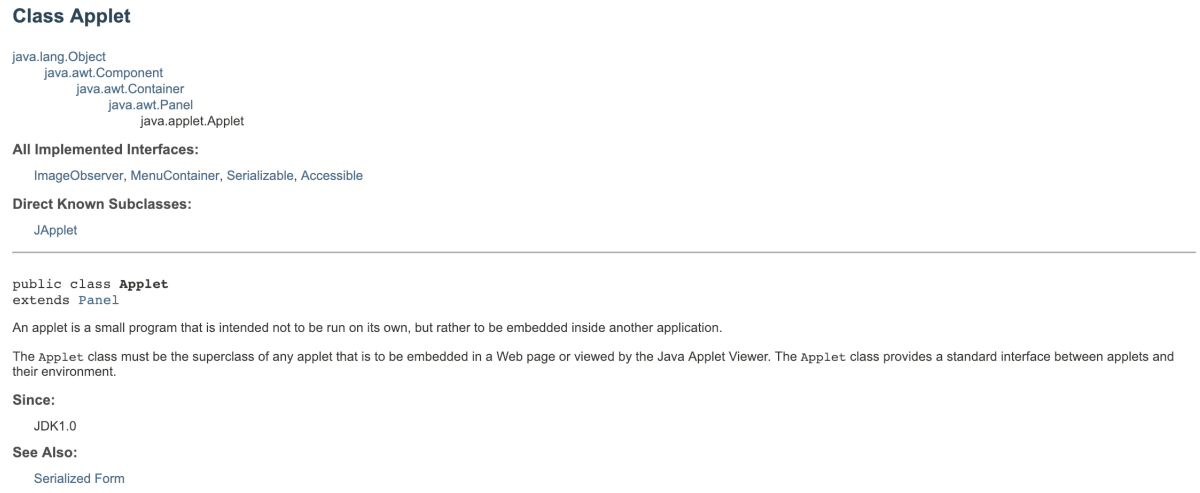
Terdapat beberapa istilah yang memiliki arti spesifik dalam konteks *Javadoc* sebagai berikut:

- *Generated Document*
  Dokumen yang dihasilkan oleh *Javadoc* adalah sebuah *file* HTML.
- *Name*
  Nama dari elemen program yang dideklarasikan atau dilakukan pemanggilan maka dapat berupa informasi lengkapnya seperti `java.lang.String` dan `java.lang.String.equals(java.lang.Object)` atau informasi pendeknya seperti `String` dan `equals(Object)`. Nama-nama tersebut dapat berupa nama *package*, kelas, *interface*, *field*, *constructor* atau *method*.
- *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 disebutkan 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*.

2.1.3 *Source Files*

Javadoc akan menghasilkan *output* yang berasal dari 4 tipe *file*, yaitu sebagai berikut:

- *Class Source Code Files*
  Setiap kelas atau *interface* memiliki dokumentasinya masing-masing.



Gambar 2.1: Hasil dokumentasi pada kelas atau *interface*

Pada gambar 2.1, *Javadoc* akan menghasilkan dokumentasinya berupa *file* HTML

- *Package Comment Files*
  Setiap *package* memiliki *file* yang berisikan komentar yang berkaitan dengan *package* tersebut. *File* tersebut terletak pada folder *package*. Contohnya *java/applet/package.html*, kemudian *Javadoc* akan menggabungkan dokumentasi tersebut menjadi sebuah ringkasan dari *package* yang ada. Terdapat 2 cara penulisan yaitu dengan sebuah *file* package.html pada listing 2.1 atau sebuah *file* package-info.java pada listing 2.2.

```
<html>
<body>
Provides the classes necessary to create an applet and the classes
an applet uses to communicate with its applet context.

@since 1.0
@see java.awt
</body>
</html>
```

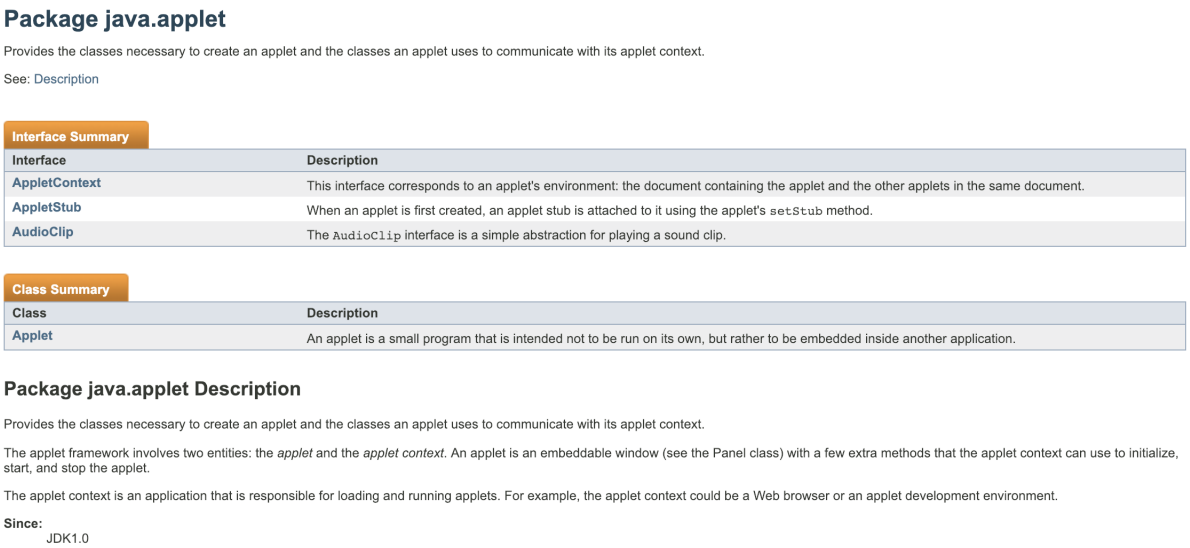
Listing 2.1: *File package.html*

```
/**
 * Provides the classes necessary to create an applet
 * and the classes an applet uses to communicate
 * with its applet context.
 *
 * @since 1.0
 * @see java.awt
 */
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* yang ada.
3. Memasukkan teks yang sudah diproses tersebut pada bagian bawah halaman dokumentasi yang dihasilkan.
4. Menyalin kalimat pertama pada *package* tersebut pada bagian atas halaman dokumentasi.



Package java.applet

Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.

See: Description

Interface Summary

Interface	Description
AppletContext	This interface corresponds to an applet's environment: the document containing the applet and the other applets in the same document.
AppletStub	When an applet is first created, an applet stub is attached to it using the applet's `setStub` method.
AudioClip	The `AudioClip` interface is a simple abstraction for playing a sound clip.

Class Summary

Class	Description
Applet	An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application.

Package java.applet Description

Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.

The applet framework involves two entities: the *applet* and the *applet context*. An applet is an embeddable window (see the Panel class) with a few extra methods that the applet context can use to initialize, start, and stop the applet.

The applet context is an application that is responsible for loading and running applets. For example, the applet context could be a Web browser or an applet development environment.

Since:
JDK1.0

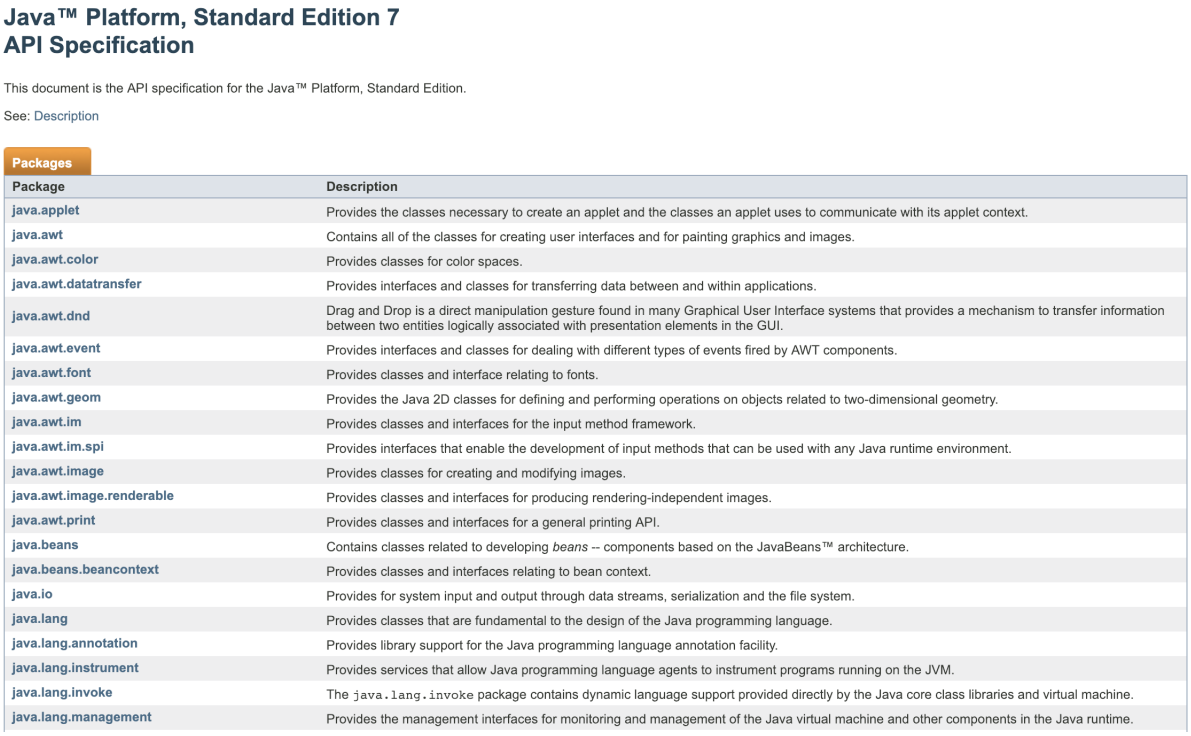
Gambar 2.2: Hasil dokumentasi pada dari *package*

Pada gambar 2.2 adalah hasil *Javadoc* mendokumentasikan setiap *package*

- *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* yang diletakkan pada bagian paling atas sebuah *source code* yang dibuat. *Javadoc* akan melakukan beberapa langkah yaitu sebagai berikut:

  1. Menyalin informasi untuk diproses.
  2. Memproses semua *tag* yang ada.
  3. Memasukan teks yang sudah diproses tersebut pada bagian bawah halaman dokumentasi yang dihasilkan.
  4. Menyalin kalimat pertama pada *package* tersebut pada bagian atas halaman dokumentasi.

  Hasil dari dokumentasi *overview* terdapat pada gambar 2.3.



Java™ Platform, Standard Edition 7
API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.invoke	The `java.lang.invoke` package contains dynamic language support provided directly by the Java core class libraries and virtual machine.
java.lang.management	Provides the management interfaces for monitoring and management of the Java virtual machine and other components in the Java runtime.

Gambar 2.3: Hasil dokumentasi pada dari *overview*

- *Miscellaneous Unprocessed Files*
  Javadoc dapat memproses berbagai macam format *file*. *File* tersebut dapat berupa sebuah *graphic files*, *Java source (.java)* dan *class (.class)* dan sebuah *file* HTML. Untuk memproses *file* tersebut, perlu membuat sebuah folder bernama *doc-files* yang diletakkan pada *package* tertentu lalu *file-file* tersebut dapat diakses dengan menggunakan lokasi direktori yang menyimpan *file* tersebut. Sebagai contoh jika terdapat sebuah gambar `myimage.jpg` yang terletak dalam folder *doc-files* maka cara mengaksesnya adalah dengan menulis `doc-files/myimage.jpg`.

2.1.4 *Generated Files*

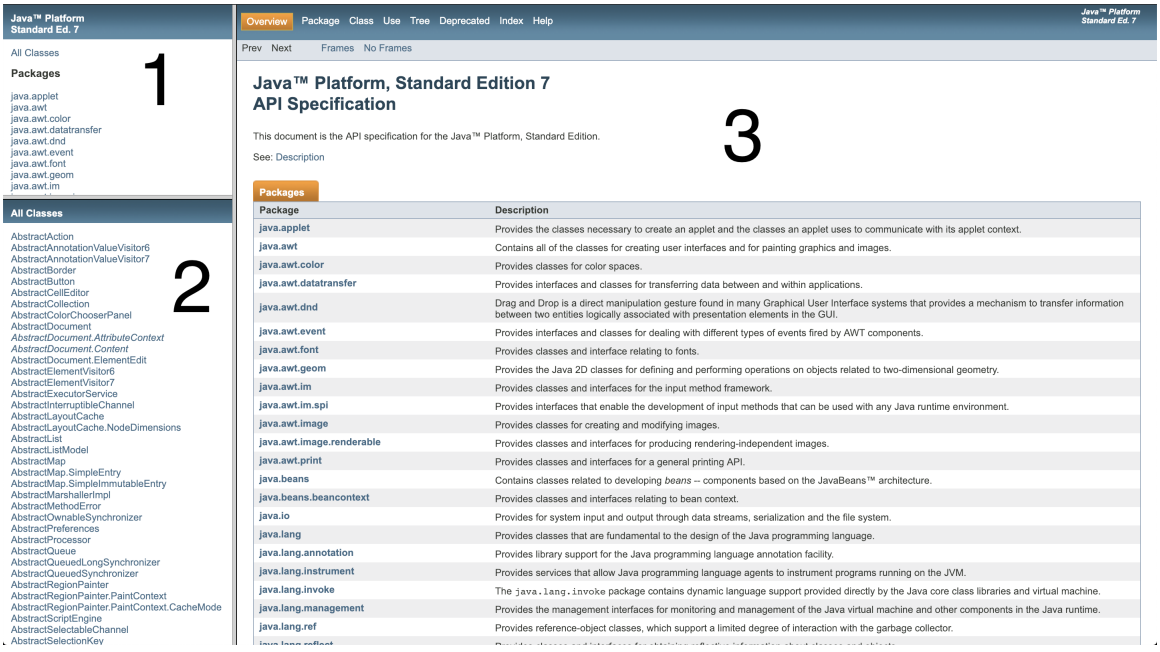
Javadoc memiliki sebuah *doclet* bawaan yang disebut dengan standar *doclet*. Standar *doclet* akan menghasilkan dokumentasi dengan format HTML. Terdapat 3 grup yaitu *Basic Content Pages*, *Cross-Reference Pages* dan *Support Files*. Setiap grup tersebut memiliki kriterianya sendiri, ketiga grup tersebut 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 untuk keseluruhan *package* (*overview-tree.html*).
  - Sebuah halaman hirarki untuk setiap *package* (*package-tree.html*).

  - Sebuah halaman *"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 kelas atau *interface* 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*.
  - Seberapa *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*.



Gambar 2.4: Frame pada hasil dokumentasi

Pada gambar 2.4 terdapat 3 buat frame. *Frame* pada HTML dapat disebut sebagai *section*. *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 yaitu *frame* 3. Jika terdapat lebih dari 2 *package*, maka *Javadoc* akan membuat 3 *frame* dari sekumpulan *package* yaitu *frame* 1, 2 dan 3. 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:

```
apidocs                                 Top directory
  index.html                            Initial page that sets up HTML frames
* overview-summary.html                 Lists all packages with first sentences summaries
  overview-tree.html                    Lists class hierarchy for all packages
  deprecated-list.html                  Lists deprecated API for all packages
    constant-values.html                Lists values of static fields for all packages
    serialized-form.html                Lists serialized form for all packages
 * overview-frame.html                  Lists all packages, used in upper-left frame
    allclasses-frame.html               Lists all classes for all packages, used in
                                        lower-left frame
    help-doc.html                       Lists user help for how these pages are organized
    index-all.html                      Default index created without -splitindex option
    index-files                         Directory created with -splitindex option
        index-<number>.html             Index files created with -splitindex option
    package-list                        Lists package names, used only for
                                        resolving external refs
    stylesheet.css                      HTML style sheet for defining fonts, colors and
                                        positions
    java                                Package directory
        applet                          Subpackage directory
            Applet.html                 Page for Applet class
            AppletContext.html          Page for AppletContext interface
            AppletStub.html             Page for AppletStub interface
            AudioClip.html              Page for AudioClip interface
        * package-summary.html          Lists classes with first sentence summaries
                                        for this package
        * package-frame.html            Lists classes in this package, used in
                                        lower left-hand frame
        * package-tree.html             Lists class hierarchy for this package
          package-use                   Lists where this package is used
            doc-files                   Directory holding image and example files
            class-use                   Directory holding pages API is used
                Applet.html             Page for uses of Applet class
                AppletContext.html      Page for uses of AppletContext interface
                AppletStub.html         Page for uses of AppletStub interface
                AudioClip.html          Page for uses of AudioClip interface
    src-html                            Source code directory
        java                            Package directory
            applet                      Subpackage directory
                Applet.html             Page for Applet source code
                AppletContext.html      Page for AppletContext source code
                AppletStub.html         Page for AppletStub source code
                AudioClip.html          Page for AudioClip source code
```

Listing 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 mengambil informasi secara spesifik [2].

2.2.1 *Interface-interface* pada Doclet

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

- `RootDoc`
  sebuah *interface* yang menjadi parameter masukan sebuah *doclet* dari perangkat lunak yang dibuat. Dari *root* tersebut semua informasi dapat diambil. *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 mengambil 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 dasar untuk membuat dan menggunakan *doclet*:

1. Membuat sebuah kelas *Java* yang akan menjadi *doclet*. Kelas *Java* tersebut harus ditambahkan *package* `com.sun.javadoc.*` untuk menggunakan *doclet* API.

2. Kelas tersebut harus diawali dengan sebuah *method* `public static boolean start` yang memiliki parameter `RootDoc`.

3. *Compile* doclet tersebut dengan menggunakan *compiler Java* yaitu `javac` yang dapat dilakukan melalui *Terminal* pada Linux/Mac atau *Command Prompt* pada Windows.

4. Jalankan perintah *Javadoc* menggunakan *option* `-doclet <class>` untuk menghasilkan *output* yang telah disesuaikan, dimana `<class>` adalah kelas *doclet* yang sudah di-*compile* pada langkah ketiga. Jika tidak menggunakan *option* `-doclet` maka *Javadoc* akan menghasilkan *output* standar yaitu berupa *file* HTML.

Kelas-kelas yang dimiliki oleh *doclet* API terdapat pada direktori `lib\tools.jar`. *Doclet* yang sudah dibuat harus di-*compile* menggunakan *file* `tools.jar` dan menambahkan *option* `-classpath` setelah *command* `javac`.

Package `com.sun.javadoc.*` terdiri dari *interface* yang mendefinisikan *doclet* API. `tools.jar` berisikan *interface* tersebut.

```
import com.sun.javadoc.*;

public class ListClass {
  public static boolean start(RootDoc doc) {
    ClassDoc[] classes = doc.classes();
    for(int i=0, i < classes.length; i++) {
      System.out.println(classes[i]);
    }
    return true;
  }
}
```

Listing 2.4: kelas ListClass.java

Potongan *program* pada Listing 2.4 adalah sebuah perogram sederhana menggunakan *doclet* untuk menampilkan nama dari kelas yang dijadikan sebagai masukan. *Method* `public static boolean start` memiliki parameter `RootDoc doc` yang akan menampung masukan dari sekumpulan *file Java* yang akan diproses lalu sekumpulan *file* tersebut akan ditampung pada `ClassDoc[] classes` sebagai `array` kemudian *array of classes* akan dijalankan sebanyak panjang *array* tersebut.

2.3 LaTeX

LaTeX adalah sebuah bahasa *markup* untuk sistem penulisan dokumen yang dikembangkan oleh Leslie B. Lamport dan dirilis pada tahun 1985 [3]. Bahasa *markup* adalah sebuah bahasa yang menganotasikan dokumen dengan cara menambahkan sintaks tertentu agar dapat dibedakan. 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. `\textbf{text}`
   Digunakan untuk menampilkan tulisan tebal.

7. `\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.

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

2.4 *Syntax Java*

Agar dokumentasi yang dihasilkan oleh *Javadoc* menghasilkan hasil yang baik dibutuhkan penulisan sintaks pada kode program java yang baik. Berikut beberapa aturan yang dapat digunakan.

1. **Kelas**
   Untuk penamaan kelas harus ditulis dengan menggunakan aturan *PascalCase*. *PascalCase* adalah aturan umum pada penulisan sebuah kode program dengan cara setiap kata diawali dengan huruf besar atau kapital. Jika ingin membuat sebuah nama kelas `myapp` maka harus ditulis seperti `MyApp`. Deklarasi kelas harus ditulis `<access_modifier> class <classname>`. Pada *Java*, *Access Modifier* terbagi menjadi 3 yaitu sebagai berikut.

   (a) *public* yang dapat diakses secara global diseluruh perangkat lunak
   (b) *private* yang tidak dapat diakses secara global diseluruh perangkat lunak
   (c) *protected* yang hanya bisa diakses jika terdapat kelas yang menjadi turunan dari kelas tersebut

   Pada bagian deklarasi kelas harus terdapat komentar yang berfungsi sebagai deksripsi kelas tersebut. Komentar diletakan didalam tanda `/** ... */`. Berikut cara penulisan kelas yang baik dan benar.

```java
/**
 * Kelas MyApp
 */
public class MyApp {

}
```

Gambar 2.5: Contoh deklarasi kelas pada *Java*

2. **Atribut**
   Untuk penamaan atribut harus ditulis dengan menggunakan aturan *CamelCase*. *CamelCase* adalah aturan umum pada penulisan sebuah kode program dengan cara setiap kata kedua diawali dengan huruf besar atau kapital. Jika ingin membuat sebuah nama atribut `islogin` maka harus ditulis seperti `isLogin`. Deklarasi atribut harus ditulis `<access_modifier> <tipe_data> <nama_atribut>`. Terdapat beberapa tipe data yang dapat digunakan yaitu sebagai berikut.

   (a) `byte` - untuk bilangan bulat dengan panjang $2^7 - 1$.
   (b) `short` - untuk bilangan bulat dengan panjang $2^{15} - 1$
   (c) `int` - untuk bilangan bulat dengan panjang $2^{31} - 1$. Umumnya tipe data ini lebih sering digunakan
   (d) `long` - untuk bilangan bulat dengan panjang $2^{63} - 1$.
   (e) `float` - untuk bilangan desimal dengan panjang 32-bit.
   (f) `double` - untuk bilangan desimal dengan panjang 64-bit.
   (g) `String` - sebuah objek pada *java* untuk kalimat.

   Pada bagian deklarasi atribut harus terdapat komentar yang berfungsi sebagai deksripsi atribut tersebut. Komentar diletakan didalam tanda `/** ... */`. Berikut cara penulisan kelas yang baik dan benar.

```java
/**
 * Atribut a
 */
private int a;
```

Gambar 2.6: Contoh deklarasi atribut pada *Java*

3. **Method**
   Untuk penamaan *method* harus ditulis dengan menggunakan aturan *CamelCase*. *CamelCase* adalah aturan umum pada penulisan sebuah kode program dengan cara setiap kata kedua diawali dengan huruf besar atau kapital. Jika ingin membuat sebuah nama method `getusercomments` maka harus ditulis seperti `getUserComments`. Dekralasi *method* harus ditulis `<access_modifier> <tipe_data_kembalian> <atribut>(<tipe_data_parameter> <nama_parameter>)`. Tipe data kembalian dan tipe data parameter sama dengan tipe data atribut.

Pada bagian deklarasi kelas harus terdapat komentar yang berfungsi sebagai deksripsi kelas tersebut. Komentar diletakan didalam tanda `/** ... */`. Berikut cara penulisan kelas yang baik dan benar.

```
/**
 * Fungsi untuk mengganti nilai A dengan yang baru
 * dan mengembalikan nilai A tersebut
 * @param newA nilai baru untuk atribut A
 * @return nilai terbaru dari atribut A
 */
public int getAndSetA(int newA) {
  this.a = newA;
  return this.a;
}
```

Gambar 2.7: Contoh deklarasi *method* pada *Java*

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.

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

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

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

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

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

Listing 3.1: Potongan kode LaTeX

Potongan kode yang terdapat pada Listing 3.1 adalah bagian isi dari sebuah dokumen LaTeX. Bagian ini nantinya akan dipindahkan ke salah satu bab pada skripsi yang akan menjelaskan perangkat lunak secara rinci. Bagian ini akan dijelaskan sebagai berikut.

1. *List* indentasi tingkat pertama
   Pada *list* tingkat 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* indentasi tingkat kedua
   Pada *list* tingkat 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* indentasi tingkat ketiga
   Pada *list* tingkat 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 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 memiliki format sebagai berikut.

```
package javadoc;

/**
 * Kelas ini MyApp
 */
public class MyApp {

    /**
     * Atribut a
     */
    private int a;
    /**
     * Atribut b
     */
    private int b;

    /**
     * Method Pertambahan
     *
     * @param a Bilangan Pertama
     * @param b Bilangan Kedua
     *
```

```
     * @return hasil penjumlahan 2 buah bilangan
     */
    public int pertambahan(int a, int b) {
        int hasil = 0;
        hasil = a + b;
        return hasil;
    }
}
```

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 per satu *file Java* dan informasi yang terdapat pada setiap *file Java* tersebut dimasukan ke dalam *file* LaTeX.

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

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

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

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

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

Listing 3.3: Contoh hasil konversi ke LaTeX

Hasil konversi pada Listing 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 mengonversi 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. Hasil PDF yang dihasilkan oleh TeXDoclet dapat dilihat pada Lampiran D.

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*.

6. `-title <title>`
   Untuk menentukan judul dari dokumentasi yang dibuat.

7. `-output <outfile>`
   Menentukan nama *file* yang akan dihasilkan. Jika perintah ini tidak digunakan maka secara otomatis akan menghasilkan file *doc.tex*.

8. `-date <date string>`
   Menentukan tanggal dari dokumen.

9. `-author <author>`
   Menentukan penulis dari dokumen.

Parameter `-output` yang terdapat pada TeXDoclet digunakan oleh perangkat lunak yang dibuat agar perangkat lunak dapat menghasilkan *output* dengan nama yang dapat disesuaikan.

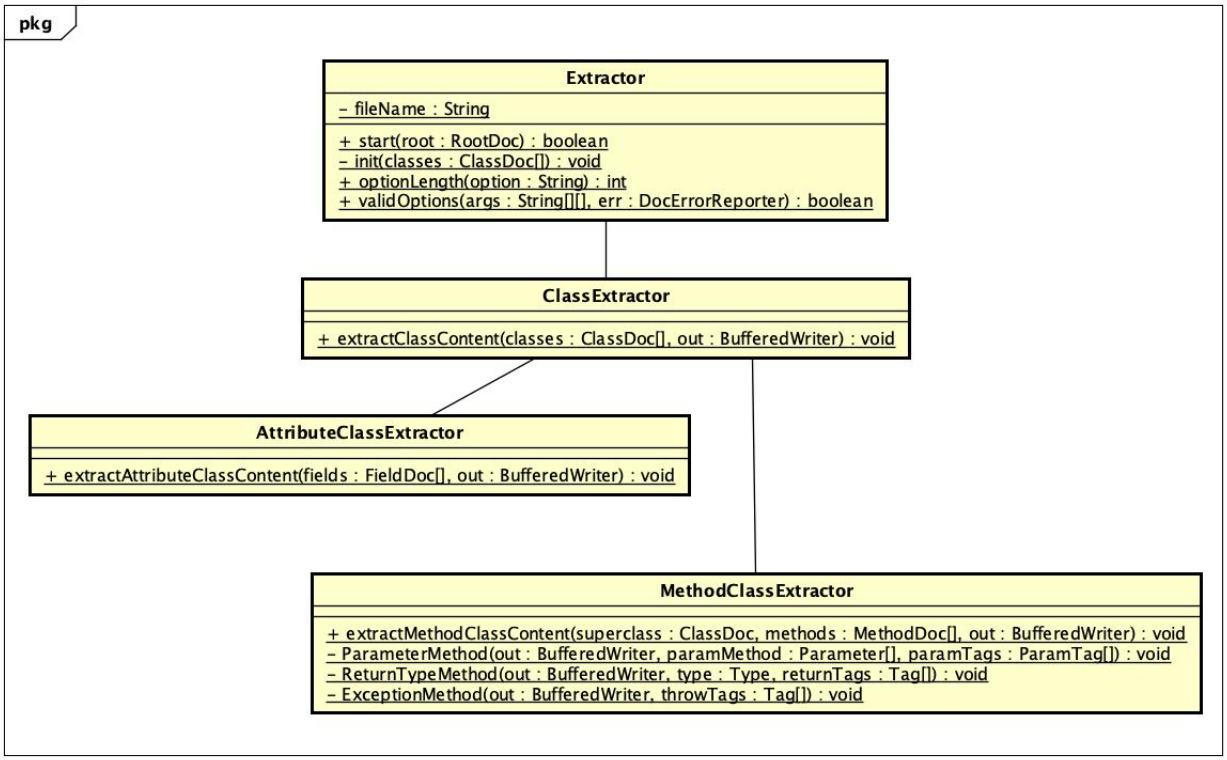
BAB 4

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 di bawah 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. `Extractor`

   Kelas ini merupakan kelas untuk menjalan *custom doclet*

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

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

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public static boolean start(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.DocErro`
  `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*

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. `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.Par 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.Ty 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. `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

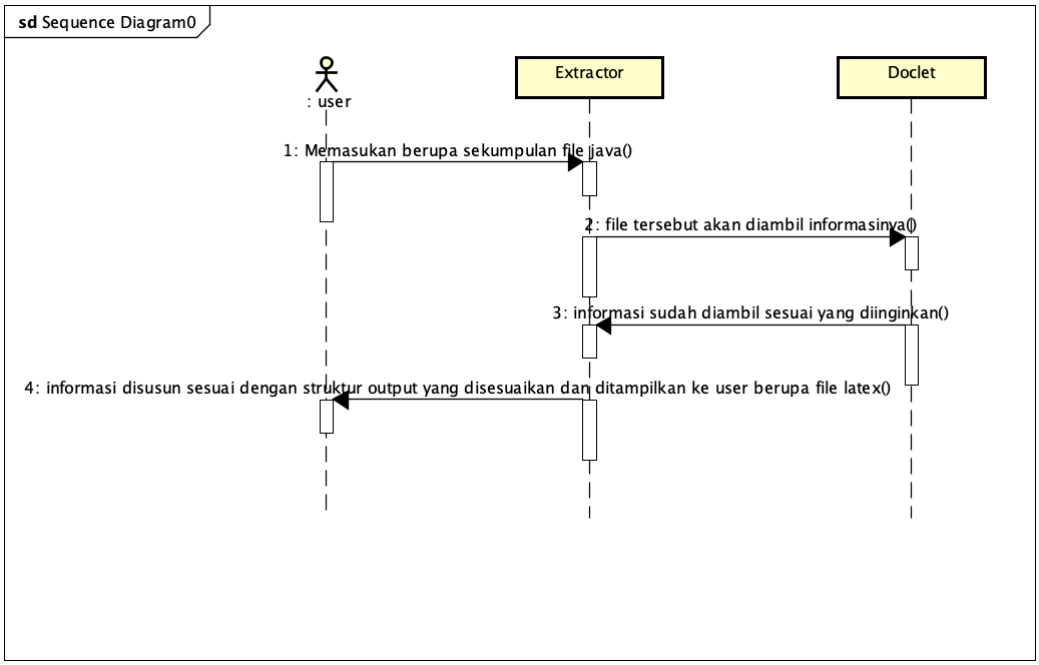
     Parameter:

     - `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*

4.2 Sequence Diagram



Gambar 4.2: Sequence Diagram

Dapat dilihat pada gambar 4.2, pada saat ingin menjalankan perangkat lunak ini user/pengguna harus memberikan masukan sekumpulan *file java*, lalu *file-file* tersebut diambil informasinya oleh *doclet*, kemudian informasi yang sudah diambil tersebut akan diletakkan sesuai tempatnya pada struktur latex yang sudah dibuat.

4.3 Rancangan Antarmuka

Rancangan antarmuka perangkat lunak adalah melalui sebuah *Terminal* pada *Linux/Mac* atau *Command Prompt* pada *Windows*. Berikut adalah antarmuka jika menggunakan *Terminal* pada *Mac*:

```
Last login: Mon Nov 26 18:49:17 on ttys002
root:~ abathz$ cd Documents/KULIAH/Skripsi/
```

Gambar 4.3: Mengarahkan ke dalam folder dari perangkat lunak

Langkah pertama adalah berpindah dari direktori awal ke direktori perangkat lunak yang dibuat. Untuk berpindah direktori diperlukan perintah `cd` atau kepanjangan dari *change directory* lalu diikuti dengan lokasi direktori yang diinginkan. Pada Gambar 4.3 direktori perangkat lunak terdapat di dalam folder Document lalu folder KULIAH lalu folder Skripsi kemudian tekan tombol *enter* lalu direktori akan langsung berpindah ke direktori yang dituju.

```
root:Skripsi abathz$ javadoc -filename file -classpath tools.jar -doclet extractor.Extractor -docletpath
GenerateJavadocToLatex.jar -sourcepath GenerateJavadocToLatex/src extractor
```

Gambar 4.4: Memasukkan perintah yang akan digunakan

Langkah kedua adalah menjalankan perangkat lunak yang dibuat. Diawali dengan perintah-perintah `javadoc` lalu dikuti 5 buah argumen. Argumen pertama yaitu `-filename` adalah *option* untuk menamai *file* sesuai dengan yang ditentukan. Sebagai contoh pada Gambar 4.4, *file* akan bernama "file", jika argumen pertama tidak dimasukkan pada *command-line* maka nama dari *file* tersebut secara otomatis menjadi "doc". Argumen kedua yaitu `-classpath` berperan sebagai penunjuk kelas-kelas yang digunakan. Argumen kedua ini bersifat *optional*, jika kode program yang akan didokumentasikan menggunakan *external library* maka argumen ini dibutuhkan. Argumen ketiga yaitu `-doclet` 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 yaitu `-docletpath` adalah *custom doclet* yang berperan untuk mengambil informasi kelas, atribut, *method* dari sekumpulan *file java*. Argumen kelima yaitu `-sourcepath` adalah lokasi sekumpulan *file java* yang akan diproses. Pada gambar 4.4, lokasi *file-file* tersebut terdapat pada folder GenerateJavadocToLatex lalu folder *src* lalu argumen keenam yaitu adalah sebuah *package* yang berisikan sekumpulan *file Java*, pada kasus ini adalah *package extractor*.

```
root:Skripsi abathz$ javadoc -filename file -classpath tools.jar -doclet extractor.Extractor -docletpath
GenerateJavadocToLatex.jar -sourcepath GenerateJavadocToLatex/src extractor
Loading source files for package extractor...
Constructing Javadoc information...
root:Skripsi abathz$
```

Gambar 4.5: Hasil tampilan jika proses konversi selesai

Perangkat lunak yang dibuat akan membaca seluruh isi *package*. Pada contoh Gambar 4.5, perangkat lunak akan melakukan pengambilan informasi terhadap *package* tersebut. Jika proses pengambilan 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 perintah menggunakan perangkat lunak dan langkah-langkah dalam pengunaan perangkat lunak. Sedangkan bagian pengujian berisi hasil pengujian fungsional dan pengujian eksperimental.

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 -filename <file-name>
        -classpath <pathlist>
        -doclet extractor.Extractor
        -docletpath GenerateJavadocToLatex.jar
        -sourcepath <pathlist>
        -subpackages <subpkglist>
        <packagenames>
```

Potongan perintah di atas memiliki 2 parameter yaitu *option* dan *packagenames*. Parameter *packagename* adalah parameter untuk *package* yang akan digunakan 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`.
- `-classpath <pathlist>` - Letak *file* `jar` yang dipergunakan. Perintah ini bersifat opsional jika kelas tersebut membutuhkan kelas yang tidak ada di dalam standar *library Java*.
- `-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. Perintah ini dapat digunakan secara opsional jika terdapat sebuah *subpackage* di dalam *package* yang menjadi masukan.
- `<packagenames>` - Nama *package* yang menjadi masukan. Lokasi *package* bergantung pada `-sourcepath` yang dituju.

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

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

5.1.1 Lingkungan Implementasi

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 Pengujian Perangkat Lunak

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

5.2.1 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
        -sourcepath OperasiMatematika/src
        operasi
```

Perintah tersebut akan mengambil informasi dari 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 diekseskusi menggunakan perintah `pdflatex` akan menghasil sebuah *file* PDF yang dapat dilihat pada Lampiran A.3.

Dapat dilihat dari hasil yang terdapat Lampiran A.3, didapatkan 5 kelas yang dikonversi menjadi 5 buah *item list enumarate* yang masing berisikan nama kelas dari 5 file tersebut. *Method-method* yang terdapat pada 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
        -classpath tools.jar
        -doclet extractor.Extractor
        -docletpath GenerateJavadocToLatex.jar
        -sourcepath GenerateJavadocToLatex/src
        extractor
```

Perintah tersebut akan mengambil informasi dari kode program yang dapat dilihat pada Lampiran B.1. Setelah perintah tersebut dieksekusi akan menghasilkan *file* LaTeX yang dapat dilihat pada Lampiran B.2. Jika *file* LaTeX tersebut diekseskusi menggunakan perintah `pdflatex` akan menghasilkan sebuah *file* PDF yang dapat dilihat pada Lampiran B.3.

Dapat dilihat dari hasil yang terdapat Lampiran B.3, didapatkan 4 kelas yang dikonversi menjadi 4 buah *item list enumarate* 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.

Pada kasus ketiga dijalankan perintah sebagai berikut.

```
javadoc -classpath tools.jar
        -doclet extractor.Extractor
        -docletpath GenerateJavadocToLatex.jar
        GenerateJavadocToLatex/src/MyApp.java
```

Perintah tersebut akan mengambil informasi dari kode program yang dapat dilihat pada Lampiran C.1. Setelah perintah tersebut dieksekusi akan menghasilkan *file* LaTeX yang dapat dilihat pada Lampiran C.2. Jika *file* LaTeX tersebut diekseskusi menggunakan perintah `pdflatex` akan menghasilkan sebuah *file* PDF yang dapat dilihat pada Lampiran C.3.

Dapat dilihat dari hasil yang terdapat pada Lampiran C.3, didapatkan kurangnya penjelasan deskripsi kelas, deskripsi atribut, deskripsi *method* dan penjelasan return value.

5.2.2 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 akan mengambil informasi dari kode program yang dapat dilihat pada Lampiran D.1. Setelah perintah tersebut dieksekusi akan menghasilkan *file* LaTeX yang dapat dilihat pada Lampiran D.2. Jika *file* LaTeX tersebut diekseskusi menggunakan perintah `pdflatex` akan menghasilkan sebuah *file* PDF yang dapat dilihat pada Lampiran D.3.

Dapat dilihat dari hasil yang terdapat Lampiran D.3, didapatkan `n` buah kelas dan `n` buah *innerclass* yang dikonversi menjadi `n` buah *item list enumarate* yang masing-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 ketidaklaziman yang ditemukan setelah mengeksekusi perintah `pdflatex`.

- Terdapat karakter yang tidak lazim yaitu tanda ¡ yang merepresentasikan tanda lebih kecil "<" dan tanda ¿ yang merepresentasikan tanda lebih besar ">". Karakter tersebut muncul pada kelas:

  1. `TahunSemester` - pada atribut kelas
  2. `Mahasiswa` - pada *method* `calculateIPKLulus()`, `calculateIPLulus()`, `calculateIPTempuh()`, `calculateIPKumulatif()`, `calculateIPKTempuh()` dan `calculateIPS()`

  Karakter yang tidak lazim tersebut muncul karena secara otomatis LaTeX menggunakan OT1 *font encoding*. Pada *Ascii Table* tanda < memiliki kode `3C` dan tanda > memiliki kode `3E` dan secara otomatis LaTeX akan memetakan tanda < dan tanda > menjadi tanda ¡ dan ¿ [4]. Untuk dapat memunculkan tanda < dan tanda > terdapat 2 cara yaitu dengan menambahkan *package* `\usepackage[T1]{fontenc}` atau menggunakan command `\textless` sebagai < dan `\textgreater` sebagai >.

- Beberapa deskripsi tidak muncul pada hasil PDF, 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 deskrpisi 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 pada hal tersebut, kesalahan yang terjadi bukan dari perangkat lunak melainkan kode program yang menjadi masukan memiliki dokumentasi yang kurang lengkap.

- Parameter setiap *method* memiliki tipe variabel yang lengkap. Setelah dilakukan pemeriksaan, tipe variabel yang bersifat *Object* akan menghasilkan tipe variabel yang *fully-qualified* seperti contohnya pada tipe variabel `String` yang akan menghasilkan `java.lang.String`.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil pembangunan perangkat lunak Konversi Javadoc ke LaTeX , didapatkanlah kesimpulan-kesimpulan sebagai berikut:

- Telah berhasil mengimplementasi *Javadoc Doclet* API untuk mengambil informasi dari sekumpulan *file Java* yang terdapat pada sebuah *package* dan mengonversikannya dari format *Javadoc* ke format LaTeX.

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 yang tidak lazim yaitu tanda ¡ yang merepresentasikan tanda lebih kecil "<" dan tanda ¿ yang merepresentasikan tanda lebih besar ">". Sebaiknya perangkat lunak dapat mengatasi *tag-tag html* jika javadoc memiliki *tag-tag* tersebut.

2. Pada saat ini, perangkat lunak menampilkan tulisan "tidak memiliki *exception*" pada bagian **Exception**. Sebaiknya mengikuti standar dari *javadoc* bahwa jika *method* yang 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.

[4] Frank Mittelbach, W. L., Robin Fairbairns (2016). *LaTeX font encodings*, **1**, 19–30.

LAMPIRAN A

PENGUJIAN TERHADAP KODE PROGRAM SEDERHANA

A.1 Kode Program

Listing A.1: OperasiMatematikaInterface.java

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package javadoc;

/**
 * Kelas Abstract OperasiMatematika. Kelas ini memiliki method {@link #calculate(int, int)}
 *
 * @author abathz
 */
public abstract class OperasiMatematikaInterface {

  /**
   * Method untuk menghasilkan perhitungan 2 buah bilangan
   * @param a Bilangan pertama
   * @param b Bilagan kedua
   * @return hasil perhitungan 2 buah bilangan {#link {@link Integer#NaN}}
   */
  public int calculate(int a, int b){return 0;}
}
```

Listing A.2: Pembagian.java

```java
package javadoc;

/**
 * Kelas ini merupakan Kelas Pembagian
 *
 * @author Adli Fariz Bonaputra
 * @see "Pembagian"
 */
public class Pembagian extends OperasiMatematikaInterface {

  /**
   * Atribut A
   */
  private int a;
  /**
   * Atribut B
   */
  private int b;

  @Override
  public int calculate(int a, int b) {
    int hasil = a / b;
    return hasil;
  }

}
```

Listing A.3: Pengurangan.java

```java
package javadoc;

/**
 * Kelas ini merupakan Kelas Pengurangan
 *
 * @author Adli Fariz Bonaputra
 * @see "Pengurangan"
 */
public class Pengurangan extends OperasiMatematikaInterface {

  /**
   * Atribut A
   */
```

```
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
```

```
\item \texttt{int b} -
Bilagan kedua
\end{itemize}
\textbf{Return Value}: hasil perhitungan 2 buah bilangan  DoubleNaN

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

\end{itemize}
\item \texttt{Pembagian}

Kelas ini merupakan Kelas Pembagian

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{int a} - Atribut A
\item \texttt{int b} - Atribut B
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public int calculate(int a, int b)}

Method untuk menghasilkan perhitungan 2 buah bilangan

\textbf{Parameter:}
\begin{itemize}
\item \texttt{int a} -
Bilangan pertama
\item \texttt{int b} -
Bilagan kedua
\end{itemize}
\textbf{Return Value}: hasil perhitungan 2 buah bilangan  DoubleNaN

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

\end{itemize}
\item \texttt{Pengurangan}

Kelas ini merupakan Kelas Pengurangan

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{int a} - Atribut A
\item \texttt{int b} - Atribut B
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public int calculate(int a, int b)}

Method untuk menghasilkan perhitungan 2 buah bilangan

\textbf{Parameter:}
\begin{itemize}
\item \texttt{int a} -
Bilangan pertama
\item \texttt{int b} -
Bilagan kedua
\end{itemize}
\textbf{Return Value}: hasil perhitungan 2 buah bilangan  DoubleNaN

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

\end{itemize}
\item \texttt{Perkalian}

Kelas ini merupakan Kelas Perkalian

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{int a} - Atribut A
\item \texttt{int b} - Atribut B
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public int calculate(int a, int b)}

Method untuk menghasilkan perhitungan 2 buah bilangan

\textbf{Parameter:}
\begin{itemize}
\item \texttt{int a} -
Bilangan pertama
\item \texttt{int b} -
Bilagan kedua
\end{itemize}
\textbf{Return Value}: hasil perhitungan 2 buah bilangan  DoubleNaN

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

\end{itemize}
\item \texttt{Pertambahan}

Kelas ini merupakan Kelas Pertambahan

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

```
\begin{itemize}
\item \texttt{public int calculate(int a, int b)}

Method untuk menghasilkan perhitungan 2 buah bilangan

\textbf{Parameter:}
\begin{itemize}
\item \texttt{int a} -
Bilangan pertama
\item \texttt{int b} -
Bilagan kedua
\end{itemize}
\textbf{Return Value}: hasil perhitungan 2 buah bilangan  DoubleNaN

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

\end{itemize}
\end{enumerate}
\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` - Bilagan 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` - Bilagan 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` - Bilagan 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` - Bilagan 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` - Bilagan kedua

     Return Value: hasil perhitungan 2 buah bilangan DoubleNaN
     Exception: Tidak memiliki *exception*

Gambar A.1: operasimatematika.pdf

LAMPIRAN B

PENGUJIAN TERHADAP KODE PROGRAM PERANGKAT LUNAK

B.1 Kode Program

Listing B.1: Extractor.java

```
1  package extractor;
2
3  import com.sun.javadoc.*;
4  import java.io.*;
5
6  /**
7   * Kelas ini merupakan kelas untuk menjalan \textit{custom doclet}
8   *
9   * @author Adli Fariz Bonaputra
10  */
11 public class Extractor {
12
13   /**
14    * atribut untuk nama \textit{file}
15    */
16   private static String fileName;
17
18   /**
19    * \textit{Method} ini berperan sebagai \textit{method} untuk menjalankan
20    * \textit{custom doclet}
21    *
22    * @param root berperan sebagai mengambil seluruh informasi spesifik dari
23    *             \textit{option} yang terdapat pada \textit{command-line} sebuah
24    *             \textit{terminal}. Selain itu berperan juga untuk mengambil informasi dari
25    *             sekumpulan \textit{file java} yang akan di proses.
26    * @return kondisi true
27    */
28   public static boolean start(RootDoc root) {
29     init(root.classes());
30     return true;
31   }
32
33   /**
34    * \textit{Method} ini berperan untuk menulis kedalam sebuah \textit{file}
35    * saat \textit{javadoc} berjalan.
36    *
37    * @param classes sebuah array yang berisikan sekumpulan \textit{file java}
38    *                yang akan di proses.
39    */
40   private static void init(ClassDoc[] classes) {
41     FileWriter file;
42     new File("output").mkdirs();
43     try {
44       if (fileName == null) {
45         file = new FileWriter("output/doc.tex");
46       } else {
47         file = new FileWriter("output/" + fileName + ".tex");
48       }
49       BufferedWriter out = new BufferedWriter(file);
50       out.write("\\documentclass{article}\n");
51       out.write("\\begin{document}\n");
52       out.write("\\begin{enumerate}\n");
53
54       ClassExtractor.extractClassContent(classes, out);
55
56       out.write("\\end{enumerate}\n");
57       out.write("\\end{document}\n");
58       out.close();
59     } catch (IOException e) { }
60   }
61
62   /**
63    * Method untuk menghitung banyak option yang digunakan pada
64    * \textit{command-line}
65    *
66    * @param option sebuah option
67    * @return panjang setiap option
68    */
```

```
69    public static int optionLength(String option) {
70      if (option.equals("-filename")) {
71        return 2;
72      }
73      return Doclet.optionLength(option);
74
75    }
76
77    /**
78     * Pengecekan option valid
79     *
80     * @param args String array 2 dimensi dari option
81     * @param err  sebuah error jika tidak terdapat option tersebut.
82     * @return bernilai true jika option tersebut dikenali, false jika option
83     * tersebut tidak dikenali
84     */
85    public static boolean validOptions(String[][] args, DocErrorReporter err) {
86      for (int i = 0; i < args.length; ++i) {
87        if (args[i][0].equals("-filename")) {
88          fileName = args[i][1];
89        }
90      }
91      return Doclet.validOptions(args, err);
92    }
93  }
```

Listing B.2: ClassExtractor.java

```
1   package extractor;
2
3   import com.sun.javadoc.*;
4   import java.io.*;
5
6   /**
7    * Kelas ini merupakan kelas untuk mengambil informasi dari sebuah kelas
8    *
9    * @author Adli Fariz Bonaputra
10   */
11  public class ClassExtractor {
12
13    /**
14     * \textit{Method} ini akan menampilkan nama kelas berserta penjelasan dari
15     * sebuah kelas
16     *
17     * @param classes sebuah array berisikan sejumlah kelas
18     * @param out     turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
19     *                file text
20     */
21    public static void extractClassContent(ClassDoc[] classes, BufferedWriter out) {
22      try {
23        for (ClassDoc classDoc : classes) {
24          Type[] interfaces = classDoc.interfaceTypes();
25          if (interfaces.length > 0) {
26            if (!classDoc.isInterface()) {
27              for (int i = 0; i < interfaces.length; i++) {
28                out.write("\\item␣\\texttt{" + classDoc.name() + "␣implements␣" + interfaces[i].typeName() + "}\n\n");
29              }
30            } else {
31              out.write("\\item␣\\texttt{" + classDoc.name() + "}\n\n");
32            }
33          } else {
34            out.write("\\item␣\\texttt{" + classDoc.name() + "}\n\n");
35          }
36
37          Tag[] inlineTags = classDoc.inlineTags();
38          for (int i = 0; i < inlineTags.length; i++) {
39            if (i == 1) {
40              out.write("\\texttt{" + inlineTags[i].text().replace("#", "") + "}");
41            } else {
42              out.write(inlineTags[i].text().replace("#", "").replace("_", "\\_").replace("&amp;", "\\&"));
43            }
44          }
45          out.write("\n\n");
46
47          FieldDoc[] fields = classDoc.fields(false);
48          AttributeClassExtractor.extractAttributeClassContent(fields, out);
49
50          MethodDoc[] methods = classDoc.methods(false);
51          MethodClassExtractor.extractMethodClassContent(classDoc, methods, out);
52        }
53      } catch (IOException e) { }
54    }
55  }
```

Listing B.3: AttributeClassExtractor.java

```
1   package extractor;
2
3   import com.sun.javadoc.*;
4   import java.io.*;
5
6   /**
7    * Kelas ini merupakan kelas untuk mengambil informasi sebuah atribut yang
8    * terdapat pada kelas
9    *
10   * @author Adli Fariz Bonaputra
```

```
 */
public class AttributeClassExtractor {

  /**
   * \textit{Method} ini akan menampilkan atribut-atribut yang dimiliki oleh
   * sebuah kelas
   *
   * @param fields sebuah array berisikan sejumlah atribut dari kelas
   * @param out turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
   * file text
   */
  public static void extractAttributeClassContent(FieldDoc[] fields, BufferedWriter out) {
    try {
      if (fields.length == 0) {
        out.write("Kelas␣ini␣tidak␣memiliki␣atribut.␣");
      } else {
        out.write("Atribut␣yang␣dimiliki␣kelas␣ini␣adalah␣sebagai␣berikut.\n");
        out.write("\\begin{itemize}\n");
        for (int j = 0; j < fields.length; j++) {
          Type type = fields[j].type();
          out.write("\\item␣\\texttt{" + type.typeName() + "␣" + fields[j].name().replace("_", "\\_") + "}␣-␣" + fields[j].
              commentText() + "\n");
        }
        out.write("\\end{itemize}\n");
      }
    } catch (IOException e) { }
  }
}
```

Listing B.4: MethodClassExtractor.java

```
package extractor;

import com.sun.javadoc.*;
import java.io.*;

/**
 * Kelas ini merupakan kelas untuk mengambil informasi sebuah \textit{method}
 * terdapat pada kelas
 *
 * @author Adli Fariz Bonaputra
 */
public class MethodClassExtractor {

  /**
   * \textit{Method} ini akan menampilkan \textit{method-method} yang dimiliki
   * oleh sebuah kelas
   *
   * @param superclass sebuah objek ClassDoc
   * @param methods    sebuah array berisikan sejumlah \textit{method} dari kelas
   * @param out        turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
   *                   file text
   */
  public static void extractMethodClassContent(ClassDoc superclass, MethodDoc[] methods, BufferedWriter out) {
    try {
      if (methods.length == 0) {
        out.write("Kelas␣ini␣tidak␣memiliki␣method.␣");
      } else {
        out.write("\\textit{Method-method}␣yang␣dimiliki␣kelas␣ini␣adalah␣sebagai␣berikut.\n");
        out.write("\\begin{itemize}\n");
        for (MethodDoc method : methods) {
          if (method.overriddenMethod() != null) {
            MethodDoc overrideMethod = method.overriddenMethod();
            Type type = overrideMethod.returnType();
            Parameter[] paramOverrideMethod = overrideMethod.parameters();
            out.write("\\item␣\\texttt{" + overrideMethod.modifiers() + "␣" + type.typeName() + "␣" + overrideMethod.name() + "("
                ;
            for (int k = 0; k < paramOverrideMethod.length; k++) {
              Type typeParam = paramOverrideMethod[k].type();
              out.write(typeParam.toString() + "␣" + paramOverrideMethod[k].name());
              if (k < paramOverrideMethod.length - 1) {
                out.write(",␣");
              }
            }
            out.write(")}\n\n");

            Tag[] inlineTags = overrideMethod.inlineTags();
            for (int i = 0; i < inlineTags.length; i++) {
              if (i == 1) {
                out.write("\\texttt{" + inlineTags[i].text().replace("#", "") + "}");
              } else {
                out.write(inlineTags[i].text().replace("#", ""));
              }
            }

            out.write("\n\n");

            ParamTag[] paramTags = overrideMethod.paramTags();
            ParameterMethod(out, paramOverrideMethod, paramTags);

            Tag[] returnTags = overrideMethod.tags("return");
            Tag[] throwTags = overrideMethod.tags("throws");
            AnnotationDesc[] override = overrideMethod.annotations();
            ReturnTypeMethod(out, type, returnTags);
            ExceptionMethod(out, throwTags);
            if (override.length != 0) {
              out.write("\\textbf{Override}:␣\\texttt{" + overrideMethod.name() + "}␣dari␣kelas␣\\texttt{" + superclass.superclass
                  ().name() + "}");
```

```
66              out.write("\n\n");
67            }
68          } else {
69            Type type = method.returnType();
70            Parameter[] paramMethod = method.parameters();
71            out.write("\\item␣\\texttt{" + method.modifiers() + "␣" + type.typeName() + "␣" + method.name() + "(");
72            for (int k = 0; k < paramMethod.length; k++) {
73              Type typeParam = paramMethod[k].type();
74              out.write(typeParam.toString() + "␣" + paramMethod[k].name());
75              if (k < paramMethod.length - 1) {
76                out.write(",␣");
77              }
78            }
79            out.write(")}\n\n");
80
81            Tag[] inlineTags = method.inlineTags();
82            for (int i = 0; i < inlineTags.length; i++) {
83              if (i == 1) {
84                out.write("\\texttt{" + inlineTags[i].text().replace("#", "") + "}");
85              } else {
86                out.write(inlineTags[i].text().replace("#", ""));
87              }
88            }
89
90            out.write("\n\n");
91
92            ParamTag[] paramTags = method.paramTags();
93            ParameterMethod(out, paramMethod, paramTags);
94
95            Tag[] returnTags = method.tags("return");
96            Tag[] throwTags = method.tags("throws");
97            AnnotationDesc[] override = method.annotations();
98            ReturnTypeMethod(out, type, returnTags);
99            ExceptionMethod(out, throwTags);
100           if (override.length != 0) {
101             out.write("\\textbf{Override}:␣\\texttt{" + method.name() + "}␣dari␣kelas␣\\texttt{" + superclass.superclass().name
                    () + "}");
102             out.write("\n\n");
103           }
104         }
105       }
106       out.write("\\end{itemize}\n");
107     }
108   } catch (IOException e) { }
109 }
110
111 /**
112  * \textit{Method} ini akan menampilkan parameter \textit{method-method} yang dimiliki
113  * oleh sebuah kelas
114  *
115  * @param out                   turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
116  * @param paramMethod           sebuah array berisikan sejumlah \textit{method} dari kelas
117  * @param paramTags             sebuah array berisikan sejumlah parameter \textit{method} dari kelas
118  */
119 private static void ParameterMethod(BufferedWriter out, Parameter[] paramMethod, ParamTag[] paramTags){
120   try {
121     if (paramTags.length == 0) {
122       out.write("\\textbf{Parameter:}\n");
123       out.write("\\begin{itemize}\n");
124       if (paramMethod.length != 0) {
125         for (int i = 0; i < paramMethod.length; i++) {
126           out.write("\\item␣\\texttt{" + paramMethod[i].toString() + "}␣-␣\n");
127         }
128       } else {
129         out.write("\\item␣Tidak␣memiliki␣parameter␣\\textit{method}\n");
130       }
131       out.write("\\end{itemize}\n");
132     } else {
133       out.write("\\textbf{Parameter:}\n");
134       out.write("\\begin{itemize}\n");
135       for (int k = 0; k < paramTags.length; k++) {
136         Type type = paramMethod[k].type();
137         String[] typeParam = type.toString().split("\\.");
138         out.write("\\item␣\\texttt{" + typeParam[typeParam.length - 1] + "␣" + paramTags[k].parameterName() + "}␣-␣\n");
139         Tag[] inlineTagsInParameter = paramTags[k].inlineTags();
140         for (int i = 0; i < inlineTagsInParameter.length; i++) {
141           if (i == 1) {
142             out.write("\\texttt{" + inlineTagsInParameter[i].text().replace("#", "") + "}");
143           } else {
144             out.write(inlineTagsInParameter[i].text());
145           }
146         }
147         out.write("\n");
148       }
149       out.write("\\end{itemize}\n");
150     }
151   } catch (IOException e) { }
152 }
153
154 /**
155  * \textit{Method} ini akan menampilkan \textit{return type} dari \textit{method-method} yang dimiliki
156  * oleh sebuah kelas
157  *
158  * @param out          turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
159  * @param type         sebuah objek Type
160  * @param returnTags   sebuah array berisikan sejumlah \textit{return type} dari \textit{method} dari kelas
161  */
162 private static void ReturnTypeMethod(BufferedWriter out, Type type, Tag[] returnTags){
163   try {
```

```
164       if (type.typeName().equals("void") || returnTags.length == 0) {
165         out.write("\\textbf{Return Value}: Tidak memiliki \\textit{return value}\n\n");
166       } else {
167         out.write("\\textbf{Return Value}: ");
168         Tag[] inlineTagsInReturnValue = returnTags[0].inlineTags();
169         for (int i = 0; i < inlineTagsInReturnValue.length; i++) {
170           if (i == 1) {
171             out.write(inlineTagsInReturnValue[i].text().replace("#", ""));
172           } else {
173             out.write(inlineTagsInReturnValue[i].text().replace("{#link", "").replace("}", "").replace("#", ""));
174           }
175         }
176         out.write("\n\n");
177       }
178     } catch (IOException e) { }
179   }
180
181   /**
182    * \textit{Method} ini akan menampilkan \textit{return type} dari \textit{method-method} yang dimiliki
183    * oleh sebuah kelas
184    *
185    * @param out          turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
186    * @param throwTags    sebuah array berisikan sejumlah \textit{exception} dari \textit{method} dari kelas
187    */
188   private static void ExceptionMethod(BufferedWriter out, Tag[] throwTags) throws IOException {
189     try {
190       if (throwTags.length == 0) {
191         out.write("\\textbf{Exception}: Tidak memiliki \\textit{exception}");
192         out.write("\n\n");
193       } else {
194         out.write("\\textbf{Exception}: " + throwTags[0].text());
195         out.write("\n\n");
196       }
197     } catch (IOException e) { }
198   }
199 }
```

B.2 Hasil Latex

Listing B.5: javadoctolatex.tex

```
1  \documentclass{article}
2  \begin{document}
3  \begin{enumerate}
4  \item \texttt{Extractor}
5
6  Kelas ini merupakan kelas untuk menjalan \textit{custom doclet}
7
8  Atribut yang dimiliki kelas ini adalah sebagai berikut.
9  \begin{itemize}
10 \item \texttt{String fileName} - atribut untuk nama \textit{file}
11 \end{itemize}
12 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
13 \begin{itemize}
14 \item \texttt{public static boolean start(com.sun.javadoc.RootDoc root)}
15
16 \textit{Method} ini berperan sebagai \textit{method} untuk menjalankan
17  \textit{custom doclet}
18
19 \textbf{Parameter:}
20 \begin{itemize}
21 \item \texttt{RootDoc root} -
22 berperan sebagai mengambil seluruh informasi spesifik dari
23              \textit{option} yang terdapat pada \textit{command-line} sebuah
24              \textit{terminal}. Selain itu berperan juga untuk mengambil informasi dari
25              sekumpulan \textit{file java} yang akan di proses.
26 \end{itemize}
27 \textbf{Return Value}: kondisi true
28
29 \textbf{Exception}: Tidak memiliki \textit{exception}
30
31 \item \texttt{private static void init(com.sun.javadoc.ClassDoc[] classes)}
32
33 \textit{Method} ini berperan untuk menulis kedalam sebuah \textit{file}
34  saat \textit{javadoc} berjalan.
35
36 \textbf{Parameter:}
37 \begin{itemize}
38 \item \texttt{ClassDoc[] classes} -
39 sebuah array yang berisikan sekumpulan \textit{file java}
40                yang akan di proses.
41 \end{itemize}
42 \textbf{Return Value}: Tidak memiliki \textit{return value}
43
44 \textbf{Exception}: Tidak memiliki \textit{exception}
45
46 \item \texttt{public static int optionLength(java.lang.String option)}
47
48 Method untuk menghitung banyak option yang digunakan pada
49  \textit{command-line}
50
51 \textbf{Parameter:}
52 \begin{itemize}
53 \item \texttt{String option} -
54 sebuah option
```

```
\end{itemize}
\textbf{Return Value}: panjang setiap option

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

\item \texttt{public static boolean validOptions(java.lang.String[][] args, com.sun.javadoc.DocErrorReporter err)}

Pengecekan option valid

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

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

\end{itemize}
\item \texttt{ClassExtractor}

Kelas ini merupakan kelas untuk mengambil informasi dari sebuah kelas

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public static void extractClassContent(com.sun.javadoc.ClassDoc[] classes, java.io.BufferedWriter out)}

\textit{Method} ini akan menampilkan nama kelas berserta penjelasan dari
 sebuah kelas

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

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

\end{itemize}
\item \texttt{MethodClassExtractor}

Kelas ini merupakan kelas untuk mengambil informasi sebuah \textit{method}
 terdapat pada kelas

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public static void extractMethodClassContent(com.sun.javadoc.ClassDoc superclass, com.sun.javadoc.MethodDoc[]
     methods, java.io.BufferedWriter out)}

\textit{Method} ini akan menampilkan \textit{method-method} yang dimiliki
 oleh sebuah kelas

\textbf{Parameter:}
\begin{itemize}
\item \texttt{ClassDoc superclass} -
sebuah objek ClassDoc
\item \texttt{MethodDoc[] methods} -
sebuah array berisikan sejumlah \textit{method} dari kelas
\item \texttt{BufferedWriter out} -
turunan dari kelas \texttt{Writer} yang digunakan untuk menulis
                    file text
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{private static void ParameterMethod(java.io.BufferedWriter out, com.sun.javadoc.Parameter[] paramMethod, com.sun.
     javadoc.ParamTag[] paramTags)}

\textit{Method} ini akan menampilkan parameter \textit{method-method} yang dimiliki
 oleh sebuah kelas

\textbf{Parameter:}
\begin{itemize}
\item \texttt{BufferedWriter out} -
turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
\item \texttt{Parameter[] paramMethod} -
sebuah array berisikan sejumlah \textit{method} dari kelas
\item \texttt{ParamTag[] paramTags} -
sebuah array berisikan sejumlah parameter \textit{method} dari kelas
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{private static void ReturnTypeMethod(java.io.BufferedWriter out, com.sun.javadoc.Type type, com.sun.javadoc.Tag[]
     returnTags)}

\textit{Method} ini akan menampilkan \textit{return type} dari \textit{method-method} yang dimiliki
 oleh sebuah kelas

\textbf{Parameter:}
```

```
\begin{itemize}
\item \texttt{BufferedWriter out} -
turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
\item \texttt{Type type} -
sebuah objek Type
\item \texttt{Tag[] returnTags} -
sebuah array berisikan sejumlah \textit{return type} dari \textit{method} dari kelas
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{private static void ExceptionMethod(java.io.BufferedWriter out, com.sun.javadoc.Tag[] throwTags)}

\textit{Method} ini akan menampilkan \textit{return type} dari \textit{method-method} yang dimiliki
 oleh sebuah kelas

\textbf{Parameter:}
\begin{itemize}
\item \texttt{BufferedWriter out} -
turunan dari kelas \texttt{Writer} yang digunakan untuk menulis file text
\item \texttt{Tag[] throwTags} -
sebuah array berisikan sejumlah \textit{exception} dari \textit{method} dari kelas
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\end{itemize}
\item \texttt{AttributeClassExtractor}

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

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public static void extractAttributeClassContent(com.sun.javadoc.FieldDoc[] fields, java.io.BufferedWriter out)}

\textit{Method} ini akan menampilkan atribut-atribut yang dimiliki oleh
 sebuah kelas

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

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

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

B.3 Hasil PDF

1. `Extractor`

   Kelas ini merupakan kelas untuk menjalan *custom doclet*

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

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

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public static boolean start(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*

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. `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.Parameter[] 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.Type 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. `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*

Gambar B.1: javadoctolatex.pdf

LAMPIRAN C

PENGUJIAN TERHADAP KODE PROGRAM SIAMODELS

C.1 Kode Program

Listing C.1: MyApp.java

```
1
2
3   public class MyApp {
4
5     private int a;
6
7     public int getAndSetA(int newA) {
8       this.a = newA;
9       return this.a;
10    }
11
12  }
```

C.2 Hasil Latex

Listing C.2: siamodels.tex

```
1  \documentclass{article}
2  \begin{document}
3  \begin{enumerate}
4  \item \texttt{MyApp}
5
6
7
8  Atribut yang dimiliki kelas ini adalah sebagai berikut.
9  \begin{itemize}
10 \item \texttt{int a} -
11 \end{itemize}
12 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
13 \begin{itemize}
14 \item \texttt{public int getAndSetA(int newA)}
15
16
17
18 \textbf{Parameter:}
19 \begin{itemize}
20 \item \texttt{int newA} -
21 \end{itemize}
22 \textbf{Return Value}: Tidak memiliki \textit{return value}
23
24 \textbf{Exception}: Tidak memiliki \textit{exception}
25
26 \end{itemize}
27 \end{enumerate}
28 \end{document}
```

C.3 Hasil PDF

1. `MyApp`

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `int a` -

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public int getAndSetA(int newA)`
     Parameter:
     - `int newA` -

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*

LAMPIRAN D

PENGUJIAN TERHADAP KODE PROGRAM SIAMODELS

D.1 Kode Program

Listing D.1: Dosen.java

```java
package id.ac.unpar.siamodels;

public class Dosen {
    private String nik;
    private String nama;

    public Dosen(String nik, String nama) throws IllegalArgumentException {
        super();
        if (nik == null && nama == null) {
            throw new IllegalArgumentException("Salah satu dari NIK atau nama harus diisi!");
        }
        this.nik = nik;
        this.nama = nama;
    }

    public String getNik() {
        return nik;
    }

    public void setNik(String nik) {
        this.nik = nik;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    /**
     * Equality check for dosen. First check NIK if both available.
     * Otherwise, check for name.
     * @return true if equal, false otherwise
     */
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Dosen other = (Dosen) obj;
        if (nik != null && other.nik != null) {
            return nik.equals(other.nik);
        }
        if (nama != null) {
            return nama.equals(other.nama);
        }
        return false;
    }


}
```

Listing D.2: InfoMataKuliah.java

```java
package id.ac.unpar.siamodels;

import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface InfoMataKuliah {
```

```
    /**
     * Jumlah bobot sks dari mata kuliah ini
     *
     * @return jumlah bobot sks
     */
    public int sks();

    /**
     * Nama mata kuliah ini
     *
     * @return nama mata kuliah
     */
    public String nama();
}
```

Listing D.3: JadwalKuliah.java

```
package id.ac.unpar.siamodels;

import java.time.DayOfWeek;
import java.time.LocalTime;

public class JadwalKuliah {
    protected MataKuliah mataKuliah;
    protected Character kelas;
    protected DayOfWeek hari;
    protected LocalTime waktuMulai;
    protected LocalTime waktuSelesai;
    protected String lokasi;
    protected Dosen pengajar;

    /**
     * Membuat jadwal kuliah baru
     * @param mataKuliah mata kuliah yang dibuat jadwalnya
     * @param kelas kelas kuliah atau null jika tidak tersedia
     * @param pengajar nama pengajar
     * @param hariString hari dalam Bahasa Indonesia (Senin, Selasa, ...)
     * @param waktuString rentang waktu kuliah (08.00-09.00 atau 08:00-09:00)
     * @param lokasi kode ruangan
     */
    public JadwalKuliah(MataKuliah mataKuliah, Character kelas, Dosen pengajar, String hariString, String waktuString,
            String lokasi) {
        this.mataKuliah = mataKuliah;
        this.kelas = kelas;
        this.waktuMulai = LocalTime.parse(waktuString.substring(0, 5).replace('.', ':'));
        this.waktuSelesai = LocalTime.parse(waktuString.substring(6, 11).replace('.', ':'));
        this.lokasi = lokasi;
        this.pengajar = pengajar;
        this.hari = indonesianToDayOfWeek(hariString);
    }

    public JadwalKuliah() {
        // void
    }

    public MataKuliah getMataKuliah() {
        return mataKuliah;
    }

    public void setMataKuliah(MataKuliah mataKuliah) {
        this.mataKuliah = mataKuliah;
    }

    public Character getKelas() {
        return kelas;
    }

    public void setKelas(Character kelas) {
        this.kelas = kelas;
    }

    public DayOfWeek getHari() {
        return hari;
    }

    public void setHari(DayOfWeek hari) {
        this.hari = hari;
    }

    public LocalTime getWaktuMulai() {
        return waktuMulai;
    }

    public void setWaktuMulai(LocalTime waktuMulai) {
        this.waktuMulai = waktuMulai;
    }

    public LocalTime getWaktuSelesai() {
        return waktuSelesai;
    }

    public void setWaktuSelesai(LocalTime waktuSelesai) {
        this.waktuSelesai = waktuSelesai;
    }

    public String getLokasi() {
        return lokasi;
    }
    }
```

```
82
83      public void setLokasi(String lokasi) {
84          this.lokasi = lokasi;
85      }
86
87      public Dosen getPengajar() {
88          return pengajar;
89      }
90
91      public void setPengajar(Dosen pengajar) {
92          this.pengajar = pengajar;
93      }
94
95      public String getWaktuString() {
96          return waktuMulai + "-" + waktuSelesai;
97      }
98
99      /**
100      * Converts Indonesian day names to {@link DayOfWeek} enumeration.
101      * @param indonesian the day name in Indonesian
102      * @return {@link DayOfWeek} object or null if not found.
103      */
104     public static DayOfWeek indonesianToDayOfWeek(String indonesian) {
105         switch (indonesian.toLowerCase()) {
106         case "senin":
107             return DayOfWeek.MONDAY;
108         case "selasa":
109             return DayOfWeek.TUESDAY;
110         case "rabu":
111             return DayOfWeek.WEDNESDAY;
112         case "kamis":
113             return DayOfWeek.THURSDAY;
114         case "jumat":
115             return DayOfWeek.FRIDAY;
116         case "sabtu":
117             return DayOfWeek.SATURDAY;
118         case "minggu":
119             return DayOfWeek.SUNDAY;
120         default:
121             return null;
122         }
123     }
124 }
```

Listing D.4: Mahasiswa.java

```
1  package id.ac.unpar.siamodels;
2
3  import java.net.URL;
4  import java.time.LocalDate;
5  import java.util.ArrayList;
6  import java.util.Comparator;
7  import java.util.HashMap;
8  import java.util.HashSet;
9  import java.util.List;
10 import java.util.Map;
11 import java.util.Set;
12 import java.util.SortedMap;
13 import java.util.TreeSet;
14
15 public class Mahasiswa {
16     protected final String npm;
17     protected String nama;
18     protected final List<Nilai> riwayatNilai;
19     protected URL photoURL;
20     protected List<JadwalKuliah> jadwalKuliahList;
21     protected SortedMap<LocalDate, Integer> nilaiTOEFL;
22
23     public Mahasiswa(String npm) throws NumberFormatException {
24         super();
25         if (!npm.matches("[0-9]{10}")) {
26             throw new NumberFormatException("NPM tidak valid: " + npm);
27         }
28         this.npm = npm;
29         this.riwayatNilai = new ArrayList<Nilai>();
30     }
31
32     public String getNama() {
33         return nama;
34     }
35
36     public void setNama(String nama) {
37         this.nama = nama;
38     }
39
40     public String getNpm() {
41         return npm;
42     }
43
44     public URL getPhotoURL() {
45         return photoURL;
46     }
47
48     public void setPhotoURL(URL photoURL) {
49         this.photoURL = photoURL;
50     }
51
52     public List<JadwalKuliah> getJadwalKuliahList() {
```

```
53          return jadwalKuliahList;
54      }
55
56      public void setJadwalKuliahList(List<JadwalKuliah> jadwalKuliahList) {
57          this.jadwalKuliahList = jadwalKuliahList;
58      }
59
60      public String getEmailAddress() {
61          return npm.substring(4, 6) + npm.substring(2, 4) + npm.substring(7, 10) + "@student.unpar.ac.id";
62      }
63
64      public List<Nilai> getRiwayatNilai() {
65          return riwayatNilai;
66      }
67
68      public SortedMap<LocalDate, Integer> getNilaiTOEFL(){
69          return nilaiTOEFL;
70      }
71
72      public void setNilaiTOEFL(SortedMap<LocalDate, Integer> nilaiTOEFL){
73          this.nilaiTOEFL = nilaiTOEFL;
74      }
75
76      /**
77       * Menghitung IPK mahasiswa sampai saat ini, dengan aturan:
78       * <ul>
79       *   <li>Kuliah yang tidak lulus tidak dihitung
80       *   <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
81       * </ul>
82       * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
83       * @return IPK Lulus, atau {#link {@link Double#NaN}} jika belum mengambil satu kuliah pun.
84       * @deprecated Gunakan {@link #calculateIPLulus()}, nama lebih konsisten dengan DPS.
85       */
86      public double calculateIPKLulus() throws ArrayIndexOutOfBoundsException {
87          return calculateIPTempuh(true);
88      }
89
90      /**
91       * Menghitung IP mahasiswa sampai saat ini, dengan aturan:
92       * <ul>
93       *   <li>Kuliah yang tidak lulus tidak dihitung
94       *   <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
95       * </ul>
96       * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
97       * @return IPK Lulus, atau {#link {@link Double#NaN}} jika belum mengambil satu kuliah pun.
98       */
99      public double calculateIPLulus() throws ArrayIndexOutOfBoundsException {
100         return calculateIPTempuh(true);
101     }
102
103     /**
104      * Menghitung IP mahasiswa sampai saat ini, dengan aturan:
105      * <ul>
106      *   <li>Perhitungan kuliah yang tidak lulus ditentukan parameter
107      *   <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
108      * </ul>
109      * @param lulusSaja set true jika ingin membuang mata kuliah tidak lulus, false jika ingin semua (sama dengan "IP N. Terbaik"
             di DPS)
110      * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
111      * @return IPK Lulus, atau {#link {@link Double#NaN}} jika belum mengambil satu kuliah pun.
112      */
113     public double calculateIPTempuh(boolean lulusSaja) throws ArrayIndexOutOfBoundsException {
114         List<Nilai> riwayatNilai = getRiwayatNilai();
115         if (riwayatNilai.size() == 0) {
116             return Double.NaN;
117         }
118         Map<String, Double> nilaiTerbaik = new HashMap<String, Double>();
119         int totalSKS = 0;
120         // Cari nilai lulus terbaik setiap kuliah
121         for (Nilai nilai: riwayatNilai) {
122             if (nilai.getNilaiAkhir() == null) {
123                 continue;
124             }
125             if (lulusSaja && nilai.getNilaiAkhir().equals('E')) {
126                 continue;
127             }
128             String kodeMK = nilai.getMataKuliah().getKode();
129             Double angkaAkhir = nilai.getAngkaAkhir();
130             int sks = nilai.getMataKuliah().getSks();
131             if (!nilaiTerbaik.containsKey(kodeMK)) {
132                 totalSKS += sks;
133                 nilaiTerbaik.put(kodeMK, sks * angkaAkhir);
134             } else if (sks * angkaAkhir > nilaiTerbaik.get(kodeMK)) {
135                 nilaiTerbaik.put(kodeMK, sks * angkaAkhir);
136             }
137         }
138         // Hitung IPK dari nilai-nilai terbaik
139         double totalNilai = 0.0;
140         for (Double nilaiAkhir: nilaiTerbaik.values()) {
141             totalNilai += nilaiAkhir;
142         }
143         return totalNilai / totalSKS;
144     }
145
146     /**
147      * Menghitung IP Kumulatif mahasiswa sampai saat ini, dengan aturan:
148      * <ul>
149      *   <li>Jika pengambilan beberapa kali, diambil semua.
150      * </ul>
```

```java
     * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
     * @return IPK Lulus, atau {#link {@link Double#NaN}} jika belum mengambil satu kuliah pun.
     */
    public double calculateIPKumulatif() throws ArrayIndexOutOfBoundsException {
        List<Nilai> riwayatNilai = getRiwayatNilai();
        if (riwayatNilai.size() == 0) {
            return Double.NaN;
        }
        double totalNilai = 0.0;
        int totalSKS = 0;
        // Cari nilai setiap kuliah
        for (Nilai nilai: riwayatNilai) {
            if (nilai.getNilaiAkhir() == null) {
                continue;
            }
            Double angkaAkhir = nilai.getAngkaAkhir();
            int sks = nilai.getMataKuliah().getSks();
            totalSKS += sks;
            totalNilai += sks * angkaAkhir;
        }
        return totalNilai / totalSKS;
    }

    /**
     * Menghitung IPK mahasiswa sampai saat ini, dengan aturan:
     * <ul>
     *   <li>Perhitungan kuliah yang tidak lulus ditentukan parameter
     *   <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
     * </ul>
     * @param lulusSaja set true jika ingin membuang mata kuliah tidak lulus
     * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
     * @return IPK Lulus, atau {#link {@link Double#NaN}} jika belum mengambil satu kuliah pun.
     * @deprecated Gunakan {@link #calculateIPKTempuh(boolean)}, nama lebih konsisten dengan DPS.
     */
    public double calculateIPKTempuh(boolean lulusSaja) throws ArrayIndexOutOfBoundsException {
        return calculateIPTempuh(lulusSaja);
    }


    /**
     * Menghitung IPS semester terakhir sampai saat ini, dengan aturan:
     * <ul>
     *   <li>Kuliah yang tidak lulus <em>dihitung</em>.
     * </ul>
     * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
     * @return nilai IPS sampai saat ini
     * @throws ArrayIndexOutOfBoundsException jika belum ada nilai satupun
     */
    public double calculateIPS() throws ArrayIndexOutOfBoundsException {
        List<Nilai> riwayatNilai = getRiwayatNilai();
        if (riwayatNilai.size() == 0) {
            throw new ArrayIndexOutOfBoundsException("Minimal harus ada satu nilai untuk menghitung IPS");
        }
        int lastIndex = riwayatNilai.size() - 1;
        TahunSemester tahunSemester = riwayatNilai.get(lastIndex).getTahunSemester();
        double totalNilai = 0;
        double totalSKS = 0;
        for (int i = lastIndex; i >= 0; i--) {
            Nilai nilai = riwayatNilai.get(i);
            if (nilai.tahunSemester.equals(tahunSemester)) {
                if (nilai.getAngkaAkhir() != null) {
                    totalNilai += nilai.getMataKuliah().getSks() * nilai.getAngkaAkhir();
                    totalSKS += nilai.getMataKuliah().getSks();
                }
            } else {
                break;
            }
        }
        return totalNilai / totalSKS;
    }

    /**
     * Menghitung jumlah SKS lulus mahasiswa saat ini.
     * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
     * @return SKS Lulus
     */
    public int calculateSKSLulus() throws ArrayIndexOutOfBoundsException {
        return calculateSKSTempuh(true);
    }

    /**
     * Menghitung jumlah SKS tempuh mahasiswa saat ini.
     * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
     * @param lulusSaja set true jika ingin membuang SKS tidak lulus
     * @return SKS tempuh
     */
    public int calculateSKSTempuh(boolean lulusSaja) throws ArrayIndexOutOfBoundsException {
        List<Nilai> riwayatNilai = getRiwayatNilai();
        Set<String> sksTerhitung = new HashSet<String>();
        int totalSKS = 0;
        // Tambahkan SKS setiap kuliah
        for (Nilai nilai: riwayatNilai) {
            if (nilai.getNilaiAkhir() == null) {
                continue;
            }
            if (lulusSaja && nilai.getNilaiAkhir().equals('E')) {
                continue;
            }
            String kodeMK = nilai.getMataKuliah().getKode();
```

```
250             if (!sksTerhitung.contains(kodeMK)) {
251                 totalSKS += nilai.getMataKuliah().getSks();
252                 sksTerhitung.add(kodeMK);
253             }
254         }
255         return totalSKS;
256     }
257 
258     /**
259      * Mendapatkan seluruh tahun semester di mana mahasiswa ini tercatat
260      * sebagai mahasiswa aktif, dengan strategi memeriksa riwayat nilainya.
261      * Jika ada satu nilai saja pada sebuah tahun semester, maka dianggap
262      * aktif pada semester tersebut.
263      * @return kumpulan tahun semester di mana mahasiswa ini aktif
264      */
265     public Set<TahunSemester> calculateTahunSemesterAktif() {
266         Set<TahunSemester> tahunSemesterAktif = new TreeSet<TahunSemester>();
267         List<Nilai> riwayatNilai = getRiwayatNilai();
268         for (Nilai nilai: riwayatNilai) {
269             tahunSemesterAktif.add(nilai.getTahunSemester());
270         }
271         return tahunSemesterAktif;
272     }
273 
274     /**
275      * Memeriksa apakah mahasiswa ini sudah lulus mata kuliah tertentu. Kompleksitas O(n).
276      * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
277      * Note: jika yang dimiliki adalah {@link MataKuliah}, gunakanlah {@link MataKuliah#getKode()}.
278      * @param kodeMataKuliah kode mata kuliah yang ingin diperiksa kelulusannya.
279      * @return true jika sudah pernah mengambil dan lulus, false jika belum
280      */
281     public boolean hasLulusKuliah(String kodeMataKuliah) {
282         for (Nilai nilai: riwayatNilai) {
283             if (nilai.getMataKuliah().getKode().equals(kodeMataKuliah)) {
284                 Character na = nilai.getNilaiAkhir();
285                 if (na != null && na >= 'A' && na <= 'D') {
286                     return true;
287                 }
288             }
289         }
290         return false;
291     }
292 
293     /**
294      * Memeriksa apakah mahasiswa ini sudah pernah menempuh mata kuliah tertentu. Kompleksitas O(n).
295      * Sebelum memanggil method ini, {@link #getRiwayatNilai()} harus sudah ada isinya!
296      * Note: jika yang dimiliki adalah {@link MataKuliah}, gunakanlah {@link MataKuliah#getKode()}.
297      * @param kodeMataKuliah kode mata kuliah yang ingin diperiksa.
298      * @return true jika sudah pernah mengambil, false jika belum
299      */
300     public boolean hasTempuhKuliah(String kodeMataKuliah) {
301         for (Nilai nilai: riwayatNilai) {
302             if (nilai.getMataKuliah().getKode().equals(kodeMataKuliah)) {
303                 return true;
304             }
305         }
306         return false;
307     }
308 
309     /**
310      * Mendapatkan tahun angkatan mahasiswa ini, berdasarkan NPM nya
311      * @return tahun angkatan
312      */
313     public int getTahunAngkatan() {
314         return Integer.parseInt(getNpm().substring(0, 4));
315     }
316 
317     @Override
318     public String toString() {
319         return "Mahasiswa [npm=" + npm + ", nama=" + nama + "]";
320     }
321 
322     /**
323      * Merepresentasikan nilai yang ada di riwayat nilai mahasiswa
324      * @author pascal
325      *
326      */
327     public static class Nilai {
328         /** Tahun dan Semester kuliah ini diambil */
329         protected final TahunSemester tahunSemester;
330         /** Mata kuliah yang diambil */
331         protected final MataKuliah mataKuliah;
332         /** Kelas kuliah */
333         protected final Character kelas;
334         /** Nilai ART */
335         protected final Double nilaiART;
336         /** Nilai UTS */
337         protected final Double nilaiUTS;
338         /** Nilai UAS */
339         protected final Double nilaiUAS;
340         /** Nilai Akhir */
341         protected final Character nilaiAkhir;
342 
343         public Nilai(TahunSemester tahunSemester, MataKuliah mataKuliah,
344                 Character kelas, Double nilaiART, Double nilaiUTS, Double nilaiUAS,
345                 Character nilaiAkhir) {
346             super();
347             this.tahunSemester = tahunSemester;
348             this.mataKuliah = mataKuliah;
```

```
349             this.kelas = kelas;
350             this.nilaiART = nilaiART;
351             this.nilaiUTS = nilaiUTS;
352             this.nilaiUAS = nilaiUAS;
353             this.nilaiAkhir = nilaiAkhir;
354         }
355
356         public MataKuliah getMataKuliah() {
357             return mataKuliah;
358         }
359
360         public Character getKelas() {
361             return kelas;
362         }
363
364         public Double getNilaiART() {
365             return nilaiART;
366         }
367
368         public Double getNilaiUTS() {
369             return nilaiUTS;
370         }
371
372         public Double getNilaiUAS() {
373             return nilaiUAS;
374         }
375
376         /**
377          * Mengembalikan nilai akhir dalam bentuk huruf (A, B, C, D, ..., atau K)
378          * @return nilai akhir dalam huruf, atau null jika tidak ada.
379          */
380         public Character getNilaiAkhir() {
381             return nilaiAkhir;
382         }
383
384         /**
385          * Mendapatkan nilai akhir dalam bentuk angka
386          * @return nilai akhir dalam angka, atau null jika {@link #getNilaiAkhir() mengembalikan 'K' atau null}
387          */
388         public Double getAngkaAkhir() {
389             if (nilaiAkhir == null) {
390                 return null;
391             }
392             switch (nilaiAkhir) {
393             case 'A':
394                 return 4.0;
395             case 'B':
396                 return 3.0;
397             case 'C':
398                 return 2.0;
399             case 'D':
400                 return 1.0;
401             case 'E':
402                 return 0.0;
403             case 'K':
404                 return null;
405             }
406             return null;
407         }
408
409         public TahunSemester getTahunSemester() {
410             return tahunSemester;
411         }
412
413         public int getTahunAjaran() {
414             return tahunSemester.getTahun();
415         }
416
417         public Semester getSemester() {
418             return tahunSemester.getSemester();
419         }
420
421         @Override
422         public String toString() {
423             return "Nilai_[tahunSemester=" + tahunSemester + ",_mataKuliah=" + mataKuliah + ",_kelas=" + kelas
424                     + ",_nilaiART=" + nilaiART + ",_nilaiUTS=" + nilaiUTS + ",_nilaiUAS=" + nilaiUAS + ",_nilaiAkhir="
425                     + nilaiAkhir + "]";
426         }
427
428         /**
429          * Pembanding antara satu nilai dengan nilai lainnya, secara
430          * kronologis waktu pengambilan.
431          * @author pascal
432          *
433          */
434         public static class ChronologicalComparator implements Comparator<Nilai> {
435
436             @Override
437             public int compare(Nilai o1, Nilai o2) {
438                 return o1.getTahunSemester().compareTo(o2.getTahunSemester());
439             }
440         }
441     }
442 }
```

Listing D.5: MataKuliah.java

```
1 package id.ac.unpar.siamodels;
```

```

public abstract class MataKuliah {
    private final String kode;
    private final String nama;
    private final Integer sks;

    public MataKuliah() {
        this.kode = this.getClass().getSimpleName();
        if (this.getClass().isAnnotationPresent(InfoMataKuliah.class)) {
            InfoMataKuliah infoMK = (InfoMataKuliah) this.getClass().getAnnotation(InfoMataKuliah.class);
            this.nama = infoMK.nama();
            this.sks = infoMK.sks();
        } else {
            this.nama = null;
            this.sks = null;
        }
    }

    public MataKuliah(String kode, String nama, int sks) {
        this.kode = kode;
        this.nama = nama;
        this.sks = sks;
    }

    public String getKode() {
        return kode;
    }

    public String getNama() {
        return nama;
    }

    public Integer getSks() {
        return sks;
    }
}
```

Listing D.6: MataKuliahFactory.java

```
package id.ac.unpar.siamodels;

import java.lang.annotation.Annotation;
import java.util.SortedMap;
import java.util.TreeMap;
import java.util.logging.Logger;

/**
 * Kelas yang bertugas membuat kelas mata kuliah, dan menyimpannya untuk bisa
 * digunakan kemudian (untuk hemat memori).
 *
 * @author pascal
 *
 */
public class MataKuliahFactory {

    /**
     * Lokasi package untuk daftar mata kuliah
     */
    public static String DEFAULT_MATAKULIAH_PACKAGE = "id.ac.unpar.siamodels.matakuliah";

    /**
     * Singleton instance to factory.
     */
    private static MataKuliahFactory instance = null;

    /**
     * Singleton instances untuk mata kuliah.
     */
    private final SortedMap<String, MataKuliah> mataKuliahCache;

    public static MataKuliahFactory getInstance() {
        if (instance == null) {
            instance = new MataKuliahFactory();
        }
        return instance;
    }

    protected MataKuliahFactory() {
        this.mataKuliahCache = new TreeMap<>();
    }

    /**
     * Membuat baru atau mendapatkan mata kuliah, jika memiliki informasi
     * nama dan jumlah SKS.
     *
     * @param kode
     *            kode mata kuliah
     * @param sks
     *            jumlah SKS
     * @param nama
     *            nama mata kuliah
     * @return objek mata kuliah
     */
    public MataKuliah createMataKuliah(String kode, int sks, String nama) {
        MataKuliah mk = this.mataKuliahCache.get(kode);
        // Coba dapatkan mata kuliah dari cache
        if (mk != null) {
            // Update jika kita punya info lebih baik
```

```
60              if (mk.getSks() == null || mk.getNama() == null) {
61                  mk = new MataKuliah(kode, nama, sks) {};
62                  this.mataKuliahCache.put(kode, mk);
63              }
64              return mk;
65          }
66
67          // Coba dapatkan dari kelas statik
68          Class<?> mkClass;
69          try {
70              mkClass = Class.forName(DEFAULT_MATAKULIAH_PACKAGE + "." + kode);
71              mk = (MataKuliah) mkClass.newInstance();
72          } catch (ClassNotFoundException e) {
73              mk = new MataKuliah(kode, nama, sks) {};
74              Logger.getGlobal().warning("Class␣is␣not␣listed:␣" + String.format("%s-%d␣%s", kode, sks, nama));
75          } catch (InstantiationException e) {
76              Logger.getGlobal().warning("Internal␣error:␣" + e.getMessage());
77              e.printStackTrace();
78          } catch (IllegalAccessException e) {
79              Logger.getGlobal().warning("Internal␣error:␣" + e.getMessage());
80              e.printStackTrace();
81          }
82          this.mataKuliahCache.put(kode, mk);
83          return mk;
84      }
85
86      /**
87       * Membuat baru atau mendapatkan mata kuliah, jika tidak memiliki informasi
88       * nama dan jumlah SKS.
89       *
90       * @param kode
91       *            kode mata kuliah
92       * @return objek mata kuliah
93       * @throws IllegalStateException
94       *             jika sks dan tidak sesuai dengan yang ada di kode
95       */
96      public MataKuliah createMataKuliah(final String kode)
97              throws IllegalStateException {
98          MataKuliah mk = this.mataKuliahCache.get(kode);
99          // Coba dapatkan mata kuliah dari cache
100         if (mk != null) {
101             return mk;
102         }
103
104         // Coba dapatkan dari kelas statik
105         Class<?> mkClass;
106         try {
107             mkClass = Class.forName(DEFAULT_MATAKULIAH_PACKAGE + "." + kode);
108             mk = (MataKuliah) mkClass.newInstance();
109         } catch (ClassNotFoundException e) {
110             throw new IllegalStateException("Mata␣kuliah␣" + kode + "␣tidak␣ditemukan!");
111         } catch (InstantiationException e) {
112             Logger.getGlobal().warning("Internal␣error:␣" + e.getMessage());
113             e.printStackTrace();
114         } catch (IllegalAccessException e) {
115             Logger.getGlobal().warning("Internal␣error:␣" + e.getMessage());
116             e.printStackTrace();
117         }
118         this.mataKuliahCache.put(kode, mk);
119         return mk;
120     }
121
122 }
```

Listing D.7: Semester.java

```
1  package id.ac.unpar.siamodels;
2
3  public enum Semester {
4      UNKNOWN5(5), TRANSFER(6), PENDEK(10), GANJIL(20), GENAP(30);
5
6      public static Semester fromString(String text) {
7          return Semester.valueOf(text.toUpperCase());
8      }
9
10     private int order;
11
12     private Semester(int order) {
13         this.order = order;
14     }
15
16     int getOrder() {
17         return order;
18     }
19
20 }
```

Listing D.8: TahunSemester.java

```
1  package id.ac.unpar.siamodels;
2
3  /**
4   * Menyimpan konstanta untuk semester beserta tahunnya di UNPAR.
5   * @author pascal
6   *
7   */
```

```java
public final class TahunSemester implements Comparable<TahunSemester> {

    /**
     * Kode semester 3 dijit, sesuai DPS:
     * <ul>
     *   <li>2 dijit pertama berupa tahun, 2 dijit terakhir</li>
     *   <li>dijit terakhir: 1 untuk ganjil, 2 untuk genap, 4 untuk pendek.
     * </ul>
     */
    private String kodeTahunSemester;

    public TahunSemester(String kodeTahunSemester) throws IllegalArgumentException {
        validateKodeSemester(kodeTahunSemester);
        this.kodeTahunSemester = kodeTahunSemester;
    }

    public TahunSemester(int tahun, Semester semester) throws IllegalArgumentException {
        char kodeSemester = '\0';
        switch (semester) {
        case GANJIL:
            kodeSemester = '1';
            break;
        case GENAP:
            kodeSemester = '2';
            break;
        case PENDEK:
            kodeSemester = '4';
            break;
        case UNKNOWN5:
            kodeSemester = '5';
            break;
        case TRANSFER:
            kodeSemester = '6';
            break;
        default:
            throw new InternalError("Incomplete TahunSemester switch case");
        }
        String kodeTahunSemester = ("" + tahun).substring(2, 4) + kodeSemester;
        validateKodeSemester(kodeTahunSemester);
        this.kodeTahunSemester = kodeTahunSemester;
    }

    public Semester getSemester() {
        switch (kodeTahunSemester.charAt(2)) {
        case '1': return Semester.GANJIL;
        case '2': return Semester.GENAP;
        case '4': return Semester.PENDEK;
        case '5': return Semester.UNKNOWN5;
        case '6': return Semester.TRANSFER;
        }
        return null;
    }

    public int getTahun() {
        return 2000 + Integer.parseInt(kodeTahunSemester.substring(0, 2));
    }

    private static void validateKodeSemester(String kodeTahunSemester) throws IllegalArgumentException {
        if (!kodeTahunSemester.matches("[0-9][0-9][12456]")) {
            throw new IllegalArgumentException("Kode semester tidak valid: " + kodeTahunSemester);
        }
    }

    /**
     * Mendapatkan kode tahun/semester sesuai aturan di DPS.
     * @return kode tahun/semester sesuai aturan di DPS.
     */
    public String getKodeDPS() {
        return kodeTahunSemester;
    }

    @Override
    public int compareTo(TahunSemester o) {
        if (this.getTahun() < o.getTahun()) {
            return -1;
        }
        if (this.getTahun() > o.getTahun()) {
            return + 1;
        }
        if (this.getSemester().getOrder() < o.getSemester().getOrder()) {
            return -1;
        }
        if (this.getSemester().getOrder() > o.getSemester().getOrder()) {
            return +1;
        }
        return 0;
    }

    @Override
    public boolean equals(Object obj) {
        if (!(obj instanceof TahunSemester)) {
            return false;
        }
        return this.compareTo((TahunSemester)obj) == 0;
    }

    @Override
    public String toString() {
        return "TahunSemester [" + getTahun() + "/" + getSemester() + "]";
```

```
107     }
108
109
110 }
```

Listing D.9: HasPraktikum.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 id.ac.unpar.siamodels.matakuliah.interfaces;
7
8  /**
9   *
10  * @author FTIS\i13037
11  */
12 public interface HasPraktikum {
13
14 }
```

Listing D.10: HasPrasyarat.java

```
1  package id.ac.unpar.siamodels.matakuliah.interfaces;
2
3  import java.util.List;
4
5  import id.ac.unpar.siamodels.Mahasiswa;
6
7  /**
8   * Mendefinisikan kelas-kelas yang memiliki prasyarat, terkustomisasi
9   * untuk seorang {@link Mahasiswa}. Jika ada tambahan, jangan lupa untuk
10  * mendaftarkannya di {@link #DEFAULT_HASPRASYARAT_CLASSES}. Jika berubah package,
11  * jangan lupa mengupdate {@link #DEFAULT_PRASYARAT_PACKAGE}.
12  * @author pascal
13  *
14  */
15 public interface HasPrasyarat {
16
17     /**
18      * Daftar dari nama kelas default seluruh turunan interface ini. Perlu didaftarkan
19      * manual, karena Java reflection tidak dapat mendeteksi otomatis.
20      */
21     public String[] DEFAULT_HASPRASYARAT_CLASSES = { "AIF102", "AIF202",
22             "AIF204", "AIF206", "AIF208", "AIF302", "AIF304", "AIF306",
23             "AIF312", "AIF314", "AIF316", "AIF318", "AIF332", "AIF339",
24             "AIF342", "AIF344", "AIF360", "AIF401", "AIF402", "AIF445",
25             "AIF458", "AIF461", "APS402" };
26
27     /**
28      * Package tempat menyimpan seluruh turunan standar interface ini. Perlu didefinisikan
29      * manual, karena Java reflection tidak dapat mendeteksi otomatis.
30      */
31     public String DEFAULT_PRASYARAT_PACKAGE = "id.ac.unpar.siamodels.matakuliah";
32
33     /**
34      * Memeriksa prasyarat-prasyarat dari kuliah, spesifik untuk mahasiswa
35      * yang dituju. Jika ada pesan-pesan khusus, akan ditambahkan pada parameter
36      * reasonsContainer.
37      * @param mahasiswa prasyarat kuliah akan diperiksa spesifik pada mahasiswa ini
38      * @param reasonsContainer pesan-pesan terkait prasyarat akan ditambahkan di sini, jika ada.
39      * @return true jika seluruh prasyarat dipenuhi, false jika tidak.
40      */
41     public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer);
42 }
```

Listing D.11: HasResponsi.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 id.ac.unpar.siamodels.matakuliah.interfaces;
7
8  /**
9   *
10  * @author FTIS\i13037
11  */
12 public interface HasResponsi {
13
14 }
```

Listing D.12: Kelulusan.java

```
1  package id.ac.unpar.siamodels.prodi.teknikinformatika;
2
3  import java.time.LocalDate;
4  import java.util.Arrays;
5  import java.util.Iterator;
6  import java.util.List;
```

```
7  import java.util.Map;
8  import java.util.Set;
9  import java.util.SortedMap;
10
11 import id.ac.unpar.siamodels.Mahasiswa;
12 import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
13 import java.util.Collection;
14
15 public class Kelulusan implements HasPrasyarat {
16
17     public static final String[] PILIHAN_WAJIB = {"AIF311", "AIF312", "AIF313", "AIF314", "AIF315", "AIF316",
18         "AIF317"};
19     public static final String[][] WAJIB = {{"AIF101", "AIF103", "AIF105", "MKU001", "MKU008", "MKU010"},
20     {"AIF102", "AIF104", "AIF106", "AMS100", "MKU009", "MKU011"},
21     {"AIF210", "AIF203", "AIF205", "AMS200", "MKU012"}, {"AIF202", "AIF204", "AIF206", "AIF208", "AIF210"},
22     {"AIF301", "AIF303", "AIF305", "MKU002"}, {"AIF302"},
23     {"AIF401", "AIF403"}, {"AIF402", "APS402"}};
24     public static final String[] AGAMA = {"MKU003", "MKU004"};
25
26     public static final int MIN_SKS_LULUS = 144;
27
28     public static final int MIN_PILIHAN_WAJIB = 4;
29
30     @Override
31     /**
32      * Melakukan pengecekan syarat kelulusan
33      *
34      * @param mahasiswa mahasiswa yang dicek
35      * @param reasonsContainer alasan2 yang ada jika tidak lulus
36      * @return boolean true jika memenuhi syarat, false jika sebaliknya
37      */
38     public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
39         // cek sks
40         boolean bisaLulus = true;
41         int sks = mahasiswa.calculateSKSLulus();
42         if (sks < MIN_SKS_LULUS) {
43             reasonsContainer.add(String.format("Anda baru lulus %d SKS, masih kurang %d SKS lagi untuk mencapai %d.", sks,
                   MIN_SKS_LULUS - sks, MIN_SKS_LULUS));
44             bisaLulus = false;
45         }
46         // cek agama
47         boolean lulusSalahSatuMKAgama = false;
48         for (int i = 0; i < AGAMA.length; i++) {
49             if (mahasiswa.hasLulusKuliah(AGAMA[i])) {
50                 lulusSalahSatuMKAgama = true;
51                 break;
52             }
53         }
54         if (!lulusSalahSatuMKAgama) {
55             reasonsContainer.add("Anda belum lulus salah satu dari MK Agama " + Arrays.toString(AGAMA));
56             bisaLulus = false;
57         }
58         // cek kuliah wajib
59         for (String[] mkWajibSemesterI : WAJIB) {
60             for (String mkWajib : mkWajibSemesterI) {
61                 if (!mahasiswa.hasLulusKuliah(mkWajib)) {
62                     reasonsContainer.add("Anda belum lulus MK wajib " + mkWajib);
63                     bisaLulus = false;
64                 }
65             }
66         }
67         // cek pilihan wajib
68         int lulusPilihanWajib = 0;
69         for (String mkPilihanWajib : PILIHAN_WAJIB) {
70             if (mahasiswa.hasLulusKuliah(mkPilihanWajib)) {
71                 lulusPilihanWajib++;
72             }
73         }
74         if (lulusPilihanWajib < MIN_PILIHAN_WAJIB) {
75             reasonsContainer.add(String.format("Anda baru lulus %d MK pilihan wajib, sedangkan Anda perlu lulus %d",
                   lulusPilihanWajib, MIN_PILIHAN_WAJIB));
76             bisaLulus = false;
77         }
78         // cek projek
79         if (!mahasiswa.hasLulusKuliah("AIF306") && !mahasiswa.hasLulusKuliah("AIF405")) {
80             reasonsContainer.add("Anda belum lulus salah satu dari MK Proyek AIF306 atau AIF304 & AIF405");
81             bisaLulus = false;
82         }
83         // cek nilai TOEFL
84         SortedMap<LocalDate, Integer> toeflScore = mahasiswa.getNilaiTOEFL();
85         if(toeflScore == null) {
86             reasonsContainer.add("Belum ada skor TOEFL.");
87             bisaLulus = false;
88         }
89         Collection nilai = toeflScore.values();
90         int maxToefl = 0;
91         Iterator i = nilai.iterator();
92         while (i.hasNext()) {
93             int val = (int) i.next();
94             if (maxToefl < val) {
95                 maxToefl = val;
96             }
97         }
98         if (!(maxToefl >= 500)) {
99             if (toeflScore.size() <= 8) {
100                reasonsContainer.add("Belum mencapai nilai TOEFL sebesar 500.");
101                bisaLulus = false;
102            } else {
103                if (maxToefl < 450) {
```

```
                    reasonsContainer.add("Belum mencapai nilai TOEFL sebesar 450.");
                    bisaLulus = false;
                } else {
                    reasonsContainer.add("Sudah lulus TOEFL dengan nilai " + maxToefl + " dan memerlukan dispensasi dari rektor
                        karena sudah mengambil tes TOEFL sebanyak "+toeflScore.size()+" kali.");
                }
            }
        }
        return bisaLulus;
    }

}
```

Listing D.13: AIF101.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;


/**
 * Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar pemrograman
 * seperti pengulangan dan percabangan, konsep dasar penyimpanan data kontigu
 * menggunakan array, konsep dasar pemrograman berorientasi objek seperti kelas
 * &amp; objek, method, dll, termasuk di dalamnya 4 prinsip dasar pemrograman
 * berorientasi objek : data abstraction, encapsulation, inheritance dan
 * polymorphism. Selain, itu diberikan masalah-masalah komputasi sederhana
 * yang harus diselesaikan menggunakan konsep-konsep yang  sudah diperkenalkan
 * dan mengimplementasikannya menggunakan bahasa pemrograman Java
 * @author Lionov, M.Sc. (lionov@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Pemrograman Berorientasi Objek", sks = 6)
public class AIF101 extends MataKuliah implements HasPraktikum {

}
```

Listing D.14: AIF102.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;

import java.util.List;

/**
 * Mata kuliah ini memperkenalkan berbagai algoritma dan teknik-teknik
 * penyelesaian masalah komputasi seperti rekursif, sorting, teknik divide dan
 * conquer, serta exhaustive search. Selain itu, pada kuliah ini juga
 * dikenalkan berbagai struktur data yang dapat digunakan untuk mendukung
 * penyelesaian masalah komputasi seperti ADT List, Stack dan Queue. Baik
 * algoritma maupun struktur data yang dikenalkan harus dapat diimplementasikan
 * dan digunakan oleh mahasiswa untuk menyelesaikan masalah dengan menggunakan
 * suatu bahasa pemrograman berorientasi objek.
 * @author husnulhakim@unpar.ac.id
 */
@InfoMataKuliah(nama = "Algoritma dan Struktur Data", sks = 4)
public class AIF102 extends MataKuliah implements HasPrasyarat, HasPraktikum {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasLulusKuliah("AIF101") && !mahasiswa.hasLulusKuliah("AIF191")) {
            reasonsContainer.add("Tidak memenuhi prasyarat lulus AIF101 atau AIF191");
            ok = false;
        }
        if (!mahasiswa.hasTempuhKuliah("AIF103")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF103");
            ok = false;
        }
        return ok;
    }

}
```

Listing D.15: AIF103.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah ini merupakan salah satu cara untuk mencapai kompetensi dasar
 * tentang matematika diskrit yang prinsipnya banyak digunakan dalam bidang
 * ilmu komputer. Selain itu, kuliah ini juga merupakan cara untuk membentuk
 * pola pikir logis yang dibutuhkan untuk menempuh kuliah-kuliah di tingkat
 * yang lebih tinggi.
 * @author Mariskha Tri Adithia (mariskha@unpar.ac.id)
```

```
 */
@InfoMataKuliah(nama = "Matematika Diskrit", sks = 3)
public class AIF103 extends MataKuliah{

}
```

Listing D.16: AIF104.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah ini memberikan pengetahuan tentang logika yang digunakan di
 * dalam ilmu komputer. Dalam kuliah ini, mahasiswa belajar untuk bisa
 * memodelkan suatu kalimat dalam kehidupan sehari-hari, ke dalam kalimat
 * dengan sintaks tertentu, yang hanya memiliki satu arti. Lalu, diperkenalkan
 * juga, bagaimana mengartikan suatu kalimat (benar atau salah) dan bagaimana
 * menentukan sifat dari kalimat tersebut.
 * @author Mariskha Tri Adithia (mariskha@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Logika Informatika", sks = 3)
public class AIF104 extends MataKuliah{


}
```

Listing D.17: AIF105.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah ini memperkenalkan kepada mahasiswa terminologi dan konsep dasar
 * yang akan banyak dipakai sepanjang kuliah di Teknik Informatika. Selain itu
 * mata kuliah ini juga mempersiapkan dan membiasakan mahasiswa dengan suasana
 * akademik yang khas perguruan tinggi seperti kedisiplinan, kerja sama,
 * kemampuan menggunakan teknologi informasi dalam pembuatan tugas, kemampuan
 * komunikasi, dsb.
 * @author Thomas Anung Basuki (anung@unpar.ac.id)
 * @author Mariskha Tri Adithia (mariskha@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Pengantar Informatika", sks = 3)
public class AIF105 extends MataKuliah{


}
```

Listing D.18: AIF106.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah ini memberikan pengetahuan tentang cara kerja komputer, dimulai
 * dari representasi data dan berbagai macam operasinya. Selanjutnya, juga
 * diperkenalkan bagaimana merepresentasikan suatu fungsi dalam rangkaian
 * gerbang logika, dan bagaimana menyederhanakannya. Berbagai rangkaian dasar
 * yang digunakan di dalam komputer juga dipekenalkan. Mahasiswa juga akan
 * mempelajari komponen komputer, misalnya register dan memori.
 * @author Mariskha Tri Adithia (mariskha@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Sistem Dijital", sks = 3)
public class AIF106 extends MataKuliah{

}
```

Listing D.19: AIF181.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Dasar-dasar Pemrograman", sks = 3)
public class AIF181 extends MataKuliah {

}
```

Listing D.20: AIF182.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;
```

```
5
6 @InfoMataKuliah(nama = "Pengantar Basis Data", sks = 2)
7 public class AIF182 extends MataKuliah {
8
9 }
```

Listing D.21: AIF183.java

```
1 package id.ac.unpar.siamodels.matakuliah;
2
3 import id.ac.unpar.siamodels.InfoMataKuliah;
4 import id.ac.unpar.siamodels.MataKuliah;
5
6 @InfoMataKuliah(nama = "Pemrograman Prosedural", sks = 4)
7 public class AIF183 extends MataKuliah {
8
9 }
```

Listing D.22: AIF191.java

```
1 package id.ac.unpar.siamodels.matakuliah;
2
3 import id.ac.unpar.siamodels.InfoMataKuliah;
4 import id.ac.unpar.siamodels.MataKuliah;
5
6 /**
7  * @deprecated Mata kuliah tidak dibuka lagi.
8  */
9 @InfoMataKuliah(nama = "Pemrograman Berorientasi Objek", sks = 3)
10 public class AIF191 extends MataKuliah {
11
12 }
```

Listing D.23: AIF192.java

```
1 package id.ac.unpar.siamodels.matakuliah;
2
3 import id.ac.unpar.siamodels.InfoMataKuliah;
4 import id.ac.unpar.siamodels.MataKuliah;
5
6 /**
7  * @deprecated Mata kuliah tidak dibuka lagi.
8  */
9 @InfoMataKuliah(nama = "Algoritma & Struktur Data", sks = 3)
10 public class AIF192 extends MataKuliah {
11
12 }
```

Listing D.24: AIF201.java

```
1 package id.ac.unpar.siamodels.matakuliah;
2
3 import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
4 import id.ac.unpar.siamodels.Mahasiswa;
5 import id.ac.unpar.siamodels.MataKuliah;
6 import id.ac.unpar.siamodels.InfoMataKuliah;
7 import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
8 import id.ac.unpar.siamodels.matakuliah.interfaces.HasResponsi;
9
10 import java.util.List;
11
12 /**
13  * Mata kuliah ini memperkenalkan prinsip-prinsip yang digunakan dalam
14  * melakukan analisa serta desain prorgram berorientasi objek. Di samping itu,
15  * mahasiswa juga belajar menggunakan kakas berupa diagram UML (Unified
16  * Modelling Language) sehingga dapat mengkomunikasikan desain secara visual.
17  * Mahasiswa juga akan mengenal beberapa software design pattern dari Gang of
18  * Four. Terakhir, mahasiswa akan belajar mengenai konsep MVC
19  * (Model-View-Controller) yang menjadi dasar dari banyak framework masa kini.
20  * Bahasa yang digunakan adalah bahasa Java, namun diusahakan tetap umum
21  * sehingga dapat diaplikasikan pada bahasa yang lain.
22  * @author Pascal (pascal@unpar.ac.id)
23  */
24 @InfoMataKuliah(nama = "Analisis & Desain Berorientasi Objek", sks = 4)
25 public class AIF201 extends MataKuliah implements HasPrasyarat, HasPraktikum, HasResponsi {
26
27     @Override
28     public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
29         boolean ok = true;
30         if (!mahasiswa.hasLulusKuliah("AIF101") && !mahasiswa.hasLulusKuliah("AIF191")) {
31             reasonsContainer.add("Tidak memenuhi prasyarat lulus AIF101 atau AIF191");
32             ok = false;
33         }
34         return ok;
35     }
36
37 }
```

Listing D.25: AIF202.java

```
1 package id.ac.unpar.siamodels.matakuliah;
```

```
2
3  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
4  import id.ac.unpar.siamodels.Mahasiswa;
5  import id.ac.unpar.siamodels.MataKuliah;
6  import id.ac.unpar.siamodels.InfoMataKuliah;
7  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
8  import id.ac.unpar.siamodels.matakuliah.interfaces.HasResponsi;
9
10
11 import java.util.List;
12
13 /**
14  * Mata kuliah ini memperkenalkan kepada mahasiswa beberapa algoritma dan
15  * struktur data, alternatif cara implementasinya, dan analisis kompleksitas
16  * waktunya. Mahasiswa diberikan beberapa masalah komputasi yang harus
17  * diselesaikan dengan menggunakan algoritma atau struktur data yang sudah
18  * diperkenalkan dan mengimplementasikannya dalam bahasa pemrograman Java.
19  * @author Joanna Helga, M.Sc. (joanna@unpar.ac.id)
20  */
21 @InfoMataKuliah(nama = "Desain & Analisis Algoritma", sks = 4)
22 public class AIF202 extends MataKuliah implements HasPrasyarat, HasResponsi, HasPraktikum {
23
24     @Override
25     public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
26         boolean ok = true;
27         if (!mahasiswa.hasLulusKuliah("AIF102") && !mahasiswa.hasLulusKuliah("AIF192")) {
28             reasonsContainer.add("Tidak memenuhi prasyarat lulus AIF102 atau AIF192");
29             ok = false;
30         }
31         int angkatan = mahasiswa.getTahunAngkatan();
32         if (angkatan >= 2012 && angkatan <= 2014) {
33             if (!mahasiswa.hasTempuhKuliah("AIF203")) {
34                 reasonsContainer.add("Angkatan " + angkatan + " tetapi tidak memenuhi prasyarat tempuh AIF103");
35                 ok = false;
36             }
37         }
38         return ok;
39     }
40
41 }
```

Listing D.26: AIF203.java

```
1  package id.ac.unpar.siamodels.matakuliah;
2
3  import id.ac.unpar.siamodels.Mahasiswa;
4  import id.ac.unpar.siamodels.MataKuliah;
5  import id.ac.unpar.siamodels.InfoMataKuliah;
6  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
7
8  import java.util.List;
9
10 /**
11  * Mata kuliah ini memperkenalkan kepada mahasiswa konsep struktur diskret yang
12  * digunakan pada bidang informatika diantaranya graph, pohon dan finite state
13  * machine
14  * @author Dr.rer.nat Cecilia Esti Nugraheni (cheni@unpar.ac.id)
15  */
16 @InfoMataKuliah(nama = "Struktur Diskrit", sks = 4)
17 public class AIF203 extends MataKuliah implements HasPrasyarat {
18
19     @Override
20     public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
21         boolean ok = true;
22         if (!mahasiswa.hasTempuhKuliah("AIF103")) {
23             reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF103");
24             ok = false;
25         }
26         return ok;
27     }
28
29 }
```

Listing D.27: AIF204.java

```
1  package id.ac.unpar.siamodels.matakuliah;
2
3  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
4  import id.ac.unpar.siamodels.Mahasiswa;
5  import id.ac.unpar.siamodels.MataKuliah;
6  import id.ac.unpar.siamodels.InfoMataKuliah;
7  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
8
9  import java.util.List;
10
11 /**
12  * Mata kuliah ini memperkenalkan konsep dan arsitektur DBMS, mengajarkan
13  * aljabar relasional dan SQL serta pemanfaatannya pada pemrograman kueri
14  * sederhana s/d relatif kompleks. Selain itu, mata kuliah ini juga mengajarkan
15  * dan mempraktekkan perancangan basisdata untuk masalah sederhana
16  * (lingkup kecil) termasuk pengembangan program aplikasinya;
17  * @author Veronica S. Moertini (moertini@unpar.ac.id)
18  */
19 @InfoMataKuliah(nama = "Manajemen Informasi dan Basis Data", sks = 4)
20 public class AIF204 extends MataKuliah implements HasPrasyarat, HasPraktikum {
21
```

```
22      @Override
23      public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
24          boolean ok = true;
25          int angkatan = mahasiswa.getTahunAngkatan();
26          if (angkatan >= 2012 && angkatan <= 2014) {
27              if (!mahasiswa.hasTempuhKuliah("AIF203")) {
28                  reasonsContainer.add("Angkatan " + angkatan + " tetapi tidak memenuhi prasyarat tempuh AIF103");
29                  ok = false;
30              }
31          }
32          return ok;
33      }
34
35  }
```

Listing D.28: AIF205.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import id.ac.unpar.siamodels.Mahasiswa;
4   import id.ac.unpar.siamodels.MataKuliah;
5   import id.ac.unpar.siamodels.InfoMataKuliah;
6   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
7
8   import java.util.List;
9
10  /**
11   * Mata kuliah ini memperkenalkan kepada mahasiswa arsitektur komputer
12   * sederhana, modern, dan Advance. Perbedaan, kelebihan dan kekurangan untuk
13   * masing-masing arsitektur. Selain itu mahasiswa juga mempelajari cara kerja
14   * dari komponen-komponen komputer, terutama memory, cache, system BUS dan
15   * input/output.
16   * @author Chandra Wijaya (chandraw@unpar.ac.id)
17   */
18  @InfoMataKuliah(nama = "Arsitektur & Organisasi Komputer", sks = 3)
19  public class AIF205 extends MataKuliah implements HasPrasyarat {
20
21      @Override
22      public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
23          boolean ok = true;
24          if (!mahasiswa.hasTempuhKuliah("AIF106")) {
25              reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF106");
26              ok = false;
27          }
28          return ok;
29      }
30
31  }
```

Listing D.29: AIF206.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import id.ac.unpar.siamodels.Mahasiswa;
4   import id.ac.unpar.siamodels.MataKuliah;
5   import id.ac.unpar.siamodels.InfoMataKuliah;
6   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
7
8   import java.util.List;
9
10  /**
11   * Mata kuliah ini memperkenalkan kepada mahasiswa mengenai konsep sistem
12   * operasi, jenis-jenis sistem operasi yang digunakan dalam kehidupan
13   * sehari-hari dan beberapa perangkat keras yang dibutuhkan pada komputer.
14   * Selain itu juga mempelajari mengenai teknik dan algoritma yang digunakan
15   * dalam pengelolaan sistem operasi.
16   * @author Chandra Wijaya (chandraw@unpar.ac.id)
17   */
18  @InfoMataKuliah(nama = "Sistem Operasi", sks = 4)
19  public class AIF206 extends MataKuliah implements HasPrasyarat{
20
21      @Override
22      public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
23          boolean ok = true;
24          if (!mahasiswa.hasTempuhKuliah("AIF102") && !mahasiswa.hasTempuhKuliah("AIF192")) {
25              reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF102 atau AIF192");
26              ok = false;
27          }
28          if (!mahasiswa.hasTempuhKuliah("AIF205")) {
29              reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF205");
30              ok = false;
31          }
32          return ok;
33      }
34
35  }
```

Listing D.30: AIF208.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import id.ac.unpar.siamodels.Mahasiswa;
4   import id.ac.unpar.siamodels.MataKuliah;
5   import id.ac.unpar.siamodels.InfoMataKuliah;
```

```java
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Mata kuliah ini memperkenalkan kepada mahasiswa tahapan rekayasa perangkat
 * lunak, terutama dengan paradigma berorientasi objek, dilengkapi dengan
 * pengenalan tentang manajemen proyek perangkat lunak.
 * Selain, itu diberikan deskripsi proyek berskala kecil yang harus dikerjakan
 * oleh mahasiswa dalam kelompok dengan menerapkan teori yang telah
 * dipelajarinya.
 * @author Thomas Anung Basuki (anung@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Rekayasa Perangkat Lunak", sks = 4)
public class AIF208 extends MataKuliah implements HasPrasyarat{

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF201")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF201");
            return false;
        }
        return true;
    }

}
```

Listing D.31: AIF210.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Interaksi Manusia Komputer", sks = 2)
public class AIF210 extends MataKuliah {


}
```

Listing D.32: AIF280.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Praktika Interaksi Manusia Komputer", sks = 1)
public class AIF280 extends MataKuliah {

}
```

Listing D.33: AIF281.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pengenalan Bidang Ilmu Teknologi Informasi & Komunikasi", sks = 2)
public class AIF281 extends MataKuliah {

}
```

Listing D.34: AIF282.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Algoritma & Struktur Data Lanjut", sks = 3)
public class AIF282 extends MataKuliah {

}
```

Listing D.35: AIF292.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * @deprecated Mata kuliah tidak dibuka lagi.
 */
@InfoMataKuliah(nama = "Desain & Analisis Algoritma", sks = 3)
public class AIF292 extends MataKuliah {

}
```

Listing D.36: AIF294.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * @deprecated Mata kuliah tidak dibuka lagi.
 */
@InfoMataKuliah(nama = "Manajemen Informasi & Basis Data", sks = 3)
public class AIF294 extends MataKuliah {

}
```

Listing D.37: AIF301.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar tentang *sistem
 * cerdas dan komponen-komponennya. " "Terdapat 4 topik utama yang dibahas yaitu
 * teknik pencarian untuk *penyelesaian masalah, representasi pengetahuan dalam
 * sistem *cerdas, pemodelan ketidakpastian dalam masalah dan teknik
 * pembelajaran mesin.
 *
 * @author Thomas Anung Basuki
 *
 */

@InfoMataKuliah(nama = "Pengantar Sistem Cerdas ", sks = 3)
public class AIF301 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF204") && !mahasiswa.hasTempuhKuliah("AIF294")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF204 atau AIF294");
            ok = false;
        }
        return ok;
    }

}
```

Listing D.38: AIF302.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Mata kuliah ini melatih mahasiswa dalam menulis ilmiah serta memperkenalkan
 * metodologi penelitian serta kakas untuk menulis ilmiah.
 *
 * @author Thomas Anung Basuki (anung@unpar.ac.id)
 */

@InfoMataKuliah(nama = "Penulisan Ilmiah", sks = 2)
public class AIF302 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        int sksLulus = mahasiswa.calculateSKSLulus();
        if (sksLulus < 84) {
            reasonsContainer.add("SKS Lulus " + sksLulus + ", belum mencapai syarat minimal 84");
            return false;
        }
        return true;
    }

}
```

Listing D.39: AIF303.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

```

```java
import java.util.List;

/**
 * Mempelajari Konsep Data, Informasi, Pengetahuan, Sistem Informasi, proses dan
 * pemodelan bisnis, jenis-jenis sistem informasi, untuk mendukung pengambilan
 * keputusan. Mempelajari trend Teknologi Informasi, tahap-tahap pembangunan
 * sistem informasi. Mempelajari pengantar : EIS, e-bisnis/e-commerce, Business
 * Intelligence, Cloud Computing dan Mobile Applications
 *
 * @author Rosa de Lima E. Padmowati (rosad5@unpar.ac.id)
 */

@InfoMataKuliah(nama = "Pengantar Sistem Informasi", sks = 3)
public class AIF303 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF204") && !mahasiswa.hasTempuhKuliah("AIF294")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF204 atau AIF294");
            ok = false;
        }
        return ok;
    }

}
```

Listing D.40: AIF304.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasResponsi;

import java.util.List;

/**
 * Mata kuliah ini memberikan kesempatan bagi mahasiswa untuk memperdalam konsep
 * tentang pengembangan sistem informasi dan mempraktekkan analisis kebutuhan,
 * analisis sistem dan perancangan sitem pada organisasi studi kasus;
 *
 * @author Veronica S. Moertini (moertini@unpar.ac.id)
 */

@InfoMataKuliah(nama = "Proyek Sistem Informasi 1", sks = 3)
public class AIF304 extends MataKuliah implements HasPrasyarat, HasPraktikum, HasResponsi {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF303")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF303");
            return false;
        }
        return true;
    }

}
```

Listing D.41: AIF305.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar jaringan
 * komputer dengan menggunakan top-down. Selain itu mengajarkan juga kepada
 * mahasiswa mengenai aplikasi-aplikasi berbasis jaringan sehingga diharapkan
 * mahasiswa dapat membuat aplikasi berbasis jaringan dengan menggunakan socket.
 * Pada akhirnya, mahasiswa akan ditugaskan untuk membangun jaringan komputer
 * LAN, baik menggunakan kabel maupun nirkabel.
 *
 * @author Chandra Wijaya (chandraw@unpar.ac.id)
 */

@InfoMataKuliah(nama = "Jaringan Komputer", sks = 4)
public class AIF305 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF206")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF206");
            ok = false;
        }
        return ok;
    }
```

```
33
34 }
```

Listing D.42: AIF306.java

```
1  package id.ac.unpar.siamodels.matakuliah;
2
3  import id.ac.unpar.siamodels.Mahasiswa;
4  import id.ac.unpar.siamodels.MataKuliah;
5  import id.ac.unpar.siamodels.InfoMataKuliah;
6  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
7  import java.util.List;
8
9  /**
10  * Mata kuliah ini bertujuan untuk memberikan pengalaman bagi mahasiswa dalam
11  * mengerjakan proyek dengan teknologi-teknologi terkini, secara berkelompok.
12  * Teknologi-teknologi yang digunakan pada kuliah ini tidak spesifik dan dapat
13  * berubah seiring perkembangan teknologi maupun disesuaikan dengan kompetensi
14  * dosen pengajar. Beberapa teknologi yang dapat dimanfaatkan antara lain: DVCS
15  * tool menggunakan Git + Github, Mobile native app (Android, iOS, dll), dan
16  * responsive web design.
17  *
18  * @author Pascal (pascal@unpar.ac.id)
19  */
20
21 @InfoMataKuliah(nama = "Proyek Informatika", sks = 6)
22 public class AIF306 extends MataKuliah implements HasPrasyarat {
23
24     @Override
25     public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
26         if (!mahasiswa.hasTempuhKuliah("AIF208")) {
27             reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF208");
28             return false;
29         }
30         return true;
31     }
32
33 }
```

Listing D.43: AIF311.java

```
1  package id.ac.unpar.siamodels.matakuliah;
2
3  import id.ac.unpar.siamodels.Mahasiswa;
4  import id.ac.unpar.siamodels.MataKuliah;
5  import id.ac.unpar.siamodels.InfoMataKuliah;
6  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
7  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
8
9  import java.util.List;
10
11 /**
12  * Kuliah Pemrograman Fungsional bertujuan untuk: 1. memperkenalkan paradigma
13  * pemrograman fungsional, yaitu sebuah pemrograman yang didasarkan pada konsep
14  * pemetaan dan fungsi matematika. Penyelesaian suatu masalah didasari atas
15  * aplikasi dari fungsi-fungsi tersebut. 2. memberikan dasar-dasar pemrograman
16  * fungsional dengan menggunakan bahasa fungsional Haskell.
17  *
18  * @author Cecilia E. Nugraheni (cheni@unpar.ac.id)
19  */
20
21 @InfoMataKuliah(nama = "Pemrograman Fungsional", sks = 2)
22 public class AIF311 extends MataKuliah implements HasPrasyarat, HasPraktikum {
23
24     @Override
25     public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
26         boolean ok = true;
27         if (!mahasiswa.hasTempuhKuliah("AIF103")) {
28             reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF103");
29             ok = false;
30         }
31         return ok;
32     }
33
34 }
```

Listing D.44: AIF312.java

```
1  package id.ac.unpar.siamodels.matakuliah;
2
3  import id.ac.unpar.siamodels.Mahasiswa;
4  import id.ac.unpar.siamodels.MataKuliah;
5  import id.ac.unpar.siamodels.InfoMataKuliah;
6  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
7  import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
8
9  import java.util.List;
10
11 /**
12  * Mata kuliah ini memberikan pengetahuan awal tentang keamanan informasi. Pada
13  * beberapa pertemuan awal, dibahas keamanan informasi secara matematis, yaitu
14  * di materi-materi seputar kriptografi dan serangannya. Lalu, dibahas pula
15  * konsep keamanan informasi pada jaringan komputer dan pada software.
16  *
```

```
 * @author Mariskha Tri Adithia (mariskha@unpar.ac.id)
 */

@InfoMataKuliah(nama = "Keamanan Informasi", sks = 2)
public class AIF312 extends MataKuliah implements HasPrasyarat, HasPraktikum {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF305")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF305");
            return false;
        }
        return true;
    }

}
```

Listing D.45: AIF313.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;

/**
 * Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar pembuatan grafik
 * dengan menggunakan komputer seperti mengenal berbagai algoritma pembuatan
 * primitif 2 dimensi seperti titik, garis, lingkaran, elips, berbagai macam
 * bentuk kurva, fraktal, konsep warna (RGB), dasar-dasar grafika 3 dimensi
 * seperti pewarnaan, pencahayaan, pemberian tekstur pada objek, transformasi,
 * animasi, dan sebagainya. Selain, itu diberikan masalah-masalah komputasi
 * sederhana yang harus diselesaikan menggunakan konsep-konsep yang sudah
 * diperkenalkan dan mengimplementasikannya menggunakan bahasa pemrograman Java.
 *
 * @author Luciana (luciana@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Grafika Komputer", sks = 2)
public class AIF313 extends MataKuliah implements HasPraktikum {

}
```

Listing D.46: AIF314.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Kuliah ini merupakan kelanjutan dari kuliah Manajemen Informasi Basisdata.
 * Pada perkuliahan ini, mahasiswa akan mempelajari teknik-teknik pengelolaan
 * basis data dan membuat program dengan basis data yang optimal/efisien.
 *
 * @author Falahah . S.
 */
@InfoMataKuliah(nama = "Pemrograman Basis Data", sks = 2)
public class AIF314 extends MataKuliah implements HasPrasyarat, HasPraktikum {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF204") && !mahasiswa.hasTempuhKuliah("AIF294")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF204 atau AIF294");
            return false;
        }
        return true;
    }


}
```

Listing D.47: AIF315.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Mata kuliah ini memperkenalkan konsep dan lingkungan pemrograman berbasis web,
 * kemudian belajar membuat aplikasi berbasis web menggunakan HTML5, CSS, Java Script
 * dan PHP. Untuk meningkatkan keterampilan pemrograman dilengkapi dengan praktikum.
 * Sedangkan untuk mendapatkan pengalaman penerapan konsep diberikan tugas besar membuat
 * program berbasis web dengan kasus yang ditentukan oleh mahasiswa.
```

```
 * @author Gede Karya(gkarya@unpar.ac.id)
 * @author Husnul Hakim (husnulhakim@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Pemrograman Berbasis Web", sks = 2)
public class AIF315 extends MataKuliah implements HasPrasyarat, HasPraktikum {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF204") && !mahasiswa.hasTempuhKuliah("AIF294")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF204 atau AIF294");
            return false;
        }
        return true;
    }
}
```

Listing D.48: AIF316.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;
/**
 * Mata kuliah ini memperkenalkan konsep-konsep dasar komputasi paralel, dimana sebuah
 * program yang berjalan secara paralel harus memiliki safety property dan liveness property.
 * Mahasiswa dikenalkan dengan beberapa teknik pemrograman multi-thread
 * seperti lock, monitor, barrier, thread pool, dan sebagainya, yang diimplementasikan
 * dalam bahasa pemrograman Java. Mahasiswa juga dikenalkan dengan beberapa metode untuk
 * menganalisis kebenaran program baik secara matematis maupun secara praktis dengan bantuan
 * model checker.
 * @author Joanna Helga, M.Sc. (joanna@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Komputasi Paralel", sks = 2)
public class AIF316 extends MataKuliah implements HasPrasyarat, HasPraktikum {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF102") && !mahasiswa.hasTempuhKuliah("AIF192")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF102 atau AIF192");
            return false;
        }
        return true;
    }
}
```

Listing D.49: AIF317.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Desain Antarmuka Grafis", sks = 2)
public class AIF317 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF210")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF210");
            return false;
        }
        return true;
    }

}
```

Listing D.50: AIF318.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;
/**
 * Mata kuliah ini memperkenalkan konsep perangkat mobile dan pemrograman pada perangkat
 * mobile. Pemrograman dikhususkan pada lingkungan J2ME dan Android.
 * Untuk meningkatkan keterampilan pemrograman dilengkapi dengan praktikum.
 * Sedangkan untuk mendapatkan pengalaman penerapan konsep diberikan tugas implementasi suatu
 * kasus pada lingkungan mobile-cloud dengan kasus yang sudah ditentukan.
 *
 * @author Gede Karya (gkarya@unpar.ac.id)
```

```
 */
@InfoMataKuliah(nama = "Pemrograman Aplikasi Bergerak", sks = 2)
public class AIF318 extends MataKuliah implements HasPrasyarat, HasPraktikum {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF102") && !mahasiswa.hasTempuhKuliah("AIF192")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF102 atau AIF192");
            ok = false;
        }
        if (!mahasiswa.hasTempuhKuliah("AIF201")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF201");
            ok = false;
        }
        return ok;
    }
}
```

Listing D.51: AIF330.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Kerja Praktek 1", sks = 2)
public class AIF330 extends MataKuliah {

}
```

Listing D.52: AIF332.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Topik Khusus Informatika 2", sks = 3)
public class AIF332 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        // Note: Topik Khusus Informatika 2: Ruby on Rails
        if (!mahasiswa.hasTempuhKuliah("AIF315")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF315");
            return false;
        }
        return true;
    }

}
```

Listing D.53: AIF334.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;


@InfoMataKuliah(nama = "Topik Khusus Sistem Informasi 2", sks = 3)
public class AIF334 extends MataKuliah {

}
```

Listing D.54: AIF335.java

```
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pembelajaran Mesin", sks = 3)
public class AIF335 extends MataKuliah {

}
```

Listing D.55: AIF336.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah ini merupakan mata kuliah lanjutan dari mata kuliah Keamanan
```

```
 * Informasi, dengan titik berat pada materi kriptografi. Mata kuliah ini
 * memperkenalkan tambahan konsep kriptografi, misalnya tentang otentikasi
 * yaitu otentikasi entitas, manajemen kunci, dan bentuk lain dari metode
 * merahasiakan pesan, yaitu dengan menggunakan secret sharing. Selanjutnya,
 * diperkenalkan juga penggunaan kriptografi pada protokol-protokol yang
 * sebenarnya banyak digunakan sehari-hari, misalnya pada e-cash, auction,
 * dan electronic voting.
 * @author Mariskha Tri Adithia (mariskha@unpar.ac.id)
 *
 */
@InfoMataKuliah(nama = "Algoritma Kriptografi", sks = 3)
public class AIF336 extends MataKuliah {

}
```

Listing D.56: AIF337.java

```
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Matematika Teknik", sks = 3)
public class AIF337 extends MataKuliah {

}
```

Listing D.57: AIF339.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Pemodelan Formal", sks = 3)
public class AIF339 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF104")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF104");
            ok = false;
        }
        if (!mahasiswa.hasTempuhKuliah("AIF208")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF208");
            ok = false;
        }
        return ok;
    }

}
```

Listing D.58: AIF341.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;

/**
 * Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar
 * jaringan dan aplikasinya di kehidupan sehari-hari. Mahasiswa
 * dikenalkan dengan teknologi-teknologi terbaru di bidang jaringan,
 * sehingga mahasiswa memiliki pengetahuan yang dapat digunakan
 * dalam kehidupan sehari-hari. Selain itu mahasiswa juga
 * diperkenalkan dengan NetAcad, sebuah layanan dari Cisco yang
 * dapat digunakan untuk memenuhi segala macam kebutuhan terkait
 * dengan Cisco Academy.
 * @author Chandra Wijaya, ST., MT. (chandraw@unpar.ac.id)
 *
 */
@InfoMataKuliah(nama = "Administrasi Jaringan Komputer 1", sks = 3)
public class AIF341 extends MataKuliah implements HasPraktikum {

}
```

Listing D.59: AIF342.java

```
package id.ac.unpar.siamodels.matakuliah;

import java.util.List;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
```

```java

@InfoMataKuliah(nama = "Administrasi Jaringan Komputer 2", sks = 3)
public class AIF342 extends MataKuliah implements HasPraktikum, HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasLulusKuliah("AIF341")) {
            reasonsContainer.add("Tidak memenuhi prasyarat lulus AIF341");
            return false;
        }
        return true;
    }

}
```

Listing D.60: AIF343.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pemrograman Kompetitif", sks = 3)
public class AIF343 extends MataKuliah {

}
```

Listing D.61: AIF344.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Pemodelan & Simulasi", sks = 3)
public class AIF344 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF102") && !mahasiswa.hasTempuhKuliah("AIF192")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF102 atau AIF192");
            ok = false;
        }
        if (!mahasiswa.hasTempuhKuliah("AMS200")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AMS200");
            ok = false;
        }
        return ok;
    }

}
```

Listing D.62: AIF347.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;

@InfoMataKuliah(nama = "E-Commerce", sks = 2)
public class AIF347 extends MataKuliah {

}
```

Listing D.63: AIF352.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;

@InfoMataKuliah(nama = "Jaringan Syaraf Tiruan", sks = 2)
public class AIF352 extends MataKuliah {

}
```

Listing D.64: AIF358.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Jaringan Komputer Lanjut", sks = 3)
public class AIF358 extends MataKuliah {

}
```

Listing D.65: AIF360.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Pemrograman Berbasis Web Lanjut", sks = 3)
public class AIF360 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF315")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF315");
            return false;
        }
        return true;
    }



}
```

Listing D.66: AIF362.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Sistem & Aplikasi Telematika", sks = 3)
public class AIF362 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF305")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF305");
            return false;
        }
        return true;
    }

}
```

Listing D.67: AIF380.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;


@InfoMataKuliah(nama = "Teori Bahasa & Otomata", sks = 3)
public class AIF380 extends MataKuliah {

}
```

Listing D.68: AIF381.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Analisis Sistem Informasi", sks = 2)
public class AIF381 extends MataKuliah {

}
```

Listing D.69: AIF382.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Gudang Data & Penambangan Data", sks = 3)
public class AIF382 extends MataKuliah {

}
```

Listing D.70: AIF383.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Praktika Grafika Komputer", sks = 1)
public class AIF383 extends MataKuliah {

}
```

Listing D.71: AIF385.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Praktika Pemrograman Berbasis Web", sks = 1)
public class AIF385 extends MataKuliah {

}
```

Listing D.72: AIF386.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Manajemen Proyek Teknologi Informasi", sks = 2)
public class AIF386 extends MataKuliah {

}
```

Listing D.73: AIF387.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pengantar Telekomunikasi", sks = 3)
public class AIF387 extends MataKuliah {

}
```

Listing D.74: AIF388.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Praktika Pemrograman Aplikasi Bergerak", sks = 1)
public class AIF388 extends MataKuliah {

}
```

Listing D.75: AIF389.java

```java

package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Kriptografi", sks = 2)
public class AIF389 extends MataKuliah {

}
```

Listing D.76: AIF401.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Skripsi 1", sks = 4)
public class AIF401 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasLulusKuliah("AIF302")) {
            reasonsContainer.add("Tidak memenuhi prasyarat lulus AIF302");
            ok = false;
        }
        int sksLulus = mahasiswa.calculateSKSLulus();
```

```
21          if (sksLulus < 108) {
22              reasonsContainer.add("SKS Lulus " + sksLulus + ", belum mencapai syarat minimal 108");
23              return false;
24          }
25          return ok;
26      }
27
28  }
```

Listing D.77: AIF402.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import id.ac.unpar.siamodels.Mahasiswa;
4   import id.ac.unpar.siamodels.MataKuliah;
5   import id.ac.unpar.siamodels.InfoMataKuliah;
6   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
7
8   import java.util.List;
9
10  @InfoMataKuliah(nama = "Skripsi 2", sks = 6)
11  public class AIF402 extends MataKuliah implements HasPrasyarat {
12
13      @Override
14      public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
15          if (mahasiswa.hasLulusKuliah("AIF401")) {
16              return true;
17          } else if (mahasiswa.calculateSKSLulus() >= 124) {
18              reasonsContainer.add("CATATAN: Mahasiswa harus mengambil juga AIF401 (tempuh bersama)");
19              return true;
20          } else {
21              reasonsContainer.add("Harus sudah mengambil AIF401 atau lulus 124 SKS");
22              return false;
23          }
24      }
25  }
```

Listing D.78: AIF403.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import java.util.List;
4
5   import id.ac.unpar.siamodels.Mahasiswa;
6   import id.ac.unpar.siamodels.MataKuliah;
7   import id.ac.unpar.siamodels.InfoMataKuliah;
8   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
9   /**
10   * 1. Memberikan wawasan kepada mahasiswa tentang kemunculan dan pemanfaatan teknologi baru,
11   * khususnya yang berkaitan dengan komputer, dan dampaknya terhadap masyarakat luas.
12   * 2. Memberikan kesadaran dan panduan bersikap kepada mahasiswa dalam menghadapi gejolak yang
13   * disebabkan oleh munculnya teknologi baru, khususnya yang berkaitan dengan komputer.
14   *
15   * @author Oerip S. Santosa (oerip@unpar.ac.id)
16   */
17  @InfoMataKuliah(nama = "Komputer & Masyarakat", sks = 2)
18  public class AIF403 extends MataKuliah implements HasPrasyarat {
19
20          @Override
21          public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
22              int sksLulus = mahasiswa.calculateSKSLulus();
23              if (sksLulus < 70) {
24                  reasonsContainer.add("SKS Lulus " + sksLulus + ", belum mencapai syarat minimal 70");
25                  return false;
26              }
27              return true;
28          }
29  }
```

Listing D.79: AIF405.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import id.ac.unpar.siamodels.Mahasiswa;
4   import id.ac.unpar.siamodels.MataKuliah;
5   import id.ac.unpar.siamodels.InfoMataKuliah;
6   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
7   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
8
9   import java.util.List;
10
11  /**
12   * Mata kuliah ini merupakan lanjutan dari Projek Sistem Informasi 1 dan
13   * memberikan kesempatan bagi mahasiswa untuk melanjutkan/mengembangkan
14   * perancangan sitem pada organisasi studi kasus, mengimplementasikan rancangan
15   * dan melakukan pengujian perangkat lunak;
16   *
17   * @author Veronica S. Moertini (moertini@unpar.ac.id)
18   */
19
20  @InfoMataKuliah(nama = "Proyek Sistem Informasi 2", sks = 3)
21  public class AIF405 extends MataKuliah implements HasPrasyarat, HasPraktikum {
22
23      @Override
```

```
24      public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
25          boolean ok = true;
26          if (!mahasiswa.hasTempuhKuliah("AIF304")) {
27              reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF304");
28              ok = false;
29          }
30          return ok;
31      }
32
33  }
```

Listing D.80: AIF438.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import id.ac.unpar.siamodels.Mahasiswa;
4   import id.ac.unpar.siamodels.MataKuliah;
5   import id.ac.unpar.siamodels.InfoMataKuliah;
6   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
7
8   import java.util.List;
9
10  /**
11   * Mata kuliah ini: Memperkenalkan karakteristik dan teknik visualisasi dari
12   * berbagai jenis data yang dapat dianalisis dengan teknik-teknik data mining;
13   * mempelajari teknik-teknik penyiapan data untuk berbagai jenis data dan teknik
14   * data mining; mempraktekkan teknik-teknik penyiapan data untuk menganalisis
15   * data nyata/simulasi dengan memanfaatkan perangkat lunak aplikasi.
16   *
17   * @author Veronica S. Moertini (moertini@unpar.ac.id)
18   */
19
20  @InfoMataKuliah(nama = "Penambangan Data", sks = 3)
21  public class AIF438 extends MataKuliah implements HasPrasyarat {
22
23      @Override
24      public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
25          boolean ok = true;
26          if (!mahasiswa.hasLulusKuliah("AIF102") && !mahasiswa.hasLulusKuliah("AIF192")) {
27              reasonsContainer.add("Tidak memenuhi prasyarat lulus AIF102 atau AIF192");
28              ok = false;
29          }
30          return ok;
31      }
32
33  }
```

Listing D.81: AIF441.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import java.util.List;
4
5   import id.ac.unpar.siamodels.InfoMataKuliah;
6   import id.ac.unpar.siamodels.Mahasiswa;
7   import id.ac.unpar.siamodels.MataKuliah;
8   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
9   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
10
11  /**
12   * Mata kuliah ini memperkenalkan kepada mahasiswa konsep jaringan lanjut
13   * terutama di layer data link dan layer network. Materi utama dari mata kuliah
14   * ini adalah pengembangan jaringan dan pengenalan fungsi-fungsi yang terdapat
15   * pada alat jaringan Cisco yang berkaitan dengan layer 2 dan layer 3.
16   *
17   * @author Chandra Wijaya, ST., MT. (chandraw@unpar.ac.id)
18   */
19
20  @InfoMataKuliah(nama = "Administrasi Jaringan Komputer 3", sks = 3)
21  public class AIF441 extends MataKuliah implements HasPraktikum, HasPrasyarat {
22
23      @Override
24      public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
25          if (!mahasiswa.hasLulusKuliah("AIF342")) {
26              reasonsContainer.add("Tidak memenuhi prasyarat lulus AIF342");
27              return false;
28          }
29          return true;
30      }
31
32  }
```

Listing D.82: AIF442.java

```
1   package id.ac.unpar.siamodels.matakuliah;
2
3   import java.util.List;
4
5   import id.ac.unpar.siamodels.InfoMataKuliah;
6   import id.ac.unpar.siamodels.Mahasiswa;
7   import id.ac.unpar.siamodels.MataKuliah;
8   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPraktikum;
9   import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
10
```

```java
@InfoMataKuliah(nama = "Administrasi_Jaringan_Komputer_4", sks = 3)
public class AIF442 extends MataKuliah implements HasPraktikum, HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasLulusKuliah("AIF441")) {
            reasonsContainer.add("Tidak_memenuhi_prasyarat_lulus_AIF441");
            return false;
        }
        return true;
    }

}
```

Listing D.83: AIF443.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Matematika_Kombinatorial", sks = 3)
public class AIF443 extends MataKuliah {

}
```

Listing D.84: AIF445.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Metode_Numerik", sks = 3)
public class AIF445 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF102") && !mahasiswa.hasTempuhKuliah("AIF192")) {
            reasonsContainer.add("Tidak_memenuhi_prasyarat_tempuh_AIF102_atau_AIF192");
            ok = false;
        }
        if (!mahasiswa.hasTempuhKuliah("AMS100")) {
            reasonsContainer.add("Tidak_memenuhi_prasyarat_tempuh_AMS100");
            ok = false;
        }
        return ok;
    }

}
```

Listing D.85: AIF446.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Kompresi_Data", sks = 3)
public class AIF446 extends MataKuliah {

}
```

Listing D.86: AIF450.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;

@InfoMataKuliah(nama = "Pengolahan_Citra", sks = 3)
public class AIF450 extends MataKuliah {

}
```

Listing D.87: AIF451.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Audit_Sistem_Informasi", sks = 3)
public class AIF451 extends MataKuliah {

}
```

Listing D.88: AIF453.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;
import java.util.List;

/**
 * Mata kuliah inimemperkenalkan kebutuhan organisasi terhadap sistem business
 * intelligent (BI) dan pemanfaatan BI untuk organisasi; memperkenalkan konsep
 * sistem business intelligent dan komponennya; Mempelajari tenik-teknik
 * analisis data bisnis dan visualisasi hasil analisis; Mempelajari konsep data
 * warehouse dan perancangannya dan fungsi OLAP; Mempraktekkan teknik-teknik
 * analisis data dan visualisasi hasil analisis.
 *
 * @author Veronica S. Moertini (moertini@unpar.ac.id)
 */

@InfoMataKuliah(nama = "Kecerdasan Bisnis", sks = 3)
public class AIF453 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = false;
        if (mahasiswa.hasTempuhKuliah("AIF204") || mahasiswa.hasTempuhKuliah("AIF294")) {
            ok = true;
        }
        if ((mahasiswa.hasTempuhKuliah("AIF102") || !mahasiswa.hasTempuhKuliah("AIF192")) && mahasiswa.calculateIPLulus() > 2.75)
             {
            ok = true;
        }
        if (!ok) {
            reasonsContainer.add("Tidak memenuhi prasyarat ((tempuh AIF204/294) atau (tempuh AIF102/AIF192 & IPK Lulus > 2.75))");
        }
        return ok;
    }

}
```

Listing D.89: AIF455.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Sistem Pendukung Keputusan", sks = 3)
public class AIF455 extends MataKuliah {

}
```

Listing D.90: AIF456.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Strategi Sistem Informasi & Arsitektur Perusahaan Berskala Besar", sks = 3)
public class AIF456 extends MataKuliah {

}
```

Listing D.91: AIF457.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Mata kuliah ini memperkenalkan konsep kewirausahaan dengan memanfaatkan teknologi, khususnya
 * teknologi informasi, sebagai basis usaha dan inovasi produk/jasa; Mempelajari
 * teknik mencari peluang dan merumuskan bidang usaha spesifik yang akan
 * diterjuni; Mempelajari konsep manajemen pemasaran, keuangan dan SDM dalam
 * kaitannya dengan berwira-usaha di bidang TI; Menyusun proposal bisnis untuk
 * berwira-usaha di bidang TI dan mempresentasikannya.
 *
 * @author Veronica S. Moertini (moertini@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Kewirausahaan Berbasis Teknologi", sks = 3)
public class AIF457 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        int sksLulus = mahasiswa.calculateSKSLulus();
        if (sksLulus < 70) {
```

```
            reasonsContainer.add("SKS Lulus " + sksLulus + ", belum mencapai syarat minimal 70");
            return false;
        }
        return ok;
    }

}
```

Listing D.92: AIF458.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Pengendalian & Audit Teknologi Informasi", sks = 3)
public class AIF458 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        if (!mahasiswa.hasTempuhKuliah("AIF315")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF315");
            return false;
        }
        return true;
    }

}
```

Listing D.93: AIF459.java

```
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Adaministrasi Basis Data", sks = 3)
public class AIF459 extends MataKuliah {

}
```

Listing D.94: AIF460.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;

@InfoMataKuliah(nama = "Manajemen Pengetahuan", sks = 2)
public class AIF460 extends MataKuliah {

}
```

Listing D.95: AIF461.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pencarian & Temu Kembali Informasi", sks = 2)
public class AIF461 extends MataKuliah {

}
```

Listing D.96: AIF462.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Manajemen Proses Bisnis", sks = 3)
public class AIF462 extends MataKuliah {

}
```

Listing D.97: AIF463.java

```
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

```

```java
@InfoMataKuliah(nama = "Jaringan Nirkabel", sks = 3)
public class AIF463 extends MataKuliah {

}
```

Listing D.98: AIF465.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Teknologi Middleware", sks = 3)
public class AIF465 extends MataKuliah {

}
```

Listing D.99: AIF468.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Teknologi Multimedia", sks = 3)
public class AIF468 extends MataKuliah {

}
```

Listing D.100: AIF469.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

/**
 * Mata kuliah ini mengajarkan kepada mahasiswa teknik-teknik untuk membuat
 * layanan berbasis web. Mahasiswa diperkenalkan dengan standar-standar seperti
 * HTTP, XML, JSON dan diajarkan untuk memanfaatkannya dalam membuat maupun
 * menggunakan layanan pihak ketiga. Dalam kuliah ini, juga akan diperkenalkan
 * minimal satu layanan pihak ketiga yang dapat dimanfaatkan mahasiswa, seperti
 * Google Places Web Service.
 * @author Pascal (pascal@unpar.ac.id)
 */
@InfoMataKuliah(nama = "Layanan Berbasis Web", sks = 3)
public class AIF469 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        boolean ok = true;
        if (!mahasiswa.hasTempuhKuliah("AIF305")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF305");
            ok = false;
        }
        if (!mahasiswa.hasTempuhKuliah("AIF315")) {
            reasonsContainer.add("Tidak memenuhi prasyarat tempuh AIF315");
            ok = false;
        }
        return ok;
    }

}
```

Listing D.101: AIF480.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pemrograman Sistem", sks = 2)
public class AIF480 extends MataKuliah {

}
```

Listing D.102: AIF483.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Teknik Kompilasi", sks = 2)
public class AIF483 extends MataKuliah {

}
```

Listing D.103: AIF484.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Kewirausahaan", sks = 3)
public class AIF484 extends MataKuliah {

}
```

Listing D.104: AIF486.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Keamanan Jaringan", sks = 3)
public class AIF486 extends MataKuliah {

}
```

Listing D.105: AKS122.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Logika Formal Dasar", sks = 2)
public class AKS122 extends MataKuliah {

}
```

Listing D.106: AKS124.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Logika Informatika", sks = 3)
public class AKS124 extends MataKuliah {

}
```

Listing D.107: AMS100.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;
/**
 * Sistem Bilangan, Fungsi, Limit dan Kekontinuan Fungsi, Turunan, Integral,
 * Penggunaan Integral, Sistem Persamaan Linear, Determinan, Vektor, Nilai dan
 * Vektor Eigen.
 * @author Taufik Limansyah, S.Si, MT.
 */
@InfoMataKuliah(nama = "Matematika Informatika", sks = 4)
public class AMS100 extends MataKuliah  {

}
```

Listing D.108: AMS190.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * @deprecated Mata kuliah tidak dibuka lagi.
 */
@InfoMataKuliah(nama = "Matematika Informatika", sks = 3)
public class AMS190 extends MataKuliah {

}
```

Listing D.109: AMS191.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * @deprecated Mata kuliah tidak dibuka lagi.
 */
@InfoMataKuliah(nama = "Kalkulus", sks = 4)
public class AMS191 extends MataKuliah {

}
```

Listing D.110: AMS200.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Probabilitas dan Statistika", sks = 3)
public class AMS200 extends MataKuliah {

}
```

Listing D.111: AMS290.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;


@InfoMataKuliah(nama = "Aljabar & Linear Matriks", sks = 3)
public class AMS290 extends MataKuliah {

}
```

Listing D.112: APS182.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Fisika Dasar", sks = 3)
public class APS182 extends MataKuliah {

}
```

Listing D.113: APS302.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Dunia Digital Dan Sains", sks = 2)
public class APS302 extends MataKuliah {

}
```

Listing D.114: APS309.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 *
 * APS302 atau APS309 ?
 *
 */
@InfoMataKuliah(nama = "Dunia Digital Dan Sains", sks = 2)
public class APS309 extends MataKuliah {

}
```

Listing D.115: APS402.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.Mahasiswa;
import id.ac.unpar.siamodels.MataKuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.matakuliah.interfaces.HasPrasyarat;

import java.util.List;

@InfoMataKuliah(nama = "Etika Profesi", sks = 2)
public class APS402 extends MataKuliah implements HasPrasyarat {

    @Override
    public boolean checkPrasyarat(Mahasiswa mahasiswa, List<String> reasonsContainer) {
        int sksLulus = mahasiswa.calculateSKSLulus();
        if (sksLulus < 90) {
            reasonsContainer.add("SKS Lulus " + sksLulus + ", belum mencapai syarat minimal 90");
            return false;
        }
        return true;
    }

}
```

Listing D.116: EAA101.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Akuntansi Keuangan Dasar I", sks = 2)
public class EAA101 extends MataKuliah {

}
```

Listing D.117: EAA102.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Akuntansi Keuangan Dasar II", sks = 2)
public class EAA102 extends MataKuliah {

}
```

Listing D.118: ESA101.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Akuntansi Keuangan Dasar", sks = 4)
public class ESA101 extends MataKuliah {

}
```

Listing D.119: ESM101.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pengantar Bisnis", sks = 3)
public class ESM101 extends MataKuliah {

}
```

Listing D.120: ESM105.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Manajemen", sks = 3)
public class ESM105 extends MataKuliah {

}
```

Listing D.121: ESM201.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Perilaku Keorganisasian", sks = 3)
public class ESM201 extends MataKuliah {

}
```

Listing D.122: ESM203.java

```java
package id.ac.unpar.siamodels.matakuliah;
import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Pengantar Manajemen Pemasaran", sks = 3)
public class ESM203 extends MataKuliah {

}
```

Listing D.123: ESM204.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Manajemen Keuangan I", sks = 3)
public class ESM204 extends MataKuliah {

}
```

Listing D.124: IIE103.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Statistika Deskriptif", sks = 3)
public class IIE103 extends MataKuliah {

}
```

Listing D.125: IIE207.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Proses Produksi Pembentukan", sks = 2)
public class IIE207 extends MataKuliah {

}
```

Listing D.126: IIE210.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Perancangan Sistem Kerja Dan Ergonomi", sks = 2)
public class IIE210 extends MataKuliah {

}
```

Listing D.127: IIE214.java

```java
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Ekologi Industri", sks = 2)
public class IIE214 extends MataKuliah {

}
```

Listing D.128: MKU001.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata Kuliah Pendidikan Pancasila berupaya menelaah/mengkaji berbagai fenomena kehidupan
 * bangsa dan Negara Indonesia sebagai sebuah ruang publik dengan menggunakan pendekatan
 * hermeneutika (filsafat) dan pendidikan nilai (pedagogik). Dengan bantuan hermenutika
 * mahasiswa diajak berpikir kritis terhadap segala bentuk ideologisme Pancasila dan melalui
 * pendidikan nilai mahasiswa dilatih untuk memiliki nilai Pancasila. Nilai pengembangan diri
 * intra-personal dan relasi inter-personal dapat tertanam melalui pendidikan Pancasila yang
 * tujuannya adalah membangun kepribadian (character building) manusia Indonesia yang utuh,
 * baik menyangkut aspek kognitif, afektif, maupun psikomotor. Dengan demikian, Pendidikan
 * Pancasila mengajak mahasiswa menilai realitas ruang publik sehari-hari secara mandiri
 * dengan panduan nilai-nilai etis Pancasila.
 *
 */
@InfoMataKuliah(nama = "Pendidikan Pancasila", sks = 2)
public class MKU001 extends MataKuliah {

}
```

Listing D.129: MKU002.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Pendidikan Kewarganegaraan menjelaskan pentingnya pemahaman tentang identitas nasional
 * Indonesia, hak dan kewajiban warga negara Indonesia serta hubungannya dengan hak dan
 * kewajiban asasi manusia. Materi kuliah mencakup juga wawasan nusantara, ketahanan nasional,
 * politik dan strategi nasional, serta implementasinya dalam kehidupan bermasyarakat, berbangsa
 * dan bernegara kesatuan Republik Indonesia.
 *
 */
@InfoMataKuliah(nama = "Pendidikan Kewarganegaraan", sks = 2)
public class MKU002 extends MataKuliah {

}
```

Listing D.130: MKU003.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah ini membentuk karakteristik mahasiswa sebagai manusia yang memiliki religiusitas
 * melalui pendalaman akan makna agama dan beragama, mendeteksi dinamika Wahyu Tuhan dan iman
 * mereka, memahami relasi dengan Tuhan dan sesama, mengenal makna keselamatan dalam konteks
 * Kerajaan Allah, dan mampu menyatakan ajaran Gereja dalam pelayanan terhadap orang miskin dan
 * terlantar.
 *
 */
@InfoMataKuliah(nama = "Pendidikan Agama (Katolik)", sks = 2)
public class MKU003 extends MataKuliah {

}
```

Listing D.131: MKU004.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Fenomenologi Agama merupakan bagian yang tak terpisahkan dari kajian filosofis, kritis,
 * rasional, dan obyektif mengenai substansi ajaran agama. Fenomenologi merupakan sebuah
 * disiplin ilmu yang secara kritis-rasional mengkaji fenomena dan dinamika kehidupan manusia
 * beragama, dari upaya menjadikan Tuhan sebagai tujuan sesembahan sampai menempatkan Tuhan
 * sebagai instrumen legitimasi untuk melakukan tindakan yang justru bertolak belakang dengan
 * kehendak Tuhan yang disembah. Sehubungan dengan itu, kritik konstruktif terhadap perilaku
 * manusia beragama menjadi salah satu poin utama dalam mata kuliah ini. Kesediaan untuk
 * melakukan otoritik terhadap agama sendiri erat terkait dengan upaya menemukan kembali nilai
 * sejati agama atau otentisitas hidup beragama.
 *
 */
@InfoMataKuliah(nama = "Pendidikan Agama (Fenomenologi)", sks = 2)
public class MKU004 extends MataKuliah {

}
```

Listing D.132: MKU008.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mendalami perilaku sehari-hari yang baik dalam bermasyarakat.
 *
 */
@InfoMataKuliah(nama = "Etika", sks = 2)
public class MKU008 extends MataKuliah{

}
```

Listing D.133: MKU009.java

```java
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Tujuan dari mata kuliah ini adalah untuk mendalami keterampilan berbahasa Indonesia, agar
 * mampu mengkomunikasikan hasil pemikiran serta meningkatkan keterampilan dalam menyusun karya
 * ilmiah. Mata kuliah Bahasa Indonesia ini dimulai dengan mempelajari penulisan kata baku dan
 * non baku serta pengungkapan pikiran dengan pungtuasi yang benar. Selanjutnya dipelajari
```

```
 * penyusunan kalimat yang baku serta menghubungkan kalimat-kalimat yang padu dalam menuangkan
 * gagasan dalam sebuah paragraf. Selain itu, dalam matakuliah ini dipelajari cara menyusun
 * surat dinas yang jelas dan komunikatif. Di akhir kuliah ini, mahasiswa diberi tugas
 * penyusunan makalah dengan benar.
 *
 */
@InfoMataKuliah(nama = "Bahasa_Indonesia", sks = 2)
public class MKU009 extends MataKuliah{

}
```

Listing D.134: MKU010.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah ini difokuskan pada pemahaman sumber referensi dalam Bahasa Inggris dan
 * pengembangan kosakata Bahasa Inggris (vocabularies). Hampir keseluruhan waktu perkuliahan
 * didedikasikan untuk menjelaskan metode mengekstraksi isi bacaan secara tepat dan melatih
 * mahasiswa untuk menerapkan metode tersebut seraya menambah kosakata-kosakata baru.
 * Mahasiswa juga dilatih untuk mempresentasikan hasil pemahamannya akan isi bahan bacaan.
 *
 */
@InfoMataKuliah(nama = "Bahasa_Inggris", sks = 2)
public class MKU010 extends MataKuliah{

}
```

Listing D.135: MKU011.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Mata kuliah estetika memberi pemahaman konseptual filosofis "seni" dalam khasanah keilmuan,
 * pembentukan kesadaran ekologis juga dalam proses pembudayaan dan peradaban. Mata kuliah ini
 * akan menjadi fondasi bagi mahasiswa untuk memahami dan mempraktekkan seni dari sudut pandang
 * filsafat, sejarah, kultural, dan global. Melalui mata kuliah ini, mahasiswa mempelajari
 * mengenai dunia manusia (manusia dan pikirannya), pluralitas dan relativitas seni, serta
 * aliran-aliran seni rupa Barat.
 *
 */
@InfoMataKuliah(nama = "Estetika", sks = 2)
public class MKU011 extends MataKuliah{

}
```

Listing D.136: MKU012.java

```
package id.ac.unpar.siamodels.matakuliah;

import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

/**
 * Perkuliahan logika ditujukan untuk memberikan dasar-dasar ketrampilan berpikir rasional dan
 * sistematik. Isinya mencakup ketrampilan berpikir deduktif dan induktif, seperti silogisme,
 * argumen analogikal dan generalisasi induktif. Pembahasan teoretis disertai pula dengan
 * pelatihan praktis yang diarahkan pada proses berpikir. Untuk menajamkan kemampuan berpikir
 * tersebut, mahasiswa dilatih pula mengidentifikasi kerancuan-kerancuan (fallacies) yang sering
 * dijumpai baik dalam kehidupan sehari-hari maupun dalam konteks akademik.
 *
 */
@InfoMataKuliah(nama = "Logika", sks = 2)
public class MKU012 extends MataKuliah{

}
```

Listing D.137: SIR104.java

```
package id.ac.unpar.siamodels.matakuliah;


import id.ac.unpar.siamodels.InfoMataKuliah;
import id.ac.unpar.siamodels.MataKuliah;

@InfoMataKuliah(nama = "Bahasa_Jepang", sks = 3)
public class SIR104 extends MataKuliah {

}
```

D.2 Hasil Latex

Listing D.138: siamodels.tex

```
\documentclass{article}
\begin{document}
\begin{enumerate}
\item \texttt{InfoMataKuliah}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public int sks()}

Jumlah bobot sks dari mata kuliah ini

\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: jumlah bobot sks

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

\item \texttt{public String nama()}

Nama mata kuliah ini

\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: nama mata kuliah

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

\end{itemize}
\item \texttt{MataKuliahFactory}

Kelas yang bertugas membuat kelas mata kuliah, dan menyimpannya untuk bisa
 digunakan kemudian (untuk hemat memori).

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{String DEFAULT\_MATAKULIAH\_PACKAGE} - Lokasi package untuk daftar mata kuliah
\item \texttt{MataKuliahFactory instance} - Singleton instance to factory.
\item \texttt{SortedMap mataKuliahCache} - Singleton instances untuk mata kuliah.
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public static MataKuliahFactory getInstance()}



\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public MataKuliah createMataKuliah(java.lang.String kode, int sks, java.lang.String nama)}

Membuat baru atau mendapatkan mata kuliah, jika memiliki informasi
 nama dan jumlah SKS.

\textbf{Parameter:}
\begin{itemize}
\item \texttt{String kode} -
kode mata kuliah
\item \texttt{int sks} -
jumlah SKS
\item \texttt{String nama} -
nama mata kuliah
\end{itemize}
\textbf{Return Value}: objek mata kuliah

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

\item \texttt{public MataKuliah createMataKuliah(java.lang.String kode)}

Membuat baru atau mendapatkan mata kuliah, jika tidak memiliki informasi
 nama dan jumlah SKS.

\textbf{Parameter:}
\begin{itemize}
\item \texttt{String kode} -
kode mata kuliah
\end{itemize}
\textbf{Return Value}: objek mata kuliah

\textbf{Exception}: IllegalStateException
             jika sks dan tidak sesuai dengan yang ada di kode

\end{itemize}
\item \texttt{Semester}



```

```
Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{Semester UNKNOWN5} -
\item \texttt{Semester TRANSFER} -
\item \texttt{Semester PENDEK} -
\item \texttt{Semester GANJIL} -
\item \texttt{Semester GENAP} -
\item \texttt{int order} -
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public static Semester values()}



\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public static Semester valueOf(java.lang.String name)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{String name} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public static Semester fromString(java.lang.String text)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{String text} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{ int getOrder()}



\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\end{itemize}
\item \texttt{TahunSemester implements Comparable}

Menyimpan konstanta untuk semester beserta tahunnya di UNPAR.

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{String kodeTahunSemester} - Kode semester 3 dijit, sesuai DPS:
 <ul>
   <li>2 dijit pertama berupa tahun, 2 dijit terakhir</li>
   <li>dijit terakhir: 1 untuk ganjil, 2 untuk genap, 4 untuk pendek.
 </ul>
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public Semester getSemester()}



\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public int getTahun()}



\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{private static void validateKodeSemester(java.lang.String kodeTahunSemester)}
```

```
197
198
199
200 \textbf{Parameter:}
201 \begin{itemize}
202 \item \texttt{String kodeTahunSemester} -
203 \end{itemize}
204 \textbf{Return Value}: Tidak memiliki \textit{return value}
205
206 \textbf{Exception}: Tidak memiliki \textit{exception}
207
208 \item \texttt{public String getKodeDPS()}
209
210 Mendapatkan kode tahun/semester sesuai aturan di DPS.
211
212 \textbf{Parameter:}
213 \begin{itemize}
214 \item Tidak memiliki parameter \textit{method}
215 \end{itemize}
216 \textbf{Return Value}: kode tahun/semester sesuai aturan di DPS.
217
218 \textbf{Exception}: Tidak memiliki \textit{exception}
219
220 \item \texttt{public int compareTo(id.ac.unpar.siamodels.TahunSemester o)}
221
222
223
224 \textbf{Parameter:}
225 \begin{itemize}
226 \item \texttt{TahunSemester o} -
227 \end{itemize}
228 \textbf{Return Value}: Tidak memiliki \textit{return value}
229
230 \textbf{Exception}: Tidak memiliki \textit{exception}
231
232 \textbf{Override}: \texttt{compareTo} dari kelas \texttt{Object}
233
234 \item \texttt{public boolean equals(java.lang.Object arg0)}
235
236
237
238 \textbf{Parameter:}
239 \begin{itemize}
240 \item \texttt{Object arg0} -
241 \end{itemize}
242 \textbf{Return Value}: Tidak memiliki \textit{return value}
243
244 \textbf{Exception}: Tidak memiliki \textit{exception}
245
246 \item \texttt{public String toString()}
247
248
249
250 \textbf{Parameter:}
251 \begin{itemize}
252 \item Tidak memiliki parameter \textit{method}
253 \end{itemize}
254 \textbf{Return Value}: Tidak memiliki \textit{return value}
255
256 \textbf{Exception}: Tidak memiliki \textit{exception}
257
258 \end{itemize}
259 \item \texttt{Mahasiswa}
260
261
262
263 Atribut yang dimiliki kelas ini adalah sebagai berikut.
264 \begin{itemize}
265 \item \texttt{String npm} -
266 \item \texttt{String nama} -
267 \item \texttt{List riwayatNilai} -
268 \item \texttt{URL photoURL} -
269 \item \texttt{List jadwalKuliahList} -
270 \item \texttt{SortedMap nilaiTOEFL} -
271 \end{itemize}
272 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
273 \begin{itemize}
274 \item \texttt{public String getNama()}
275
276
277
278 \textbf{Parameter:}
279 \begin{itemize}
280 \item Tidak memiliki parameter \textit{method}
281 \end{itemize}
282 \textbf{Return Value}: Tidak memiliki \textit{return value}
283
284 \textbf{Exception}: Tidak memiliki \textit{exception}
285
286 \item \texttt{public void setNama(java.lang.String nama)}
287
288
289
290 \textbf{Parameter:}
291 \begin{itemize}
292 \item \texttt{String nama} -
293 \end{itemize}
294 \textbf{Return Value}: Tidak memiliki \textit{return value}
295
```

```
296 \textbf{Exception}: Tidak memiliki \textit{exception}
297
298 \item \texttt{public String getNpm()}
299
300
301
302 \textbf{Parameter:}
303 \begin{itemize}
304 \item Tidak memiliki parameter \textit{method}
305 \end{itemize}
306 \textbf{Return Value}: Tidak memiliki \textit{return value}
307
308 \textbf{Exception}: Tidak memiliki \textit{exception}
309
310 \item \texttt{public URL getPhotoURL()}
311
312
313
314 \textbf{Parameter:}
315 \begin{itemize}
316 \item Tidak memiliki parameter \textit{method}
317 \end{itemize}
318 \textbf{Return Value}: Tidak memiliki \textit{return value}
319
320 \textbf{Exception}: Tidak memiliki \textit{exception}
321
322 \item \texttt{public void setPhotoURL(java.net.URL photoURL)}
323
324
325
326 \textbf{Parameter:}
327 \begin{itemize}
328 \item \texttt{URL photoURL} -
329 \end{itemize}
330 \textbf{Return Value}: Tidak memiliki \textit{return value}
331
332 \textbf{Exception}: Tidak memiliki \textit{exception}
333
334 \item \texttt{public List getJadwalKuliahList()}
335
336
337
338 \textbf{Parameter:}
339 \begin{itemize}
340 \item Tidak memiliki parameter \textit{method}
341 \end{itemize}
342 \textbf{Return Value}: Tidak memiliki \textit{return value}
343
344 \textbf{Exception}: Tidak memiliki \textit{exception}
345
346 \item \texttt{public void setJadwalKuliahList(java.util.List jadwalKuliahList)}
347
348
349
350 \textbf{Parameter:}
351 \begin{itemize}
352 \item \texttt{java.util.List jadwalKuliahList} -
353 \end{itemize}
354 \textbf{Return Value}: Tidak memiliki \textit{return value}
355
356 \textbf{Exception}: Tidak memiliki \textit{exception}
357
358 \item \texttt{public String getEmailAddress()}
359
360
361
362 \textbf{Parameter:}
363 \begin{itemize}
364 \item Tidak memiliki parameter \textit{method}
365 \end{itemize}
366 \textbf{Return Value}: Tidak memiliki \textit{return value}
367
368 \textbf{Exception}: Tidak memiliki \textit{exception}
369
370 \item \texttt{public List getRiwayatNilai()}
371
372
373
374 \textbf{Parameter:}
375 \begin{itemize}
376 \item Tidak memiliki parameter \textit{method}
377 \end{itemize}
378 \textbf{Return Value}: Tidak memiliki \textit{return value}
379
380 \textbf{Exception}: Tidak memiliki \textit{exception}
381
382 \item \texttt{public SortedMap getNilaiTOEFL()}
383
384
385
386 \textbf{Parameter:}
387 \begin{itemize}
388 \item Tidak memiliki parameter \textit{method}
389 \end{itemize}
390 \textbf{Return Value}: Tidak memiliki \textit{return value}
391
392 \textbf{Exception}: Tidak memiliki \textit{exception}
393
394 \item \texttt{public void setNilaiTOEFL(java.util.SortedMap nilaiTOEFL)}
```

```
395
396
397
398 \textbf{Parameter:}
399 \begin{itemize}
400 \item \texttt{java.util.SortedMap nilaiTOEFL} -
401 \end{itemize}
402 \textbf{Return Value}: Tidak memiliki \textit{return value}
403
404 \textbf{Exception}: Tidak memiliki \textit{exception}
405
406 \item \texttt{public double calculateIPKLulus()}
407
408 Menghitung IPK mahasiswa sampai saat ini, dengan aturan:
409  <ul>
410    <li>Kuliah yang tidak lulus tidak dihitung
411    <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
412  </ul>
413  Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
414
415 \textbf{Parameter:}
416 \begin{itemize}
417 \item Tidak memiliki parameter \textit{method}
418 \end{itemize}
419 \textbf{Return Value}: IPK Lulus, atau  DoubleNaN jika belum mengambil satu kuliah pun.
420
421 \textbf{Exception}: Tidak memiliki \textit{exception}
422
423 \item \texttt{public double calculateIPLulus()}
424
425 Menghitung IP mahasiswa sampai saat ini, dengan aturan:
426  <ul>
427    <li>Kuliah yang tidak lulus tidak dihitung
428    <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
429  </ul>
430  Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
431
432 \textbf{Parameter:}
433 \begin{itemize}
434 \item Tidak memiliki parameter \textit{method}
435 \end{itemize}
436 \textbf{Return Value}: IPK Lulus, atau  DoubleNaN jika belum mengambil satu kuliah pun.
437
438 \textbf{Exception}: Tidak memiliki \textit{exception}
439
440 \item \texttt{public double calculateIPTempuh(boolean lulusSaja)}
441
442 Menghitung IP mahasiswa sampai saat ini, dengan aturan:
443  <ul>
444    <li>Perhitungan kuliah yang tidak lulus ditentukan parameter
445    <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
446  </ul>
447
448 \textbf{Parameter:}
449 \begin{itemize}
450 \item \texttt{boolean lulusSaja} -
451 set true jika ingin membuang mata kuliah tidak lulus, false jika ingin semua (sama dengan "IP N. Terbaik" di DPS)
452  Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
453 \end{itemize}
454 \textbf{Return Value}: IPK Lulus, atau  DoubleNaN jika belum mengambil satu kuliah pun.
455
456 \textbf{Exception}: Tidak memiliki \textit{exception}
457
458 \item \texttt{public double calculateIPKumulatif()}
459
460 Menghitung IP Kumulatif mahasiswa sampai saat ini, dengan aturan:
461  <ul>
462    <li>Jika pengambilan beberapa kali, diambil semua.
463  </ul>
464  Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
465
466 \textbf{Parameter:}
467 \begin{itemize}
468 \item Tidak memiliki parameter \textit{method}
469 \end{itemize}
470 \textbf{Return Value}: IPK Lulus, atau  DoubleNaN jika belum mengambil satu kuliah pun.
471
472 \textbf{Exception}: Tidak memiliki \textit{exception}
473
474 \item \texttt{public double calculateIPKTempuh(boolean lulusSaja)}
475
476 Menghitung IPK mahasiswa sampai saat ini, dengan aturan:
477  <ul>
478    <li>Perhitungan kuliah yang tidak lulus ditentukan parameter
479    <li>Jika pengambilan beberapa kali, diambil <em>nilai terbaik</em>.
480  </ul>
481
482 \textbf{Parameter:}
483 \begin{itemize}
484 \item \texttt{boolean lulusSaja} -
485 set true jika ingin membuang mata kuliah tidak lulus
486  Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
487 \end{itemize}
488 \textbf{Return Value}: IPK Lulus, atau  DoubleNaN jika belum mengambil satu kuliah pun.
489
490 \textbf{Exception}: Tidak memiliki \textit{exception}
491
492 \item \texttt{public double calculateIPS()}
493
```

```
Menghitung IPS semester terakhir sampai saat ini, dengan aturan:
 <ul>
   <li>Kuliah yang tidak lulus <em>dihitung</em>.
 </ul>
 Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!

\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: nilai IPS sampai saat ini

\textbf{Exception}: ArrayIndexOutOfBoundsException jika belum ada nilai satupun

\item \texttt{public int calculateSKSLulus()}

Menghitung jumlah SKS lulus mahasiswa saat ini.
 Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!

\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: SKS Lulus

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

\item \texttt{public int calculateSKSTempuh(boolean lulusSaja)}

Menghitung jumlah SKS tempuh mahasiswa saat ini.
 Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!

\textbf{Parameter:}
\begin{itemize}
\item \texttt{boolean lulusSaja} -
set true jika ingin membuang SKS tidak lulus
\end{itemize}
\textbf{Return Value}: SKS tempuh

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

\item \texttt{public Set calculateTahunSemesterAktif()}

Mendapatkan seluruh tahun semester di mana mahasiswa ini tercatat
 sebagai mahasiswa aktif, dengan strategi memeriksa riwayat nilainya.
 Jika ada satu nilai saja pada sebuah tahun semester, maka dianggap
 aktif pada semester tersebut.

\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: kumpulan tahun semester di mana mahasiswa ini aktif

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

\item \texttt{public boolean hasLulusKuliah(java.lang.String kodeMataKuliah)}

Memeriksa apakah mahasiswa ini sudah lulus mata kuliah tertentu. Kompleksitas O(n).
 Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah mengandung nilai per mata kuliah!
 Note: jika yang dimiliki adalah MataKuliah, gunakanlah MataKuliahgetKode().

\textbf{Parameter:}
\begin{itemize}
\item \texttt{String kodeMataKuliah} -
kode mata kuliah yang ingin diperiksa kelulusannya.
\end{itemize}
\textbf{Return Value}: true jika sudah pernah mengambil dan lulus, false jika belum

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

\item \texttt{public boolean hasTempuhKuliah(java.lang.String kodeMataKuliah)}

Memeriksa apakah mahasiswa ini sudah pernah menempuh mata kuliah tertentu. Kompleksitas O(n).
 Sebelum memanggil method ini, \texttt{getRiwayatNilai()} harus sudah ada isinya!
 Note: jika yang dimiliki adalah MataKuliah, gunakanlah MataKuliahgetKode().

\textbf{Parameter:}
\begin{itemize}
\item \texttt{String kodeMataKuliah} -
kode mata kuliah yang ingin diperiksa.
\end{itemize}
\textbf{Return Value}: true jika sudah pernah mengambil, false jika belum

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

\item \texttt{public int getTahunAngkatan()}

Mendapatkan tahun angkatan mahasiswa ini, berdasarkan NPM nya

\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: tahun angkatan

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

\item \texttt{public String toString()}
```

```
593
594
595
596 \textbf{Parameter:}
597 \begin{itemize}
598 \item Tidak memiliki parameter \textit{method}
599 \end{itemize}
600 \textbf{Return Value}: Tidak memiliki \textit{return value}
601
602 \textbf{Exception}: Tidak memiliki \textit{exception}
603
604 \end{itemize}
605 \item \texttt{Mahasiswa.Nilai}
606
607 Merepresentasikan nilai yang ada di riwayat nilai mahasiswa
608
609 Atribut yang dimiliki kelas ini adalah sebagai berikut.
610 \begin{itemize}
611 \item \texttt{TahunSemester tahunSemester} - Tahun dan Semester kuliah ini diambil
612 \item \texttt{MataKuliah mataKuliah} - Mata kuliah yang diambil
613 \item \texttt{Character kelas} - Kelas kuliah
614 \item \texttt{Double nilaiART} - Nilai ART
615 \item \texttt{Double nilaiUTS} - Nilai UTS
616 \item \texttt{Double nilaiUAS} - Nilai UAS
617 \item \texttt{Character nilaiAkhir} - Nilai Akhir
618 \end{itemize}
619 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
620 \begin{itemize}
621 \item \texttt{public MataKuliah getMataKuliah()}
622
623
624
625 \textbf{Parameter:}
626 \begin{itemize}
627 \item Tidak memiliki parameter \textit{method}
628 \end{itemize}
629 \textbf{Return Value}: Tidak memiliki \textit{return value}
630
631 \textbf{Exception}: Tidak memiliki \textit{exception}
632
633 \item \texttt{public Character getKelas()}
634
635
636
637 \textbf{Parameter:}
638 \begin{itemize}
639 \item Tidak memiliki parameter \textit{method}
640 \end{itemize}
641 \textbf{Return Value}: Tidak memiliki \textit{return value}
642
643 \textbf{Exception}: Tidak memiliki \textit{exception}
644
645 \item \texttt{public Double getNilaiART()}
646
647
648
649 \textbf{Parameter:}
650 \begin{itemize}
651 \item Tidak memiliki parameter \textit{method}
652 \end{itemize}
653 \textbf{Return Value}: Tidak memiliki \textit{return value}
654
655 \textbf{Exception}: Tidak memiliki \textit{exception}
656
657 \item \texttt{public Double getNilaiUTS()}
658
659
660
661 \textbf{Parameter:}
662 \begin{itemize}
663 \item Tidak memiliki parameter \textit{method}
664 \end{itemize}
665 \textbf{Return Value}: Tidak memiliki \textit{return value}
666
667 \textbf{Exception}: Tidak memiliki \textit{exception}
668
669 \item \texttt{public Double getNilaiUAS()}
670
671
672
673 \textbf{Parameter:}
674 \begin{itemize}
675 \item Tidak memiliki parameter \textit{method}
676 \end{itemize}
677 \textbf{Return Value}: Tidak memiliki \textit{return value}
678
679 \textbf{Exception}: Tidak memiliki \textit{exception}
680
681 \item \texttt{public Character getNilaiAkhir()}
682
683 Mengembalikan nilai akhir dalam bentuk huruf (A, B, C, D, ..., atau K)
684
685 \textbf{Parameter:}
686 \begin{itemize}
687 \item Tidak memiliki parameter \textit{method}
688 \end{itemize}
689 \textbf{Return Value}: nilai akhir dalam huruf, atau null jika tidak ada.
690
691 \textbf{Exception}: Tidak memiliki \textit{exception}
```

```
692
693 \item \texttt{public Double getAngkaAkhir()}
694
695 Mendapatkan nilai akhir dalam bentuk angka
696
697 \textbf{Parameter:}
698 \begin{itemize}
699 \item Tidak memiliki parameter \textit{method}
700 \end{itemize}
701 \textbf{Return Value}: nilai akhir dalam angka, atau null jika getNilaiAkhir() mengembalikan 'K' atau null
702
703 \textbf{Exception}: Tidak memiliki \textit{exception}
704
705 \item \texttt{public TahunSemester getTahunSemester()}
706
707
708
709 \textbf{Parameter:}
710 \begin{itemize}
711 \item Tidak memiliki parameter \textit{method}
712 \end{itemize}
713 \textbf{Return Value}: Tidak memiliki \textit{return value}
714
715 \textbf{Exception}: Tidak memiliki \textit{exception}
716
717 \item \texttt{public int getTahunAjaran()}
718
719
720
721 \textbf{Parameter:}
722 \begin{itemize}
723 \item Tidak memiliki parameter \textit{method}
724 \end{itemize}
725 \textbf{Return Value}: Tidak memiliki \textit{return value}
726
727 \textbf{Exception}: Tidak memiliki \textit{exception}
728
729 \item \texttt{public Semester getSemester()}
730
731
732
733 \textbf{Parameter:}
734 \begin{itemize}
735 \item Tidak memiliki parameter \textit{method}
736 \end{itemize}
737 \textbf{Return Value}: Tidak memiliki \textit{return value}
738
739 \textbf{Exception}: Tidak memiliki \textit{exception}
740
741 \item \texttt{public String toString()}
742
743
744
745 \textbf{Parameter:}
746 \begin{itemize}
747 \item Tidak memiliki parameter \textit{method}
748 \end{itemize}
749 \textbf{Return Value}: Tidak memiliki \textit{return value}
750
751 \textbf{Exception}: Tidak memiliki \textit{exception}
752
753 \end{itemize}
754 \item \texttt{Mahasiswa.Nilai.ChronologicalComparator implements Comparator}
755
756 Pembanding antara satu nilai dengan nilai lainnya, secara
757  kronologis waktu pengambilan.
758
759 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
760 \begin{itemize}
761 \item \texttt{public int compare(id.ac.unpar.siamodels.Mahasiswa.Nilai o1, id.ac.unpar.siamodels.Mahasiswa.Nilai o2)}
762
763
764
765 \textbf{Parameter:}
766 \begin{itemize}
767 \item \texttt{Mahasiswa.Nilai o1} -
768 \item \texttt{Mahasiswa.Nilai o2} -
769 \end{itemize}
770 \textbf{Return Value}: Tidak memiliki \textit{return value}
771
772 \textbf{Exception}: Tidak memiliki \textit{exception}
773
774 \textbf{Override}: \texttt{compare} dari kelas \texttt{Object}
775
776 \end{itemize}
777 \item \texttt{JadwalKuliah}
778
779
780
781 Atribut yang dimiliki kelas ini adalah sebagai berikut.
782 \begin{itemize}
783 \item \texttt{MataKuliah mataKuliah} -
784 \item \texttt{Character kelas} -
785 \item \texttt{DayOfWeek hari} -
786 \item \texttt{LocalTime waktuMulai} -
787 \item \texttt{LocalTime waktuSelesai} -
788 \item \texttt{String lokasi} -
789 \item \texttt{Dosen pengajar} -
790 \end{itemize}
```

```
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public MataKuliah getMataKuliah()}


\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public void setMataKuliah(id.ac.unpar.siamodels.MataKuliah mataKuliah)}


\textbf{Parameter:}
\begin{itemize}
\item \texttt{MataKuliah mataKuliah} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public Character getKelas()}


\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public void setKelas(java.lang.Character kelas)}


\textbf{Parameter:}
\begin{itemize}
\item \texttt{Character kelas} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public DayOfWeek getHari()}


\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public void setHari(java.time.DayOfWeek hari)}


\textbf{Parameter:}
\begin{itemize}
\item \texttt{DayOfWeek hari} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public LocalTime getWaktuMulai()}


\textbf{Parameter:}
\begin{itemize}
\item Tidak memiliki parameter \textit{method}
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public void setWaktuMulai(java.time.LocalTime waktuMulai)}


\textbf{Parameter:}
\begin{itemize}
\item \texttt{LocalTime waktuMulai} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public LocalTime getWaktuSelesai()}
```

```
890
891
892
893 \textbf{Parameter:}
894 \begin{itemize}
895 \item Tidak memiliki parameter \textit{method}
896 \end{itemize}
897 \textbf{Return Value}: Tidak memiliki \textit{return value}
898
899 \textbf{Exception}: Tidak memiliki \textit{exception}
900
901 \item \texttt{public void setWaktuSelesai(java.time.LocalTime waktuSelesai)}
902
903
904
905 \textbf{Parameter:}
906 \begin{itemize}
907 \item \texttt{LocalTime waktuSelesai} -
908 \end{itemize}
909 \textbf{Return Value}: Tidak memiliki \textit{return value}
910
911 \textbf{Exception}: Tidak memiliki \textit{exception}
912
913 \item \texttt{public String getLokasi()}
914
915
916
917 \textbf{Parameter:}
918 \begin{itemize}
919 \item Tidak memiliki parameter \textit{method}
920 \end{itemize}
921 \textbf{Return Value}: Tidak memiliki \textit{return value}
922
923 \textbf{Exception}: Tidak memiliki \textit{exception}
924
925 \item \texttt{public void setLokasi(java.lang.String lokasi)}
926
927
928
929 \textbf{Parameter:}
930 \begin{itemize}
931 \item \texttt{String lokasi} -
932 \end{itemize}
933 \textbf{Return Value}: Tidak memiliki \textit{return value}
934
935 \textbf{Exception}: Tidak memiliki \textit{exception}
936
937 \item \texttt{public Dosen getPengajar()}
938
939
940
941 \textbf{Parameter:}
942 \begin{itemize}
943 \item Tidak memiliki parameter \textit{method}
944 \end{itemize}
945 \textbf{Return Value}: Tidak memiliki \textit{return value}
946
947 \textbf{Exception}: Tidak memiliki \textit{exception}
948
949 \item \texttt{public void setPengajar(id.ac.unpar.siamodels.Dosen pengajar)}
950
951
952
953 \textbf{Parameter:}
954 \begin{itemize}
955 \item \texttt{Dosen pengajar} -
956 \end{itemize}
957 \textbf{Return Value}: Tidak memiliki \textit{return value}
958
959 \textbf{Exception}: Tidak memiliki \textit{exception}
960
961 \item \texttt{public String getWaktuString()}
962
963
964
965 \textbf{Parameter:}
966 \begin{itemize}
967 \item Tidak memiliki parameter \textit{method}
968 \end{itemize}
969 \textbf{Return Value}: Tidak memiliki \textit{return value}
970
971 \textbf{Exception}: Tidak memiliki \textit{exception}
972
973 \item \texttt{public static DayOfWeek indonesianToDayOfWeek(java.lang.String indonesian)}
974
975 Converts Indonesian day names to \texttt{DayOfWeek} enumeration.
976
977 \textbf{Parameter:}
978 \begin{itemize}
979 \item \texttt{String indonesian} -
980 the day name in Indonesian
981 \end{itemize}
982 \textbf{Return Value}: DayOfWeek object or null if not found.
983
984 \textbf{Exception}: Tidak memiliki \textit{exception}
985
986 \end{itemize}
987 \item \texttt{MataKuliah}
988
```

```
989
990
991 Atribut yang dimiliki kelas ini adalah sebagai berikut.
992 \begin{itemize}
993 \item \texttt{String kode} -
994 \item \texttt{String nama} -
995 \item \texttt{Integer sks} -
996 \end{itemize}
997 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
998 \begin{itemize}
999 \item \texttt{public String getKode()}
1000
1001
1002
1003 \textbf{Parameter:}
1004 \begin{itemize}
1005 \item Tidak memiliki parameter \textit{method}
1006 \end{itemize}
1007 \textbf{Return Value}: Tidak memiliki \textit{return value}
1008
1009 \textbf{Exception}: Tidak memiliki \textit{exception}
1010
1011 \item \texttt{public String getNama()}
1012
1013
1014
1015 \textbf{Parameter:}
1016 \begin{itemize}
1017 \item Tidak memiliki parameter \textit{method}
1018 \end{itemize}
1019 \textbf{Return Value}: Tidak memiliki \textit{return value}
1020
1021 \textbf{Exception}: Tidak memiliki \textit{exception}
1022
1023 \item \texttt{public Integer getSks()}
1024
1025
1026
1027 \textbf{Parameter:}
1028 \begin{itemize}
1029 \item Tidak memiliki parameter \textit{method}
1030 \end{itemize}
1031 \textbf{Return Value}: Tidak memiliki \textit{return value}
1032
1033 \textbf{Exception}: Tidak memiliki \textit{exception}
1034
1035 \end{itemize}
1036 \item \texttt{Dosen}
1037
1038
1039
1040 Atribut yang dimiliki kelas ini adalah sebagai berikut.
1041 \begin{itemize}
1042 \item \texttt{String nik} -
1043 \item \texttt{String nama} -
1044 \end{itemize}
1045 \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
1046 \begin{itemize}
1047 \item \texttt{public String getNik()}
1048
1049
1050
1051 \textbf{Parameter:}
1052 \begin{itemize}
1053 \item Tidak memiliki parameter \textit{method}
1054 \end{itemize}
1055 \textbf{Return Value}: Tidak memiliki \textit{return value}
1056
1057 \textbf{Exception}: Tidak memiliki \textit{exception}
1058
1059 \item \texttt{public void setNik(java.lang.String nik)}
1060
1061
1062
1063 \textbf{Parameter:}
1064 \begin{itemize}
1065 \item \texttt{String nik} -
1066 \end{itemize}
1067 \textbf{Return Value}: Tidak memiliki \textit{return value}
1068
1069 \textbf{Exception}: Tidak memiliki \textit{exception}
1070
1071 \item \texttt{public String getNama()}
1072
1073
1074
1075 \textbf{Parameter:}
1076 \begin{itemize}
1077 \item Tidak memiliki parameter \textit{method}
1078 \end{itemize}
1079 \textbf{Return Value}: Tidak memiliki \textit{return value}
1080
1081 \textbf{Exception}: Tidak memiliki \textit{exception}
1082
1083 \item \texttt{public void setNama(java.lang.String nama)}
1084
1085
1086
1087 \textbf{Parameter:}
```

```
\begin{itemize}
\item \texttt{String nama} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\item \texttt{public boolean equals(java.lang.Object arg0)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Object arg0} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\end{itemize}
\item \texttt{MKU008}

Mendalami perilaku sehari-hari yang baik dalam bermasyarakat.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{IIE210}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF203 implements HasPrasyarat}

Mata kuliah ini memperkenalkan kepada mahasiswa konsep struktur diskret yang
 digunakan pada bidang informatika diantaranya graph, pohon dan finite state
 machine

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF311 implements HasPrasyarat}

\item \texttt{AIF311 implements HasPraktikum}

Kuliah Pemrograman Fungsional bertujuan untuk: 1. memperkenalkan paradigma
 pemrograman fungsional, yaitu sebuah pemrograman yang didasarkan pada konsep
 pemetaan dan fungsi matematika. Penyelesaian suatu masalah didasari atas
 aplikasi dari fungsi-fungsi tersebut. 2. memberikan dasar-dasar pemrograman
 fungsional dengan menggunakan bahasa fungsional Haskell.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF192}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF468}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{IIE103}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF385}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF106}

Mata kuliah ini memberikan pengetahuan tentang cara kerja komputer, dimulai
```

```
1187  dari representasi data dan berbagai macam operasinya. Selanjutnya, juga
1188  diperkenalkan bagaimana merepresentasikan suatu fungsi dalam rangkaian
1189  gerbang logika, dan bagaimana menyederhanakannya. Berbagai rangkaian dasar
1190  yang digunakan di dalam komputer juga dipekenalkan. Mahasiswa juga akan
1191  mempelajari komponen komputer, misalnya register dan memori.
1192
1193 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF281}
1194
1195
1196
1197 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{EAA102}
1198
1199
1200
1201 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF405 implements HasPrasyarat}
1202
1203 \item \texttt{AIF405 implements HasPraktikum}
1204
1205 Mata kuliah ini merupakan lanjutan dari Projek Sistem Informasi 1 dan
1206  memberikan kesempatan bagi mahasiswa untuk melanjutkan/mengembangkan
1207  perancangan sitem pada organisasi studi kasus, mengimplementasikan rancangan
1208  dan melakukan pengujian perangkat lunak;
1209
1210 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
1211 \begin{itemize}
1212 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
1213
1214
1215
1216 \textbf{Parameter:}
1217 \begin{itemize}
1218 \item \texttt{Mahasiswa mahasiswa} -
1219 \item \texttt{java.util.List reasonsContainer} -
1220 \end{itemize}
1221 \textbf{Return Value}: Tidak memiliki \textit{return value}
1222
1223 \textbf{Exception}: Tidak memiliki \textit{exception}
1224
1225 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
1226
1227 \end{itemize}
1228 \item \texttt{APS182}
1229
1230
1231
1232 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{MKU004}
1233
1234 Fenomenologi Agama merupakan bagian yang tak terpisahkan dari kajian filosofis, kritis,
1235  rasional, dan obyektif mengenai substansi ajaran agama. Fenomenologi merupakan sebuah
1236  disiplin ilmu yang secara kritis-rasional mengkaji fenomena dan dinamika kehidupan manusia
1237  beragama, dari upaya menjadikan Tuhan sebagai tujuan sesembahan sampai menempatkan Tuhan
1238  sebagai instrumen legitimasi untuk melakukan tindakan yang justru bertolak belakang dengan
1239  kehendak Tuhan yang disembah. Sehubungan dengan itu, kritik konstruktif terhadap perilaku
1240  manusia beragama menjadi salah satu poin utama dalam mata kuliah ini. Kesediaan untuk
1241  melakukan otoritik terhadap agama sendiri erat terkait dengan upaya menemukan kembali nilai
1242  sejati agama atau otentisitas hidup beragama.
1243
1244 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{MKU012}
1245
1246 Perkuliahan logika ditujukan untuk memberikan dasar-dasar ketrampilan berpikir rasional dan
1247  sistematik. Isinya mencakup ketrampilan berpikir deduktif dan induktif, seperti silogisme,
1248  argumen analogikal dan generalisasi induktif. Pembahasan teoretis disertai pula dengan
1249  pelatihan praktis yang diarahkan pada proses berpikir. Untuk menajamkan kemampuan berpikir
1250  tersebut, mahasiswa dilatih pula mengidentifikasi kerancuan-kerancuan (fallacies) yang sering
1251  dijumpai baik dalam kehidupan sehari-hari maupun dalam konteks akademik.
1252
1253 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF389}
1254
1255
1256
1257 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AMS190}
1258
1259
1260
1261 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AMS191}
1262
1263
1264
1265 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AMS200}
1266
1267
1268
1269 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF330}
1270
1271
1272
1273 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF388}
1274
1275
1276
1277 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF465}
1278
1279
1280
1281 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF453 implements HasPrasyarat}
1282
1283 Mata kuliah inimemperkenalkan kebutuhan organisasi terhadap sistem business
1284  intelligent (BI) dan pemanfaatan BI untuk organisasi; memperkenalkan konsep
1285  sistem business intelligent dan komponennya; Mempelajari tenik-teknik
```

```
 analisis data bisnis dan visualisasi hasil analisis; Mempelajari konsep data
 warehouse dan perancangannya dan fungsi OLAP; Mempraktekkan teknik-teknik
 analisis data dan visualisasi hasil analisis.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF280}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF445 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF469 implements HasPrasyarat}

Mata kuliah ini mengajarkan kepada mahasiswa teknik-teknik untuk membuat
 layanan berbasis web. Mahasiswa diperkenalkan dengan standar-standar seperti
 HTTP, XML, JSON dan diajarkan untuk memanfaatkannya dalam membuat maupun
 menggunakan layanan pihak ketiga. Dalam kuliah ini, juga akan diperkenalkan
 minimal satu layanan pihak ketiga yang dapat dimanfaatkan mahasiswa, seperti
 Google Places Web Service.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF486}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{IIE207}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF202 implements HasPrasyarat}

\item \texttt{AIF202 implements HasResponsi}

\item \texttt{AIF202 implements HasPraktikum}

Mata kuliah ini memperkenalkan kepada mahasiswa beberapa algoritma dan
 struktur data, alternatif cara implementasinya, dan analisis kompleksitas
 waktunya. Mahasiswa diberikan beberapa masalah komputasi yang harus
 diselesaikan dengan menggunakan algoritma atau struktur data yang sudah
 diperkenalkan dan mengimplementasikannya dalam bahasa pemrograman Java.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}

```

```
1385
1386
1387 \textbf{Parameter:}
1388 \begin{itemize}
1389 \item \texttt{Mahasiswa mahasiswa} -
1390 \item \texttt{java.util.List reasonsContainer} -
1391 \end{itemize}
1392 \textbf{Return Value}: Tidak memiliki \textit{return value}
1393
1394 \textbf{Exception}: Tidak memiliki \textit{exception}
1395
1396 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
1397
1398 \end{itemize}
1399 \item \texttt{AIF347}
1400
1401
1402
1403 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{MKU009}
1404
1405 Tujuan dari mata kuliah ini adalah untuk mendalami keterampilan berbahasa Indonesia, agar
1406  mampu mengkomunikasikan hasil pemikiran serta meningkatkan keterampilan dalam menyusun karya
1407  ilmiah. Mata kuliah Bahasa Indonesia ini dimulai dengan mempelajari penulisan kata baku dan
1408  non baku serta pengungkapan pikiran dengan pungtuasi yang benar. Selanjutnya dipelajari
1409  penyusunan kalimat yang baku serta menghubungkan kalimat-kalimat yang padu dalam menuangkan
1410  gagasan dalam sebuah paragraf. Selain itu, dalam matakuliah ini dipelajari cara menyusun
1411  surat dinas yang jelas dan komunikatif. Di akhir kuliah ini, mahasiswa diberi tugas
1412  penyusunan makalah dengan benar.
1413
1414 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{ESM203}
1415
1416
1417
1418 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{APS402 implements HasPrasyarat}
1419
1420
1421
1422 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
1423 \begin{itemize}
1424 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
1425
1426
1427
1428 \textbf{Parameter:}
1429 \begin{itemize}
1430 \item \texttt{Mahasiswa mahasiswa} -
1431 \item \texttt{java.util.List reasonsContainer} -
1432 \end{itemize}
1433 \textbf{Return Value}: Tidak memiliki \textit{return value}
1434
1435 \textbf{Exception}: Tidak memiliki \textit{exception}
1436
1437 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
1438
1439 \end{itemize}
1440 \item \texttt{AIF306 implements HasPrasyarat}
1441
1442 Mata kuliah ini bertujuan untuk memberikan pengalaman bagi mahasiswa dalam
1443  mengerjakan proyek dengan teknologi-teknologi terkini, secara berkelompok.
1444  Teknologi-teknologi yang digunakan pada kuliah ini tidak spesifik dan dapat
1445  berubah seiring perkembangan teknologi maupun disesuaikan dengan kompetensi
1446  dosen pengajar. Beberapa teknologi yang dapat dimanfaatkan antara lain: DVCS
1447  tool menggunakan Git + Github, Mobile native app (Android, iOS, dll), dan
1448  responsive web design.
1449
1450 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
1451 \begin{itemize}
1452 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
1453
1454
1455
1456 \textbf{Parameter:}
1457 \begin{itemize}
1458 \item \texttt{Mahasiswa mahasiswa} -
1459 \item \texttt{java.util.List reasonsContainer} -
1460 \end{itemize}
1461 \textbf{Return Value}: Tidak memiliki \textit{return value}
1462
1463 \textbf{Exception}: Tidak memiliki \textit{exception}
1464
1465 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
1466
1467 \end{itemize}
1468 \item \texttt{AKS122}
1469
1470
1471
1472 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{MKU002}
1473
1474 Pendidikan Kewarganegaraan menjelaskan pentingnya pemahaman tentang identitas nasional
1475  Indonesia, hak dan kewajiban warga negara Indonesia serta hubungannya dengan hak dan
1476  kewajiban asasi manusia. Materi kuliah mencakup juga wawasan nusantara, ketahanan nasional,
1477  politik dan strategi nasional, serta implementasinya dalam kehidupan bermasyarakat, berbangsa
1478  dan bernegara kesatuan Republik Indonesia.
1479
1480 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF462}
1481
1482
1483
```

```
Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF360 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF337}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF458 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF301 implements HasPrasyarat}

Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar tentang *sistem
 cerdas dan komponen-komponennya. " "Terdapat 4 topik utama yang dibahas yaitu
 teknik pencarian untuk *penyelesaian masalah, representasi pengetahuan dalam
 sistem *cerdas, pemodelan ketidakpastian dalam masalah dan teknik
 pembelajaran mesin.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF182}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{ESM204}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF205 implements HasPrasyarat}

Mata kuliah ini memperkenalkan kepada mahasiswa arsitektur komputer
 sederhana, modern, dan Advance. Perbedaan, kelebihan dan kekurangan untuk
 masing-masing arsitektur. Selain itu mahasiswa juga mempelajari cara kerja
 dari komponen-komponen komputer, terutama memory, cache, system BUS dan
 input/output.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
```

```
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF317 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF383}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF442 implements HasPraktikum}

\item \texttt{AIF442 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{ESM101}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF403 implements HasPrasyarat}

1. Memberikan wawasan kepada mahasiswa tentang kemunculan dan pemanfaatan teknologi baru,
 khususnya yang berkaitan dengan komputer, dan dampaknya terhadap masyarakat luas.
 2. Memberikan kesadaran dan panduan bersikap kepada mahasiswa dalam menghadapi gejolak yang
 disebabkan oleh munculnya teknologi baru, khususnya yang berkaitan dengan komputer.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF402 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
```

```
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF455}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF443}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF101 implements HasPraktikum}

Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar pemrograman
 seperti pengulangan dan percabangan, konsep dasar penyimpanan data kontigu
 menggunakan array, konsep dasar pemrograman berorientasi objek seperti kelas
 \& objek, method, dll, termasuk di dalamnya 4 prinsip dasar pemrograman
 berorientasi objek : data abstraction, encapsulation, inheritance dan
 polymorphism. Selain, itu diberikan masalah-masalah komputasi sederhana
 yang harus diselesaikan menggunakan konsep-konsep yang  sudah diperkenalkan
 dan mengimplementasikannya menggunakan bahasa pemrograman Java

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF382}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF480}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF316 implements HasPrasyarat}

\item \texttt{AIF316 implements HasPraktikum}

Mata kuliah ini memperkenalkan konsep-konsep dasar komputasi paralel, dimana sebuah
 program yang berjalan secara paralel harus memiliki safety property dan liveness property.
 Mahasiswa dikenalkan dengan beberapa teknik pemrograman multi-thread
 seperti lock, monitor, barrier, thread pool, dan sebagainya, yang diimplementasikan
 dalam bahasa pemrograman Java. Mahasiswa juga dikenalkan dengan beberapa metode untuk
 menganalisis kebenaran program baik secara matematis maupun secara praktis dengan bantuan
 model checker.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF438 implements HasPrasyarat}

Mata kuliah ini: Memperkenalkan karakteristik dan teknik visualisasi dari
 berbagai jenis data yang dapat dianalisis dengan teknik-teknik data mining;
 mempelajari teknik-teknik penyiapan data untuk berbagai jenis data dan teknik
 data mining; mempraktekkan teknik-teknik penyiapan data untuk menganalisis
 data nyata/simulasi dengan memanfaatkan perangkat lunak aplikasi.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF204 implements HasPrasyarat}

\item \texttt{AIF204 implements HasPraktikum}

Mata kuliah ini memperkenalkan konsep dan arsitektur DBMS, mengajarkan
```

```
 aljabar relasional dan SQL serta pemanfaatannya pada pemrograman kueri
 sederhana s/d relatif kompleks. Selain itu, mata kuliah ini juga mengajarkan
 dan mempraktekkan perancangan basisdata untuk masalah sederhana
 (lingkup kecil) termasuk pengembangan program aplikasinya;

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF341 implements HasPraktikum}

Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar
 jaringan dan aplikasinya di kehidupan sehari-hari. Mahasiswa
 dikenalkan dengan teknologi-teknologi terbaru di bidang jaringan,
 sehingga mahasiswa memiliki pengetahuan yang dapat digunakan
 dalam kehidupan sehari-hari. Selain itu mahasiswa juga
 diperkenalkan dengan NetAcad, sebuah layanan dari Cisco yang
 dapat digunakan untuk memenuhi segala macam kebutuhan terkait
 dengan Cisco Academy.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF183}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AKS124}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF459}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF336}

Mata kuliah ini merupakan mata kuliah lanjutan dari mata kuliah Keamanan
 Informasi, dengan titik berat pada materi kriptografi. Mata kuliah ini
 memperkenalkan tambahan konsep kriptografi, misalnya tentang otentikasi
 yaitu otentikasi entitas, manajemen kunci, dan bentuk lain dari metode
 merahasiakan pesan, yaitu dengan menggunakan secret sharing. Selanjutnya,
 diperkenalkan juga penggunaan kriptografi pada protokol-protokol yang
 sebenarnya banyak digunakan sehari-hari, misalnya pada e-cash, auction,
 dan electronic voting.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF463}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF208 implements HasPrasyarat}

Mata kuliah ini memperkenalkan kepada mahasiswa tahapan rekayasa perangkat
 lunak, terutama dengan paradigma berorientasi objek, dilengkapi dengan
 pengenalan tentang manajemen proyek perangkat lunak.
 Selain, itu diberikan deskripsi proyek berskala kecil yang harus dikerjakan
 oleh mahasiswa dalam kelompok dengan menerapkan teori yang telah
 dipelajarinya.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{MKU003}

Mata kuliah ini membentuk karakteristik mahasiswa sebagai manusia yang memiliki religiusitas
 melalui pendalaman akan makna agama dan beragama, mendeteksi dinamika Wahyu Tuhan dan iman
 mereka, memahami relasi dengan Tuhan dan sesama, mengenal makna keselamatan dalam konteks
 Kerajaan Allah, dan mampu menyatakan ajaran Gereja dalam pelayanan terhadap orang miskin dan
 terlantar.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{APS309}

APS302 atau APS309 ?
```

```
1880
1881 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF335}
1882
1883
1884
1885 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF362 implements HasPrasyarat}
1886
1887
1888
1889 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
1890 \begin{itemize}
1891 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
1892
1893
1894
1895 \textbf{Parameter:}
1896 \begin{itemize}
1897 \item \texttt{Mahasiswa mahasiswa} -
1898 \item \texttt{java.util.List reasonsContainer} -
1899 \end{itemize}
1900 \textbf{Return Value}: Tidak memiliki \textit{return value}
1901
1902 \textbf{Exception}: Tidak memiliki \textit{exception}
1903
1904 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
1905
1906 \end{itemize}
1907 \item \texttt{AIF460}
1908
1909
1910
1911 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF358}
1912
1913
1914
1915 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AMS100}
1916
1917 Sistem Bilangan, Fungsi, Limit dan Kekontinuan Fungsi, Turunan, Integral,
1918  Penggunaan Integral, Sistem Persamaan Linear, Determinan, Vektor, Nilai dan
1919  Vektor Eigen.
1920
1921 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF401 implements HasPrasyarat}
1922
1923
1924
1925 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
1926 \begin{itemize}
1927 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
1928
1929
1930
1931 \textbf{Parameter:}
1932 \begin{itemize}
1933 \item \texttt{Mahasiswa mahasiswa} -
1934 \item \texttt{java.util.List reasonsContainer} -
1935 \end{itemize}
1936 \textbf{Return Value}: Tidak memiliki \textit{return value}
1937
1938 \textbf{Exception}: Tidak memiliki \textit{exception}
1939
1940 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
1941
1942 \end{itemize}
1943 \item \texttt{AIF456}
1944
1945
1946
1947 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{SIR104}
1948
1949
1950
1951 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF339 implements HasPrasyarat}
1952
1953
1954
1955 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
1956 \begin{itemize}
1957 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
1958
1959
1960
1961 \textbf{Parameter:}
1962 \begin{itemize}
1963 \item \texttt{Mahasiswa mahasiswa} -
1964 \item \texttt{java.util.List reasonsContainer} -
1965 \end{itemize}
1966 \textbf{Return Value}: Tidak memiliki \textit{return value}
1967
1968 \textbf{Exception}: Tidak memiliki \textit{exception}
1969
1970 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
1971
1972 \end{itemize}
1973 \item \texttt{AIF102 implements HasPrasyarat}
1974
1975 \item \texttt{AIF102 implements HasPraktikum}
1976
1977 Mata kuliah ini memperkenalkan berbagai algoritma dan teknik-teknik
1978  penyelesaian masalah komputasi seperti rekursif, sorting, teknik divide dan
```

```
 conquer, serta exhaustive search. Selain itu, pada kuliah ini juga
 dikenalkan berbagai struktur data yang dapat digunakan untuk mendukung
 penyelesaian masalah komputasi seperti ADT List, Stack dan Queue. Baik
 algoritma maupun struktur data yang dikenalkan harus dapat diimplementasikan
 dan digunakan oleh mahasiswa untuk menyelesaikan masalah dengan menggunakan
 suatu bahasa pemrograman berorientasi objek.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF381}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF483}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF315 implements HasPrasyarat}

\item \texttt{AIF315 implements HasPraktikum}

Mata kuliah ini memperkenalkan konsep dan lingkungan pemrograman berbasis web,
 kemudian belajar membuat aplikasi berbasis web menggunakan HTML5, CSS, Java Script
 dan PHP. Untuk meningkatkan keterampilan pemrograman dilengkapi dengan praktikum.
 Sedangkan untuk mendapatkan pengalaman penerapan konsep diberikan tugas besar membuat
 program berbasis web dengan kasus yang ditentukan oleh mahasiswa.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF342 implements HasPraktikum}

\item \texttt{AIF342 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{IIE214}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF303 implements HasPrasyarat}

Mempelajari Konsep Data, Informasi, Pengetahuan, Sistem Informasi, proses dan
 pemodelan bisnis, jenis-jenis sistem informasi, untuk mendukung pengambilan
 keputusan. Mempelajari trend Teknologi Informasi, tahap-tahap pembangunan
 sistem informasi. Mempelajari pengantar : EIS, e-bisnis/e-commerce, Business
 Intelligence, Cloud Computing dan Mobile Applications

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
```

```
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF302 implements HasPrasyarat}

Mata kuliah ini melatih mahasiswa dalam menulis ilmiah serta memperkenalkan
 metodologi penelitian serta kakas untuk menulis ilmiah.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF181}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF210}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF343}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF206 implements HasPrasyarat}

Mata kuliah ini memperkenalkan kepada mahasiswa mengenai konsep sistem
 operasi, jenis-jenis sistem operasi yang digunakan dalam kehidupan
 sehari-hari dan beberapa perangkat keras yang dibutuhkan pada komputer.
 Selain itu juga mempelajari mengenai teknik dan algoritma yang digunakan
 dalam pengelolaan sistem operasi.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF314 implements HasPrasyarat}

\item \texttt{AIF314 implements HasPraktikum}

Kuliah ini merupakan kelanjutan dari kuliah Manajemen Informasi Basisdata.
 Pada perkuliahan ini, mahasiswa akan mempelajari teknik-teknik pengelolaan
 basis data dan membuat program dengan basis data yang optimal/efisien.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

```
2177
2178 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
2179
2180 \end{itemize}
2181 \item \texttt{AIF380}
2182
2183
2184
2185 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF103}
2186
2187 Mata kuliah ini merupakan salah satu cara untuk mencapai kompetensi dasar
2188  tentang matematika diskrit yang prinsipnya banyak digunakan dalam bidang
2189  ilmu komputer. Selain itu, kuliah ini juga merupakan cara untuk membentuk
2190  pola pikir logis yang dibutuhkan untuk menempuh kuliah-kuliah di tingkat
2191  yang lebih tinggi.
2192
2193 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF292}
2194
2195
2196
2197 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF441 implements HasPraktikum}
2198
2199 \item \texttt{AIF441 implements HasPrasyarat}
2200
2201 Mata kuliah ini memperkenalkan kepada mahasiswa konsep jaringan lanjut
2202  terutama di layer data link dan layer network. Materi utama dari mata kuliah
2203  ini adalah pengembangan jaringan dan pengenalan fungsi-fungsi yang terdapat
2204  pada alat jaringan Cisco yang berkaitan dengan layer 2 dan layer 3.
2205
2206 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
2207 \begin{itemize}
2208 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
2209
2210
2211
2212 \textbf{Parameter:}
2213 \begin{itemize}
2214 \item \texttt{Mahasiswa mahasiswa} -
2215 \item \texttt{java.util.List reasonsContainer} -
2216 \end{itemize}
2217 \textbf{Return Value}: Tidak memiliki \textit{return value}
2218
2219 \textbf{Exception}: Tidak memiliki \textit{exception}
2220
2221 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
2222
2223 \end{itemize}
2224 \item \texttt{AIF457 implements HasPrasyarat}
2225
2226 Mata kuliah ini memperkenalkan konsep kewirausahaan dengan memanfaatkan teknologi, khususnya
2227  teknologi informasi, sebagai basis usaha dan inovasi produk/jasa; Mempelajari
2228  teknik mencari peluang dan merumuskan bidang usaha spesifik yang akan
2229  diterjuni; Mempelajari konsep manajemen pemasaran, keuangan dan SDM dalam
2230  kaitannya dengan berwira-usaha di bidang TI; Menyusun proposal bisnis untuk
2231  berwira-usaha di bidang TI dan mempresentasikannya.
2232
2233 Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
2234 \begin{itemize}
2235 \item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}
2236
2237
2238
2239 \textbf{Parameter:}
2240 \begin{itemize}
2241 \item \texttt{Mahasiswa mahasiswa} -
2242 \item \texttt{java.util.List reasonsContainer} -
2243 \end{itemize}
2244 \textbf{Return Value}: Tidak memiliki \textit{return value}
2245
2246 \textbf{Exception}: Tidak memiliki \textit{exception}
2247
2248 \textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}
2249
2250 \end{itemize}
2251 \item \texttt{MKU001}
2252
2253 Mata Kuliah Pendidikan Pancasila berupaya menelaah/mengkaji berbagai fenomena kehidupan
2254  bangsa dan Negara Indonesia sebagai sebuah ruang publik dengan menggunakan pendekatan
2255  hermeneutika (filsafat) dan pendidikan nilai (pedagogik). Dengan bantuan hermenutika
2256  mahasiswa diajak berpikir kritis terhadap segala bentuk ideologisme Pancasila dan melalui
2257  pendidikan nilai mahasiswa dilatih untuk memiliki nilai Pancasila. Nilai pengembangan diri
2258  intra-personal dan relasi inter-personal dapat tertanam melalui pendidikan Pancasila yang
2259  tujuannya adalah membangun kepribadian (character building) manusia Indonesia yang utuh,
2260  baik menyangkut aspek kognitif, afektif, maupun psikomotor. Dengan demikian, Pendidikan
2261  Pancasila mengajak mahasiswa menilai realitas ruang publik sehari-hari secara mandiri
2262  dengan panduan nilai-nilai etis Pancasila.
2263
2264 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{ESA101}
2265
2266
2267
2268 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AMS290}
2269
2270
2271
2272 Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF461}
2273
2274
2275
```

```
Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF318 implements HasPrasyarat}

\item \texttt{AIF318 implements HasPraktikum}

Mata kuliah ini memperkenalkan konsep perangkat mobile dan pemrograman pada perangkat
 mobile. Pemrograman dikhususkan pada lingkungan J2ME dan Android.
 Untuk meningkatkan keterampilan pemrograman dilengkapi dengan praktikum.
 Sedangkan untuk mendapatkan pengalaman penerapan konsep diberikan tugas implementasi suatu
 kasus pada lingkungan mobile-cloud dengan kasus yang sudah ditentukan.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF334}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{ESM105}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF450}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF446}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF104}

Mata kuliah ini memberikan pengetahuan tentang logika yang digunakan di
 dalam ilmu komputer. Dalam kuliah ini, mahasiswa belajar untuk bisa
 memodelkan suatu kalimat dalam kehidupan sehari-hari, ke dalam kalimat
 dengan sintaks tertentu, yang hanya memiliki satu arti. Lalu, diperkenalkan
 juga, bagaimana mengartikan suatu kalimat (benar atau salah) dan bagaimana
 menentukan sifat dari kalimat tersebut.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF387}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF313 implements HasPraktikum}

Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar pembuatan grafik
 dengan menggunakan komputer seperti mengenal berbagai algoritma pembuatan
 primitif 2 dimensi seperti titik, garis, lingkaran, elips, berbagai macam
 bentuk kurva, fraktal, konsep warna (RGB), dasar-dasar grafika 3 dimensi
 seperti pewarnaan, pencahayaan, pemberian tekstur pada objek, transformasi,
 animasi, dan sebagainya. Selain, itu diberikan masalah-masalah komputasi
 sederhana yang harus diselesaikan menggunakan konsep-konsep yang sudah
 diperkenalkan dan mengimplementasikannya menggunakan bahasa pemrograman Java.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF344 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF201 implements HasPrasyarat}

\item \texttt{AIF201 implements HasPraktikum}

\item \texttt{AIF201 implements HasResponsi}

Mata kuliah ini memperkenalkan prinsip-prinsip yang digunakan dalam
 melakukan analisa serta desain prorgram berorientasi objek. Di samping itu,
 mahasiswa juga belajar menggunakan kakas berupa diagram UML (Unified
```

```
 Modelling Language) sehingga dapat mengkomunikasikan desain secara visual.
 Mahasiswa juga akan mengenal beberapa software design pattern dari Gang of
 Four. Terakhir, mahasiswa akan belajar mengenai konsep MVC
 (Model-View-Controller) yang menjadi dasar dari banyak framework masa kini.
 Bahasa yang digunakan adalah bahasa Java, namun diusahakan tetap umum
 sehingga dapat diaplikasikan pada bahasa yang lain.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF352}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF305 implements HasPrasyarat}

Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar jaringan
 komputer dengan menggunakan top-down. Selain itu mengajarkan juga kepada
 mahasiswa mengenai aplikasi-aplikasi berbasis jaringan sehingga diharapkan
 mahasiswa dapat membuat aplikasi berbasis jaringan dengan menggunakan socket.
 Pada akhirnya, mahasiswa akan ditugaskan untuk membangun jaringan komputer
 LAN, baik menggunakan kabel maupun nirkabel.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{MKU010}

Mata kuliah ini difokuskan pada pemahaman sumber referensi dalam Bahasa Inggris dan
 pengembangan kosakata Bahasa Inggris (vocabularies). Hampir keseluruhan waktu perkuliahan
 didedikasikan untuk menjelaskan metode mengekstraksi isi bacaan secara tepat dan melatih
 mahasiswa untuk menerapkan metode tersebut seraya menambah kosakata-kosakata baru.
 Mahasiswa juga dilatih untuk mempresentasikan hasil pemahamannya akan isi bahan bacaan.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{MKU011}

Mata kuliah estetika memberi pemahaman konseptual filosofis "seni" dalam khasanah keilmuan,
 pembentukan kesadaran ekologis juga dalam proses pembudayaan dan peradaban. Mata kuliah ini
 akan menjadi fondasi bagi mahasiswa untuk memahami dan mempraktekkan seni dari sudut pandang
 filsafat, sejarah, kultural, dan global. Melalui mata kuliah ini, mahasiswa mempelajari
 mengenai dunia manusia (manusia dan pikirannya), pluralitas dan relativitas seni, serta
 aliran-aliran seni rupa Barat.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF332 implements HasPrasyarat}



Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF304 implements HasPrasyarat}

\item \texttt{AIF304 implements HasPraktikum}

```

```
\item \texttt{AIF304 implements HasResponsi}

Mata kuliah ini memberikan kesempatan bagi mahasiswa untuk memperdalam konsep
 tentang pengembangan sistem informasi dan mempraktekkan analisis kebutuhan,
 analisis sistem dan perancangan sitem pada organisasi studi kasus;

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{ESM201}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF312 implements HasPrasyarat}

\item \texttt{AIF312 implements HasPraktikum}

Mata kuliah ini memberikan pengetahuan awal tentang keamanan informasi. Pada
 beberapa pertemuan awal, dibahas keamanan informasi secara matematis, yaitu
 di materi-materi seputar kriptografi dan serangannya. Lalu, dibahas pula
 konsep keamanan informasi pada jaringan komputer dan pada software.

Kelas ini tidak memiliki atribut. \textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{MataKuliah}

\end{itemize}
\item \texttt{AIF484}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF191}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF386}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF105}

Mata kuliah ini memperkenalkan kepada mahasiswa terminologi dan konsep dasar
 yang akan banyak dipakai sepanjang kuliah di Teknik Informatika. Selain itu
 mata kuliah ini juga mempersiapkan dan membiasakan mahasiswa dengan suasana
 akademik yang khas perguruan tinggi seperti kedisiplinan, kerja sama,
 kemampuan menggunakan teknologi informasi dalam pembuatan tugas, kemampuan
 komunikasi, dsb.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF294}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF282}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{AIF451}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{APS302}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{EAA101}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{HasPrasyarat}

Mendefinisikan kelas-kelas yang memiliki prasyarat, terkustomisasi
```

```
 untuk seorang \texttt{Mahasiswa}. Jika ada tambahan, jangan lupa untuk
 mendaftarkannya di DEFAULT\_HASPRASYARAT\_CLASSES. Jika berubah package,
 jangan lupa mengupdate DEFAULT\_PRASYARAT\_PACKAGE.

Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{String DEFAULT\_HASPRASYARAT\_CLASSES} - Daftar dari nama kelas default seluruh turunan interface ini. Perlu
     didaftarkan
 manual, karena Java reflection tidak dapat mendeteksi otomatis.
\item \texttt{String DEFAULT\_PRASYARAT\_PACKAGE} - Package tempat menyimpan seluruh turunan standar interface ini. Perlu
     didefinisikan
 manual, karena Java reflection tidak dapat mendeteksi otomatis.
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}

Memeriksa prasyarat-prasyarat dari kuliah, spesifik untuk mahasiswa
 yang dituju. Jika ada pesan-pesan khusus, akan ditambahkan pada parameter
 reasonsContainer.

\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
prasyarat kuliah akan diperiksa spesifik pada mahasiswa ini
\item \texttt{List reasonsContainer} -
pesan-pesan terkait prasyarat akan ditambahkan di sini, jika ada.
\end{itemize}
\textbf{Return Value}: true jika seluruh prasyarat dipenuhi, false jika tidak.

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

\end{itemize}
\item \texttt{HasPraktikum}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{HasResponsi}



Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method. \item \texttt{Kelulusan implements HasPrasyarat}



Atribut yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{String PILIHAN\_WAJIB} -
\item \texttt{String WAJIB} -
\item \texttt{String AGAMA} -
\item \texttt{int MIN\_SKS\_LULUS} -
\item \texttt{int MIN\_PILIHAN\_WAJIB} -
\end{itemize}
\textit{Method-method} yang dimiliki kelas ini adalah sebagai berikut.
\begin{itemize}
\item \texttt{public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)}



\textbf{Parameter:}
\begin{itemize}
\item \texttt{Mahasiswa mahasiswa} -
\item \texttt{java.util.List reasonsContainer} -
\end{itemize}
\textbf{Return Value}: Tidak memiliki \textit{return value}

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

\textbf{Override}: \texttt{checkPrasyarat} dari kelas \texttt{Object}

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

D.3 Hasil PDF

1. `InfoMataKuliah`

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

   - `public int sks()`
     Jumlah bobot sks dari mata kuliah ini
     Parameter:
     – Tidak memiliki parameter *method*
     Return Value: jumlah bobot sks
     Exception: Tidak memiliki *exception*
   - `public String nama()`
     Nama mata kuliah ini
     Parameter:
     – Tidak memiliki parameter *method*
     Return Value: nama mata kuliah
     Exception: Tidak memiliki *exception*

2. `MataKuliahFactory`

   Kelas yang bertugas membuat kelas mata kuliah, dan menyimpannya untuk bisa digunakan kemudian (untuk hemat memori).

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `String DEFAULT_MATAKULIAH_PACKAGE` - Lokasi package untuk daftar mata kuliah
   - `MataKuliahFactory instance` - Singleton instance to factory.
   - `SortedMap mataKuliahCache` - Singleton instances untuk mata kuliah.

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public static MataKuliahFactory getInstance()`
     Parameter:
     – Tidak memiliki parameter *method*
     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public MataKuliah createMataKuliah(java.lang.String kode, int sks, java.lang.String nama)`
     Membuat baru atau mendapatkan mata kuliah, jika memiliki informasi nama dan jumlah SKS.
     Parameter:
     – `String kode` - kode mata kuliah

- `int sks` - jumlah SKS
- `String nama` - nama mata kuliah

Return Value: objek mata kuliah
Exception: Tidak memiliki *exception*

- `public MataKuliah createMataKuliah(java.lang.String kode)`
  Membuat baru atau mendapatkan mata kuliah, jika tidak memiliki informasi nama dan jumlah SKS.
  Parameter:
  - `String kode` - kode mata kuliah

  Return Value: objek mata kuliah
  Exception: IllegalStateException jika sks dan tidak sesuai dengan yang ada di kode

3. `Semester`

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `Semester UNKNOWN5` -
   - `Semester TRANSFER` -
   - `Semester PENDEK` -
   - `Semester GANJIL` -
   - `Semester GENAP` -
   - `int order` -

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public static Semester values()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public static Semester valueOf(java.lang.String name)`
     Parameter:
     - `String name` -

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public static Semester fromString(java.lang.String text)`
     Parameter:
     - `String text` -

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*

- `int getOrder()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

4. `TahunSemester implements Comparable`

   Menyimpan konstanta untuk semester beserta tahunnya di UNPAR.

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `String kodeTahunSemester` - Kode semester 3 dijit, sesuai DPS: ¡ul¿ ¡li¿2 dijit pertama berupa tahun, 2 dijit terakhir¡/li¿ ¡li¿dijit terakhir: 1 untuk ganjil, 2 untuk genap, 4 untuk pendek. ¡/ul¿

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public Semester getSemester()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public int getTahun()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `private static void validateKodeSemester(java.lang.String kodeTahunSemester)`
     Parameter:
     - `String kodeTahunSemester` -

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public String getKodeDPS()`
     Mendapatkan kode tahun/semester sesuai aturan di DPS.
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: kode tahun/semester sesuai aturan di DPS.
     Exception: Tidak memiliki *exception*
   - `public int compareTo(id.ac.unpar.siamodels.TahunSemester o)`
     Parameter:

– `TahunSemester o` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*
Override: `compareTo` dari kelas `Object`

- `public boolean equals(java.lang.Object arg0)`
  Parameter:
  – `Object arg0` -
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public String toString()`
  Parameter:
  – Tidak memiliki parameter *method*
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

5. `Mahasiswa`

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `String npm` -
   - `String nama` -
   - `List riwayatNilai` -
   - `URL photoURL` -
   - `List jadwalKuliahList` -
   - `SortedMap nilaiTOEFL` -

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public String getNama()`
     Parameter:
     – Tidak memiliki parameter *method*
     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public void setNama(java.lang.String nama)`
     Parameter:
     – `String nama` -
     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public String getNpm()`
     Parameter:

– Tidak memiliki parameter *method*

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*

- `public URL getPhotoURL()`
  Parameter:
  – Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setPhotoURL(java.net.URL photoURL)`
  Parameter:
  – `URL photoURL` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public List getJadwalKuliahList()`
  Parameter:
  – Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setJadwalKuliahList(java.util.List jadwalKuliahList)`
  Parameter:
  – `java.util.List jadwalKuliahList` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public String getEmailAddress()`
  Parameter:
  – Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public List getRiwayatNilai()`
  Parameter:
  – Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public SortedMap getNilaiTOEFL()`
  Parameter:
  – Tidak memiliki parameter *method*

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

- `public void setNilaiTOEFL(java.util.SortedMap nilaiTOEFL)`
  Parameter:
  - `java.util.SortedMap nilaiTOEFL` -

  Return Value: Tidak memiliki *return value*

  Exception: Tidak memiliki *exception*

- `public double calculateIPKLulus()`
  Menghitung IPK mahasiswa sampai saat ini, dengan aturan: ¡ul¿ ¡li¿Kuliah yang tidak lulus tidak dihitung ¡li¿Jika pengambilan beberapa kali, diambil ¡em¿nilai terbaik¡/em¿. ¡/ul¿ Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: IPK Lulus, atau DoubleNaN jika belum mengambil satu kuliah pun.

  Exception: Tidak memiliki *exception*

- `public double calculateIPLulus()`
  Menghitung IP mahasiswa sampai saat ini, dengan aturan: ¡ul¿ ¡li¿Kuliah yang tidak lulus tidak dihitung ¡li¿Jika pengambilan beberapa kali, diambil ¡em¿nilai terbaik¡/em¿. ¡/ul¿ Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: IPK Lulus, atau DoubleNaN jika belum mengambil satu kuliah pun.

  Exception: Tidak memiliki *exception*

- `public double calculateIPTempuh(boolean lulusSaja)`
  Menghitung IP mahasiswa sampai saat ini, dengan aturan: ¡ul¿ ¡li¿Perhitungan kuliah yang tidak lulus ditentukan parameter ¡li¿Jika pengambilan beberapa kali, diambil ¡em¿nilai terbaik¡/em¿. ¡/ul¿

  Parameter:
  - `boolean lulusSaja` - set true jika ingin membuang mata kuliah tidak lulus, false jika ingin semua (sama dengan "IP N. Terbaik" di DPS) Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Return Value: IPK Lulus, atau DoubleNaN jika belum mengambil satu kuliah pun.

  Exception: Tidak memiliki *exception*

- `public double calculateIPKumulatif()`

  Menghitung IP Kumulatif mahasiswa sampai saat ini, dengan aturan: ¡ul¿ ¡li¿Jika pengambilan beberapa kali, diambil semua. ¡/ul¿ Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Parameter:

  – Tidak memiliki parameter *method*

  Return Value: IPK Lulus, atau DoubleNaN jika belum mengambil satu kuliah pun.

  Exception: Tidak memiliki *exception*

- `public double calculateIPKTempuh(boolean lulusSaja)`

  Menghitung IPK mahasiswa sampai saat ini, dengan aturan: ¡ul¿ ¡li¿Perhitungan kuliah yang tidak lulus ditentukan parameter ¡li¿Jika pengambilan beberapa kali, diambil ¡em¿nilai terbaik¡/em¿. ¡/ul¿

  Parameter:

  – `boolean lulusSaja` - set true jika ingin membuang mata kuliah tidak lulus Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Return Value: IPK Lulus, atau DoubleNaN jika belum mengambil satu kuliah pun.

  Exception: Tidak memiliki *exception*

- `public double calculateIPS()`

  Menghitung IPS semester terakhir sampai saat ini, dengan aturan: ¡ul¿ ¡li¿Kuliah yang tidak lulus ¡em¿dihitung¡/em¿. ¡/ul¿ Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Parameter:

  – Tidak memiliki parameter *method*

  Return Value: nilai IPS sampai saat ini

  Exception: ArrayIndexOutOfBoundsException jika belum ada nilai satupun

- `public int calculateSKSLulus()`

  Menghitung jumlah SKS lulus mahasiswa saat ini. Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Parameter:

  – Tidak memiliki parameter *method*

  Return Value: SKS Lulus

  Exception: Tidak memiliki *exception*

- `public int calculateSKSTempuh(boolean lulusSaja)`

  Menghitung jumlah SKS tempuh mahasiswa saat ini. Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah!

  Parameter:

  - `boolean lulusSaja` - set true jika ingin membuang SKS tidak lulus

  Return Value: SKS tempuh

  Exception: Tidak memiliki *exception*

- `public Set calculateTahunSemesterAktif()`

  Mendapatkan seluruh tahun semester di mana mahasiswa ini tercatat sebagai mahasiswa aktif, dengan strategi memeriksa riwayat nilainya. Jika ada satu nilai saja pada sebuah tahun semester, maka dianggap aktif pada semester tersebut.

  Parameter:

  - Tidak memiliki parameter *method*

  Return Value: kumpulan tahun semester di mana mahasiswa ini aktif

  Exception: Tidak memiliki *exception*

- `public boolean hasLulusKuliah(java.lang.String kodeMataKuliah)`

  Memeriksa apakah mahasiswa ini sudah lulus mata kuliah tertentu. Kompleksitas O(n). Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah mengandung nilai per mata kuliah! Note: jika yang dimiliki adalah MataKuliah, gunakanlah MataKuliahgetKode().

  Parameter:

  - `String kodeMataKuliah` - kode mata kuliah yang ingin diperiksa kelulusannya.

  Return Value: true jika sudah pernah mengambil dan lulus, false jika belum

  Exception: Tidak memiliki *exception*

- `public boolean hasTempuhKuliah(java.lang.String kodeMataKuliah)`

  Memeriksa apakah mahasiswa ini sudah pernah menempuh mata kuliah tertentu. Kompleksitas O(n). Sebelum memanggil method ini, `getRiwayatNilai()` harus sudah ada isinya! Note: jika yang dimiliki adalah MataKuliah, gunakanlah MataKuliahgetKode().

  Parameter:

  - `String kodeMataKuliah` - kode mata kuliah yang ingin diperiksa.

  Return Value: true jika sudah pernah mengambil, false jika belum

  Exception: Tidak memiliki *exception*

- `public int getTahunAngkatan()`
  Mendapatkan tahun angkatan mahasiswa ini, berdasarkan NPM nya
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: tahun angkatan
  Exception: Tidak memiliki *exception*
- `public String toString()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

6. `Mahasiswa.Nilai`

   Merepresentasikan nilai yang ada di riwayat nilai mahasiswa

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `TahunSemester tahunSemester` - Tahun dan Semester kuliah ini diambil
   - `MataKuliah mataKuliah` - Mata kuliah yang diambil
   - `Character kelas` - Kelas kuliah
   - `Double nilaiART` - Nilai ART
   - `Double nilaiUTS` - Nilai UTS
   - `Double nilaiUAS` - Nilai UAS
   - `Character nilaiAkhir` - Nilai Akhir

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public MataKuliah getMataKuliah()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public Character getKelas()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public Double getNilaiART()`
     Parameter:

- Tidak memiliki parameter *method*

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*

- `public Double getNilaiUTS()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public Double getNilaiUAS()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public Character getNilaiAkhir()`
  Mengembalikan nilai akhir dalam bentuk huruf (A, B, C, D, ..., atau K)
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: nilai akhir dalam huruf, atau null jika tidak ada.
  Exception: Tidak memiliki *exception*
- `public Double getAngkaAkhir()`
  Mendapatkan nilai akhir dalam bentuk angka
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: nilai akhir dalam angka, atau null jika getNilaiAkhir() mengembalikan 'K' atau null
  Exception: Tidak memiliki *exception*
- `public TahunSemester getTahunSemester()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public int getTahunAjaran()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

- `public Semester getSemester()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public String toString()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

7. `Mahasiswa.Nilai.ChronologicalComparator implements Comparator`

   Pembanding antara satu nilai dengan nilai lainnya, secara kronologis waktu pengambilan.

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

   - `public int compare(id.ac.unpar.siamodels.Mahasiswa.Nilai o1, id.ac.unpar.siamodels.Mahasiswa.Nilai o2)`
     Parameter:
     - `Mahasiswa.Nilai o1` -
     - `Mahasiswa.Nilai o2` -

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
     Override: `compare` dari kelas `Object`

8. `JadwalKuliah`

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `MataKuliah mataKuliah` -
   - `Character kelas` -
   - `DayOfWeek hari` -
   - `LocalTime waktuMulai` -
   - `LocalTime waktuSelesai` -
   - `String lokasi` -
   - `Dosen pengajar` -

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public MataKuliah getMataKuliah()`
     Parameter:

– Tidak memiliki parameter *method*

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*

- `public void setMataKuliah(id.ac.unpar.siamodels.MataKuliah mataKuliah)`
  Parameter:
  – `MataKuliah mataKuliah` -
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public Character getKelas()`
  Parameter:
  – Tidak memiliki parameter *method*
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setKelas(java.lang.Character kelas)`
  Parameter:
  – `Character kelas` -
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public DayOfWeek getHari()`
  Parameter:
  – Tidak memiliki parameter *method*
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setHari(java.time.DayOfWeek hari)`
  Parameter:
  – `DayOfWeek hari` -
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public LocalTime getWaktuMulai()`
  Parameter:
  – Tidak memiliki parameter *method*
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setWaktuMulai(java.time.LocalTime waktuMulai)`
  Parameter:
  – `LocalTime waktuMulai` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*

- `public LocalTime getWaktuSelesai()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setWaktuSelesai(java.time.LocalTime waktuSelesai)`
  Parameter:
  - `LocalTime waktuSelesai` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public String getLokasi()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setLokasi(java.lang.String lokasi)`
  Parameter:
  - `String lokasi` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public Dosen getPengajar()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setPengajar(id.ac.unpar.siamodels.Dosen pengajar)`
  Parameter:
  - `Dosen pengajar` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public String getWaktuString()`
  Parameter:
  - Tidak memiliki parameter *method*

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

- `public static DayOfWeek indonesianToDayOfWeek(java.lang.String indonesian)`
  Converts Indonesian day names to `DayOfWeek` enumeration.
  Parameter:
  - `String indonesian` - the day name in Indonesian

  Return Value: DayOfWeek object or null if not found.
  Exception: Tidak memiliki *exception*

9. `MataKuliah`

   Atribut yang dimiliki kelas ini adalah sebagai berikut.

   - `String kode` -
   - `String nama` -
   - `Integer sks` -

   Method-method yang dimiliki kelas ini adalah sebagai berikut.

   - `public String getKode()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public String getNama()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*
   - `public Integer getSks()`
     Parameter:
     - Tidak memiliki parameter *method*

     Return Value: Tidak memiliki *return value*
     Exception: Tidak memiliki *exception*

10. `Dosen`

    Atribut yang dimiliki kelas ini adalah sebagai berikut.

    - `String nik` -
    - `String nama` -

    Method-method yang dimiliki kelas ini adalah sebagai berikut.

    - `public String getNik()`
      Parameter:

– Tidak memiliki parameter *method*

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*

- `public void setNik(java.lang.String nik)`
  Parameter:
  – `String nik` -
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public String getNama()`
  Parameter:
  – Tidak memiliki parameter *method*
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public void setNama(java.lang.String nama)`
  Parameter:
  – `String nama` -
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
- `public boolean equals(java.lang.Object arg0)`
  Parameter:
  – `Object arg0` -
  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*

11. `MKU008`

    Mendalami perilaku sehari-hari yang baik dalam bermasyarakat.

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

12. `IIE210`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

13. `AIF203 implements HasPrasyarat`

    Mata kuliah ini memperkenalkan kepada mahasiswa konsep struktur diskret yang digunakan pada bidang informatika diantaranya graph, pohon dan finite state machine

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:

- `Mahasiswa mahasiswa` -
- `java.util.List reasonsContainer` -

Return Value: Tidak memiliki *return value*

Exception: Tidak memiliki *exception*

Override: `checkPrasyarat` dari kelas `MataKuliah`

14. `AIF311 implements HasPrasyarat`

15. `AIF311 implements HasPraktikum`

    Kuliah Pemrograman Fungsional bertujuan untuk: 1. memperkenalkan paradigma pemrograman fungsional, yaitu sebuah pemrograman yang didasarkan pada konsep pemetaan dan fungsi matematika. Penyelesaian suatu masalah didasari atas aplikasi dari fungsi-fungsi tersebut. 2. memberikan dasar-dasar pemrograman fungsional dengan menggunakan bahasa fungsional Haskell.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

      Parameter:

      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*

      Exception: Tidak memiliki *exception*

      Override: `checkPrasyarat` dari kelas `MataKuliah`

16. `AIF192`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

17. `AIF468`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

18. `IIE103`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

19. `AIF385`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

20. `AIF106`

    Mata kuliah ini memberikan pengetahuan tentang cara kerja komputer, dimulai dari representasi data dan berbagai macam operasinya. Selanjutnya, juga diperkenalkan bagaimana merepresentasikan suatu fungsi dalam

rangkaian gerbang logika, dan bagaimana menyederhanakannya. Berbagai rangkaian dasar yang digunakan di dalam komputer juga dipekenalkan. Mahasiswa juga akan mempelajari komponen komputer, misalnya register dan memori.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

21. `AIF281`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

22. `EAA102`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

23. `AIF405 implements HasPrasyarat`

24. `AIF405 implements HasPraktikum`

    Mata kuliah ini merupakan lanjutan dari Projek Sistem Informasi 1 dan memberikan kesempatan bagi mahasiswa untuk melanjutkan/mengembangkan perancangan sitem pada organisasi studi kasus, mengimplementasikan rancangan dan melakukan pengujian perangkat lunak;

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

25. `APS182`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

26. `MKU004`

    Fenomenologi Agama merupakan bagian yang tak terpisahkan dari kajian filosofis, kritis, rasional, dan obyektif mengenai substansi ajaran agama. Fenomenologi merupakan sebuah disiplin ilmu yang secara kritis-rasional mengkaji fenomena dan dinamika kehidupan manusia beragama, dari upaya menjadikan Tuhan sebagai tujuan sesembahan sampai menempatkan Tuhan sebagai instrumen legitimasi untuk melakukan tindakan yang justru bertolak belakang dengan kehendak Tuhan yang disembah. Sehubungan dengan itu, kritik konstruktif terhadap perilaku manusia beragama

menjadi salah satu poin utama dalam mata kuliah ini. Kesediaan untuk melakukan otoritik terhadap agama sendiri erat terkait dengan upaya menemukan kembali nilai sejati agama atau otentisitas hidup beragama.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

27. `MKU012`

    Perkuliahan logika ditujukan untuk memberikan dasar-dasar ketrampilan berpikir rasional dan sistematik. Isinya mencakup ketrampilan berpikir deduktif dan induktif, seperti silogisme, argumen analogikal dan generalisasi induktif. Pembahasan teoretis disertai pula dengan pelatihan praktis yang diarahkan pada proses berpikir. Untuk menajamkan kemampuan berpikir tersebut, mahasiswa dilatih pula mengidentifikasi kerancuan-kerancuan (fallacies) yang sering dijumpai baik dalam kehidupan sehari-hari maupun dalam konteks akademik.

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

28. `AIF389`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

29. `AMS190`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

30. `AMS191`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

31. `AMS200`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

32. `AIF330`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

33. `AIF388`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

34. `AIF465`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

35. `AIF453 implements HasPrasyarat`

    Mata kuliah inimemperkenalkan kebutuhan organisasi terhadap sistem business intelligent (BI) dan pemanfaatan BI untuk organisasi; memperkenalkan konsep sistem business intelligent dan komponennya; Mempelajari tenik-teknik analisis data bisnis dan visualisasi hasil analisis; Mempelajari konsep data warehouse dan perancangannya dan fungsi OLAP; Mempraktekkan teknik-teknik analisis data dan visualisasi hasil analisis.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

36. `AIF280`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

37. `AIF445 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

38. `AIF469 implements HasPrasyarat`

    Mata kuliah ini mengajarkan kepada mahasiswa teknik-teknik untuk membuat layanan berbasis web. Mahasiswa diperkenalkan dengan standar-standar seperti HTTP, XML, JSON dan diajarkan untuk memanfaatkannya dalam membuat maupun menggunakan layanan pihak ketiga. Dalam kuliah ini, juga akan diperkenalkan minimal satu layanan pihak ketiga yang dapat dimanfaatkan mahasiswa, seperti Google Places Web Service.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

39. `AIF486`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

40. `IIE207`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

41. `AIF202 implements HasPrasyarat`

42. `AIF202 implements HasResponsi`

43. `AIF202 implements HasPraktikum`

    Mata kuliah ini memperkenalkan kepada mahasiswa beberapa algoritma dan struktur data, alternatif cara implementasinya, dan analisis kompleksitas waktunya. Mahasiswa diberikan beberapa masalah komputasi yang harus diselesaikan dengan menggunakan algoritma atau struktur data yang sudah diperkenalkan dan mengimplementasikannya dalam bahasa pemrograman Java.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*

      Exception: Tidak memiliki *exception*

      Override: `checkPrasyarat` dari kelas `MataKuliah`

44. `AIF347`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

45. `MKU009`

    Tujuan dari mata kuliah ini adalah untuk mendalami keterampilan berbahasa Indonesia, agar mampu mengkomunikasikan hasil pemikiran serta meningkatkan keterampilan dalam menyusun karya ilmiah. Mata kuliah Bahasa Indonesia ini dimulai dengan mempelajari penulisan kata baku dan non baku serta pengungkapan pikiran dengan pungtuasi yang benar. Selanjutnya dipelajari penyusunan kalimat yang baku serta menghubungkan kalimat-kalimat yang padu dalam menuangkan gagasan dalam sebuah paragraf. Selain itu, dalam matakuliah ini dipelajari cara menyusun surat dinas yang jelas dan komunikatif. Di akhir kuliah ini, mahasiswa diberi tugas penyusunan makalah dengan benar.

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

46. `ESM203`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

47. `APS402 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

48. `AIF306 implements HasPrasyarat`

    Mata kuliah ini bertujuan untuk memberikan pengalaman bagi mahasiswa dalam mengerjakan proyek dengan teknologi-teknologi terkini, secara berkelompok. Teknologi-teknologi yang digunakan pada kuliah ini tidak spesifik dan dapat berubah seiring perkembangan teknologi maupun disesuaikan dengan kompetensi dosen pengajar. Beberapa teknologi yang dapat dimanfaatkan antara lain: DVCS tool menggunakan Git + Github, Mobile native app (Android, iOS, dll), dan responsive web design.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

49. `AKS122`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

50. `MKU002`

    Pendidikan Kewarganegaraan menjelaskan pentingnya pemahaman tentang identitas nasional Indonesia, hak dan kewajiban warga negara Indonesia serta hubungannya dengan hak dan kewajiban asasi manusia. Materi kuliah mencakup juga wawasan nusantara, ketahanan nasional, politik

dan strategi nasional, serta implementasinya dalam kehidupan bermasyarakat, berbangsa dan bernegara kesatuan Republik Indonesia.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

51. `AIF462`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

52. `AIF360 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

53. `AIF337`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

54. `AIF458 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

55. `AIF301 implements HasPrasyarat`

    Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar tentang *sistem cerdas dan komponen-komponennya. ” ”Terdapat 4 topik utama yang dibahas yaitu teknik pencarian untuk *penyelesaian masalah, representasi pengetahuan dalam sistem *cerdas, pemodelan ketidakpastian dalam masalah dan teknik pembelajaran mesin.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

  Parameter:

  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*

  Exception: Tidak memiliki *exception*

  Override: `checkPrasyarat` dari kelas `MataKuliah`

56. `AIF182`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

57. `ESM204`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

58. `AIF205 implements HasPrasyarat`

    Mata kuliah ini memperkenalkan kepada mahasiswa arsitektur komputer sederhana, modern, dan Advance. Perbedaan, kelebihan dan kekurangan untuk masing-masing arsitektur. Selain itu mahasiswa juga mempelajari cara kerja dari komponen-komponen komputer, terutama memory, cache, system BUS dan input/output.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

      Parameter:

      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*

      Exception: Tidak memiliki *exception*

      Override: `checkPrasyarat` dari kelas `MataKuliah`

59. `AIF317 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

      Parameter:

      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*
Override: `checkPrasyarat` dari kelas `MataKuliah`

60. `AIF383`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

61. `AIF442 implements HasPraktikum`

62. `AIF442 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

63. `ESM101`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

64. `AIF403 implements HasPrasyarat`

    1. Memberikan wawasan kepada mahasiswa tentang kemunculan dan pemanfaatan teknologi baru, khususnya yang berkaitan dengan komputer, dan dampaknya terhadap masyarakat luas. 2. Memberikan kesadaran dan panduan bersikap kepada mahasiswa dalam menghadapi gejolak yang disebabkan oleh munculnya teknologi baru, khususnya yang berkaitan dengan komputer.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

65. `AIF402 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

66. `AIF455`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

67. `AIF443`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

68. `AIF101 implements HasPraktikum`

    Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar pemrograman seperti pengulangan dan percabangan, konsep dasar penyimpanan data kontigu menggunakan array, konsep dasar pemrograman berorientasi objek seperti kelas & objek, method, dll, termasuk di dalamnya 4 prinsip dasar pemrograman berorientasi objek : data abstraction, encapsulation, inheritance dan polymorphism. Selain, itu diberikan masalah-masalah komputasi sederhana yang harus diselesaikan menggunakan konsep-konsep yang sudah diperkenalkan dan mengimplementasikannya menggunakan bahasa pemrograman Java

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

69. `AIF382`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

70. `AIF480`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

71. `AIF316 implements HasPrasyarat`

72. `AIF316 implements HasPraktikum`

    Mata kuliah ini memperkenalkan konsep-konsep dasar komputasi paralel, dimana sebuah program yang berjalan secara paralel harus memiliki safety property dan liveness property. Mahasiswa dikenalkan dengan beberapa teknik pemrograman multi-thread seperti lock, monitor, barrier, thread

pool, dan sebagainya, yang diimplementasikan dalam bahasa pemrograman Java. Mahasiswa juga dikenalkan dengan beberapa metode untuk menganalisis kebenaran program baik secara matematis maupun secara praktis dengan bantuan model checker.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

73. `AIF438 implements HasPrasyarat`

    Mata kuliah ini: Memperkenalkan karakteristik dan teknik visualisasi dari berbagai jenis data yang dapat dianalisis dengan teknik-teknik data mining; mempelajari teknik-teknik penyiapan data untuk berbagai jenis data dan teknik data mining; mempraktekkan teknik-teknik penyiapan data untuk menganalisis data nyata/simulasi dengan memanfaatkan perangkat lunak aplikasi.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

74. `AIF204 implements HasPrasyarat`

75. `AIF204 implements HasPraktikum`

    Mata kuliah ini memperkenalkan konsep dan arsitektur DBMS, mengajarkan aljabar relasional dan SQL serta pemanfaatannya pada pemrograman kueri sederhana s/d relatif kompleks. Selain itu, mata kuliah ini juga mengajarkan dan mempraktekkan perancangan basisdata untuk masalah sederhana (lingkup kecil) termasuk pengembangan program aplikasinya;

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

  Parameter:

  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*

  Exception: Tidak memiliki *exception*

  Override: `checkPrasyarat` dari kelas `MataKuliah`

76. `AIF341 implements HasPraktikum`

    Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar jaringan dan aplikasinya di kehidupan sehari-hari. Mahasiswa dikenalkan dengan teknologi-teknologi terbaru di bidang jaringan, sehingga mahasiswa memiliki pengetahuan yang dapat digunakan dalam kehidupan sehari-hari. Selain itu mahasiswa juga diperkenalkan dengan NetAcad, sebuah layanan dari Cisco yang dapat digunakan untuk memenuhi segala macam kebutuhan terkait dengan Cisco Academy.

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

77. `AIF183`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

78. `AKS124`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

79. `AIF459`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

80. `AIF336`

    Mata kuliah ini merupakan mata kuliah lanjutan dari mata kuliah Keamanan Informasi, dengan titik berat pada materi kriptografi. Mata kuliah ini memperkenalkan tambahan konsep kriptografi, misalnya tentang otentikasi yaitu otentikasi entitas, manajemen kunci, dan bentuk lain dari metode merahasiakan pesan, yaitu dengan menggunakan secret sharing. Selanjutnya, diperkenalkan juga penggunaan kriptografi pada protokol-protokol yang sebenarnya banyak digunakan sehari-hari, misalnya pada e-cash, auction, dan electronic voting.

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

81. `AIF463`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

82. `AIF208 implements HasPrasyarat`

    Mata kuliah ini memperkenalkan kepada mahasiswa tahapan rekayasa perangkat lunak, terutama dengan paradigma berorientasi objek, dilengkapi dengan pengenalan tentang manajemen proyek perangkat lunak. Selain, itu diberikan deskripsi proyek berskala kecil yang harus dikerjakan oleh mahasiswa dalam kelompok dengan menerapkan teori yang telah dipelajarinya.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

83. `MKU003`

    Mata kuliah ini membentuk karakteristik mahasiswa sebagai manusia yang memiliki religiusitas melalui pendalaman akan makna agama dan beragama, mendeteksi dinamika Wahyu Tuhan dan iman mereka, memahami relasi dengan Tuhan dan sesama, mengenal makna keselamatan dalam konteks Kerajaan Allah, dan mampu menyatakan ajaran Gereja dalam pelayanan terhadap orang miskin dan terlantar.

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

84. `APS309`

    APS302 atau APS309 ?

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

85. `AIF335`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

86. `AIF362 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      - `Mahasiswa mahasiswa` -

– `java.util.List reasonsContainer` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*
Override: `checkPrasyarat` dari kelas `MataKuliah`

87. `AIF460`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

88. `AIF358`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

89. `AMS100`

    Sistem Bilangan, Fungsi, Limit dan Kekontinuan Fungsi, Turunan, Integral, Penggunaan Integral, Sistem Persamaan Linear, Determinan, Vektor, Nilai dan Vektor Eigen.

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

90. `AIF401 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      – `Mahasiswa mahasiswa` -
      – `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

91. `AIF456`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

92. `SIR104`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

93. `AIF339 implements HasPrasyarat`

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      – `Mahasiswa mahasiswa` -

– `java.util.List reasonsContainer` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*
Override: `checkPrasyarat` dari kelas `MataKuliah`

94. `AIF102 implements HasPrasyarat`

95. `AIF102 implements HasPraktikum`

    Mata kuliah ini memperkenalkan berbagai algoritma dan teknik-teknik penyelesaian masalah komputasi seperti rekursif, sorting, teknik divide dan conquer, serta exhaustive search. Selain itu, pada kuliah ini juga dikenalkan berbagai struktur data yang dapat digunakan untuk mendukung penyelesaian masalah komputasi seperti ADT List, Stack dan Queue. Baik algoritma maupun struktur data yang dikenalkan harus dapat diimplementasikan dan digunakan oleh mahasiswa untuk menyelesaikan masalah dengan menggunakan suatu bahasa pemrograman berorientasi objek.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
      Parameter:
      – `Mahasiswa mahasiswa` -
      – `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*
      Exception: Tidak memiliki *exception*
      Override: `checkPrasyarat` dari kelas `MataKuliah`

96. `AIF381`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

97. `AIF483`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

98. `AIF315 implements HasPrasyarat`

99. `AIF315 implements HasPraktikum`

    Mata kuliah ini memperkenalkan konsep dan lingkungan pemrograman berbasis web, kemudian belajar membuat aplikasi berbasis web menggunakan HTML5, CSS, Java Script dan PHP. Untuk meningkatkan keterampilan pemrograman dilengkapi dengan praktikum. Sedangkan untuk mendapatkan pengalaman penerapan konsep diberikan tugas besar membuat program berbasis web dengan kasus yang ditentukan oleh mahasiswa.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

  Parameter:

  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*

  Exception: Tidak memiliki *exception*

  Override: `checkPrasyarat` dari kelas `MataKuliah`

100. `AIF342 implements HasPraktikum`

101. `AIF342 implements HasPrasyarat`

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

     - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

       Parameter:

       - `Mahasiswa mahasiswa` -
       - `java.util.List reasonsContainer` -

       Return Value: Tidak memiliki *return value*

       Exception: Tidak memiliki *exception*

       Override: `checkPrasyarat` dari kelas `MataKuliah`

102. `IIE214`

     Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

103. `AIF303 implements HasPrasyarat`

     Mempelajari Konsep Data, Informasi, Pengetahuan, Sistem Informasi, proses dan pemodelan bisnis, jenis-jenis sistem informasi, untuk mendukung pengambilan keputusan. Mempelajari trend Teknologi Informasi, tahap-tahap pembangunan sistem informasi. Mempelajari pengantar : EIS, e-bisnis/e-commerce, Business Intelligence, Cloud Computing dan Mobile Applications

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

     - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

       Parameter:

       - `Mahasiswa mahasiswa` -
       - `java.util.List reasonsContainer` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*
Override: `checkPrasyarat` dari kelas `MataKuliah`

104. `AIF302 implements HasPrasyarat`

Mata kuliah ini melatih mahasiswa dalam menulis ilmiah serta memperkenalkan metodologi penelitian serta kakas untuk menulis ilmiah.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

105. `AIF181`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

106. `AIF210`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

107. `AIF343`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

108. `AIF206 implements HasPrasyarat`

Mata kuliah ini memperkenalkan kepada mahasiswa mengenai konsep sistem operasi, jenis-jenis sistem operasi yang digunakan dalam kehidupan sehari-hari dan beberapa perangkat keras yang dibutuhkan pada komputer. Selain itu juga mempelajari mengenai teknik dan algoritma yang digunakan dalam pengelolaan sistem operasi.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*
Override: `checkPrasyarat` dari kelas `MataKuliah`

109. `AIF314 implements HasPrasyarat`

110. `AIF314 implements HasPraktikum`

Kuliah ini merupakan kelanjutan dari kuliah Manajemen Informasi Basis-data. Pada perkuliahan ini, mahasiswa akan mempelajari teknik-teknik pengelolaan basis data dan membuat program dengan basis data yang optimal/efisien.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

111. `AIF380`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

112. `AIF103`

Mata kuliah ini merupakan salah satu cara untuk mencapai kompetensi dasar tentang matematika diskrit yang prinsipnya banyak digunakan dalam bidang ilmu komputer. Selain itu, kuliah ini juga merupakan cara untuk membentuk pola pikir logis yang dibutuhkan untuk menempuh kuliah-kuliah di tingkat yang lebih tinggi.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

113. `AIF292`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

114. `AIF441 implements HasPraktikum`

115. `AIF441 implements HasPrasyarat`

Mata kuliah ini memperkenalkan kepada mahasiswa konsep jaringan lanjut terutama di layer data link dan layer network. Materi utama dari mata kuliah ini adalah pengembangan jaringan dan pengenalan fungsi-fungsi yang terdapat pada alat jaringan Cisco yang berkaitan dengan layer 2 dan layer 3.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

116. `AIF457 implements HasPrasyarat`

Mata kuliah ini memperkenalkan konsep kewirausahaan dengan memanfaatkan teknologi, khususnya teknologi informasi, sebagai basis usaha dan inovasi produk/jasa; Mempelajari teknik mencari peluang dan merumuskan bidang usaha spesifik yang akan diterjuni; Mempelajari konsep manajemen pemasaran, keuangan dan SDM dalam kaitannya dengan berwira-usaha di bidang TI; Menyusun proposal bisnis untuk berwira-usaha di bidang TI dan mempresentasikannya.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

117. `MKU001`

Mata Kuliah Pendidikan Pancasila berupaya menelaah/mengkaji berbagai fenomena kehidupan bangsa dan Negara Indonesia sebagai sebuah ruang publik dengan menggunakan pendekatan hermeneutika (filsafat) dan pendidikan nilai (pedagogik). Dengan bantuan hermenutika mahasiswa diajak berpikir kritis terhadap segala bentuk ideologisme Pancasila dan melalui pendidikan nilai mahasiswa dilatih untuk memiliki nilai Pancasila. Nilai pengembangan diri intra-personal dan relasi interpersonal dapat tertanam melalui pendidikan Pancasila yang tujuannya adalah membangun kepribadian (character building) manusia Indonesia yang utuh, baik menyangkut aspek kognitif, afektif, maupun psikomotor.

Dengan demikian, Pendidikan Pancasila mengajak mahasiswa menilai realitas ruang publik sehari-hari secara mandiri dengan panduan nilai-nilai etis Pancasila.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

118. `ESA101`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

119. `AMS290`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

120. `AIF461`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

121. `AIF318 implements HasPrasyarat`

122. `AIF318 implements HasPraktikum`

    Mata kuliah ini memperkenalkan konsep perangkat mobile dan pemrograman pada perangkat mobile. Pemrograman dikhususkan pada lingkungan J2ME dan Android. Untuk meningkatkan keterampilan pemrograman dilengkapi dengan praktikum. Sedangkan untuk mendapatkan pengalaman penerapan konsep diberikan tugas implementasi suatu kasus pada lingkungan mobile-cloud dengan kasus yang sudah ditentukan.

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

    - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

      Parameter:

      - `Mahasiswa mahasiswa` -
      - `java.util.List reasonsContainer` -

      Return Value: Tidak memiliki *return value*

      Exception: Tidak memiliki *exception*

      Override: `checkPrasyarat` dari kelas `MataKuliah`

123. `AIF334`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

124. `ESM105`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

125. `AIF450`

    Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

126. `AIF446`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

127. `AIF104`

Mata kuliah ini memberikan pengetahuan tentang logika yang digunakan di dalam ilmu komputer. Dalam kuliah ini, mahasiswa belajar untuk bisa memodelkan suatu kalimat dalam kehidupan sehari-hari, ke dalam kalimat dengan sintaks tertentu, yang hanya memiliki satu arti. Lalu, diperkenalkan juga, bagaimana mengartikan suatu kalimat (benar atau salah) dan bagaimana menentukan sifat dari kalimat tersebut.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

128. `AIF387`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

129. `AIF313 implements HasPraktikum`

Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar pembuatan grafik dengan menggunakan komputer seperti mengenal berbagai algoritma pembuatan primitif 2 dimensi seperti titik, garis, lingkaran, elips, berbagai macam bentuk kurva, fraktal, konsep warna (RGB), dasar-dasar grafika 3 dimensi seperti pewarnaan, pencahayaan, pemberian tekstur pada objek, transformasi, animasi, dan sebagainya. Selain, itu diberikan masalah-masalah komputasi sederhana yang harus diselesaikan menggunakan konsep-konsep yang sudah diperkenalkan dan mengimplementasikannya menggunakan bahasa pemrograman Java.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

130. `AIF344 implements HasPrasyarat`

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

131. `AIF201 implements HasPrasyarat`

132. `AIF201 implements HasPraktikum`

133. `AIF201 implements HasResponsi`

Mata kuliah ini memperkenalkan prinsip-prinsip yang digunakan dalam melakukan analisa serta desain prorgram berorientasi objek. Di samping itu, mahasiswa juga belajar menggunakan kakas berupa diagram UML (Unified Modelling Language) sehingga dapat mengkomunikasikan desain secara visual. Mahasiswa juga akan mengenal beberapa software design pattern dari Gang of Four. Terakhir, mahasiswa akan belajar mengenai konsep MVC (Model-View-Controller) yang menjadi dasar dari banyak framework masa kini. Bahasa yang digunakan adalah bahasa Java, namun diusahakan tetap umum sehingga dapat diaplikasikan pada bahasa yang lain.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

134. `AIF352`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

135. `AIF305 implements HasPrasyarat`

Mata kuliah ini memperkenalkan kepada mahasiswa konsep dasar jaringan komputer dengan menggunakan top-down. Selain itu mengajarkan juga kepada mahasiswa mengenai aplikasi-aplikasi berbasis jaringan sehingga diharapkan mahasiswa dapat membuat aplikasi berbasis jaringan dengan menggunakan socket. Pada akhirnya, mahasiswa akan ditugaskan untuk membangun jaringan komputer LAN, baik menggunakan kabel maupun nirkabel.

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

Return Value: Tidak memiliki *return value*
Exception: Tidak memiliki *exception*
Override: `checkPrasyarat` dari kelas `MataKuliah`

136. `MKU010`

Mata kuliah ini difokuskan pada pemahaman sumber referensi dalam Bahasa Inggris dan pengembangan kosakata Bahasa Inggris (vocabularies). Hampir keseluruhan waktu perkuliahan didedikasikan untuk menjelaskan metode mengekstraksi isi bacaan secara tepat dan melatih mahasiswa untuk menerapkan metode tersebut seraya menambah kosakata-kosakata baru. Mahasiswa juga dilatih untuk mempresentasikan hasil pemahamannya akan isi bahan bacaan.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

137. `MKU011`

Mata kuliah estetika memberi pemahaman konseptual filosofis "seni" dalam khasanah keilmuan, pembentukan kesadaran ekologis juga dalam proses pembudayaan dan peradaban. Mata kuliah ini akan menjadi fondasi bagi mahasiswa untuk memahami dan mempraktekkan seni dari sudut pandang filsafat, sejarah, kultural, dan global. Melalui mata kuliah ini, mahasiswa mempelajari mengenai dunia manusia (manusia dan pikirannya), pluralitas dan relativitas seni, serta aliran-aliran seni rupa Barat.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

138. `AIF332 implements HasPrasyarat`

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `MataKuliah`

139. `AIF304 implements HasPrasyarat`

140. `AIF304 implements HasPraktikum`

141. `AIF304 implements HasResponsi`

Mata kuliah ini memberikan kesempatan bagi mahasiswa untuk memperdalam konsep tentang pengembangan sistem informasi dan mempraktekkan analisis kebutuhan, analisis sistem dan perancangan sitem pada organisasi studi kasus;

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

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

  Parameter:

  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*

  Exception: Tidak memiliki *exception*

  Override: `checkPrasyarat` dari kelas `MataKuliah`

142. `ESM201`

     Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

143. `AIF312 implements HasPrasyarat`

144. `AIF312 implements HasPraktikum`

     Mata kuliah ini memberikan pengetahuan awal tentang keamanan informasi. Pada beberapa pertemuan awal, dibahas keamanan informasi secara matematis, yaitu di materi-materi seputar kriptografi dan serangannya. Lalu, dibahas pula konsep keamanan informasi pada jaringan komputer dan pada software.

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

     - `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

       Parameter:

       - `Mahasiswa mahasiswa` -
       - `java.util.List reasonsContainer` -

       Return Value: Tidak memiliki *return value*

       Exception: Tidak memiliki *exception*

       Override: `checkPrasyarat` dari kelas `MataKuliah`

145. `AIF484`

     Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

146. `AIF191`

     Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

147. `AIF386`

     Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

148. `AIF105`

Mata kuliah ini memperkenalkan kepada mahasiswa terminologi dan konsep dasar yang akan banyak dipakai sepanjang kuliah di Teknik Informatika. Selain itu mata kuliah ini juga mempersiapkan dan membiasakan mahasiswa dengan suasana akademik yang khas perguruan tinggi seperti kedisiplinan, kerja sama, kemampuan menggunakan teknologi informasi dalam pembuatan tugas, kemampuan komunikasi, dsb.

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

149. `AIF294`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

150. `AIF282`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

151. `AIF451`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

152. `APS302`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

153. `EAA101`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

154. `HasPrasyarat`

Mendefinisikan kelas-kelas yang memiliki prasyarat, terkustomisasi untuk seorang `Mahasiswa`. Jika ada tambahan, jangan lupa untuk mendaftarkannya di DEFAULT_HASPRASYARAT_CLASSES. Jika berubah package, jangan lupa mengupdate DEFAULT_PRASYARAT_PACKAGE.

Atribut yang dimiliki kelas ini adalah sebagai berikut.

- `String DEFAULT_HASPRASYARAT_CLASSES` - Daftar dari nama kelas default seluruh turunan interface ini. Perlu didaftarkan manual, karena Java reflection tidak dapat mendeteksi otomatis.
- `String DEFAULT_PRASYARAT_PACKAGE` - Package tempat menyimpan seluruh turunan standar interface ini. Perlu didefinisikan manual, karena Java reflection tidak dapat mendeteksi otomatis.

Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`

  Memeriksa prasyarat-prasyarat dari kuliah, spesifik untuk mahasiswa yang dituju. Jika ada pesan-pesan khusus, akan ditambahkan pada parameter reasonsContainer.

  Parameter:

- `Mahasiswa mahasiswa` - prasyarat kuliah akan diperiksa spesifik pada mahasiswa ini
- `List reasonsContainer` - pesan-pesan terkait prasyarat akan ditambahkan di sini, jika ada.

Return Value: true jika seluruh prasyarat dipenuhi, false jika tidak.
Exception: Tidak memiliki *exception*

155. `HasPraktikum`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

156. `HasResponsi`

Kelas ini tidak memiliki atribut. Kelas ini tidak memiliki method.

157. `Kelulusan implements HasPrasyarat`

Atribut yang dimiliki kelas ini adalah sebagai berikut.

- `String PILIHAN_WAJIB` -
- `String WAJIB` -
- `String AGAMA` -
- `int MIN_SKS_LULUS` -
- `int MIN_PILIHAN_WAJIB` -

Method-method yang dimiliki kelas ini adalah sebagai berikut.

- `public boolean checkPrasyarat(id.ac.unpar.siamodels.Mahasiswa mahasiswa, java.util.List reasonsContainer)`
  Parameter:
  - `Mahasiswa mahasiswa` -
  - `java.util.List reasonsContainer` -

  Return Value: Tidak memiliki *return value*
  Exception: Tidak memiliki *exception*
  Override: `checkPrasyarat` dari kelas `Object`

Gambar D.1: javadoctolatex.pdf

LAMPIRAN E
HASIL PDF TEXDOCLET

TeXDoclet Java Documentation

Created with Javadoc TeXDoclet Doclet

Greg Wonderly S"oren Caspersen Stefan Marx

October 3, 2012

Contents

Class Hierarchy

Classes

- java.lang.Object
  - com.sun.javadoc.Doclet
    - org.stfm.texdoclet.TeXDoclet (in 1.10, page 19)
  - javax.swing.text.html.HTMLEditorKit.ParserCallback
    - org.stfm.texdoclet.HTMLtoLaTeXBackEnd (in 1.4, page 6)
  - org.stfm.texdoclet.ClassHierachy (in 1.2, page 4)
  - org.stfm.texdoclet.HelpOutput (in 1.3, page 5)
  - org.stfm.texdoclet.InterfaceHierachy (in 1.5, page 8)
  - org.stfm.texdoclet.MarkdownTest (in 1.6, page 9)
  - org.stfm.texdoclet.Package (in 1.7, page 13)
  - org.stfm.texdoclet.TableInfo (in 1.8, page 15)
  - org.stfm.texdoclet.TestFilter (in 1.9, page 18)

Interfaces

- org.stfm.texdoclet.ClassFilter (in 1.1, page 4)

Chapter 1

Package org.stfm.texdoclet

Package Contents *Page*

Interfaces

Classes

This doclet is based on the doclet originally created by Greg Wonderly of C2 technologies Inc. and its revision by XO Software. The project of Greg Wonderly is available here : http://java.net/projects/texdoclet.

1.1 Interface ClassFilter

This interface can be implemented and a class name provided to the Doclet to filter which classes are and are not included in the output document.

1.1.1 Declaration

public interface ClassFilter

1.1.2 All known subinterfaces

TestFilter (in 1.9, page 18)

1.1.3 All classes known to implement interface

TestFilter (in 1.9, page 18)

1.1.4 Method summary

includeClass(ClassDoc) Filters the ClassDoc passed.

1.1.5 Methods

- **includeClass**
  `boolean` **includeClass**`(com.sun.javadoc.ClassDoc` **cd**`)`
  - **Description**
    Filters the ClassDoc passed. If true is returned, the passed class will be included into the output. If false is returned, this document will not be included.

1.2 Class ClassHierachy

Manages and prints a class hierarchy. Use `add` to add another class to the hierarchy. Use `printTree` to print the corresponding LaTeX.

1.2.1 Declaration

public class ClassHierachy
extends java.lang.Object

1.2.2 Field summary

root

1.2.3 Constructor summary

ClassHierachy() Creates new ClassHierachy

1.2.4 Method summary

add(ClassDoc) Adds another class to the hierachy
printBranch(RootDoc, SortedMap, double, double) Prints a branch of the tree.
printTree(RootDoc, double) Prints the LaTeX corresponding to the tree.

1.2.5 Fields

- public java.util.SortedMap **root**

1.2.6 Constructors

- **ClassHierachy**
  `public` **ClassHierachy**`()`
  - **Description**
    Creates new ClassHierachy

1.2.7 Methods

- **add**
  `protected java.util.SortedMap` **add**`(com.sun.javadoc.ClassDoc` **cls**`)`
  - **Description**
    Adds another class to the hierachy
- **printBranch**
  `protected void` **printBranch**`(com.sun.javadoc.RootDoc` **rootDoc**`, java.util.SortedMap` **map**`, double` **indent**`, double` **overviewindent**`)`
  - **Description**
    Prints a branch of the tree. The branch is printed using `TeXDoclet.os`.
- **printTree**
  `public void` **printTree**`(com.sun.javadoc.RootDoc` **rootDoc**`, double` **overviewindent**`)`
  - **Description**
    Prints the LaTeX corresponding to the tree. The tree is printed using `TeXDoclet.os`.

1.3 Class HelpOutput

1.3.1 Declaration

public class HelpOutput
extends java.lang.Object

1.3.2 Constructor summary

HelpOutput()

1.3.3 Method summary

printHelp()

1.3.4 Constructors

- **HelpOutput**
  `public` **HelpOutput**`()`

1.3.5 Methods

- **printHelp**
  `protected static void` **printHelp**`()`

1.4 Class HTMLtoLaTeXBackEnd

This class implements a `ParserCallback` that translates HTML to the corresponding LaTeX. Not all tags a processed but the most common are.

HTML links to files located in the doc-files directory (appendix_a.html, appendix_b.txt) are transformed to references to the appendix, whereby the referenced files itself are included in the appendix.

1.4.1 See also

– `javax.swing.text.html.parser.ParserDelegator`

1.4.2 Declaration

public class HTMLtoLaTeXBackEnd
extends javax.swing.text.html.HTMLEditorKit.ParserCallback

1.4.3 Constructor summary

HTMLtoLaTeXBackEnd(StringBuffer) Constructs a new instance.

1.4.4 Method summary

fixText(String) Converts a HTML string into LaTeX using an instance of `HTMLtoLaTeXBackEnd`.

handleEndTag(HTML.Tag, int) This method handles HTML tags that mark an ending (e.g.

handleSimpleTag(HTML.Tag, MutableAttributeSet, int) This method handles simple HTML tags (e.g.

handleStartTag(HTML.Tag, MutableAttributeSet, int) This method handles HTML tags that mark a beginning (e.g.
handleText(char[], int) This method handles all other text.

1.4.5 Constructors

- **HTMLtoLaTeXBackEnd**
  `public` **HTMLtoLaTeXBackEnd**`(java.lang.StringBuffer` **ret**`)`
  - **Description**
    Constructs a new instance.
  - **Parameters**
    * `StringBuffer` – The `StringBuffer` where the translated HTML is appended.

1.4.6 Methods

- **fixText**
  `public static java.lang.String` **fixText**`(java.lang.String` **str**`)`
  - **Description**
    Converts a HTML string into LaTeX using an instance of `HTMLtoLaTeXBackEnd`.
- **handleEndTag**
  `public void` **handleEndTag**`(javax.swing.text.html.HTML.Tag` **tag**`, int` **pos**`)`
  - **Description**
    This method handles HTML tags that mark an ending (e.g. </P>-tags). It is called by the parser whenever such a tag is encountered.
- **handleSimpleTag**
  `public void` **handleSimpleTag**`(javax.swing.text.html.HTML.Tag` **tag**`,`
  `javax.swing.text.MutableAttributeSet` **attrSet**`, int` **pos**`)`
  - **Description**
    This method handles simple HTML tags (e.g. <HR>-tags). It is called by the parser whenever such a tag is encountered.
- **handleStartTag**
  `public void` **handleStartTag**`(javax.swing.text.html.HTML.Tag` **tag**`,`
  `javax.swing.text.MutableAttributeSet` **attrSet**`, int` **pos**`)`
  - **Description**
    This method handles HTML tags that mark a beginning (e.g. <P>-tags). It is called by the parser whenever such a tag is encountered.
- **handleText**
  `public void` **handleText**`(char[]` **data**`, int` **pos**`)`
  - **Description**
    This method handles all other text.

1.4.7 Members inherited from class HTMLEditorKit.ParserCallback

`javax.swing.text.html.HTMLEditorKit.ParserCallback`
flush, handleComment, handleEndOfLineString, handleEndTag, handleError, handleSimpleTag, handleStartTag, handleText, IMPLIED

1.5 Class InterfaceHierachy

Manages and prints a interface hierarchy. Use `add` to add another interface to the hierarchy. Use `printTree` to print the corresponding LaTeX.

1.5.1 Declaration

public class InterfaceHierachy
extends java.lang.Object

1.5.2 Field summary

root

1.5.3 Constructor summary

InterfaceHierachy() Creates new InterfaceHierachy

1.5.4 Method summary

add(ClassDoc) Adds another interface to the hierachy
printBranch(RootDoc, SortedMap, double, double) Prints a branch of the tree.
printTree(RootDoc, double) Prints the LaTeX corresponding to the tree.

1.5.5 Fields

- public java.util.SortedMap **root**

1.5.6 Constructors

- **InterfaceHierachy**
  `public` **InterfaceHierachy**`()`
  - **Description**
    Creates new InterfaceHierachy

1.5.7 Methods

- **add**
  `protected java.util.SortedMap` **add**`(com.sun.javadoc.ClassDoc` **cls**`)`

- **Description**

  Adds another interface to the hierachy

- **printBranch**
  `protected void` **printBranch**`(com.sun.javadoc.RootDoc` **rootDoc**`, java.util.SortedMap` **map**`, double` **indent**`, double` **overviewindent**`)`

  - **Description**

    Prints a branch of the tree. The branch is printed using `TeXDoclet.os`.

- **printTree**
  `public void` **printTree**`(com.sun.javadoc.RootDoc` **rootDoc**`, double` **overviewindent**`)`

  - **Description**

    Prints the LATEX corresponding to the tree. The tree is printed using `TeXDoclet.os`.

1.6 Class MarkdownTest

This class is just for testing the Mardown processing output.

a) Some text

Markdown code :

```
some text some text some text some text some text some text some
text some text some text some text some text some text some text
some text some text some text with 2 ending spaces

text some text some text some text some text some text some text
some text some text some text some text some text some text some text

text some text some text some text some text some text some text
some text some text
```

results in :

some text some text some text some text some text some text some text some text some text some text some text some text some text some text some text some text with 2 ending spaces

text some text some text some text some text some text some text some text some text some text some text some text some text some text

text some text some text some text some text some text some text some text some text

b) Lists

Markdown code :

```
unsorted :
```

```
- item1
    - item11
    - item12
- item2

or :

+ item1
    + item12
        + item13

sorted :

1. item1
    1. item11
    2. item12
2. item2
    - item21
    - item22
3. item3

lists with paragraphs :

1. some text some text some text some text some text some text some text some
text some text some text

    some text some text some text some text some text some text

2. some text some text some text some text some text some text
```

results in :

unsorted :

- item1
    - item11
    - item12
- item2

or :

- item1
    - item12
        - item13

sorted :

1. item1

   (a) item11
   (b) item12
2. item2
   - item21
   - item22
3. item3

lists with paragraphs :

1. some text some text some text some text some text some text some text some text some text some text

   some text some text some text some text some text some text
2. some text some text some text some text some text some text

c) Blockquotes

Markdown code :

```
some text some text some text some text some text some text

> some quoting text
>
> > some nested quoting text
>
> some quoting text
>
> ###### header in blockquote
>
> a list in blockquote :
>
> 1. item1
> 2. item2
>     1. item21
>     2. item22
> 3. item3
>
> some quoting text
>
>    code in blockquote
```

results in :

some text some text some text some text some text some text
some quoting text
some nested quoting text
some quoting text

header in blockquote

a list in blockquote :

1. item1
2. item2
   (a) item21
   (b) item22
3. item3

> some quoting text
> code in blockquote

d) Preformatted text

Markdown code :

```
some preformatted :

    code line 1
    code line 2
```

results in :

some preformatted :

```
code line 1
code line 2
```

e) Horizontal rules

Markdown code :

```
***
```

results in :

---

f) Emphasis

Markdown code :

```
*single asterisks* (em)

_single underscores_ (em)

**double asterisks** (strong)

__double underscores__ (strong)
```

results in :
single asterisks (em)
single underscores (em)
double asterisks (strong)
double underscores (strong)

h) Code

Markdown code :

```
some code : ‘TeXDoclet extends Doclet‘ and ‘‘There is a literal backtick (‘)
here.‘‘
```

results in :
some code : `TeXDoclet extends Doclet` and `There is a literal backtick (‘) here.`

1.6.1 Declaration

public class MarkdownTest
extends java.lang.Object

1.6.2 Constructor summary

MarkdownTest()

1.6.3 Constructors

- **MarkdownTest**
  `public` **MarkdownTest**`()`

1.7 Class Package

This class is used to manage the contents of a Java package. It accepts ClassDoc objects and examines them and groups them according to whether they are classes, interfaces, exceptions or errors. The accumulated Vectors can then be processed to get to all of the elements of the package that fall into each category. If needed the classes, interfaces, exceptions and errors can be sorted using the `sort` method.

1.7.1 See also

– `Package.sort()` (in 1.7.8, page 15)

1.7.2 Declaration

public class Package
extends java.lang.Object

1.7.3 Field summary

classes The classes this package has in it
errors The errors this package has in it
exceptions The exceptions this package has in it
interfaces The interfaces this package has in it
pkg The name of the package this object is for
pkgDoc

1.7.4 Constructor summary

Package(String, PackageDoc) Construct a new object corresponding to the passed package name.

1.7.5 Method summary

addElement(ClassDoc) Adds a ClassDoc element to this package.
sort() Sorts the vectors of classes, interfaces exceptions and errors.

1.7.6 Fields

- protected com.sun.javadoc.PackageDoc **pkgDoc**
- protected java.lang.String **pkg**
  - The name of the package this object is for
- protected java.util.Vector **classes**
  - The classes this package has in it
- protected java.util.Vector **interfaces**
  - The interfaces this package has in it
- protected java.util.Vector **exceptions**
  - The exceptions this package has in it
- protected java.util.Vector **errors**
  - The errors this package has in it

1.7.7 Constructors

- **Package**
  `public` **Package**`(java.lang.String` **pkg**`, com.sun.javadoc.PackageDoc` **doc**`)`
  - **Description**
    Construct a new object corresponding to the passed package name.
  - **Parameters**
    * `pkg` – the package name to use

1.7.8 Methods

- **addElement**
  `public void` **addElement**`(com.sun.javadoc.ClassDoc` **cd**`)`
  - **Description**
    Adds a ClassDoc element to this package.
  - **Parameters**
    * `cd` – the object to add to this package
- **sort**
  `public void` **sort**`()`
  - **Description**
    Sorts the vectors of classes, interfaces exceptions and errors.

1.8 Class TableInfo

This class provides support for converting HTML tables into LaTeX tables. Some of the things **NOT** implemented include the following:

- valign attributes are not processed, but align= is.
- rowspan attributes are not processed, but colspan= is.
- the argument to border= in the table tag is not used to control line size

Here is an example table.

Column 1 Heading	Column two heading	Column three heading
data	Span two columns	
more data	right	left

A nested table example		
Column one Heading	**Column two heading**	**Column three heading**
data	Span two columns	
more data	right	left
1 2 3 4	first line second line third line fourth line	

1.8.1 Declaration

public class TableInfo
extends java.lang.Object

1.8.2 Constructor summary

TableInfo()

1.8.3 Method summary

endCol() Ends the current column.
endRow() Ends the current row.
endTable() Ends the table, closing the last row as needed
startCol(MutableAttributeSet) Starts a new column, possibly closing the current column if needed
startHeadCol(MutableAttributeSet) Starts a new Heading column, possibly closing the current column if needed.
startRow(MutableAttributeSet) Starts a new row, possibly closing the current row if needed

startTable(StringBuffer, MutableAttributeSet) Constructs a new table object and starts processing of the table by scanning the <table> passed to count columns.

1.8.4 Constructors

- **TableInfo**
  `public` **TableInfo**`()`

1.8.5 Methods

- **endCol**
  `public void` **endCol**`()`
  - **Description**
    Ends the current column.
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
- **endRow**
  `public void` **endRow**`()`
  - **Description**
    Ends the current row.
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
- **endTable**
  `public java.lang.StringBuffer` **endTable**`()`
  - **Description**
    Ends the table, closing the last row as needed
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
- **startCol**
  `public void` **startCol**`(javax.swing.text.MutableAttributeSet` **attrSet**`)`
  - **Description**
    Starts a new column, possibly closing the current column if needed
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
    * `p` – the properties from the <td> tag
- **startHeadCol**
  `public void` **startHeadCol**`(javax.swing.text.MutableAttributeSet` **attrSet**`)`

  - **Description**
    Starts a new Heading column, possibly closing the current column if needed. A Heading column has a Bold Face font directive around it.
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
    * `p` – The properties from the <th> tag

- **startRow**
  `public void` **startRow**`(javax.swing.text.MutableAttributeSet` **attrSet**`)`
  - **Description**
    Starts a new row, possibly closing the current row if needed
  - **Parameters**
    * `ret` – The output buffer to put LaTeX into.
    * `p` – The properties from the <tr> tag

- **startTable**
  `public java.lang.StringBuffer` **startTable**`(java.lang.StringBuffer` **org**`,`
  `javax.swing.text.MutableAttributeSet` **attrSet**`)`
  - **Description**
    Constructs a new table object and starts processing of the table by scanning the <table> passed to count columns.
  - **Parameters**
    * `p` – properties found on the <table> tag
    * `ret` – the result buffer that will contain the output
    * `table` – the input string that has the entire table definition in it.
    * `off` – the offset into <table> where scanning should start

1.9 Class TestFilter

This class filters out classes beginning with "Test" when applied to the Doclet.

1.9.1 Declaration

public class TestFilter
extends java.lang.Object
implements ClassFilter

1.9.2 Constructor summary

TestFilter()

1.9.3 Method summary

includeClass(ClassDoc) Returns false if class name starts with "Test".

1.9.4 Constructors

- **TestFilter**
  `public` **TestFilter**`()`

1.9.5 Methods

- **includeClass**
  `public boolean` **includeClass**`(com.sun.javadoc.ClassDoc` **cd**`)`
  - **Description**
    Returns false if class name starts with "Test".

1.10 Class TeXDoclet

This class provides a Java `javadoc` Doclet which generates a LaTeX 2_ε document out of the java classes that it is used on. This is convenient for creating printable documentation complete with cross reference information.

Supported HTML tags

<a> including an additional attribut "doprinturl". Since the output of the doclet should be printable, the href attribut of tags is printed in parentheses following the link if attribut "doprinturl" is set. Sometimes this is undesirable, and omitting "doprinturl" attribut will prevent this.

<dl> with the associated <dt><dd></dl>tags

<p> but not align=center...yet

 but not clear=xxx

<table> including all the associated <td><th><tr></td></th></tr>

 ordered lists

 unordered lists

<font> font coloring

<pre> preformatted text

<code> fixed point fonts

<i> italized fonts

<b> bold fonts

<sub> subscript

<sup> superscript

 center

<img> image located in java sources (<img src="package path/image name">)

1. example converted from JPG: (image file not found)
2. example converted from GIF: (image file not found)

<img> image located in the www: (see image at http://upload.wikimedia.org/wikipedia/commons/9/92/LaTeX_

Extra tags

<TEX>

A new tag is defined: <TEX>. This tag is useful for passing TEX code directly to the TEX compiler. The following code:

```
<TEX txt="\[ F\left( x \right) = \int_{ - \infty }^x {\frac{1}{{\sqrt {2\pi }
          }}e^{ - \frac{{z^2 }}{2}} dz} \]">
<BR><BR><B>This alternative text will appear if the javadoc/HTML is parsed
by any other doclet/browser</B><BR><BR></TEX>
```

will produce the following result:

$$F\left(x\right) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$$

The "alternative" text is ignored by the TeXDoclet, but useful if you want to use both the TeXDoclet and a regular HTML based doclet.

<PRE format="markdown">

Instead of writing your java documentation in often hard to read HTML code you can make use of Markdown syntax. The HTML <PRE> tag is used therefore to prevent your IDE from automatically reordering your Markdown documentation text. Markdown parsing is based on the Pegdown implementation. The following code :

```
<PRE format="markdown">

some text some text some text some text some text some text some text

##### Lists

- item1
    1. item11
    2. item12
- item1

##### Text formatting
```

```
_emphasis_ and __strong__ and some `code` :

    code line 1
    code line 2

some text some text some text some text some text some text some text

<PRE>
```

will produce the following :

some text some text some text some text some text some text some text

Lists

- item1
  1. item11
  2. item12
- item1

Text formatting

emphasis and **strong** and some `code` :

```
code line 1
code line 2
```

some text some text some text some text some text some text some text

1.10.1 See also

- `HTMLtoLaTeXBackEnd` (in 1.4, page 6)
- `TeXDoclet.start(RootDoc)` (in 1.10.8, page 23)

1.10.2 Declaration

public class TeXDoclet
extends com.sun.javadoc.Doclet

1.10.3 Field summary

BOLD
CHAPTER_LEVEL
ITALIC
os Writer for writing to output file
SECTION_LEVEL
SUBSECTION_LEVEL
TRUETYPE

1.10.4 Constructor summary

TeXDoclet()

1.10.5 Method summary

finish()
init()
initSections()
main(String[])
optionLength(String) Returns how many arguments would be consumed if `option` is a recognized option.
start(RootDoc) Called by the framework to format the entire document
validOptions(String[][], DocErrorReporter) Checks the passed options and their arguments for validity.

1.10.6 Fields

- public static final java.lang.String **SECTION_LEVEL**
- public static final java.lang.String **CHAPTER_LEVEL**
- public static final java.lang.String **SUBSECTION_LEVEL**
- public static final java.lang.String **BOLD**
- public static final java.lang.String **TRUETYPE**
- public static final java.lang.String **ITALIC**
- public static java.io.PrintWriter **os**
  - Writer for writing to output file

1.10.7 Constructors

- **TeXDoclet**
  `public` **TeXDoclet**`()`

1.10.8 Methods

- **finish**
  `public static void` **finish**`()`
- **init**
  `public static void` **init**`()`
- **initSections**
  `public static void` **initSections**`()`
- **main**
  `public static void` **main**`(java.lang.String[]` **args**`)`

- **optionLength**
  `public static int` **optionLength**`(java.lang.String` **option**`)`
  - **Description**
    Returns how many arguments would be consumed if `option` is a recognized option.
  - **Parameters**
    * `option` – the option to check
- **start**
  `public static boolean` **start**`(com.sun.javadoc.RootDoc` **root**`)`
  - **Description**
    Called by the framework to format the entire document
  - **Parameters**
    * `root` – the root of the starting document
- **validOptions**
  `public static boolean` **validOptions**`(java.lang.String[][]` **args**`,`
  `com.sun.javadoc.DocErrorReporter` **err**`)`
  - **Description**
    Checks the passed options and their arguments for validity.
  - **Parameters**
    * `args` – the arguments to check
    * `err` – the interface to use for reporting errors

1.10.9 Members inherited from class Doclet

`com.sun.javadoc.Doclet`
languageVersion, optionLength, start, validOptions

Gambar E.1: TeXDoclet_report.pdf

LAMPIRAN F

HASIL PDF TEXDOCLET

TeXDoclet Java Documentation

Created with Javadoc TeXDoclet Doclet

Greg Wonderly S"oren Caspersen Stefan Marx

October 3, 2012

Contents

Class Hierarchy

Classes

- java.lang.Object
  - com.sun.javadoc.Doclet
    - org.stfm.texdoclet.TeXDoclet (in 1.10, page 19)
  - javax.swing.text.html.HTMLEditorKit.ParserCallback
    - org.stfm.texdoclet.HTMLtoLaTeXBackEnd (in 1.4, page 6)
  - org.stfm.texdoclet.ClassHierachy (in 1.2, page 4)
  - org.stfm.texdoclet.HelpOutput (in 1.3, page 5)
  - org.stfm.texdoclet.InterfaceHierachy (in 1.5, page 8)
  - org.stfm.texdoclet.MarkdownTest (in 1.6, page 9)
  - org.stfm.texdoclet.Package (in 1.7, page 13)
  - org.stfm.texdoclet.TableInfo (in 1.8, page 15)
  - org.stfm.texdoclet.TestFilter (in 1.9, page 18)

Interfaces

- org.stfm.texdoclet.ClassFilter (in 1.1, page 4)

Chapter 1

Package org.stfm.texdoclet

Package Contents *Page*

Interfaces

Classes

This doclet is based on the doclet originally created by Greg Wonderly of C2 technologies Inc. and its revision by XO Software. The project of Greg Wonderly is available here : http://java.net/projects/texdoclet.

1.1 Interface ClassFilter

This interface can be implemented and a class name provided to the Doclet to filter which classes are and are not included in the output document.

1.1.1 Declaration

public interface ClassFilter

1.1.2 All known subinterfaces

TestFilter (in 1.9, page 18)

1.1.3 All classes known to implement interface

TestFilter (in 1.9, page 18)

1.1.4 Method summary

includeClass(ClassDoc) Filters the ClassDoc passed.

1.1.5 Methods

- **includeClass**
  `boolean` **includeClass**`(com.sun.javadoc.ClassDoc` **cd**`)`
  - **Description**
    Filters the ClassDoc passed. If true is returned, the passed class will be included into the output. If false is returned, this document will not be included.

1.2 Class ClassHierachy

Manages and prints a class hierarchy. Use `add` to add another class to the hierarchy. Use `printTree` to print the corresponding LaTeX.

1.2.1 Declaration

public class ClassHierachy
extends java.lang.Object

1.2.2 Field summary

root

1.2.3 Constructor summary

ClassHierachy() Creates new ClassHierachy

1.2.4 Method summary

add(ClassDoc) Adds another class to the hierachy
printBranch(RootDoc, SortedMap, double, double) Prints a branch of the tree.
printTree(RootDoc, double) Prints the LaTeX corresponding to the tree.

1.2.5 Fields

- public java.util.SortedMap **root**

1.2.6 Constructors

- **ClassHierachy**
  `public` **ClassHierachy**`()`
  - **Description**
    Creates new ClassHierachy

1.2.7 Methods

- **add**
  `protected java.util.SortedMap` **add**`(com.sun.javadoc.ClassDoc` **cls**`)`
  - **Description**
    Adds another class to the hierachy
- **printBranch**
  `protected void` **printBranch**`(com.sun.javadoc.RootDoc` **rootDoc**`, java.util.SortedMap` **map**`, double` **indent**`, double` **overviewindent**`)`
  - **Description**
    Prints a branch of the tree. The branch is printed using `TeXDoclet.os`.
- **printTree**
  `public void` **printTree**`(com.sun.javadoc.RootDoc` **rootDoc**`, double` **overviewindent**`)`
  - **Description**
    Prints the LaTeX corresponding to the tree. The tree is printed using `TeXDoclet.os`.

1.3 Class HelpOutput

1.3.1 Declaration

public class HelpOutput
extends java.lang.Object

1.3.2 Constructor summary

HelpOutput()

1.3.3 Method summary

printHelp()

1.3.4 Constructors

- **HelpOutput**
  `public` **HelpOutput**`()`

1.3.5 Methods

- **printHelp**
  `protected static void` **printHelp**`()`

1.4 Class HTMLtoLaTeXBackEnd

This class implements a `ParserCallback` that translates HTML to the corresponding LaTeX. Not all tags a processed but the most common are.

HTML links to files located in the doc-files directory (appendix_a.html, appendix_b.txt) are transformed to references to the appendix, whereby the referenced files itself are included in the appendix.

1.4.1 See also

– `javax.swing.text.html.parser.ParserDelegator`

1.4.2 Declaration

public class HTMLtoLaTeXBackEnd
extends javax.swing.text.html.HTMLEditorKit.ParserCallback

1.4.3 Constructor summary

HTMLtoLaTeXBackEnd(StringBuffer) Constructs a new instance.

1.4.4 Method summary

fixText(String) Converts a HTML string into LaTeX using an instance of `HTMLtoLaTeXBackEnd`.

handleEndTag(HTML.Tag, int) This method handles HTML tags that mark an ending (e.g.

handleSimpleTag(HTML.Tag, MutableAttributeSet, int) This method handles simple HTML tags (e.g.

handleStartTag(HTML.Tag, MutableAttributeSet, int) This method handles HTML tags that mark a beginning (e.g.
handleText(char[], int) This method handles all other text.

1.4.5 Constructors

- **HTMLtoLaTeXBackEnd**
  `public` **HTMLtoLaTeXBackEnd**`(java.lang.StringBuffer` **ret**`)`
  - **Description**
    Constructs a new instance.
  - **Parameters**
    * `StringBuffer` – The `StringBuffer` where the translated HTML is appended.

1.4.6 Methods

- **fixText**
  `public static java.lang.String` **fixText**`(java.lang.String` **str**`)`
  - **Description**
    Converts a HTML string into LaTeX using an instance of `HTMLtoLaTeXBackEnd`.
- **handleEndTag**
  `public void` **handleEndTag**`(javax.swing.text.html.HTML.Tag` **tag**`, int` **pos**`)`
  - **Description**
    This method handles HTML tags that mark an ending (e.g. </P>-tags). It is called by the parser whenever such a tag is encountered.
- **handleSimpleTag**
  `public void` **handleSimpleTag**`(javax.swing.text.html.HTML.Tag` **tag**`,`
  `javax.swing.text.MutableAttributeSet` **attrSet**`, int` **pos**`)`
  - **Description**
    This method handles simple HTML tags (e.g. <HR>-tags). It is called by the parser whenever such a tag is encountered.
- **handleStartTag**
  `public void` **handleStartTag**`(javax.swing.text.html.HTML.Tag` **tag**`,`
  `javax.swing.text.MutableAttributeSet` **attrSet**`, int` **pos**`)`
  - **Description**
    This method handles HTML tags that mark a beginning (e.g. <P>-tags). It is called by the parser whenever such a tag is encountered.
- **handleText**
  `public void` **handleText**`(char[]` **data**`, int` **pos**`)`
  - **Description**
    This method handles all other text.

1.4.7 Members inherited from class HTMLEditorKit.ParserCallback

`javax.swing.text.html.HTMLEditorKit.ParserCallback`
flush, handleComment, handleEndOfLineString, handleEndTag, handleError, handleSimpleTag, handleStartTag, handleText, IMPLIED

1.5 Class InterfaceHierachy

Manages and prints a interface hierarchy. Use `add` to add another interface to the hierarchy. Use `printTree` to print the corresponding LaTeX.

1.5.1 Declaration

public class InterfaceHierachy
extends java.lang.Object

1.5.2 Field summary

root

1.5.3 Constructor summary

InterfaceHierachy() Creates new InterfaceHierachy

1.5.4 Method summary

add(ClassDoc) Adds another interface to the hierachy
printBranch(RootDoc, SortedMap, double, double) Prints a branch of the tree.
printTree(RootDoc, double) Prints the LaTeX corresponding to the tree.

1.5.5 Fields

- public java.util.SortedMap **root**

1.5.6 Constructors

- **InterfaceHierachy**
  `public` **InterfaceHierachy**`()`
  - **Description**
    Creates new InterfaceHierachy

1.5.7 Methods

- **add**
  `protected java.util.SortedMap` **add**`(com.sun.javadoc.ClassDoc` **cls**`)`

  - **Description**
    Adds another interface to the hierachy

- **printBranch**
  `protected void` **printBranch**`(com.sun.javadoc.RootDoc` **rootDoc**`, java.util.SortedMap` **map**`, double` **indent**`, double` **overviewindent**`)`
  - **Description**
    Prints a branch of the tree. The branch is printed using `TeXDoclet.os`.

- **printTree**
  `public void` **printTree**`(com.sun.javadoc.RootDoc` **rootDoc**`, double` **overviewindent**`)`
  - **Description**
    Prints the LaTeX corresponding to the tree. The tree is printed using `TeXDoclet.os`.

1.6 Class MarkdownTest

This class is just for testing the Mardown processing output.

a) Some text

Markdown code :

```
some text some text some text some text some text some text some
text some text some text some text some text some text some text
some text some text some text with 2 ending spaces

text some text some text some text some text some text some text
some text some text some text some text some text some text some text

text some text some text some text some text some text some text
some text some text
```

results in :

some text some text some text some text some text some text some text some text some text some text some text some text some text some text some text some text with 2 ending spaces

text some text some text some text some text some text some text some text some text some text some text some text some text some text

text some text some text some text some text some text some text some text some text

b) Lists

Markdown code :

```
unsorted :
```

```
- item1
    - item11
    - item12
- item2

or :

+ item1
    + item12
        + item13

sorted :

1. item1
    1. item11
    2. item12
2. item2
    - item21
    - item22
3. item3

lists with paragraphs :

1. some text some text some text some text some text some text some text some
text some text some text

    some text some text some text some text some text some text

2. some text some text some text some text some text some text
```

results in :

unsorted :

- item1
    - item11
    - item12
- item2

or :

- item1
    - item12
        - item13

sorted :

1. item1

   (a) item11
   (b) item12
2. item2
   - item21
   - item22
3. item3

lists with paragraphs :

1. some text some text some text some text some text some text some text some text some text some text

   some text some text some text some text some text some text
2. some text some text some text some text some text some text

c) Blockquotes

Markdown code :

```
some text some text some text some text some text some text

> some quoting text
>
> > some nested quoting text
>
> some quoting text
>
> ###### header in blockquote
>
> a list in blockquote :
>
> 1. item1
> 2. item2
>     1. item21
>     2. item22
> 3. item3
>
> some quoting text
>
>    code in blockquote
```

results in :

some text some text some text some text some text some text
some quoting text
some nested quoting text
some quoting text

header in blockquote

a list in blockquote :

1. item1
2. item2
   (a) item21
   (b) item22
3. item3

> some quoting text
> code in blockquote

d) Preformatted text

Markdown code :

```
some preformatted :

    code line 1
    code line 2
```

results in :

some preformatted :

```
code line 1
code line 2
```

e) Horizontal rules

Markdown code :

```
***
```

results in :

---

f) Emphasis

Markdown code :

```
*single asterisks* (em)

_single underscores_ (em)

**double asterisks** (strong)

__double underscores__ (strong)
```

results in :
single asterisks (em)
single underscores (em)
double asterisks (strong)
double underscores (strong)

h) Code

Markdown code :

```
some code : ‘TeXDoclet extends Doclet‘ and ‘‘There is a literal backtick (‘)
here.‘‘
```

results in :
some code : `TeXDoclet extends Doclet` and `There is a literal backtick (‘) here.`

1.6.1 Declaration

public class MarkdownTest
extends java.lang.Object

1.6.2 Constructor summary

MarkdownTest()

1.6.3 Constructors

- **MarkdownTest**
  `public` **MarkdownTest**`()`

1.7 Class Package

This class is used to manage the contents of a Java package. It accepts ClassDoc objects and examines them and groups them according to whether they are classes, interfaces, exceptions or errors. The accumulated Vectors can then be processed to get to all of the elements of the package that fall into each category. If needed the classes, interfaces, exceptions and errors can be sorted using the `sort` method.

1.7.1 See also

– `Package.sort()` (in 1.7.8, page 15)

1.7.2 Declaration

public class Package
extends java.lang.Object

1.7.3 Field summary

classes The classes this package has in it
errors The errors this package has in it
exceptions The exceptions this package has in it
interfaces The interfaces this package has in it
pkg The name of the package this object is for
pkgDoc

1.7.4 Constructor summary

Package(String, PackageDoc) Construct a new object corresponding to the passed package name.

1.7.5 Method summary

addElement(ClassDoc) Adds a ClassDoc element to this package.
sort() Sorts the vectors of classes, interfaces exceptions and errors.

1.7.6 Fields

- protected com.sun.javadoc.PackageDoc **pkgDoc**
- protected java.lang.String **pkg**
  - The name of the package this object is for
- protected java.util.Vector **classes**
  - The classes this package has in it
- protected java.util.Vector **interfaces**
  - The interfaces this package has in it
- protected java.util.Vector **exceptions**
  - The exceptions this package has in it
- protected java.util.Vector **errors**
  - The errors this package has in it

1.7.7 Constructors

- **Package**
  `public` **Package**`(java.lang.String` **pkg**`, com.sun.javadoc.PackageDoc` **doc**`)`
  - **Description**
    Construct a new object corresponding to the passed package name.
  - **Parameters**
    * `pkg` – the package name to use

1.7.8 Methods

- **addElement**
  `public void` **addElement**`(com.sun.javadoc.ClassDoc` **cd**`)`
  - **Description**
    Adds a ClassDoc element to this package.
  - **Parameters**
    * `cd` – the object to add to this package
- **sort**
  `public void` **sort**`()`
  - **Description**
    Sorts the vectors of classes, interfaces exceptions and errors.

1.8 Class TableInfo

This class provides support for converting HTML tables into LaTeX tables. Some of the things **NOT** implemented include the following:

- valign attributes are not processed, but align= is.
- rowspan attributes are not processed, but colspan= is.
- the argument to border= in the table tag is not used to control line size

Here is an example table.

Column 1 Heading	Column two heading	Column three heading
data	Span two columns	
more data	right	left

A nested table example		
Column one Heading	**Column two heading**	**Column three heading**
data	Span two columns	
more data	right	left
1 2 3 4	first line second line third line fourth line	

1.8.1 Declaration

public class TableInfo
extends java.lang.Object

1.8.2 Constructor summary

TableInfo()

1.8.3 Method summary

endCol() Ends the current column.
endRow() Ends the current row.
endTable() Ends the table, closing the last row as needed
startCol(MutableAttributeSet) Starts a new column, possibly closing the current column if needed
startHeadCol(MutableAttributeSet) Starts a new Heading column, possibly closing the current column if needed.
startRow(MutableAttributeSet) Starts a new row, possibly closing the current row if needed

startTable(StringBuffer, MutableAttributeSet) Constructs a new table object and starts processing of the table by scanning the <table> passed to count columns.

1.8.4 Constructors

- **TableInfo**
  `public` **TableInfo**`()`

1.8.5 Methods

- **endCol**
  `public void` **endCol**`()`
  - **Description**
    Ends the current column.
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
- **endRow**
  `public void` **endRow**`()`
  - **Description**
    Ends the current row.
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
- **endTable**
  `public java.lang.StringBuffer` **endTable**`()`
  - **Description**
    Ends the table, closing the last row as needed
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
- **startCol**
  `public void` **startCol**`(javax.swing.text.MutableAttributeSet` **attrSet**`)`
  - **Description**
    Starts a new column, possibly closing the current column if needed
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
    * `p` – the properties from the <td> tag
- **startHeadCol**
  `public void` **startHeadCol**`(javax.swing.text.MutableAttributeSet` **attrSet**`)`

  - **Description**
    Starts a new Heading column, possibly closing the current column if needed. A Heading column has a Bold Face font directive around it.
  - **Parameters**
    * `ret` – The output buffer to put LaTeX 2_ε into.
    * `p` – The properties from the <th> tag

- **startRow**
  `public void` **startRow**`(javax.swing.text.MutableAttributeSet` **attrSet**`)`
  - **Description**
    Starts a new row, possibly closing the current row if needed
  - **Parameters**
    * `ret` – The output buffer to put LaTeX into.
    * `p` – The properties from the <tr> tag

- **startTable**
  `public java.lang.StringBuffer` **startTable**`(java.lang.StringBuffer` **org**`,`
  `javax.swing.text.MutableAttributeSet` **attrSet**`)`
  - **Description**
    Constructs a new table object and starts processing of the table by scanning the <table> passed to count columns.
  - **Parameters**
    * `p` – properties found on the <table> tag
    * `ret` – the result buffer that will contain the output
    * `table` – the input string that has the entire table definition in it.
    * `off` – the offset into <table> where scanning should start

1.9 Class TestFilter

This class filters out classes beginning with "Test" when applied to the Doclet.

1.9.1 Declaration

public class TestFilter
extends java.lang.Object
implements ClassFilter

1.9.2 Constructor summary

TestFilter()

1.9.3 Method summary

includeClass(ClassDoc) Returns false if class name starts with "Test".

1.9.4 Constructors

- **TestFilter**
  `public` **TestFilter**`()`

1.9.5 Methods

- **includeClass**
  `public boolean` **includeClass**`(com.sun.javadoc.ClassDoc` **cd**`)`
  - **Description**
    Returns false if class name starts with "Test".

1.10 Class TeXDoclet

This class provides a Java `javadoc` Doclet which generates a LaTeX 2_ε document out of the java classes that it is used on. This is convenient for creating printable documentation complete with cross reference information.

Supported HTML tags

<a> including an additional attribut "doprinturl". Since the output of the doclet should be printable, the href attribut of tags is printed in parentheses following the link if attribut "doprinturl" is set. Sometimes this is undesirable, and omitting "doprinturl" attribut will prevent this.

<dl> with the associated <dt><dd></dl>tags

<p> but not align=center...yet

 but not clear=xxx

<table> including all the associated <td><th><tr></td></th></tr>

 ordered lists

 unordered lists

<font> font coloring

<pre> preformatted text

<code> fixed point fonts

<i> italized fonts

<b> bold fonts

<sub> subscript

<sup> superscript

 center

<img> image located in java sources (<img src="package path/image name">)

1. example converted from JPG: (image file not found)
2. example converted from GIF: (image file not found)

<img> image located in the www: (see image at http://upload.wikimedia.org/wikipedia/commons/9/92/LaTeX_

Extra tags

<TEX>

A new tag is defined: <TEX>. This tag is useful for passing TEX code directly to the TEX compiler. The following code:

```
<TEX txt="\[ F\left( x \right) = \int_{ - \infty }^x {\frac{1}{{\sqrt {2\pi }
          }}e^{ - \frac{{z^2 }}{2}} dz} \]">
<BR><BR><B>This alternative text will appear if the javadoc/HTML is parsed
by any other doclet/browser</B><BR><BR></TEX>
```

will produce the following result:

$$F\left(x\right) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$$

The "alternative" text is ignored by the TeXDoclet, but useful if you want to use both the TeXDoclet and a regular HTML based doclet.

<PRE format="markdown">

Instead of writing your java documentation in often hard to read HTML code you can make use of Markdown syntax. The HTML <PRE> tag is used therefore to prevent your IDE from automatically reordering your Markdown documentation text. Markdown parsing is based on the Pegdown implementation. The following code :

```
<PRE format="markdown">

some text some text some text some text some text some text some text

##### Lists

- item1
    1. item11
    2. item12
- item1

##### Text formatting
```

```
_emphasis_ and __strong__ and some `code` :

    code line 1
    code line 2

some text some text some text some text some text some text some text

<PRE>
```

will produce the following :

some text some text some text some text some text some text some text

Lists

- item1
  1. item11
  2. item12
- item1

Text formatting

emphasis and **strong** and some `code` :

```
code line 1
code line 2
```

some text some text some text some text some text some text some text

1.10.1 See also

- `HTMLtoLaTeXBackEnd` (in 1.4, page 6)
- `TeXDoclet.start(RootDoc)` (in 1.10.8, page 23)

1.10.2 Declaration

public class TeXDoclet
extends com.sun.javadoc.Doclet

1.10.3 Field summary

BOLD
CHAPTER_LEVEL
ITALIC
os Writer for writing to output file
SECTION_LEVEL
SUBSECTION_LEVEL
TRUETYPE

1.10.4 Constructor summary

TeXDoclet()

1.10.5 Method summary

finish()
init()
initSections()
main(String[])
optionLength(String) Returns how many arguments would be consumed if `option` is a recognized option.
start(RootDoc) Called by the framework to format the entire document
validOptions(String[][], DocErrorReporter) Checks the passed options and their arguments for validity.

1.10.6 Fields

- public static final java.lang.String **SECTION_LEVEL**
- public static final java.lang.String **CHAPTER_LEVEL**
- public static final java.lang.String **SUBSECTION_LEVEL**
- public static final java.lang.String **BOLD**
- public static final java.lang.String **TRUETYPE**
- public static final java.lang.String **ITALIC**
- public static java.io.PrintWriter **os**
  - Writer for writing to output file

1.10.7 Constructors

- **TeXDoclet**
  `public` **TeXDoclet**`()`

1.10.8 Methods

- **finish**
  `public static void` **finish**`()`
- **init**
  `public static void` **init**`()`
- **initSections**
  `public static void` **initSections**`()`
- **main**
  `public static void` **main**`(java.lang.String[]` **args**`)`

- **optionLength**
  `public static int` **optionLength**`(java.lang.String` **option**`)`
  - **Description**
    Returns how many arguments would be consumed if `option` is a recognized option.
  - **Parameters**
    * `option` – the option to check
- **start**
  `public static boolean` **start**`(com.sun.javadoc.RootDoc` **root**`)`
  - **Description**
    Called by the framework to format the entire document
  - **Parameters**
    * `root` – the root of the starting document
- **validOptions**
  `public static boolean` **validOptions**`(java.lang.String[][]` **args**`,`
  `com.sun.javadoc.DocErrorReporter` **err**`)`
  - **Description**
    Checks the passed options and their arguments for validity.
  - **Parameters**
    * `args` – the arguments to check
    * `err` – the interface to use for reporting errors

1.10.9 Members inherited from class Doclet

`com.sun.javadoc.Doclet`
languageVersion, optionLength, start, validOptions