TUGAS AKHIR - KI1502

RANCANG BANGUN ARSITEKTUR MODULAR ALBUM **FOTO DIGITAL 'FOTOKITA'** BERBASIS DESKTOP

A HEYNOUM DALA RIF'AT
NRP 5112100084

Dosen Pembimbing I
Rizky Januar Akbar, S.Kom., M.Eng.

Dosen Pembimbing II
Dr.tech. Ir. R. V. Hari Ginardi, M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

UNDERGRADUATE THESES - KI1502

MODULAR ARCHITECTURE DESIGN AND IMPLEMENTATION OF DESKTOP-BASED **DIGITAL PHOTO ALBUM 'FOTOKITA'**

A HEYNOUM DALA RIF'AT
NRP 5112100084

Supervisor I
Rizky Januar Akbar, S.Kom., M.Eng.

Supervisor II
Dr.tech. Ir. R. V. Hari Ginardi, M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

LEMBAR PENGESAHAN

RANCANG BANGUN ARSITEKTUR MODULAR ALBUM FOTO DIGITAL 'FOTOKITA' BERBASIS DESKTOP

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
A. Heynoum Dala Rif'at
NRP . 5112 100 084

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Rizky Januar Akbar, S.Kom., M.Eng.
NIP. 19870103201404100 1 (Pembimbing 1) .....

2. Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.
NIP. 196505181992031003 (Pembimbing 2) .....

SURABAYA
JUNI 2016

RANCANG BANGUN ARSITEKTUR MODULAR ALBUM FOTO DIGITAL 'FOTOKITA' BERBASIS DESKTOP

Nama : **A. Heynoum Dala Rif'at**
NRP : **5112100084**
Jurusan : **Teknik Informatika**
Fakultas Teknologi Informasi ITS
Dosen Pembimbing 1 : **Rizky Januar Akbar, S.Kom.,M.Eng.**
Dosen Pembimbing 2 : **Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.**

ABSTRAK

Seiring dengan fenomena semakin maraknya aktifitas pengabadian momen, dibutuhkannya teknologi yang memadai untuk mendukung kebutuhan. Beberapa solusi sudah ditawarkan oleh percetakan atau studio foto untuk menawarkan jasa membuat album foto. Tetapi di Indonesia belum ada aplikasi album digital berbasis desktop yang dapat memudahkan pengguna secara langsung untuk membuat album fotonya sendiri.

Pada tugas akhir ini, permasalahan tersebut akan ditangani dengan membuat album foto digital yang bersifat modular yang dapat mengakomodasi kebutuhan pengguna dan ketidakpastian perkembangan fitur di masa depan, misalnya penambahan frame, efek dan tools pada aplikasi. Perangkat lunak ini bersifat desktop dan terdapat abstraksi/aturan yang perlu diimplementasi agar modul dapat diintegrasi. Perangkat lunak akan dapat menambah, menghapus instalasi atau mengubah status modul-modul dari album foto digital tersebut tanpa melakukan perubahan pada modul lain.

Pengujian dilakukan dengan melakukan penambahan, penghapusan instalasi dan pengubahan status modul. Dari hasil pengujian, aplikasi yang dirancang dan diimplementasikan telah memenuhi semua kebutuhan fungsional.

Kata kunci: Album Foto Digital, Perangkat Lunak Modular

MODULAR ARCHITECTURE DESIGN AND IMPLEMENTATION OF DESKTOP-BASED DIGITAL PHOTO ALBUM 'FOTOKITA'

Name	**: A. Heynoum Dala Rifat**
ID	**: 5112100084**
Department	**: Department of Informatics Faculty of Information Technology ITS**
Supervisor I	**: Rizky Januar Akbar, S.Kom., M.Eng.**
Supervisor II	**: Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.**

ABSTRACT

Along with the increasingly widespread phenomenon of perpetuation of moments, adequate technology to support the needs is required. Some solutions have been offered by printing and photo studio to offer services to create photo albums, but in Indonesia there is no desktop-based digital photo album yet to facilitate the users to create their own album directly using their own personal computer.

In this undergraduate thesis, those problems will be handled by designing and implementing modular architecture for digital photo album to facilitate the client's needs along with the uncertainty of features development in the future, such as addition or change of frames, effect and tools of the application. The aforementioned software will be desktop-based along with some abstraction/rules that have to be implemented so the module can be integrated. The software can install, uninstall and change the status of the modules without making any change to other modules.

Testing is done by creating, uninstalling and changing status of modul. From the results, the designed framework and its implementation has fulfilled all functional requirements.

Keywords: Digital Photo Album**, **Modular Software

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul: **"Rancang Bangun Arsitektur Modular Album Foto Digital 'Fotokita' berbasis Desktop"**

Melalui lembar ini, penulis ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Bapak, Ibu, Ade dan Bintang yang selalu mendukung tiap pilihan dan mendoakan yang terbik.
3. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku pembimbing I yang sealu memberikan motivasi dan membimbing penulis selama pengerjaan tugas akhir.
4. Bapak Dr.tech. Ir. R.V. Hari Ginardi, M.Sc. selaku pembimbing II yang telah membantu dan membimbing penulis mulai selama pengerjaan tugas akhir mulai dari ide awal.
5. Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. selaku dosen wali yang selalu mendukung penulis sejak awal perkuliahan.
6. Ir. Siti Rochimah, MT., Ph.D. selaku Kepala Laboratorium Rekayasa Perangkat Lunak (RPL) yang selalu memberikan dukungan bagi penulis selama penulis menjadi administrator Laboratorium RPL.
7. Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Dr. Radityo Anggoro, S.Kom., M.Sc. selaku koordinator TA dan segenap dosen Teknik Informatika yang telah memberika ilmunya.
8. Keluarga angkat penulis, keluarga Bücker: Franz-Josef, Anne, Jenny, Annika-Holger-Jakob dan Christina-

Matthias-Ole yang selalu menyemangati penulis dari jarak jauh.

9. Kelompok bit.ly/cinTA_114; Rina, Dwi, Fadri dan Ridwan yang selalu menemani, memotivasi dan mendengarkan keluh-kesah penulis selama mengerjakan tugas akhir ini.
10. Admin RPL 2012, Sasa, Dery, Reva, Aranda dan DS yang selalu menyemangati.
11. Mas Jay, Mas Icang dan Prad yang disela-sela kesibukannya turut membantu penulis ketika penulis menghadapi kesusahan dalam mengerjakan tugas akhir.
12. Arin, Eva, Anno, Nisa dan Lina yang selalu mendukung dan mendengarkan keluh kesah penulis.
13. Atlantis, Aero dan penghuni B875.
14. Admin-admin RPL yang lain yang selalu berusaha untuk menciptakan suasana kondusif selama pengerjaan TA di lab.
15. Teman-teman seangkatan 2012 yang telah membantu, berbagi ilmu, menjaga kebersamaan dan memberi motivasi kepada penulis.
16. Ali Ariff, yang selalu mendoakan dan membantu penulis ketika penulis mengalami kesulitan baik dalam pengerjaan tugas akhir maupun perkuliahan.
17. Serta semua pihak yang telah urut membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak kekurangan, sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan, metodologi dan sistematika penulisan yang digunakan dalam pembuatan buku tugas akhir ini.

1.1 Latar Belakang

Perkembangan teknologi di bidang digital sudah berkembang dengan sangat pesat, terutama dalam bidang fotografi. Dalam hal mencetak foto, yang dulu harus menggunakan film dan kemudian mencetaknya melalui proses yang rumit, sekarang bisa dilakukan dengan sangat mudah dengan memindahkan berkas yang ada di dalam kartu memori ke dalam komputer dan mencetaknya pun hanya membutuhkan sebuah mesin cetak (*printer*) dan kertas foto, dan foto-foto tersebut lalu diletakkan ke dalam album foto. Proses tersebut pada saat ini dapat dilakukan dengan menggunakan komputer melalui aplikasi.

Seiring dengan fenomena semakin maraknya pengabadian momen, dibutuhkannya teknologi yang memadai untuk mendukung kebutuhan. Beberapa percetakan atau studio foto sudah menyediakan layanan untuk melakukan pengeditan foto dan album, tetapi terkadang hasil yang ada tidak sesuai dengan harapan klien dan memakan biaya lebih. Solusi untuk klien yang menginginkan untuk mencetak foto album sesuai dengan selera masing-masing sudah tersedia, tetapi untuk di Indonesia sendiri sarana yang tersedia masih berupa pengeditan album foto secara *online*, sedangkan kecepatan jaringan belum bisa dikatakan mampu untuk melakukan semua kegiatan secara *online*.

Aplikasi album foto digital berbasis desktop diperlukan untuk memudahkan pengerjaan tersebut. Selain itu, untuk dapat menarik perhatian pengguna, sebuah aplikasi harus mengikuti perkembangan zaman dan mode, dan dikarenakan adanya ketidakpastian perkembangan fitur, misalnya penambahan *frame,*

layout, efek dan tema sesuai dengan kebutuhan dan selera pengguna, maka dibutuhkan aplikasi album foto digital yang modular, dimana nantinya modul-modul dalam album foto digital dapat diperbaharui, dihapus, atau diganti dengan mudah.

Tujuan dikembangkannya aplikasi ini adalah untuk memberikan solusi kemudahan dalam melakukan evolusi dan modularitas sistem, sehingga dapat mengakomodasi kebutuhan pengguna kedepannya.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimanakah membangun album foto digital secara modular?
2. Bagaimanakah cara penerapan modul sehingga modul dapat berfungsi pada sistem?
3. Bagaimanakah daur hidup modul pada Fotokita?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Pengerjaan aplikasi tidak termasuk pengerjaan modul-modul Fotokita dimana modul tersebut akan dikerjakan pada tugas akhir lain yaitu “Rancang Bangun Aplikasi Album Foto Digital ‘Fotokita’ Berbasis Desktop”.
2. Pengerjaan aplikasi meliputi pengerjaan antarmuka pengguna.
3. Modul yang ingin ditambah perlu mengikuti aturan modularitas yang telah ditetapkan.
4. Modul yang ada pada Fotokita adalah modul *frame*, modul *tool* dan modul efek.
5. Aplikasi dibuat dengan menggunakan bahasa pemrograman Java

1.4 Tujuan

Tugas akhir ini mempunyai beberapa tujuan, yaitu sebagai berikut:

1. Membuat album foto digital yang modular.
2. Membuat aplikasi yang dapat menerapkan modul yang telah dibuat.
3. Membuat aplikasi yang dapat mengakomodasi kebutuhan mengguna untuk menambah, mengubah dan menghapus instalasi modul.

1.5 Manfaat

Manfaat yang diharapkan dari hasil pembangunan arsitektur modular album foto digital ini antara lain adalah sebagai berikut:

1. Aplikasi yang modular dapat mengakomodasi kebutuhan pengguna kedepannya.
2. Pengguna dapat memiliki banyak variasi dari fitur yang ada.
3. Biaya dan tenaga yang diperlukan pengguna untuk membuat album foto berkurang.

1.6 Metodologi

Ada beberapa tahapan dalam pengerjaan tugas akhir ini, yaitu sebagai berikut:

1. Studi Literatur

   Tahap ini merupakan tahap pengumpulan informasi dan pembelajaran yang akan digunakan pada tugas akhir ini. Studi literatur meliputi diskusi dan pemahaman mengenai topik tugas akhir ini, diantaranya mengenai:

   1. Modularitas perangkat lunak
   2. Java *Standard Widget Toolkit* (SWT)

2. Analisis dan Perancangan Sistem

   Adapun pembagian tahap analisa kebutuhan dan perancangan arsitektur modular album foto digital Fotokita adalah sebagai berikut:

   a. Mempelajari kebutuhan arsitektur modular secara umum

Pada tahap ini yaitu mempelajari kebutuhan aplikasi arsitektur modular desktop secara umum melalui aplikasi serupa yang dikembangkan.

b. Mempelajari kebutuhan album foto digital Fotokita secara garis besar

Tahap ini dilakukan agar dapat membuat rancang bangun arsitektur modular yang sesuai kebutuhan pada tahap selanjutnya.

c. Merancang arsitektur modular album foto digital

Pada tahap ini dilakukan perancangan struktur kelas dalam bentuk *class diagram* dan *package diagram.*

3. Implementasi perangkat lunak

Pada tahap ini dilakukan pembuatan elemen perangkat lunak. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman Java.

4. Pengujian dan evaluasi

Pengujian sistem ini dilakukan dengan menggunakan metode *blackbox.* Pengujian *blackbox* merupakan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinisikan [1].

5. Penyusunan buku tugas akhir

Tahap ini merupakan tahap penyusunan laporan berupa buku sebagai dokumentasi pengerjaan tugas akhir yang mencakup seluruh dasar teori, desain, implementasi serta hasil pengujian yang telah dilakukan. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Metodologi
    f. Sistematika Penulisan
2. Tinjauan Pustaka

3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7 Sistematika Penulisan

Penulisan buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada kakas.

Bab IV Implementasi

Bab ini berisi hasil penerapan rancangan sistem penyelesaian permasalahan dalam tugas akhir ini dalam bentuk sumber kode beserta penjelasannya.

Bab V Uji Coba dan Evaluasi

Bab ini berisi pengujian fungsionalitas dengan metode pengujian *black box* untuk mengetahui kesesuaian hasil keluaran sistem.

Bab VI Kesimpulan dan Saran

Bab berisi kesimpulan pengerjaan tugas akhir ini dan saran untuk pengembangan kedepannya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Album Foto Digital

Album foto digital adalah sekumpulan foto yang dibukukan dalam suatu wadah. Album foto digital yang digital adalah aplikasi perangkat lunak dimana pengguna dapat memindahkan berkas foto dari hardisk ke dalam basis data utama aplikasi tersebut. Perangkat lunak Album foto digital biasanya memungkinkan pengguna untuk melihat, mengubah, dan mengatur foto dengan menggunakan antarmuka buku seperti yang menyerupai album foto tradisional.

Pada saat ini di Indonesia sudah terdapat beberapa usaha yang bergerak dalam bidang album foto digital seperti *Photobook* Indonesia[1], *Picbit Photobook*[2] dan *SnappyPhotobook*[3] tetapi masih terdapat beberapa kekurangan, yaitu:

- *Editor* berbasis *web* sehingga membutuhkan koneksi internet untuk mengedit album foto.
- Waktu untuk memuat halaman *editor* relatif lama sehingga tidak memungkinkan untuk mengedit foto dengan koneksi internet yang lambat.

2.2 Modularitas

Secara umum, pemrograman yang modular adalah teknik perancangan perangkat lunak yang menekankan kepada pemisahan

[1] http://www.photobookindonesia.com/
[2] http://picbitphotobook.com/
[3] http://www.snapyphotobook.com/site/

fungsi program ke modul-modul yang mandiri yang dapat dintegrasikan ke aplikasi yang lebih besar.

Menurut sudut pandang ilmu teknik, modularitas secara umum memiliki tiga tujuan, yaitu:

- Untuk mengelola kompleksitas suatu pengembangan
- Untuk memungkinkan pengerjaan secara pararel
- Untuk mengakomodasi ketidakpastian yang akan terjadi di masa depan.

Modularitas mengakomodasi ketidakpastian yang akan terjadi di masa depan karena elemen tertentu pada rencang bangun modular dapat diubah selama aturan-aturan perancangan ditaati. Oleh karena itu, dalam arsitektur modular, modul baru dapat mengganti modul lama dengan mudah dan dengan biaya yang rendah [2].

Modularitas terdiri dari dua aspek: *model runtime* dan *model development*. *Model runtime* fokus kepada bagaimana mengatur sistem perangkat lunak pada saat runtime, sementara *model development* adalah bagaimana para pengembang menggunakan kerangka kerja untuk membangun perangkat lunak mereka [2].

Dalam pembuatan modularitas pada tugas akhir ini, pemrograman berbasis *interface (interface-based programming*) diperlukan. Pemrograman berbasis *interface* adalah pola arsitektur untuk melaksanakan pemrograman modular pada tingkat komponen dalam bahasa pemrograman berorientasi objek yang tidak memiliki sistem modul, sebagai contoh Java. Java tidak memiliki sistem modul pada tingkat komponen. Java memiliki sistem *package*, tetapi komponen perangkat lunak Java biasanya terdiri dari beberapa *packages* Java. Pemrograman berbasis *interface* memberikan keuntungan.

Interface modul menyatakan unsur-unsur yang akan disediakan dan dibutuhkan oleh modul. Implementasinya berupa kode yang bekerja yang merespon unsur-unsur yang terdapat pada *interface*.

2.3 Java *Service Loader*

Kelas *ServiceLoader* merupakan kelas dari java.util yang dapat membaca konfigurasi berkas yang tersimpan dalam berkas arsip Java (JAR) dan menemukan implementasi dari sebuah implementasi, kemudian membuat implementasi tersebut tersedia sebagai daftar objek untuk dipilih [3]. Untuk tujuan pemuatan, *service* direpresentasikan oleh satu jenis *interface* atau *abstract class* tunggal. *Concrete class* juga dapat digunakan, tetapi tidak direkomendasikan. Sebuah penyedia *service* mengandung satu atau lebih *concrete class* yang meng*extend* tipe servis ini dengan data dan kode yang spesifik kepada penyedia [4].

Kelas penyedia biasanya tidak bukan seluruh penyedia itu sendiri melainkan proksi yang berisi informasi yang cukup untuk menentukan apakan layanan mampu memenuhi permintaan tertentu bersama dengan kode yang dapat membuat penyedia yang sebenarnya pada permintaan [4].

2.4 Java *Standard Widget Toolkit* (SWT)

Standard Widget Toolkit (SWT) adalah sebuah *widget toolkit* grafis yang digunakan pada platform Java. SWT Java merupakan salah satu alternative untuk menampilkan elemen GUI selain Java *Absract Window Toolkit* (AWT) dan Swing.

Untuk menampilkan elemen GUI, implementasi SWT mengakses pustaka asli GUI dari sistem operasi menggunakan JNI (*Java Native Interface*). Program yang menggunakan SWT portabel, tetapi implementasi dari *toolkit* unik pada setiap platform [5].

2.5 Java *Properties*

Kelas properti pada java merupakan satu set yang persistent dari properti. Properti dapat disimpan dan dimuat darri sebuah *stream*. Setiap kunci dan nilai yang dikandung dalam daftar properti adalah *string* [6]. Kelas properti merupakan kelas yang

aman dari *thread*. Beberapa *thread* dapat berbagi properti tunggal tanpa perlu sinkronisasi eksternal [6].

Kelas properti digunakan oleh banyak kelas yang lain [7]. Beberapa metode dan konstruktor pada properti yang digunakan pada tugas akhir ini adalah:

- Properties()
- String getProperty(String key)
- Void load(InputStream streamIN) throws IOException
- Object setProperty(String key, String value)
- Void store(OutputStream streamOut, String description)

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas analisis dan perancangan perangkat lunak dari album foto digital Fotokita modular. Hasil dari proses ini berupa diagram yang akan digunakan sebagai acuan untuk proses implementasi perangkat lunak. Selain digunakan sebagai acuan untuk proses selanjutnya, beberapa diagram hasil dari proses perancangan digunakan sebagai dokumentasi dari implementasi perangkat lunak. Diagram yang dihasilkan pada proses ini disajikan dalam bentuk *Unified Modelling Language* (UML).

3.1 Analisis

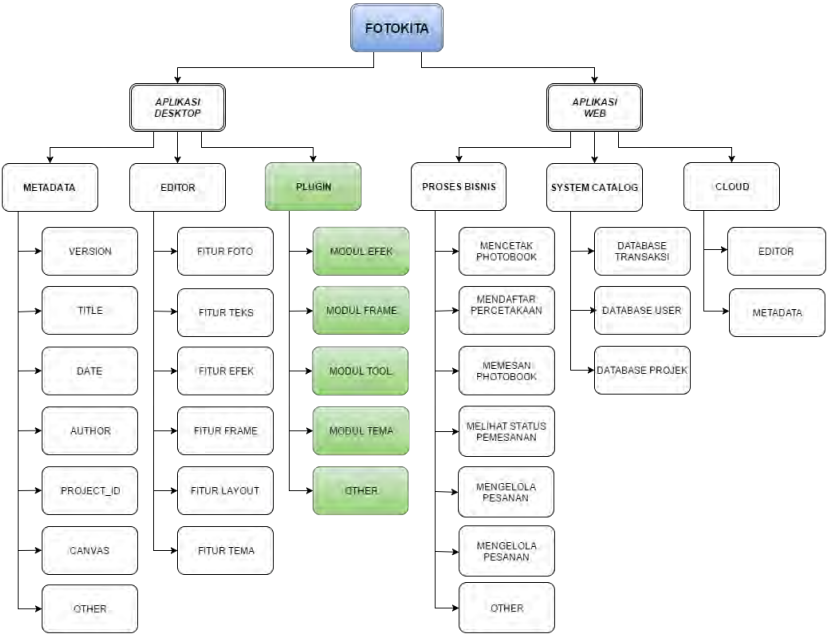
Tahap analisis dibagi menjadi beberapa bagian, antara lain pendeskripsian perangkat lunak, penggambaran dan penjelasan kasus penggunaan dan penjelasan alur aktivitas tiap kasus penggunaan dalam bentuk diagram aktivitas.

3.1.1 Deskripsi Umum Perangkat Lunak

Fotokita terdiri dari dua *platform* yaitu *desktop* dan *website* seperti yang ditunjukkan pada Gambar 3.1. Aplikasi berbasis *desktop* berfungsi sebagai *editor* foto album dan *website* berfungsi untuk menangani proses bisnis yang terjadi antara pelanggan dan percetakan dalam proses pencetakan album foto. Bagian kotak yang berwarna hijau merupakan bagian yang dikerjakan pada tugas akhir ini ditambah dengan perancangan arsitektur Fotokita secara modular dengan membagi aplikasi menjadi elemen kecil yang dijelaskan pada Subbab 3.2.1.

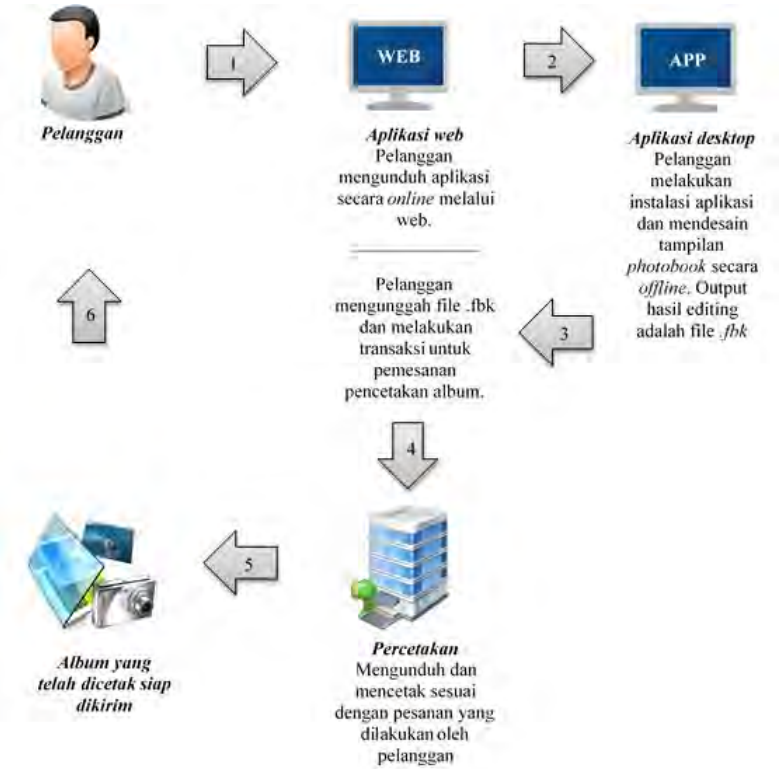
Secara garis besar, aplikasi utama Fotokita memiliki alur seperti berikut: pelanggan dapat mengunduh aplikasi Fotokita secara *online* melalui web. Dengan Fotokita, pelanggan dapat melakukan instalasi aplikasi dan mendesain album foto digital secara *offline*. Keluaran hasil editan adalah berkas .fbk yang hanya

bisa dibuka dengan menggunakan aplikasi Fotokita. Selanjutnya, untuk dapat dicetak, pelanggan harus mengunggah berkas .fbk tersebut ke *website* untuk melakukan pemesanan pencetakan album. Percetakan lalu mengunduh dan mencetak album tersebut sesuai dengan order yang dilakukan pelanggan. Setelah itu, percetakan mengirim album foto kepada pelanggan. Abstraksi alur aplikasi Fotokita digambarkan pada Gambar 3.2.



Gambar 3.1 *Grand Design* aplikasi Fotokita

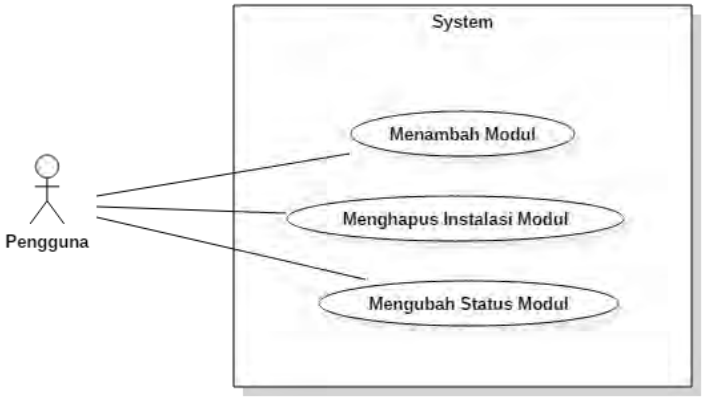
Fotokita modular merupakan perangkat lunak yang mengintegrasikan modul-modul bawaan pada aplikasi Fotokita yaitu modul *frame*, modul *tool* dan modul efek. Fitur administratif perangkat lunak utama merupakan fitur bagi pengguna untuk menambah modul, menghapus instalasi modul dan mengaktifkan/mengnonaktifkan modul.



Gambar 3.2 Deskripsi Umum Perangkat Lunak

3.1.2 Kasus Penggunaan

Bagian ini menjelaskan analisa dan perancangan kasus-kasus penggunaan yang terdapat pada fitur administratif pada perangkat lunak. Tiap kasus penggunaan akan disertakan spesifikasi dan diagram aktivitasnya. Kasus penggunaan dari aplikasi Fotokita dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram Kasus Penggunaan

3.1.2.1 Aktor

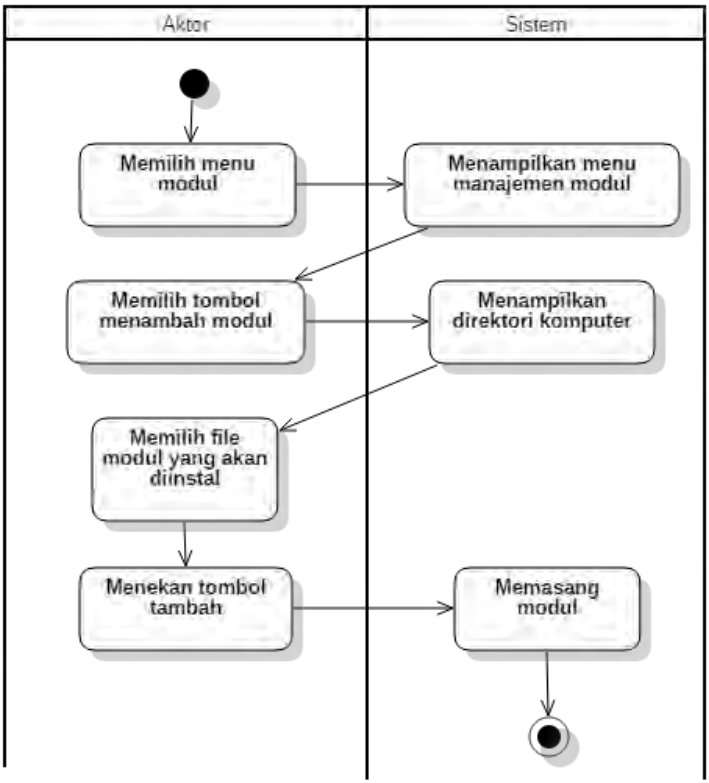
Aktor adalah entitas luar yang terlibat langsung dalam penggunaan perangkat lunak. Pada aplikasi ini aktor adalah pengguna. Pengguna dapat melakukan perubahan pada sistem manajemen modul yaitu menambah modul, menghapus modul dan mengubah status modul. Setiap aktivitas yang dilakukan oleh pengguna hanya akan berpengaruh pada aplikasi di desktop yang digunakan pengguna saat melakukan aktivitas.

3.1.2.2 Kasus Penggunaan Menambah Modul

Modul yang dibuat wajib dimuat ke dalam berkas Java *Archive* (.jar) kemudian aktor mengunggah berkas modul tersebut ke dalam perangkat lunak. Modul yang dibangun harus mengikuti konvensi dan memiliki properti yang telah ditentukan pada aplikasi sehingga bisa dibaca. Properti yang telah didefinisikan di dalam modul akan otomatis dibaca sehingga aktor tidak perlu mengisi kembali properti modul tersebut melalui formulir desktop. Tabel spesifikasi kasus penggunaan menambah modul dapat dilihat pada Tabel 3.1. Gambar 3.4 menggambarkan diagram aktivitas dari kasus penggunaan menambah modul.

Tabel 3.1 Spesifikasi Kasus Penggunaan Menambah Modul

Komponen	Deskripsi
Nama	Menambah modul
Nomor	UC-001
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor menambah modul ke dalam perangkat lunak album foto digital Fotokita
Tipe	Fungsional
Aktor	Pengguna
Kondisi Awal	Modul belum terdaftar
Kondisi Akhir	Modul terdaftar
Alur Normal	1. Aktor memilih menu modul yang ingin ditambah 2. Sistem menampilkan menu manajemen modul 3. Aktor memilih tombol menambah modul 4. Sistem menampilkan direktori komputer yang digunakan 5. Aktor memilih modul yang ingin ditambahkan 6. Aktor memilih tombol tambah 7. Sistem memasang (*install*) modul
Alur Alternatif	7.1 Modul tidak sesuai 7.2 Sistem mengeluarkan pesan error



Gambar 3.4 Diagram Aktivitas Menambah Modul

3.1.2.3 Kasus Penggunaan Menghapus Instalasi Modul

Pada kasus penggunaan ini, aktor menghapus instalasi modul yang sudah tidak diinginkan. Ketika modul dihapus, modul tetap akan berada dalam sistem namun tidak akan dimuat. Untuk menambahkan modul kembali ke dalam sistem pengguna diharuskan untuk melakukan proses penambahan modul dari awal

jika ingin memakai modul yang sudah dihapus instalasinya. Tabel spesifikasi kasus penggunaan menghapus modul dapat dilihat pada Tabel 3.2. Gambar 3.5 menggambarkan diagram aktivitas dari kasus penggunaan menghapus modul.

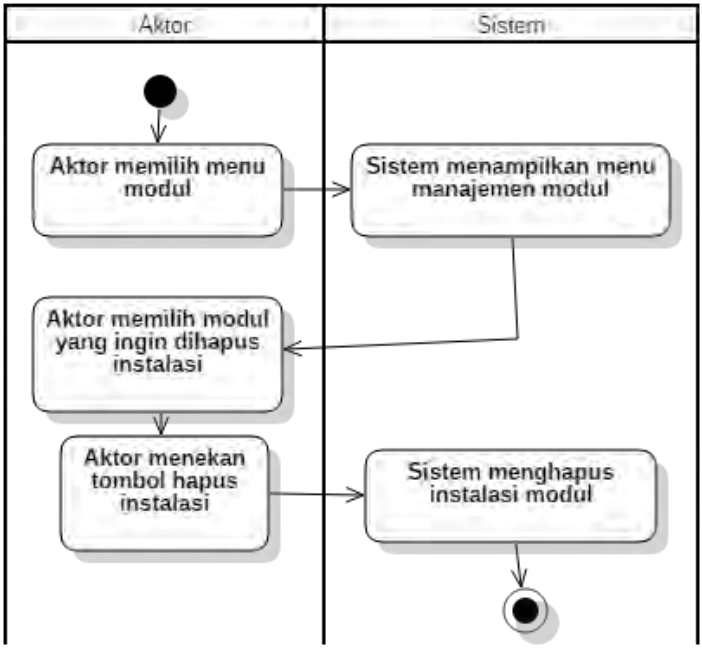
Tabel 3.2 Kasus Penggunaan Menghapus Instalasi Modul

Komponen	Deskripsi
Nama	Mengelola Modul
Nomor	UC-002
Deskripsi	Kasus penggunaan ini digunakan oleh pengguna untuk menghapus satu modul yang sudah terpasang pada sistem.
Tipe	Fungsional
Aktor	Pengguna
Kondisi Awal	Modul masih ada pada sistem.
Kondisi Akhir	Modul sudah terhapus dari sistem.
Alur Normal	1. Aktor memilih menu modul 2. Sistem menampilkan menu manajemen modul 3. Aktor memilih modul yang ingin dihapus instalsinya dengan cara klik kiri mouse pada baris dalam tabel 4. Aktor menekan tombol hapus instalasi 5. Sistem menghapus instalasi modul
Alur Alternatif	

3.1.2.4 Kasus Penggunaan Mengubah Status Modul

Pada kasus penggunaan ini, aktor mengubah status modul yang sudah terpasang pada sistem. Aktor dapat mengubah status modul dari aktif menjadi tidak aktif atau sebaliknya. Jika modul aktif, maka modul terpasang dan terlihat oleh pengguna, dan jika modul tidak aktif, maka modul tetap terpasang tetapi tidak tampil pada user interface fotokita. Tabel spesifikasi kasus penggunaan

mengubah status modul dapat dilihat pada Tabel 3.3. Gambar 3.6 menggambarkan diagram aktivitas dari kasus penggunaan mengubah status modul.

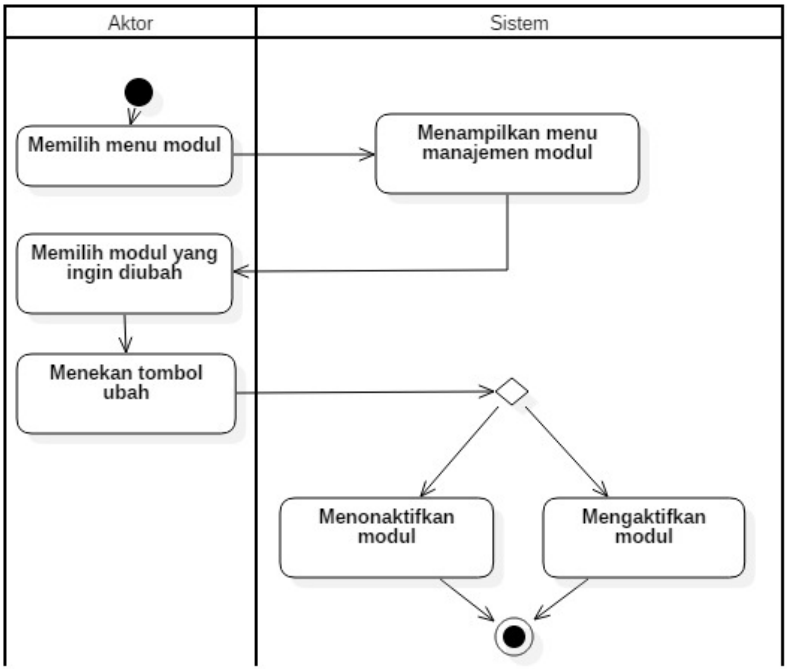


Gambar 3.5 Diagram Aktivitas Menghapus Modul

Tabel 3.3 Spesifikasi Kasus Penggunaan Mengubah Status Modul

Komponen	Deskripsi
Nama	Mengelola Status Modul
Nomor	UC-003
Deskripsi	Kasus penggunaan ini digunakan oleh pengguna untuk mengubah status modul yang sudah terpasang menjadi aktif atau tidak aktif.

Tipe	Fungsional
Aktor	Pengguna
Kondisi Awal	Modul aktif/tidak aktif
Kondisi Akhir	Modul tidak aktif/aktif
Alur Normal	1. Aktor memilih menu modul 2. Sistem menampilkan menu manajemen modul 3. Aktor memilih modul yang ingin diubah statusnya 4. Aktor menekan tombol ubah 5. Sistem mengaktifkan modul
Alur Alternatif	5.1. Sistem menonaktifkan modul



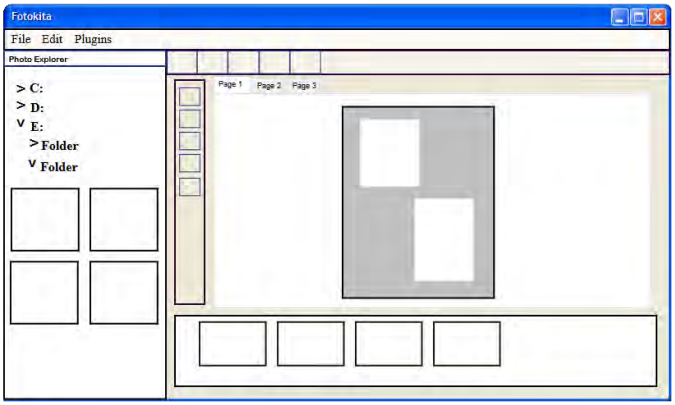
Gambar 3.6 Diagram Aktivitas Mengubah Status Modul

3.2 Perancangan

Pada subbab perancangan akan dijelaskan mengenai perancangan modularitas dan perancangan antarmuka pengguna.

3.2.1 Perancangan *Interface* Fotokita

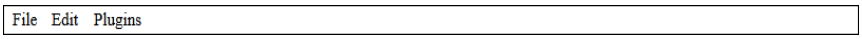
Fotokita merupakan aplikasi *standalone* yang dapat dijalankan mandiri. Untuk membuat class diagram dan juga menentukan bagian apa saja yang bisa dijadikan *plugin* dan juga atribut yang dibutuhkan untuk mengembangkan modul, rancangan aplikasi Fotokita dipecah menjadi beberapa bagian. Secara umum, fotokita memiliki tampilan antarmuka seperti pada Gambar 3.7.



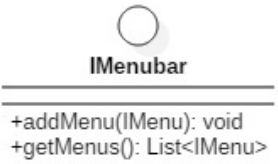
Gambar 3.7 Rancangan Antarmuka Aplikasi Fotokita

3.2.1.1 Menubar

Menubar adalah strip horizontal yang berisi daftar menu yang tersedia seperti pada Gambar 3.8. *Interface* yang akan diimplementasika digambarkan pada Gambar 3.9.



Gambar 3.8 Menubar



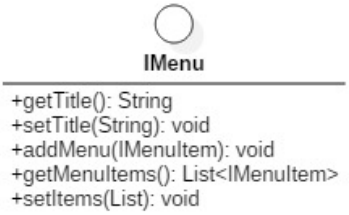
Gambar 3.9 Interface IMenubar

3.2.1.1.1 Menu

Menu berisi daftar tindakan yang dapat dilakukan pengguna yang dikelompokkan berdasarkan fungsi. Elemen dari menu digambarkan pada Tabel 3.4. Dari daftar menu yang digambarkan pada Gambar 3.7, menu *file*, menu *edit* dan menu *plugins* masing-masing berisi beberapa kesamaan, yaitu masing-masing memiliki nama dan berisi satu atau lebih *sub*-menu. Oleh karena itu dapat dibuat satu *interface* IMenu yang digambarkan pada Gambar 3.10.

Tabel 3.4 Elemen Menu

Elemen	Keterangan
Nama	Merupakan nama menu
Menu item	Isi dari menu yang masing-masing memiliki fungsi ketika ditekan.

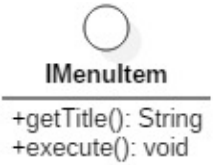


Gambar 3.10 Interface IMenu

3.2.1.1.2 Menu Item

Menu item merupakan isi dari menu yang masing-masing memiliki fungsi ketika ditekan. Dari Tabel 3.5, Tabel 3.6 dan Tabel

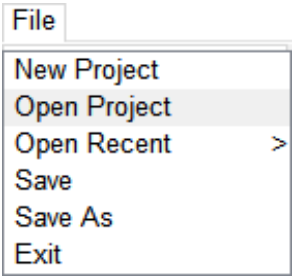
3.7**,** terdapat beberapa kesamaan yaitu setiap menu item memerlukan nama dan juga fungsi, yang dapat disimpulkan dengan sebuah *interface* pada Gambar 3.11.



Gambar 3.11 Interface IMenuItem

3.2.1.1.2.1 Menu Item File

Menu item merupakan isi dari menu yang masing-masing memiliki fungsi ketika ditekan. Menu item dari menu file digambarkan pada Gambar 3.12 dan fungsinya dapat dilihat pada Tabel 3.5.



Gambar 3.12 Rancangan Antarmuka *MenuItem* pada Menu File

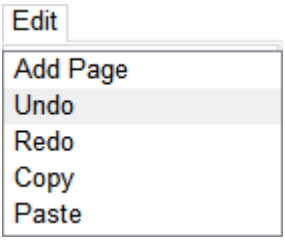
Tabel 3.5 Menu Item beserta Fungsinya dari Menu File

Menu Item	Fungsi ketika ditekan
New Project	Membuka *dialog window* baru yang berisi ukuran kertas dan jumlah halaman project yang akan dibuat
Open Project	Membuka *dialog window* baru yang berisi direktori komputer. Hanya berkas yang

	bertipe .fbk dan *folder* yang akan ditampilkan.
Open Recent	Ketika di *hover* akan menampilkan submenu yang berisi daftar *project* yang pernah dibukan diurutkan berdasarkan waktu
Save	Menyimpan *project*. Untuk penyimpanan pertama kali, muncul *dialog window* untuk mengubah nama penyimpanan *project*.
Save As	Membuka *dialog window* yang berisi direktori penyimpanan.
Exit	• Jika *project* belum disimpan, maka akan menawarkan pilihan untuk menyimpan *project*. • Jika sudah disimpan, maka langsung menutup aplikasi.

3.2.1.1.2.2 Menu Item – Edit

Menu item dari menu edit bersi fungsi-fungsi yang berpengaruh pada *canvas.* Menu item dari menu edit digambarkan pada Gambar 3.13 dan fungsinya dapat dilihat pada Tabel 3.6.



Gambar 3.13 Rancangan Antarmuka Menu Item pada Menu Edit

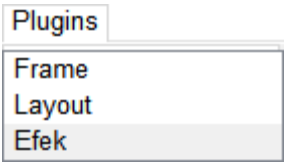
Tabel 3.6 Menu Item beserta Fungsinya dari Menu Edit

Menu Item	Fungsi ketika ditekan
Add Page	Menambah satu halaman baru pada *project*

Undo	Mengembalikan *state canvas* sebelum aksi terakhir
Redo	Mengembalikan *state canvas* setelah aksi terakhir
Copy	Menyalin objek yang dipilih
Paste	Menampilkan objek yang telah di*copy*

3.2.1.1.2.3 Menu Item – Plugins

Menu item dari menu plugins bersi fungsi di mana pengguna dapat melakukan manajemen terhadap modul yang disediakan. Menu item dari menu plugins digambarkan pada Gambar 3.14 dan fungsinya dapat dilihat pada Tabel 3.7.



Gambar 3.14 Rancangan Antarmuka Menu Item pada Menu Plugins

Tabel 3.7 Menu Item beserta Fungsinya dari Menu Plugins

Menu Item	Fungsi ketika ditekan
Frame	Menampilkan window baru yang berisi manajemen modul *frame*. Dalam window ini, pengguna dapat menambahkan, menghapus dan mengubah status plugin.
Tool	Menampilkan window baru yang berisi manajemen modul *tool*. Dalam window ini, pengguna dapat menambahkan, menghapus dan mengubah status plugin.
Efek	Menampilkan window baru yang berisi manajemen modul efek. Dalam window ini,

	pengguna dapat menambahkan, menghapus dan mengubah status plugin.

3.2.1.2 Toolbox dan Tools

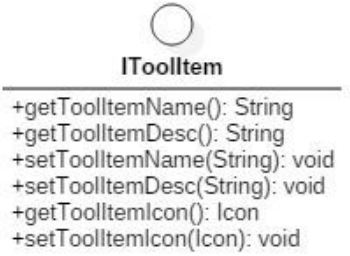
Toolbox merupakan strip vertical yang berisi daftar *tools* yang dapat digunakan untuk memanipulasi objek pada *canvas.* Gambar 3.16 menunjungan rancangan antarmuka *toolbox*. *Toolitem* merupakan bagian dari *toolbox* yang berfungsi untuk melakukan manipulasi objek pada *canvas*. Fungsi-fungsi *tools* yang ada pada toolbox dijabarkan pada Tabel 3.8. Gambar 3.15 dan Gambar 3.17 masing-masing menggambarkan *interface* dari *IToolbox* dan *IToolItem*.

Gambar 3.15 Interface *IToolbox*

Gambar 3.16 Rancangan Antarmuka Toolbox

Tabel 3.8 Tool Item beserta fungsinya dari Toolbox

Tools	**Fungsi**
Select	• Memilih objek yang berada di paling atas dengan klik kiri pada mouse • Ketika objek dipilih akan muncul penanda (kotak-kotak) pada beberapa titik objek
Move	Memindahkan objek ke posisi baru
Teks	• Menambahkan teks baru pada canvas • Mengubah teks yang sudah ada
Zoom in	Memperbesar *view* pada *canvas*
Zoom out	Mengecilkan *view* pada *canvas*



Gambar 3.17 Interace IToolItem

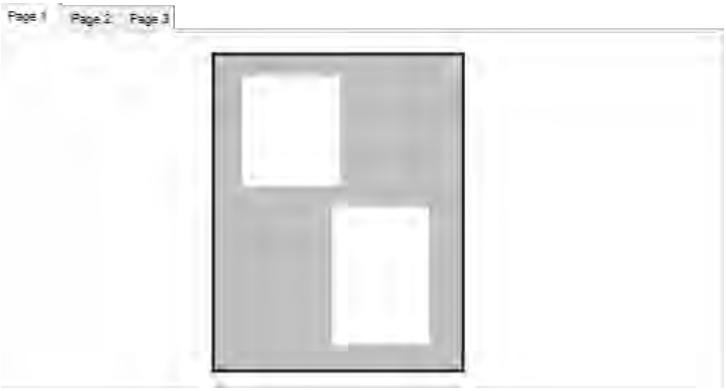
3.2.1.3 Toolbar dan Bar

Toolbar merupakan strip horizontal yang terletak di bawah menubar. *Toolbar* berisi daftar bar yang merupakan icon beberapa menu item pada menu file dan menu edit. *Toolbar* digambarkan pada Gambar 3.18 dan fungsi masing-masing bar dijabarkan pada Tabel 3.5 dan Tabel 3.6.

Gambar 3.18 Rancangan Antarmuka Toolbar

3.2.1.4 InitTab dan Page

InitTab merupakan wadah untuk menampung *page. Page* atau *canvas* merupakan wadah dimana objek-objek ditempatkan. *Page* berisi objek-objek yang dapat dimanipulasi oleh pengguna. InittTab dan Page digambarkan pada Gambar 3.19. Kelas Interface dari InitTab dan Page dapat dilihat pada Gambar 3.20.



Gambar 3.19 Rancangan Antarmuka InitTab dan Page

Gambar 3.20 Interface IInitTab dan IPage

3.2.1.5 Foto

Foto merupakan elemen utama dari apliaksi Fotokita. Pengguna dapat menambahkan dan memanipulasi foto yang

terdapat pada penyimpanan eksternal maupun internal. Contoh foto dapat dilihat pada Gambar 3.21**.**

Atribut yang dibutuhkan setiap foto adalah:

- Koordinat x awal
- Koordinat y awal
- Panjang foto
- Lebar foto
- Nama foto
- ID unik foto
- Gambar

Gambar 3.21 Foto

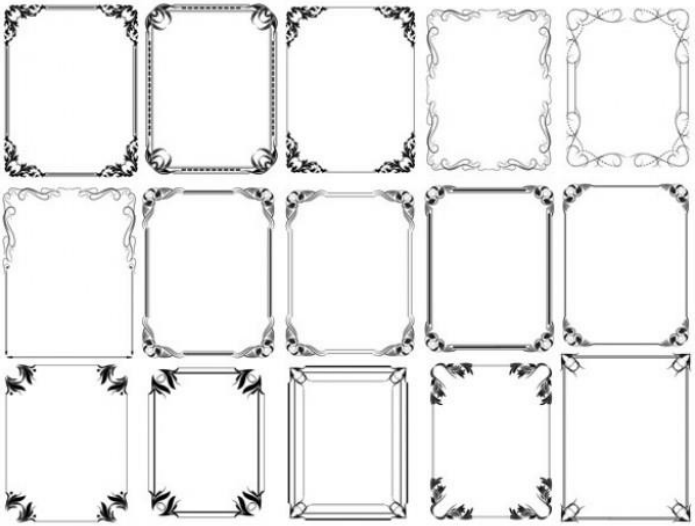
3.2.1.6 Frame

Frame atau bingkai merupakan tepi dekoratif yang digunakan untuk menghias foto. Pada aplikasi Fotokita, *frame* bersifat tidak wajib dipasang, tetapi pengguna dapat mengganti

frame sesuai dengan selera. Beberapa contoh frame dapat dilihat pada Gambar 3.22.

Atribut yang dibutuhkan setiap *frame* adalah:

- Koordinat x awal
- Koordinat y awal
- Panjang *frame*
- Lebar *frame*
- Nama *frame*
- ID unik *frame*



Gambar 3.22 *Frame*

3.2.1.7 Layout dan *Photoholder*

Layout merupakan tampilan tidak kasat mata saat halaman dicetak, *layout* berisi sekumpulan objek dan *photoholder*. *Photoholder* merupakan wadah dimana foto akan ditempatkan. Setiap halaman dapat berisi layout yang berbeda, dan setiap layout terdiri dari *photoholder* dengan jumlah dan posisi yang berbeda.

Atribut yang dibutuhkan oleh *photoholder* adalah:

- Koordinat x awal
- Koordinat y awal
- Panjang *photoholder*
- Lebar *photoholder*
- Nama *photoholder*
- ID unik *photoholder*

3.2.1.8 *Enhancement* (Efek)

Fitur *enhancement* atau efek *filter* tidak bersifat wajib, tetapi pengguna dapat menambahkan fitur tersebut ke dalam foto untuk melakukan peningkatan kualitas pada foto atau sesuai dengan selera pengguna.

3.2.1.9 Teks

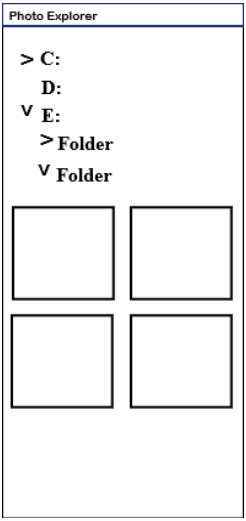
Teks merupakan sekumpulan tulisan yang dapat ditambahkan oleh penguna ke dalam *page*. Seperti foto, teks dapat diubah ukuran maupun konteks isinya.

Atribut yang dibutuhkan oleh teks adalah:

- Koordinat x awal
- Koordinat y awal
- Panjang teks
- Lebar teks
- ID unik teks
- Isi teks
- *Style* (cetak tebal, cetak miring, garis bawah)
- Ukuran teks
- Jenis *font*

3.2.1.10 Photo Explorer

Photo explorer merupakan *explorer* yang berisi direktori yang menujukkan isi direktori foto yang ada di sebuah *folder*. Ketika direktori tersebut di tekan, *photo explorer* akan menampilkan *preview* foto. Jika ditekan dua kali, maka akan keluar *edit window* dimana pengguna dapat mengedit foto tersebut (menambahkan efek dan *frame*). Ilustrasi *photo explorer* dapat dilihat pada Gambar 3.23.



Gambar 3.23 Rancangan Antarmuka *Photo Explorer*

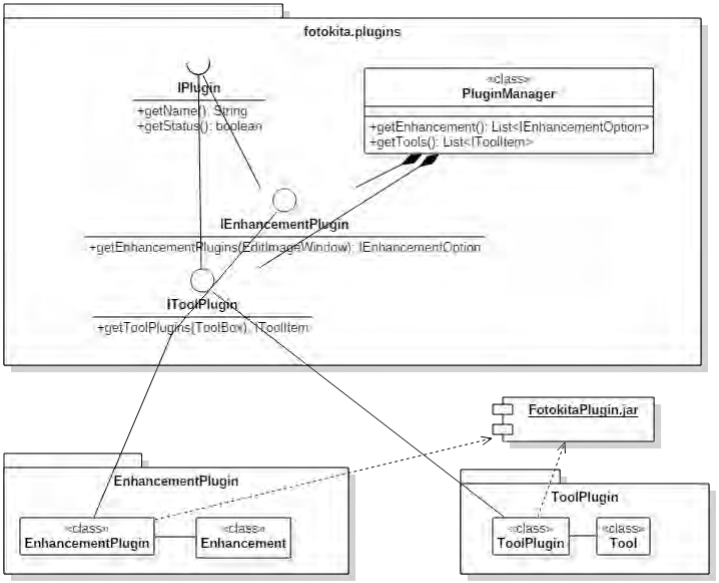
3.2.2 Perancangan Diagram Kelas

Pada Gambar 3.24 menunjukkan model arsitektur modular Fotokita. Aplikasi Fotokita memiliki *package plugins* yang berisi *interface* dan kelas yang menangani sistem modularitas pada Fotokita dan *package window* yang berisi antarmuka.

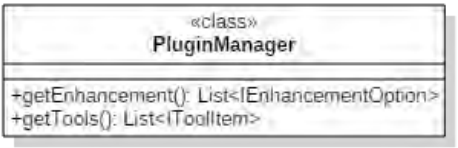
3.2.2.1 Kelas *PluginManager*

Kelas *PluginManager* yang dapat dilihat pada Gambar 3.25 merupakan kelas penghubung antara *plugin* yang berbentuk *java*

archives (.jar) dan Fotokita yang berfungsi untuk memanggil dan memasang *plugin* yang terdapat pada folder Plugins.



Gambar 3.24 Diagram Kelas Fotokita

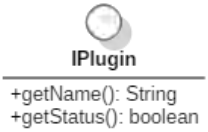


Gambar 3.25 Kelas *PluginManager*

3.2.2.2 Interface IPlugin

Interface IPlugin yang dapat dilihat pada Gambar 3.26 merupakan *interface* yang berisi fungi yang harus di*extends* oleh

interface tiap modul. *IPlugin* berisi fungsi untuk mengambil nama modul.



Gambar 3.26 *Interface IPlugin*

3.2.2.3 Interface IEnhancementPlugin

Gambar 3.27 *Interface IEnhancementPlugin*

Interface IEnhancementPlugin dapat dilihat pada Gambar 3.27. Setiap *java archive* modul *Enhancement* wajib memiliki satu kelas yang mengimplementasikan *IEnhancementPlugin*. *IEnhancementPlugin* berisi fungsi yang menghubungkan Fotokita dan kelas *AEnhancementOption* pada *package widget.edit* Fotokita.

3.2.2.4 Interface IToolPlugin

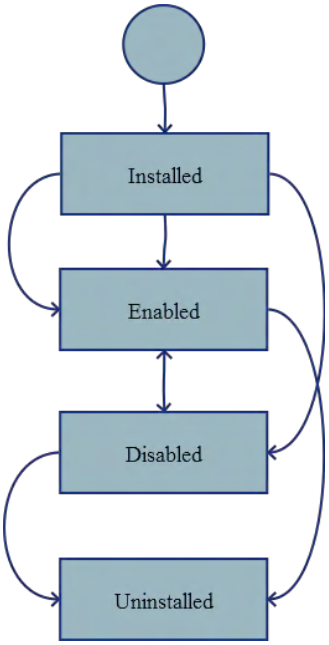
Gambar 3.28 *Interface IToolPlugin*

Interface IToolPlugin ditunjukkan pada Gambar 3.28. setia berkas arsip java modul *Tool* wajib memiliki satu kelas yang mengimplementasikan *IToolPlugin. IToolPlugin* berisi fungsi yang menghubungkan Fotokita pada kelas abstrak ATool pada *package widget.tool.toolbox* Fotokita.

3.2.3 Perancangan Modularitas

Pada bagian ini akan dibahas daur hidup modul (*development life cycle*) dan arsitektur yang harus digunakan modul *(development environment*).

3.2.3.1 Daur Hidup Modul (*Development Life Cycle)*



Gambar 3.29 Daur Hidup Modul

Daur hidup modul pada Fotokita dapat dilihat pada Gambar 3.29. Daur hidup modul dalam aplikasi ini adalah:

- *Installed*
  Modul telah berhasil dipasang.
- *Enabled*
  Modul terpasang dan diaktifkan.
- *Disabled*
  Modul terpasang tetapi tidak aktif.
- *Uninstalled*
  Modul telah dihapus pemasangannya.

3.2.3.2 *Development Environment*

Modul yang dikembangkan wajib memiliki komponen yang mendefinisikan properti modul. Modul akan menggunakan beberapa kelas pada perangkat lunak utama. Karena itu, modul memiliki ketergantungan/dependensi dengan komponen lain dari perangkat utama. Komponen yang menjadi dependensi yaitu:

1. **fotokita.plugin**
   Modul yang dibuat wajib memiliki suatu kelas yang merealisasikan kelas interface *IModulPlugin* (jika modul *enhancement* maka merealisasikan kelas interface *IEnhancementPlugin*) dan kelas *IPlugin*.
2. **fotokita.widget.edit.*enhancement***
   Efek yang dibuat wajib memiliki satu kelas yang mengimplementasikan kelas abstrak efek yang berada pada komponen ini.
3. **Fotokita.widget.tool.toolbox**
   Tool yang dibuat wajib memiliki satu kelas yang mengimplementasikan kelas abstrak *tool* yang berada pada komponen ini.

Setelah modul dibuat, modul di-*build* ke dalam berkas arsip Java (*java archives .jar)* untuk kemudian ditambahkan ke dalam perangkat lunak utama.

3.2.4 Perancangan Data

Pada bagian ini akan dibahas mengenai data yang tersimpan melalui aplikasi terkait dengan daur hidup modul. Dengan menggunakan *Java Properties*, setiap berkas arsip java yang berhasil dipasang pada aplikasi akan memiliki satu berkas *properties* yang berisi beberapa data yang dijelaskan pada

Tabel 3.9 Perancangan Berkas *Properties*

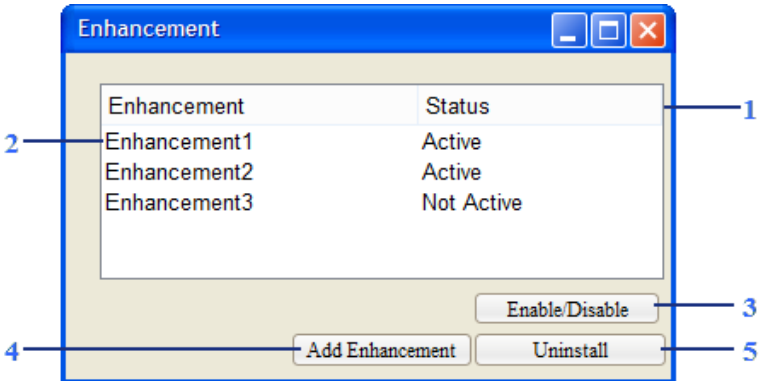
Nama *Property*	Keterangan
pluginName	Nama berkas arsip java
pluginPath	Direktori berkas arsip java
enabled	Modul terpasang atau tidak terpasang
active	Status modul

3.2.5 Perancangan Antarmuka Pengguna

Perancangan antarmuka pengguna merupakan hal yang penting dalam melakukan perancangan aplikasi. Antarmuka pengguna yang berhubungan langsung dengan aktor harus memiliki kemudahan-kemudahan dan tampilan yang rapi dan menarik bagi penggunanya. Sistem memiliki 3 antarmuka pengguna, yaitu *enhancement management window, tool management window* dan *frame management window*.

3.2.5.1 Rancangan Antarmuka *Enhancement Management Window*

Window ini digunakan untuk kasus penggunaan menambah *enhancement*, menghapus *enhancement* dan mengubah status *enhancement*. Pada *window* ini terdapat tabel yang berisi daftar *enhancement* yang telah ditambahkan ke dalam aplikasi fotokita beserta statusnya apakah aktif atau tidak. Rancangan antarmuka dapat dilihat pada Gambar 3.30. Penjelasan dari elemen rancangan antarmuka berada pada Tabel 3.10.



Gambar 3.30 Rancangan Antarmuka *Enhancement Plugin Window*

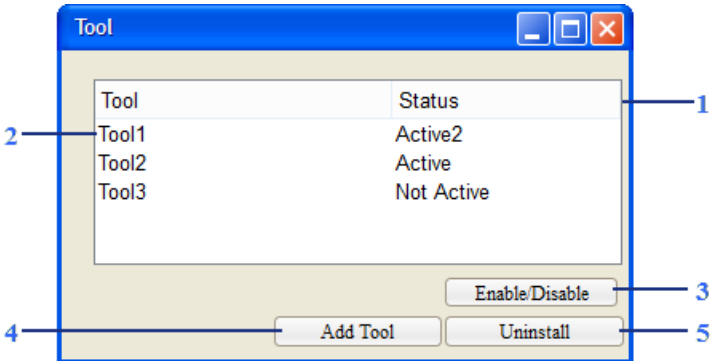
Tabel 3.10 Penjelasan Antarmuka *Enhancement Management Window*

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Ke-luaran
1	*tableEnhancement*	*Table*	Menampilkan plugins.	*Table*
2	*enhancement*	*Table item*	Menampilkan nama plugins beserta status.	*TableItemClick*
3	*statusEnhancementButton*	*Button*	Tombol aksi untuk melakukan perubahan *life*	*ButtonClick*

			cycle enhancement ke/dari aktif/non-aktif	
4	*addButton*	*Button*	Tombol aksi untuk menambahkan plugin.	*ButtonClick*
5	*deleteButton*	*Button*	Tombol aksi untuk menghapus plugin yang terpilih.	*ButtonClick*

3.2.5.2 Rancangan Antarmuka *Tool Management Window*

Window ini digunakan untuk kasus penggunaan menambah *tool*, menghapus *tool* dan mengubah status *tool*. Pada *window* ini terdapat tabel yang berisi daftar *tool* yang telah ditambahkan ke dalam aplikasi fotokita beserta statusnya apakah aktif atau tidak. Rancangan antarmuka dapat dilihat pada Gambar 3.30. Penjelasan dari elemen rancangan antarmuka berada pada Tabel 3.10.



Gambar 3.31 Rancangan Antarmuka *Tool Plugin Window*

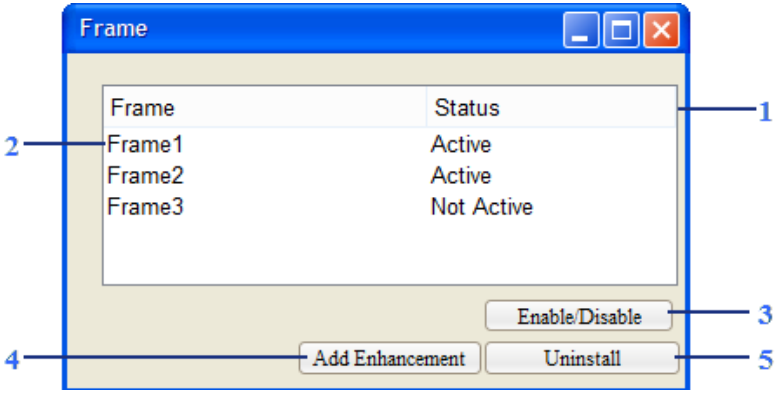
Tabel 3.11 Penjelasan Antarmuka *Tool Management Window*

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Keluaran
1	*tableTool*	*Table*	Menampilkan plugins.	*Table*
2	*tool*	*Table item*	Menampilkan nama plugins beserta status.	*TableItemClick*
3	*statusToolButton*	*Button*	Tombol aksi untuk melakukan perubahan *life cycle tool* ke/dari aktif/non-aktif	*ButtonClick*
4	*addButton*	*Button*	Tombol aksi untuk menambahkan plugin.	*ButtonClick*
5	*deleteButton*	*Button*	Tombol aksi untuk	*ButtonClick*

			menghapus plugin yang terpilih.	

3.2.5.3 Rancangan Antarmuka *Frame Management Window*

Window ini digunakan untuk kasus penggunaan menambah *frame*, menghapus *frame* dan mengubah status *frame*. Pada *window* ini terdapat tabel yang berisi daftar *frame* yang telah ditambahkan ke dalam aplikasi fotokita beserta statusnya apakah aktif atau tidak. Rancangan antarmuka dapat dilihat pada Gambar 3.30. Penjelasan dari elemen rancangan antarmuka berada pada Tabel 3.10.



Gambar 3.32 Rancangan Antarmuka *Frame Plugin Window*

Tabel 3.12 Penjelasan Antarmuka *Frame Management Window*

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Keluaran
1	*tableFrame*	*Table*	Menampilkan plugins.	*Table*

2	*frame*	*Table item*	Menampilkan nama plugins beserta status.	*TableItemClick*
3	*statusFrame Button*	*Button*	Tombol aksi untuk melakukan perubahan *life cycle frame* ke/dari aktif/non-aktif	*ButtonClick*
4	*addButton*	*Button*	Tombol aksi untuk menambahkan plugin.	*ButtonClick*
5	*deleteButton*	*Button*	Tombol aksi untuk menghapus plugin yang terpilih.	*ButtonClick*

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas tahap-tahap implementasi dari Fotokita berdasarkan perancangan yang telah dibahas pada bab sebelumnya dan aturan pembuatan modul perangkat lunak Fotokita.

Implementasi yang dijelaskan meliputi lingkungan pembangunan perangkat lunak, kode sumber utama dan implementasi antaruka pengguna. Bahasa pemrograman yang digunakan adalah Java.

4.1 Lingkungan Implementasi

Dalam merancang perangkat lunak ini digunakan beberapa perangkat pendukung yang terdiri dari perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam membangun web layanan transaksi *photobook* adalah sebagai berikut:

Tipe : HP Pavilion 7
Prosesor : Intel ® Core ™ i7-4510U CPU @ 2.00 GHz
Memori : 8.00 GB RAM
Sistem Operasi : Windows 8.1 Professional 64 bit

4.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan dalam Fotokita modular adalah sebagai berikut:

- Eclipse Mars digunakan sebagai IDE
- StarUML 2.7.0 digunakan untuk membuat digram kasus penggunaan, diagram aktivitas dan diagram kelas.

4.2 Implementasi Proses Bisnis

Pada subbab ini akan dijelaskan mengenai implementasi proses bisnis yang dibahas pada subbab 3.1.2.

4.2.1 Implementasi Kasus Penggunaan Menambah Modul

Kode Sumber 4.1 *Pseudocode* Kasus Penggunaan Menambah ModulKode Sumber 4.1 menunjukkan *pseudocode* dari kasus penggunaan menambah modul. Ketika tombol *add* pada jendela manajemen modul ditekan, *file dialog* akan muncul dan menampilkan berkas yang mempunyai ekstensi .jar. Ketika berkas arsip Java dipilih dan tombol *open* ditekan, berkas arsip Java akan diperiksa terlebih dahulu apakah sudah memenuhi aturan modularitas dan/atau berkas tersebut belum ada, jika iya maka berkas tersebut akan disalin ke dalam *folder* Plugins/Modul pada proyek, dan jika tidak maka pesan error akan dimunculkan. Aplikasi juga akan membuat satu berkas *properties* yang berisi informasi berkas arsip Java tersebut, yaitu nama plugin, lokasi plugin, status plugin dan aktif/tidaknya plugin. Berkas *properties* ini berfungsi untuk manajemen modul. Kode program tiap modul dapat dilihat pada Kode Sumber A.7.1, Kode Sumber A.7.2 dan Kode Sumber A.7.3 pada lampiran.

4.2.2 Implementasi Kasus Penggunaan Menghapus Instalasi Modul

Kode Sumber 4.2 menunjukkan *pseudocode* dari kasus penggunaan menghapus instalasi modul. Ketika salah satu baris dari dari tabel ditekan dan tombol *uninstall* pada jendela manajemen modul ditekan, sistem akan membuka *berkas properties* dari modul yang ingin dihapus instalasi, lalu akan merubah properti *enabled* menjadi *no*. Untuk melihat perubahan, aplikasi harus di*restart* ulang terlebih dahulu. Berkas arsip Java pada *folder Plugins*/Modul dapat dihapus oleh pengguna secara

manual. Kode program tiap modul dapat dilihat pada Kode Sumber A.7.4 pada lampiran.

```
read File's path
if file exist
  show error message;
else
  try
  load Plugin;
  if Plugin true
    try
      move File (.Jar) to Plugins/Modul
      try
        set Plugin's property
        store Plugin's property
        show message box
      catch IOException
        print stack trace
   catch IOException
      print stack trace
  else
    show message box
  catch MalformedURLException
    print stack trace
  catch ServiceConfigurationError
    show message box
```

Kode Sumber 4.1 *Pseudocode* Kasus Penggunaan Menambah Modul

4.2.3 Implementasi Kasus Penggunaan Mengubah Status Modul

Kode Sumber 4.3 menunjukkan implementasi dari kasus penggunaan mengubah status modul. Ketika salah satu baris dari table dipilih dan tombol *change status* pada Gambar 3.30 ditekan, sistem memeriksa properti *active* pada *berkas properties* modul yang dipilih. Jika properti *active* bernilai *true*, nilai properti diubah menjadi *false* dan sebaliknya jika properti *active* bernilai *false,* nilai properti diubah menajdi *true*.

```
get the selected file
try
  load file's property
  set property "enabled" to no
  store property
  show message box
catch FileNotFoundException
  print stack trace
catch IOException
  print stack trace
```

Kode Sumber 4.2 *Pseudocode* Kasus Penggunaan Menghapus Instalasi Modul

4.3 Implementasi Antarmuka Pengguna

Pada subbab ini akan menjelaskan dan menampilkan tampilan *window* untuk modul sesuai dengan rancangan antarmuka yang terdapat pada bab 3.

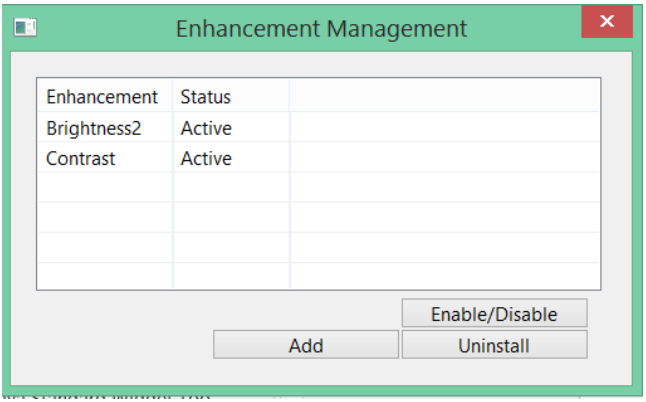
Gambar 4.1, Gambar 4.2 dan Gambar 4.3 menunjukkan antarmuka jendela untuk manajemen per modul. Masing-masing modul pada fotokita memiliki jendela manajemen modul sendiri. Jendela ini digunakan untuk kasus penggunaan UC-01, UC-02 dan UC-03.

4.4 Implementasi Data

Pada bagian ini, implementasi yang dijelaskan adalah implementasi dari perancangan data yang dijelaskan pada Subbab 3.2.4. Kode Sumber 4.4 merupakan implementasi dari pembuatan berkas *properties*. Pembuatan berkas *properties* terjadi pada kasus penggunaan menambah modul, hanya jika modul yang ditambahkan sudah sesuai dengan aturan modularitas yang ditetapkan. Berkas *properties* berada di direktori dan memiliki nama yang sama dengan berkas arsip java. Setiap berkas arsip java akan memiliki satu berkas *properties*.

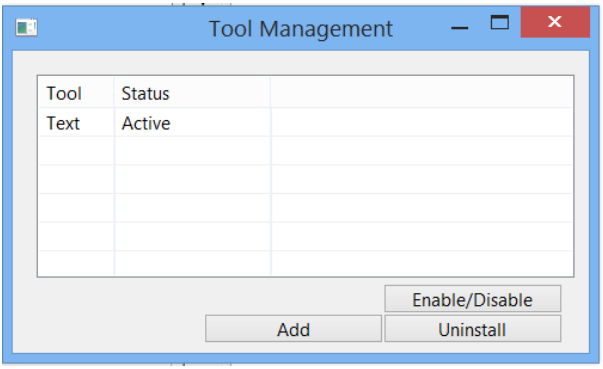
```
get the selected file
try
  load file's property
  if property "active" is true
    set property "active" to
false
  else
    set property "active" to
true
  store property
catch FileNotFoundException
  print stack trace
catch IOException
  print stack trace
```

Kode Sumber 4.3 *Pseudocode* Kasus Penggunaan Mengubah Status Modul

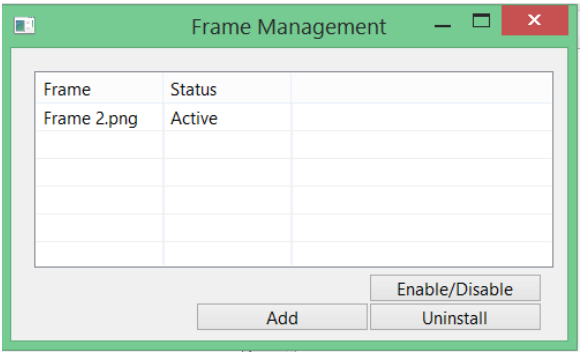


Gambar 4.1 Antarmuka Jendela Manajemen Modul *Enhancement*

Kode Sumber 4.5 merupakan implementasi dari pembacaan berkas *properties* yang digunakan pada kasus penggunaan mengubah status modul. Jika pengguna menekan tombol *enable/disable*, status modul pada berkas *properties* akan diperiksa lalu diubah.



Gambar 4.2 Antarmuka Jendela Manajemen Modul *Tool*



Gambar 4.3 Antarmuka Jendela Manajemen Modul *Frame*

.

```
1.  Properties pluginConfig = new Properties();
2.  FileOutputStream output = new FileOutputStream("Plu
    gins/Enhancement/" + sourceFileNameOnly + ".propert
    ies");
3.  pluginConfig.setProperty("pluginName", sourceFileNa
    meOnly);
4.  pluginConfig.setProperty("pluginPath" , destFiles);

5.  pluginConfig.setProperty("enabled", "yes");
6.  pluginConfig.setProperty("active", "true");
7.  pluginConfig.store(output, null);
```

```
8.  output.close();
```

Kode Sumber 4.4 Implementasi Pembuatan Berkas *Properties*

```
1.  FileInputStream in;
2.  in = new FileInputStream(propName);
3.  loadProp.load(in);
4.  in.close();
5.
6.  if(loadProp.getProperty("active").equals("true"))
7.  {
8.      loadProp.setProperty("active", "false");
9.      messageBox.setMessage("Plugin is disabled!");
10. }
11. else
12. {
13.     loadProp.setProperty("active", "true");
14.     messageBox.setMessage("Plugin is enabled!");
15. }
16. FileOutputStream out = new FileOutputStream(propNam
    e);
17. loadProp.store(out, null);
18. out.close();
```

Kode Sumber 4.5 Implementasi Pembacaan Berkas *Properties* pada Kasus Penggunaan Mengubah Status Modul

Kode Sumber 4.6 merupakan impelementasi pembacaan berkas *properties* pada kasus penggunaan menghapus instalasi modul. Ketika modul dihapus instalasinya, *property "enabled"* akan diubah menjadi *"no"*.

```
1.  in = new FileInputStream(propName);
2.  loadProp.load(in);
3.  in.close();
4.  FileOutputStream out = new FileOutputStream(propNam
    e);
5.  loadProp.setProperty("enabled", "no");
6.  loadProp.store(out, null);
```

```
7.  out.close();
```

Kode Sumber 4.6 Implementasi Pembacaan Berkas *Properties* pada Kasus Penggunaan Menghapus Instalasi Modul

Kode Sumber 4.7 merupakan pembacaan berkas *properties* pada pemuatan modul. Sebelum antarmuka pengguna sebuah modul ditampilkan, sebagai contoh *icon* pada *tool*, berkas *properties* dari modul tersebut akan diperiksa terlebih dahulu nilai pada *property "active"*. Jika bernilai "*false*" maka modul tidak akan dimuat.

```
1.  List<IEnhancementPlugin> plugins = PluginManager.ge
    tEnhancementPlugins();
2.  for (IEnhancementPlugin p : plugins)
3.  {
4.    try
5.    {
6.    FileInputStream in = new FileInputStream("Plugins
    /Enhancement/" + p.getName()+ ".properties");
7.    loadProp.load(in);
8.    in.close();
9.       if(loadProp.getProperty("active").equals("true
    "))
10.      {
11.        addEnhancementOption(p.getEnhacement(this));

12.      }
13.     } catch (IOException e) {
14.         // TODO Auto-generated catch block
15.         e.printStackTrace();
16.     }
17. }
```

Kode Sumber 4.7 Implementasi Pembacaan Berkas *Properties* untuk Memuat Modul

4.5 Implementasi *Interface Plugins*

Pada bagian ini, implementasi yang dijelaskan adalah implementasi *interface* pada *package plugins*. Beberapa *interface* dan kelas realisasi dari aplikasi utama Fotokita yang dibahas pada Subbab 3.2.1 diimplementasikan pada tugas akhir yang berjudul "*Rancang Bangun Aplikasi Editor Album Foto Digital ‚Fotokita' Berbasis Destkop*" [8] dan "*Rancang Bangun Aplikasi Album Foto Digital ‚Fotokita' dengan Penyimpanan Dinamis*" [9]. Manajemen modul *Frame* dilakukan tidak dengan menggunakan berkas arsip java. Penambahan *frame* dilakukan dengan menambahkan berkas gambar *frame* yang berekstensi .png.

4.5.1 Implementasi *Interface IPlugin*

Kode Sumber 4.8 menunjukkan implementasi dari *interface IPlugin*. *IPlugin* berisi fungsi getName() yang akan mengembalikan nama dari modul dan getStatus() yang aka mengembalikan status modul. Setiap modul yang dibuat harus mengimplementasikan *IPlugin*.

```
1.  package plugins;
2.
3.  public interface IPlugin
4.  {
5.      public String getName();
6.      public boolean getStatus();
7.  }
```

Kode Sumber 4.8 Implementasi *Interface IPlugin*

4.5.2 Implementasi *Interface IToolPlugin*

Kode Sumber 4.9 menunjukkan implementasi dari *interface IToolPlugin*. *IToolPlugin* meng*extend interface IPlugin* dan berisi fungsi getToolItem(ToolBox parent) yang akan mengembalikan *tool* yang akan ditambahkan. Kode Sumber 4.10 dan Kode Sumber

4.11 merupakan contoh realisasi *interface IToolPlugin*. Kode Sumber 4.9 Implementasi *Interface IToolPlugin*.

```
1.  package plugins;
2.
3.  import interfaces.IToolItem;
4.  import widget.tool.toolbox.ToolBox;
5.
6.  public interface IToolPlugin extends IPlugin
7.  {
8.      IToolItem getToolItem(ToolBox parent);
9.  }
```

Kode Sumber 4.9 Implementasi *Interface IToolPlugin*

```
1.  package widget.tool.toolbox;
2.
3.  import interfaces.IToolItem;
4.  import pointer;
5.
6.  public class PointerToolPlugin implements IToolPlug
    in
7.  {
8.
9.      @Override
10.     public String getName()
11.     {
12.         return "PointerTool";
13.     }
14.
15.     @Override
16.     public IToolItem getToolItem(ToolBox parent)
17.     {
18.         return new PointerTool(parent);
19.     }
20.
21. }
```

Kode Sumber 4.10 Realisasi *Interface IToolPlugin* untuk *Pointer Tool*

```
1.  package text;
2.
3.  import interfaces.IToolItem;
4.  import plugins.IToolPlugin;
5.  import widget.tool.toolbox.ToolBox;
6.
7.  public class TextToolPlugin implements IToolPlugin

8.  {
9.
10.     @Override
11.     public String getName()
12.     {
13.         return "Text";
14.     }
15.
16.     @Override
17.     public IToolItem getToolItem(ToolBox parent)
18.     {
19.         return new TextTool(parent);
20.     }
21. }
```

Kode Sumber 4.11 Realisasi *Interface IToolPlugin* untuk *Text Tool*

```
1.  package text;
2.
3.  import org.eclipse.swt.events.MouseEvent;
4.
5.  import widget.tool.toolbox.ATool;
6.  import widget.tool.toolbox.ToolBox;
7.  import window.AddTextWindow;
8.
9.  public class TextTool extends ATool
10. {
11.
12.     public TextTool(ToolBox parent)
13.     {
14.         super(parent);
15.     }
16.
17.     @Override
```

```
18.     public void initialize()
19.     {
20.         setIconName("text-3x.png");
21.         super.initialize();
22.     }
23.
24.     @Override
25.     public void mouseDoubleClick(MouseEvent e) {
26.         // TODO Auto-generated method stub
27.
28.     }
29.
30.     @Override
31.     public void mouseDown(MouseEvent e)
32.     {
33.         System.out.println("koordinat X = " + e.x);

34.         System.out.println("koordinat Y = " + e.y);

35.         AddTextWindow textWindow = new AddTextWindo
    w(e.x, e.y);
36.         textWindow.pack();
37.         textWindow.show();
38.     }
39.
40.     @Override
41.     public void mouseUp(MouseEvent e) {
42.         // TODO Auto-generated method stub
43.     }
44.
45.     @Override
46.     public void mouseMove(MouseEvent e) {
47.         // TODO Auto-generated method stub
48.     }
49. }
```

Kode Sumber 4.12 Implementasi *Text Tool*

4.5.3 Implementasi *Interface IEnhancementPlugin*

Kode Sumber 4.13 menunjukkan implementasi dari *interface IEnhancementPlugin*. *IEnhancementPlugin* meng*extend*

interface IPlugin dan berisi fungsi getEnhancementItem(EditImageWindow editWindow) yang akan mengembalikan *Enhancement* yang akan ditambahkan. Kode Sumber 4.14, Kode Sumber 4.16 dan Kode Sumber 4.18merupakan realisasi *interface IEnhancementPlugin*.

```
1.  package plugins;
2.
3.  import interfaces.IEnhancementOption;
4.  import window.EditImageWindow;
5.
6.  public interface IEnhancementPlugin extends IPlugin

7.  {
8.      IEnhancementOption getEnhacement(EditImageWindo
    w editWindow);
9.  }
```

Kode Sumber 4.13 Implementasi *Interface IEnhancementPlugin*

```
1.  package brightness;
2.
3.  import interfaces.IEnhancementOption;
4.  import plugins.IEnhancementPlugin;
5.  import window.EditImageWindow;
6.
7.  public class BrightnessPlugin implements IEnhanceme
    ntPlugin
8.  {
9.
10.     @Override
11.     public String getName()
12.     {
13.         return "Brightness";
14.     }
15.
16.     @Override
17.     public IEnhancementOption getEnhacement(EditIma
    geWindow editWindow)
18.     {
```

```
19.         return new BrightnessEnhancement(editWindow
    );
20.     }
21.
22. }
```

Kode Sumber 4.14 Realisasi *Interface IEnhancementPlugin* untuk *Brightness Enhancement*

```
1.  package brightness;
2.
3.  import org.eclipse.swt.events.SelectionEvent;
4.  import org.eclipse.swt.graphics.ImageData;
5.  import org.eclipse.swt.widgets.Composite;
6.  import org.opencv.core.Core;
7.  import org.opencv.core.Mat;
8.  import org.opencv.highgui.Highgui;
9.
10. import widget.edit.enhancement.AEnhancementOption;

11. import window.EditImageWindow;
12.
13. public class BrightnessEnhancement extends AEnhance
    mentOption
14. {
15.
16.     public EditImageWindow editWindow;
17.
18.     public BrightnessEnhancement(Composite parent)

19.     {
20.         super(parent);
21.         this.editWindow = (EditImageWindow) parent;
22.     }
23.
24.     @Override
25.     public void initialize()
26.     {
27.       setEnhancementName("Brightness");
28.       this.getButton().addSelectionListener(this);
29.     }
```

```
30.
31.     @Override
32.     public void widgetSelected(SelectionEvent e)
33.     {
34.       double alpha = 1;
35.       double beta = 31;
36.       try
37.       {
38.         System.loadLibrary( Core.NATIVE_LIBRARY_NAM
    E );
39.          Mat source =  Highgui.imread(editWindow.ge
    tImagePath(),Highgui.CV_LOAD_IMAGE_COLOR);
40.          Mat destination = new Mat(source.rows(),so
    urce.cols(),
41.
42.           source.type());
43.           source.convertTo(destination, -
    1, alpha, beta);
44.           Highgui.imwrite("temp/brightness.jpg", de
    stination);
45.
46.           ImageData imgData = new ImageData("temp/b
    rightness.jpg");
47.           editWindow.setImageData(imgData);
48.          editWindow.getImageEditor().reCallPicture(
    imgData);
49.          editWindow.setEffect("brightness");
50.       }catch (Exception error)
51.         {
52.           System.out.println("error: " + error.getM
    essage());
53.           }
54.     }
55.
56.     @Override
57.     public void widgetDefaultSelected(SelectionEven
    t e) {
58.         // TODO Auto-generated method stub
59.     }
60. }
```

Kode Sumber 4.15 Implementasi kelas *BrightnessEnhancement*

Kode Sumber 4.15 merupakan kode untuk memberikan efek *Brightness* pada foto yang dipilih. Kode tersebut akan disalurkan ke aplikasi utama melalui fungsi *getEnhacement()* pada Kode Sumber 4.14.

```
1.  package contrast;
2.
3.  import interfaces.IEnhancementOption;
4.  import plugins.IEnhancementPlugin;
5.  import window.EditImageWindow;
6.
7.  public class Plugin implements IEnhancementPlugin
8.  {
9.      @Override
10.     public String getName()
11.     {
12.         return "Contrast";
13.     }
14.
15.     @Override
16.     public IEnhancementOption getEnhacement(EditIma
    geWindow editWindow)
17.     {
18.        return new ContrastEnhancement(editWindow);
19.     }
20. }
```

Kode Sumber 4.16 Realisasi *Interface IEnhancementPlugin* untuk *Contrast Enhancement*

```
1.  package contrast;
2.
3.  import java.util.ArrayList;
4.  import java.util.List;
5.  import org.eclipse.swt.events.SelectionEvent;
6.  import org.eclipse.swt.graphics.ImageData;
7.  import org.eclipse.swt.widgets.Composite;
8.  import org.opencv.core.Core;
9.  import org.opencv.core.Mat;
```

```
10. import org.opencv.highgui.Highgui;
11. import org.opencv.imgproc.Imgproc;
12.
13. import widget.edit.enhancement.AEnhancementOption;

14. import window.EditImageWindow;
15.
16. public class ContrastEnhancement extends AEnhanceme
    ntOption
17. {
18.     public EditImageWindow editWindow;
19.
20.     public ContrastEnhancement(Composite parent)
21.     {
22.         super(parent);
23.         this.editWindow = (EditImageWindow) parent;
24.     }
25.
26.     @Override
27.     public void initialize()
28.     {
29.       setEnhancementName("Contrast");
30.       this.getButton().addSelectionListener(this);

31.     }
32.
33.     @Override
34.     public void widgetSelected(SelectionEvent e)
35.     {
36.       System.out.println("contrast selected");
37.       System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

38.       Mat source = Highgui.imread(editWindow.getIma
    gePath(), Highgui.CV_LOAD_IMAGE_COLOR);
39.       List<Mat> bgr = new ArrayList<>();
40.       List<Mat> dest = new ArrayList<>();
41.       Mat destination = new Mat(source.rows(), sour
    ce.cols(), source.type());
42.       Core.split(source, bgr);
43.       Core.split(source, dest);
44.      Imgproc.equalizeHist(bgr.get(1), dest.get(1));
         Core.merge(dest, destination);
```

```
45.         Highgui.imwrite("temp/contrast.jpg", destin
    ation);
46.      ImageData imgData = new ImageData("temp/contra
    st.jpg");
47.       editWindow.setImageData(imgData);
48.       editWindow.getImageEditor().reCallPicture(img
    Data);
49.        editWindow.setEffect("contrast");
50.     }
51.     @Override
52.     public void widgetDefaultSelected(SelectionEven
    t e) {
53.        // TODO Autogenerated method stub
54.     }
55. }
```

Kode Sumber 4.17 Implementasi kelas *ContrastEnhancement*

Kode Sumber 4.17 merupakan kode untuk memberikan efek *Contrast* pada foto yang dipilih. Kode tersebut akan disalurkan ke aplikasi utama melalui fungsi *getEnhacement()* pada Kode Sumber 4.16.

```
1.  package sharp;
2.
3.  import interfaces.IEnhancementOption;
4.  import plugins.IEnhancementPlugin;
5.  import window.EditImageWindow;
6.
7.  public class SharpPlugin implements IEnhancementPlu
    gin
8.  {
9.
10.     @Override
11.     public String getName()
12.     {
13.         return "Sharp";
14.     }
15.
16.     @Override
```

```
17.     public IEnhancementOption getEnhacement(EditIma
    geWindow editWindow)
18.     {
19.         return new SharpEnhancement(editWindow);
20.     }
21.
22. }
```

Kode Sumber 4.18 Realisasi *Interface IEnhancementPlugin* untuk *Sharp Enhancement*

```
1.  package sharp;
2.
3.  import org.eclipse.swt.events.SelectionEvent;
4.  import org.eclipse.swt.graphics.ImageData;
5.  import org.eclipse.swt.widgets.Composite;
6.  import org.opencv.core.Core;
7.  import org.opencv.core.Mat;
8.  import org.opencv.core.Size;
9.  import org.opencv.highgui.Highgui;
10. import org.opencv.imgproc.Imgproc;
11.
12. import widget.edit.enhancement.AEnhancementOption;

13. import window.EditImageWindow;
14.
15. public class SharpEnhancement extends AEnhancementO
    ption
16. {
17.
18.     public EditImageWindow editWindow;
19.
20.     public SharpEnhancement(Composite parent)
21.     {
22.         super(parent);
23.         this.editWindow = (EditImageWindow) parent;
24.     }
25.
26.     @Override
27.     public void initialize() {
28.        setEnhancementName("Sharp");
29.        this.getButton().addSelectionListener(this);
```

```
30.     }
31.
32.     @Override
33.     public void widgetSelected(SelectionEvent e)
34.     {
35.         System.loadLibrary( Core.NATIVE_LIBRARY_NA
    ME );
36.         Mat source = Highgui.imread(editWindow.get
    ImagePath(),
37.         Highgui.CV_LOAD_IMAGE_COLOR);
38.         Mat destination = new Mat(source.rows(),so
    urce.cols(),source.type());
39.         Imgproc.GaussianBlur(source, destination,
    new Size(0,0), 10);
40.         Core.addWeighted(source, 1.5, destination,
     -0.5, 0, destination);
41.         Highgui.imwrite("temp/sharp.jpg", destinat
    ion);
42.
43.         ImageData imgData = new ImageData("temp/sh
    arp.jpg");
44.         editWindow.setImageData(imgData);
45.         editWindow.getImageEditor().reCallPicture(
    imgData);
46.         editWindow.setEffect("sharp");
47.     }
48.
49.     @Override
50.     public void widgetDefaultSelected(SelectionEven
    t e) {
51.         // TODO Auto-generated method stub
52.     }
53.
54. }
```

Kode Sumber 4.19 Implementasi kelas *SharpEnhancement*

Kode Sumber 4.19 merupakan kode untuk memberikan efek *Sharp* pada foto yang dipilih. Kode tersebut akan disalurkan ke aplikasi utama melalui fungsi *getEnhacement()* pada Kode Sumber 4.18.

4.6 Implementasi Modularitas

Pada bagian ini implementasi yang dijelaskan meliputi pembuatan modul beserta aturan yang perlu diimplementasi dan pembacaan modul pada perangkat lunak utama.

4.6.1 Pembuatan Modul

Penjelasan cara pembuatan modul meliputi aturan apa saja yang diperlukan sebuah modul agar dapat digabungkan dengan perangkat lunak utama dan langkah-langkah pembuatan modul yang mengikuti aturan-aturan yang telah dijelaskan. Berikut aturan-aturan yang perlu diimplementasikan pada modul.

1. Modul wajib di-*build* sebagai berkas arsip Java (*.jar*).
2. Modul wajib memiliki satu kelas yang mengimplementasikan *IEnhancementPlugin* atau *IToolPlugin*.
3. Modul wajib memiliki satu berkas pada folder META-INF/Services yang berjudul *plugins.IEnhancementPlugin,* atau *IToolPlugin* yang berisi satu baris dengan nama *package* lengkap dengan kelas berkas yang akan diimplementasi.
4. Pada *project* modul *Fotokita.jar* wajib untuk dimasukkan ke *build path*. Tetapi pada saat di-*build*, *FotokitaPlugin.jar* tidak wajib untuk ikut di arsipkan.
5. Penamaan berkas arsip java harus sesuai dengan nama pada fungsi *getName()*.
6. Tidak boleh ada *error* pada *project*.

Adapun langkah-langkah untuk membuat modul adalah:

1. Membuat *project* baru pada *Eclipse*
2. Klik kanan pada *project*, pilih *Build Path -> Configure Build Path*
3. Pada *tab Libraries* menekan tombol *Add External JARs* dan memilih berkas Fotokita.jar

4. Membuat kode, sebagai contoh Kode Sumber 4.11, Kode Sumber 4.12 dan Kode Sumber A.7.8 yang ditempatkan dalam satu atau lebih *package*.
5. Membuat *folder* META-INF/Services dengan nama berkas sesuai dengan modul yang ingin dibuat, *plugins.IEnhancementPlugin* untuk modul efek dan *plugins.IToolPlugin* untuk modul *tool*. Isi berkas tersebut berisi satu baris berisi nama kelas lengkap dengan *package*. Sebagai contoh untuk membuat *text tool* maka isi dari *plugins.IToolPlugin* adalah *text.TextToolPlugin*.
6. Klik kanan pada *project,* pilih *Export,* lalu pilih *JAR File* sebagai *export destination*, lalu tekan tombol *Next*, dan centang *package* maupun berkas yang akan diekspor.
7. Pilih destinasi berkas, lalu klik tombol *Finish*.

4.6.2 Pembacaan Modul

Pembacaan berkas modul dilakukan pada kelas *PluginManager* dalam fungsi getEnhancementPlugins, getToolPlugins dan getFramePlugins yang masing-masing akan mengembalikan list yang berisi modul yang sesuai dengan aturan. Kode Sumber 4.20 menunjukkan *pseudocode* untuk fungsi pembacaan modul. Untuk modul *frame*, daftar *frame* tidak perlu dimuat karena berupa gambar (.png). Kode Sumber A.7.6 dam Kode Sumber A.7.7 pada lampiran merupakan implementasi dari Kode Sumber 4.20.

```
get the folder path
get the files with extension.jar (.png for frame)
convert the files' path into URL
create new URCLassLoader
load the URL
create new Iterator for the loaded URL
create new List for plugins
while iterator has next
  load the iterator's file's property
  if property "enabled" equals "yes"
    add to the List
return List
```

Kode Sumber 4.20 *Pseudocode* fungsi getEnhancementPlugins() dan getToolPlugins()

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan pengujian dan evaluasi dari perangkat lunak ini yang meliputi rincian lingkungan uji coba dan pengujian pada semua fungsionalitas administratif dan semua menu pada modul yang terintegrasi. Aspek yang diperhatikan dalam pengujian perangkat lunak ini adalah terpenuhinya fungsionalitas administratif dan integrasi modul ke dalam perangkat lunak.

5.1 Lingkungan Pelaksanaan Pengujian

Lingkungan pelaksanaan pengujian merupakan perangkat keras dan perangkat lunak yang digunakan selama melakukan pengujian. Pengujian dilakukan dengan menggunakan satu lingkungan pengujian. Rincian lingkungan pengujian ditunjukkan pada Tabel 5.1.

Tabel 5.1 Lingkungan Uji Coba

Spesifikasi	Deskripsi
CPU	Intel ® Core ™ i7-4510U CPU @ 2.00 GHz
RAM	8.00 GB
Sistem Operasi	Windows 8.1 Professional 64 bit

5.2 Dasar Pengujian

Pengujian yang dilakukan berupa pengujian fungsionalitas. Pengujian fungsionalitas dilakukan dengan model *blackbox* untuk menguji proses penambahan modul, proses penghapusan instalasi modul dan proses pengubahan status modul.

5.3 Pengujian Fungsionalitas

Uji coba fungsionalitas administratif adalah uji coba yang dilakukan terhadap fungsionalitas yang dapat dilakukan oleh

pengguna. Fungsionalitas tersebut merupakan kasus penggunaan yang sebelumnya telah dijelaskan pada Subbab 3.2.3. Penjelasan uji coba meliputi scenario uji coba dan hasil uji coba.

Uji coba dilakukan dengan metode *black box* yang artinya fungsionalitas diperiksa apakah terpenuhi atau tidak tanpa melihat struktur internal ataupun metode yang digunakan dalam pengerjaan fungsionalitas tersebut.

5.3.1 Kasus Pengujian Menambah Modul

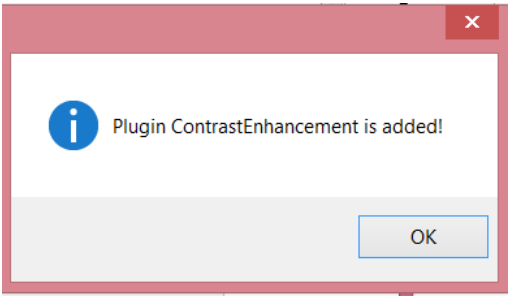
Pada kasus uji ini terhadap tiga skenario yaitu jika modul yang ditambahkan belum ada, jika properti modul yang ingin ditambahkan tidak terpenuhi dan jika modul yang ingin ditambahkan sudah terdaftar. Properti yang dimaksud sudah dijelaskan pada Subbab 3.2.3. Properti tidak memenuhi artinya salah satu atau lebih properti yang diperlukan tidak ada atau kesalahan penamahan pada properti. Jika properti tidak memenuhi sistem akan mengeluarkan pesan *error* seperti pada Gambar 5.2 dan jika semua memenuhi maka sistem akan mengeluarkan pesan seperti pada Gambar 5.1. Gambar 5.3 menunjukkan *edit image window* sebelum modul ditambahkan dan Gambar 5.4 menunjukkan *edit image window* setelah modul ditambahkan. Rincian kasus uji ini ditunjukkan pada Tabel 5.2.

Tabel 5.2 Kasus Uji Menambah Modul

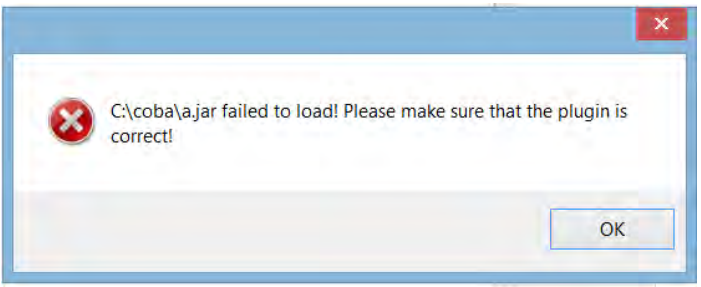
ID	**UJ-001**
Kasus Penggunaan	Menambah modul
Nama	Pengujian menambah modul
Tujuan Pengujian	Menguji apakah modul dapat ditambah ke dalam perangkat lunak
Skenario 1	*Aktor menambahkan modul dengan semua properti yang memenuhi*
Kondisi Awal	Modul belum terdaftar
Data Uji	• Nama modul: *EnhancementPlugin*

	• Nama submodul: *ContrastEnhancementPlugin* • Isi berkas pada META-INF/services/plugins.IEnhancementPlugin: enhancementPlugin.ContrastEnhancementPlugin
Langkah Pengujian	1. Aktor memilih salah satu submenu pada menu *plugins* 2. Aktor memilih berkas modul yang ingin ditambah
Hasil yang Diharapkan	Modul berhasil didaftarkan
Hasil yang Didapat	Modul terdaftar
Skenario 2	*Aktor menambahkan modul dengan properti yang kurang atau salah*
Kondisi Awal	Modul belum terdaftar atau modul sudah terdaftar
Data Uji	• Nama modul: *EnhancementPlugin* • Nama submodul: *ContrastEnhancementPlugin* Isi berkas pada META-INF/services/plugins.IEnhancementPlugin: enhancementPlugin
Langkah Pengujian	1. Aktor memilih salah satu submenu pada menu *plugins* 2. Aktor memilih berkas modul yang ingin ditambah
Hasil yang Diharapkan	Penambahan modul gagal dan mengeluarkan pesan error
Hasil yang Didapat	Penambahan modul gagal dan mengeluarkan pesan error
Skenario 3	*Aktor menambahkan modul yang sudah ada*

Kondisi Awal	Modul sudah terdaftar
Data Uji	• Nama modul: *EnhancementPlugin* • Nama submodul: *ContrastEnhancementPlugin* Isi *file* pada META-INF/services/plugins.IEnhancementPlugin: enhancementPlugin.ContrastEnhancementPlugin
Langkah Pengujian	1. Aktor memilih salah satu submenu pada menu *plugins* 2. Aktor memilih berkas modul yang ingin ditambah
Hasil yang Diharapkan	Penambahan modul berhasil dan modul yang lama diperbarui
Hasil yang Didapat	Penambahan modul berhasil dan modul yang lama diperbarui



Gambar 5.1 Pesan ketika modul berhasil ditambahkan



Gambar 5.2 *Error* jika properti modul tidak lengkap

Gambar 5.3 *Edit image window* sebelum *ContrastEnhancement* ditambahkan

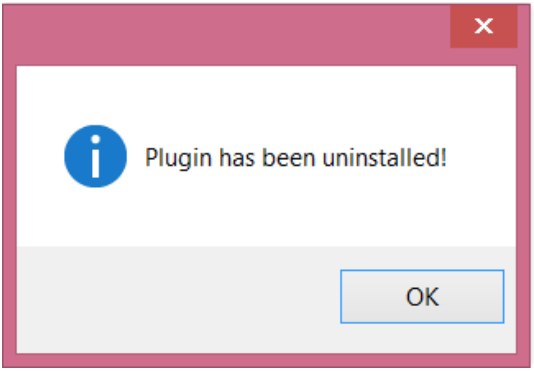
Gambar 5.4 *Edit image window* setelah *ContrastEnhancement* ditambahkan

5.3.2 Kasus Pengujian Menghapus Instalasi Modul

Pada kasus uji ini terdapat satu skenario yaitu menghapus instalasi modul yang sudah terinstal. Jika modul berhasil dihapus siste mengeluarkan kotak pesan seperti pada Gambar 5.5. Rincian kasus uji ini ditunjukkan pada Tabel 5.3.

Tabel 5.3 Kasus Uji Menghapus Instalasi Modul

ID	**UJ-002**
Kasus Penggunaan	Menghapus instalasi modul
Nama	Pengujian menghapus instalasi modul
Tujuan Pengujian	Menguji apakah modul bisa dihapus instalasinya
Skenario 1	*Aktor menghapus instalasi modul yang sudah terinstal*
Kondisi Awal	Modul terdaftar
Data Uji	• Nama modul: *EnhancementPlugin* • Nama submodul: *ContrastEnhancementPlugin*
Langkah Pengujian	1. Aktor memilih modul yang ingin dihapus instalasinya 2. Aktor menekan tombol *uninstall*
Hasil yang Diharapkan	Modul berhasil dihapus instalasinya
Hasil yang Didapat	Modul terhapus



Gambar 5.5 Kotak pesan ketika modul berhasil dihapus instalasi

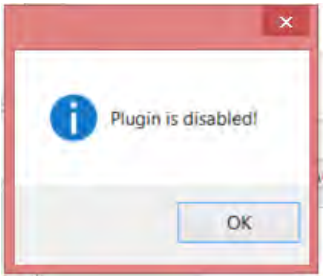
5.3.3 Kasus Pengujian Mengubah Status Modul

Pada kasus uji ini terdapat dua skenario yaitu mengubah status modul menjadi „enabled“ pada modul yang tidak aktif dan mengubah status modul menjadi „disabled“ pada modul yang aktif. Jika modul berhasil dinonaktifkan sistem akan mengeluarkan kotak pesan seperti pada Gambar 5.6 dan pada jendela manajemen modul status akan berubah seperti pada Gambar 5.7. Rincian kasus uji ini ditunjukkan pada Tabel 5.4.

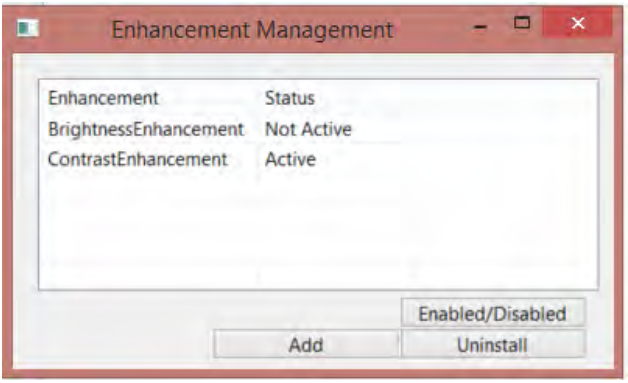
Tabel 5.4 Kasus Uji Mengubah Status Modul

ID	**UJ-003**
Kasus Penggunaan	Mengubah Status Modul
Nama	Pengujian mengubah status modul
Tujuan Pengujian	Menguji apakah modul bisa diubah statusnya
Skenario 1	*Aktor mengubah status modul yang aktif menjadi disabled*
Kondisi Awal	Modul terinstall dan aktif
Data Uji	• Nama modul: *EnhancementPlugin* • Nama submodul: *BrightnessEnhancement*

Langkah Pengujian	1. Aktor memilih modul yang ingin dihapus instalasinya 2. Aktor menekan tombol „*Enabled/Disabled*“
Hasil yang Diharapkan	Modul berhasi di nonaktifkan
Hasil yang Didapat	Modul tidak aktif
Skenario 2	*Aktor mengubah status modul yang tidak aktif menjadi „enabled“*
Kondisi Awal	Modul terinstall tetapi tidak aktif
Data Uji	• Nama modul: *EnhancementPlugin* • Nama submodul: *BrightnessEnhancement*
Langkah Pengujian	1. Aktor memilih modul yang ingin dihapus instalasinya 2. Aktor menekan tombol „*Enabled/Disabled*
Hasil yang Diharapkan	Modul berhasil diaktifkan
Hasil yang Didapat	Modul aktif



Gambar 5.6 Kotak pesan ketika modul dinonaktfikan



Gambar 5.7 Status berubah

5.4 Evaluasi

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 5.5. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik kesimpulan bahwa fungsionalitas dari program telah bisa bekerja sesuai dengan yang diharapkan dan menunjukkan hasil yang benar.

Tabel 5.5 Hasil Pengujian Fungsionalitas Sistem

ID	Nama	Skenario	Hasil
UJ-001	Menambah Modul	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
UJ-002	Menghapus Instalasi Modul	Skenario 1	Berhasil
UJ-003	Mengubah Status Modul	Skenario 1	Berhasil
		Skenario 2	Berhasil

LAMPIRAN A. KODE SUMBER

```
1.  public void widgetSelected(SelectionEvent e)
2.  {
3.    FileDialog fd = new FileDialog(s, SWT.OPEN);
4.    fd.setText("Add Enhancement");
5.    fd.setFilterPath("C:/");
6.    String[] filterExt = { "*.jar"};
7.    fd.setFilterExtensions(filterExt);
8.    String selected = fd.open();
9.
10.   String path = "file:///" + selected;
11.
12.   File sourceFile = new File(selected);
13.   String sourceFileName = sourceFile.getName();
14.   File destFile = new File("Plugins/Enhancement/" + source
    FileName);
15.
16.   try
17.   {
18.     List<IEnhancementPlugin> loadPlugin = PluginManager.ch
    eckPlugin(path);
19.
20.     if (loadPlugin.size()!=0)
21.     {
22.       String destFiles = destFile.toString();
23.       String[] temp = sourceFileName.split(".jar");
24.       String sourceFileNameOnly = temp[0];
25.
26.       try
27.       {
28.        copyJAR(sourceFile, destFile);
29.        try
30.        {
31.         Properties pluginConfig = new Properties();
32.         FileOutputStream output = new FileOutputStream("Pl
    ugins/Enhancement/" + sourceFileNameOnly + ".properties");

33.         pluginConfig.setProperty("pluginName", sourceFileN
    ameOnly);
```

```
34.       pluginConfig.setProperty("pluginPath" , destFiles);

35.       pluginConfig.setProperty("enabled", "yes");
36.       pluginConfig.setProperty("active", "true");
37.       pluginConfig.store(output, null);
38.       output.close();
39.       messageBox.setMessage("Plugin " + sourceFileNameOnly
     + " is added!");
40.       int open = messageBox.open();
41.
42.       switch (open)
43.       {
44.          case SWT.OK:
45.          break;
46.       }
47.      } catch(IOException io)
48.        {
49.          io.printStackTrace();
50.        }
51.
52.      } catch (IOException e1)
53.        {
54.           e1.printStackTrace();
55.        }
56.      }
57.        else
58.        {
59.          int style = SWT.ICON_ERROR;
60.          MessageBox messageBoxError = new MessageBox(s, st
    yle);
61.          messageBoxError.setMessage(selected + " failed to
     load! Please make sure that the plugin is correct!");
62.          int open = messageBoxError.open();
63.
64.         switch (open)
65.         {
66.          case SWT.OK:
67.          System.out.println("SWT.OK");
68.          break;
69.         }
70.        }
71.      } catch (MalformedURLException e2)
72.     {
```

```
73.       e2.printStackTrace();
74.     } catch (ServiceConfigurationError e2)
75.      {
76.        int style = SWT.ICON_ERROR;
77.        MessageBox messageBoxError = new MessageBox(s, styl
    e);
78.        messageBoxError.setMessage(selected + " failed to l
    oad! Please make sure that the plugin is correct!");
79.        int open = messageBoxError.open();
80.
81.        switch (open)
82.       {
83.         case SWT.OK:
84.         System.out.println("SWT.OK");
85.         break;
86.       }
87.     }
88.  }
```

Kode Sumber A.7.1 Implementasi Penambahan Modul *Enhancement*

```
1.  public void widgetSelected(SelectionEvent e)
2.  {
3.    FileDialog fd = new FileDialog(s, SWT.OPEN);
4.    fd.setText("Add Tools");
5.    fd.setFilterPath("C:/");
6.    String[] filterExt = { "*.jar"};
7.    fd.setFilterExtensions(filterExt);
8.    String selected = fd.open();
9.
10.   String path = "file:///" + selected;
11.
12.   File sourceFile = new File(selected);
13.   String sourceFileName = sourceFile.getName();
14.   File destFile = new File("Plugins/Tools/" + sourceFileNa
    me);
15.
16.   try
17.   {
```

```
18.     List<IEnhancementPlugin> loadPlugin = PluginManager.ch
    eckPlugin(path);
19.
20.     if (loadPlugin.size()!=0)
21.     {
22.       String destFiles = destFile.toString();
23.       String[] temp = sourceFileName.split(".jar");
24.       String sourceFileNameOnly = temp[0];
25.
26.       try
27.       {
28.        copyJAR(sourceFile, destFile);
29.        try
30.        {
31.         Properties pluginConfig = new Properties();
32.         FileOutputStream output = new FileOutputStream("Pl
    ugins/Tools/" + sourceFileNameOnly + ".properties");
33.         pluginConfig.setProperty("pluginName", sourceFileN
    ameOnly);
34.       pluginConfig.setProperty("pluginPath" , destFiles);

35.       pluginConfig.setProperty("enabled", "yes");
36.       pluginConfig.setProperty("active", "true");
37.       pluginConfig.store(output, null);
38.       output.close();
39.       messageBox.setMessage("Plugin " + sourceFileNameOnly
     + " is added!");
40.       int open = messageBox.open();
41.
42.       switch (open)
43.       {
44.          case SWT.OK:
45.          break;
46.       }
47.      } catch(IOException io)
48.        {
49.          io.printStackTrace();
50.        }
51.
52.      } catch (IOException e1)
53.        {
54.           e1.printStackTrace();
55.        }
```

```
56.       }
57.         else
58.         {
59.           int style = SWT.ICON_ERROR;
60.           MessageBox messageBoxError = new MessageBox(s, st
    yle);
61.           messageBoxError.setMessage(selected + " failed to
     load! Please make sure that the plugin is correct!");
62.           int open = messageBoxError.open();
63.
64.          switch (open)
65.          {
66.           case SWT.OK:
67.           System.out.println("SWT.OK");
68.           break;
69.          }
70.         }
71.       } catch (MalformedURLException e2)
72.      {
73.        e2.printStackTrace();
74.      } catch (ServiceConfigurationError e2)
75.       {
76.         int style = SWT.ICON_ERROR;
77.         MessageBox messageBoxError = new MessageBox(s, styl
    e);
78.         messageBoxError.setMessage(selected + " failed to l
    oad! Please make sure that the plugin is correct!");
79.         int open = messageBoxError.open();
80.
81.         switch (open)
82.        {
83.          case SWT.OK:
84.          System.out.println("SWT.OK");
85.          break;
86.        }
87.      }
88.   }
```

Kode Sumber A.7.2 Implementasi Penambahan Modul Tools

```
1.  public void widgetSelected(SelectionEvent e)
```

```
2.  {
3.    FileDialog fd = new FileDialog(s, SWT.OPEN);
4.    fd.setText("Add Enhancement");
5.    fd.setFilterPath("C:/");
6.    String[] filterExt = { "*.jar"};
7.    fd.setFilterExtensions(filterExt);
8.    String selected = fd.open();
9.
10.   String path = "file:///" + selected;
11.
12.   File sourceFile = new File(selected);
13.   String sourceFileName = sourceFile.getName();
14.   File destFile = new File("Plugins/Frame/" + sourceFileNa
    me);
15.
16.   try
17.   {
18.     List<IEnhancementPlugin> loadPlugin = PluginManager.ch
    eckPlugin(path);
19.
20.     if (loadPlugin.size()!=0)
21.     {
22.       String destFiles = destFile.toString();
23.       String[] temp = sourceFileName.split(".jar");
24.       String sourceFileNameOnly = temp[0];
25.
26.       try
27.       {
28.        copyJAR(sourceFile, destFile);
29.        try
30.        {
31.         Properties pluginConfig = new Properties();
32.         FileOutputStream output = new FileOutputStream("Pl
    ugins/Frame/" + sourceFileNameOnly + ".properties");
33.         pluginConfig.setProperty("pluginName", sourceFileN
    ameOnly);
34.       pluginConfig.setProperty("pluginPath" , destFiles);

35.       pluginConfig.setProperty("enabled", "yes");
36.       pluginConfig.setProperty("active", "true");
37.       pluginConfig.store(output, null);
38.       output.close();
```

```
39.       messageBox.setMessage("Plugin " + sourceFileNameOnly
     + " is added!");
40.       int open = messageBox.open();
41.
42.       switch (open)
43.       {
44.          case SWT.OK:
45.          break;
46.       }
47.      } catch(IOException io)
48.        {
49.          io.printStackTrace();
50.        }
51.
52.      } catch (IOException e1)
53.        {
54.           e1.printStackTrace();
55.        }
56.      }
57.        else
58.        {
59.          int style = SWT.ICON_ERROR;
60.          MessageBox messageBoxError = new MessageBox(s, st
    yle);
61.          messageBoxError.setMessage(selected + " failed to
     load! Please make sure that the plugin is correct!");
62.          int open = messageBoxError.open();
63.
64.         switch (open)
65.         {
66.          case SWT.OK:
67.          System.out.println("SWT.OK");
68.          break;
69.         }
70.        }
71.      } catch (MalformedURLException e2)
72.     {
73.       e2.printStackTrace();
74.     } catch (ServiceConfigurationError e2)
75.      {
76.        int style = SWT.ICON_ERROR;
77.        MessageBox messageBoxError = new MessageBox(s, styl
    e);
```

```
78.         messageBoxError.setMessage(selected + " failed to l
    oad! Please make sure that the plugin is correct!");
79.         int open = messageBoxError.open();
80.
81.         switch (open)
82.        {
83.          case SWT.OK:
84.          System.out.println("SWT.OK");
85.          break;
86.        }
87.      }
88.   }
```

Kode Sumber A.7.3 Implementasi Penambahan Modul Frame

```
1.   public void widgetSelected(SelectionEvent e)
2.   {
3.     int selected = table.getSelectionIndex();
4.     String propName = propList[selected].toString();
5.     FileInputStream in;
6.     try
7.     {
8.       in = new FileInputStream(propName);
9.       loadProp.load(in);
10.      in.close();
11.
12.      FileOutputStream out = new FileOutputStream(propName);

13.      loadProp.setProperty("enabled", "no");
14.      loadProp.store(out, null);
15.      out.close();
16.
17.      messageBox.setMessage("Plugin has been disabled!");
18.      int open = messageBox.open();
19.
20.      switch (open)
21.      {
22.        case SWT.OK:
23.        System.out.println("SWT.OK");
24.        break;
25.       }
```

```
26.  } catch (FileNotFoundException e1)
27.    {
28.       e1.printStackTrace();
29.    } catch (IOException e1)
30.    {
31.       e1.printStackTrace();
32.    }
33.  }
```

Kode Sumber A.7.4 Implementasi Kasus Penggunaan Menghapus Instalasi Modul

```
1.   public void widgetSelected(SelectionEvent e)
2.  {
3.     int selected = table.getSelectionIndex();
4.     String propName = propList[selected].toString();
5.     FileInputStream in;
6.     try {
7.       in = new FileInputStream(propName);
8.       loadProp.load(in);
9.        in.close();
10.       if(loadProp.getProperty("active").equals("true"))
11.       {
12.         loadProp.setProperty("active", "false");
13.       }
14.       else
15.       {
16.         loadProp.setProperty("active", "true");
17.       }
18.      FileOutputStream out = new FileOutputStream(propName)
    ;
19.      loadProp.store(out, null);
20.      out.close();
21.    } catch (FileNotFoundException e1) {
22.        e1.printStackTrace();
23.      } catch (IOException e1) {
24.        e1.printStackTrace();
25.      }  }
```

Kode Sumber A.7.5 Implementasi Kasus Penggunaan Mengubah Status Modul

```
1.  public static List<IEnhancementPlugin> getEnhancementPlugi
    ns()
2.  {
3.    File pluginLoc = new File("Plugins/Enhancement");
4.    Properties loadProp = new Properties();
5.
6.    File[] pluginList = pluginLoc.listFiles(new FileFilter()

7.    {
8.      @Override
9.      public boolean accept(File pathname)
10.     {
11.       return pathname.getPath().toLowerCase().endsWith(".j
    ar");
12.     }
13.   });
14.
15.   URL[] urls = new URL[pluginList.length];
16.
17.   for (int i = 0; i < pluginList.length; i++)
18.   {
19.     try
20.     {
21.       urls[i] = pluginList[i].toURI().toURL();
22.     }
23.     catch (MalformedURLException e)
24.     {
25.       e.printStackTrace();
26.     }
27.   }
28.
29.   URLClassLoader ucl = new URLClassLoader(urls);
30.
31.   ServiceLoader<IEnhancementPlugin> plugin = ServiceLoad
    er.load(IEnhancementPlugin.class, ucl);
32.
33.   Iterator<IEnhancementPlugin> it = plugin.iterator();
34.
35.   List<IEnhancementPlugin> plugins = new ArrayList<>();
36.
37.   while(it.hasNext())
38.   {
39.   IEnhancementPlugin p = it.next();
```

```
40.     FileInputStream in;
41.     try
42.     {
43.       in = new FileInputStream("Plugins/Enhancement/" + p.
    getName()+ ".properties");
44.       loadProp.load(in);
45.       in.close();
46.       if(loadProp.getProperty("enabled").equals("yes"))
47.       {
48.         plugins.add(p);
49.       }
50.      }
51.      catch (FileNotFoundException e)
52.      {
53.         e.printStackTrace();
54.      }
55.      catch (IOException e)
56.      {
57.         e.printStackTrace();
58.      }
59.     }
60.
61.   return plugins;
62. }
```

Kode Sumber A.7.6 Implementasi Pembacaan Modul untuk Modul Efek

```
1.  public static List<IToolPlugin> getToolPlugins()
2.  {
3.    File pluginLoc = new File("Plugins/Tool");
4.    Properties loadProp = new Properties();
5.
6.    File[] pluginList = pluginLoc.listFiles(new FileFilter()

7.    {
8.      @Override
9.      public boolean accept(File pathname)
10.     {
11.       return pathname.getPath().toLowerCase().endsWith(".j
    ar");
12.     }
```

```
13.     });
14.
15.     URL[] urls = new URL[pluginList.length];
16.
17.     for (int i = 0; i < pluginList.length; i++)
18.     {
19.       try
20.       {
21.         urls[i] = pluginList[i].toURI().toURL();
22.       }
23.       catch (MalformedURLException e)
24.       {
25.         e.printStackTrace();
26.       }
27.      }
28.
29.      URLClassLoader ucl = new URLClassLoader(urls);
30.
31.      ServiceLoader<IToolPlugin> plugin = ServiceLoader.load
    (IToolPlugin.class, ucl);
32.
33.      Iterator<IToolPlugin> it = plugin.iterator();
34.
35.     List<IToolPlugin> plugins = new ArrayList<>();
36.
37.     while(it.hasNext())
38.     {
39.      IToolPlugin p = it.next();
40.      FileInputStream in;
41.      try
42.      {
43.        in = new FileInputStream("Plugins/Tool/" + p.getName
    ()+ ".properties");
44.        loadProp.load(in);
45.        in.close();
46.        if(loadProp.getProperty("enabled").equals("yes"))
47.        {
48.          plugins.add(p);
49.        }
50.       }
51.       catch (FileNotFoundException e)
52.       {
53.          e.printStackTrace();
```

```
54.       }
55.       catch (IOException e)
56.       {
57.          e.printStackTrace();
58.       }
59.      }
60.
61.    return plugins;
62. }
```

Kode Sumber A.7.7 Implementasi Pembacaan Modul untuk Modul *Tool*

```
1.  package text;
2.
3.  import org.eclipse.swt.SWT;
4.  import org.eclipse.swt.custom.StyledText;
5.  import org.eclipse.swt.events.SelectionEvent;
6.  import org.eclipse.swt.events.SelectionListener;
7.  import org.eclipse.swt.graphics.Font;
8.  import org.eclipse.swt.layout.GridData;
9.  import org.eclipse.swt.layout.GridLayout;
10. import org.eclipse.swt.widgets.Button;
11.
12. import element.Paragraph;
13. import interfaces.IColorPicker;
14. import interfaces.IFontSelector;
15. import interfaces.ISizeSelector;
16. import widget.edit.ColorPicker;
17. import widget.texting.FontSelector;
18. import widget.texting.SizeSelector;
19. import window.AWindow;
20. import window.MainWindow;
21.
22. public class AddTextWindow extends AWindow implements Sele
    ctionListener
23. {
24.
25.     private StyledText textArea;
26.     int x;
27.     int y;
28.     private Button addButton;
```

```
29.     private IFontSelector fontSelector;
30.     private ISizeSelector sizeSelector;
31.     private IColorPicker colorPicker;
32.
33.     public AddTextWindow(int x, int y)
34.     {
35.         this.x = x;
36.         this.y = y;
37.     }
38.
39.     @Override
40.     public void initialize()
41.     {
42.         this.setSize(400, 200);
43.         this.setText("Add Text");
44.
45.         GridLayout layout = new GridLayout();
46.         layout.numColumns = 4;
47.         this.setLayout(layout);
48.
49.         GridData styledText = new GridData();
50.         styledText.grabExcessHorizontalSpace = true;
51.         styledText.grabExcessVerticalSpace = true;
52.         styledText.horizontalAlignment = SWT.FILL;
53.         styledText.verticalAlignment = SWT.FILL;
54.         styledText.horizontalSpan = 4;
55.         textArea = new StyledText(this, SWT.MULTI | SWT.WR
    AP | SWT.BORDER | SWT.H_SCROLL | SWT.V_SCROLL);
56.         textArea.setLayoutData(styledText);
57.
58.         setFontSelector(new FontSelector(this, SWT.READ_ON
    LY, getTextArea()));
59.         setSizeSelector(new SizeSelector(this, SWT.READ_ON
    LY, getTextArea(), getFontSelector()));
60.         getFontSelector().setSizeSelector(sizeSelector);
61.         setColorPicker(new ColorPicker(this));
62.
63.         addButton = new Button(this, SWT.PUSH);
64.         GridData button = new GridData();
65.         button.horizontalAlignment = SWT.END;
66.         button.horizontalSpan = 2;
67.         addButton.setText("Add");
68.         addButton.setLayoutData(button);
```

```
69.         getAddButton().addSelectionListener(this);
70.     }
71.
72.     public StyledText getTextArea()
73.     {
74.         return textArea;
75.     }
76.
77.     public void setTextArea(StyledText textArea)
78.     {
79.         this.textArea = textArea;
80.     }
81.
82.     public Button getAddButton()
83.     {
84.         return addButton;
85.     }
86.
87.     public void setAddButton(Button addButton)
88.     {
89.         this.addButton = addButton;
90.     }
91.
92.     @Override
93.     public void widgetSelected(SelectionEvent e)
94.     {
95.         System.out.println(getTextArea().getText());
96.         Font font = new Font(this.getDisplay(), getFontSel
    ector().getNameSelected(getFontSelector().getIndexSelected
    ()), Integer.parseInt(getSizeSelector().getNameSelected(ge
    tSizeSelector().getIndexSelected())), SWT.NORMAL);
97.         Paragraph paragraph = new Paragraph(x,y,100,50, ge
    tTextArea().getText(), font, getColorPicker().getColor());

98.         MainWindow.getInstance().getInitTab().getActivePag
    e().addElement(paragraph);
99.         MainWindow.getInstance().getInitTab().getActivePag
    e().draw();
100.        System.out.println(paragraph.getColor().toString()
    );
101.        this.dispose();
102.    }
103.
```

```
104.    @Override
105.    public void widgetDefaultSelected(SelectionEvent e) {}

106.
107.    public IFontSelector getFontSelector()
108.    {
109.        return fontSelector;
110.    }
111.
112.    public void setFontSelector(IFontSelector fontSelector
    )
113.    {
114.        this.fontSelector = fontSelector;
115.    }
116.
117.    public ISizeSelector getSizeSelector()
118.    {
119.        return sizeSelector;
120.    }
121.
122.    public void setSizeSelector(ISizeSelector sizeSelector
    )
123.    {
124.        this.sizeSelector = sizeSelector;
125.    }
126.
127.    public IColorPicker getColorPicker()
128.    {
129.        return colorPicker;
130.    }
131.
132.    public void setColorPicker(IColorPicker colorPicker)
133.    {
134.        this.colorPicker = colorPicker;
135.    }}
```

Kode Sumber A.7.8 *Text Tool Editor Window*

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

Pengerjaan arsitektur modular album foto digital Fotokita ini diawali dengan membuat rancangan aplikasi album foto digital, dan memecah rancangan tersebut menjadi unsur-unsur. Setelah itu kesamaan beberapa unsur yang sejenis dicari dan dibuat menjadi kelas *interface*. Dari kelas-kelas *interface* tersebut, aturan modularitas dibuat dan setiap modul yang akan dimasukkan ke dalam aplikasi Fotokita harus mengimplementasikan aturan modularitas tersebut.

Pengujian fungsionalitas dilakukan dengan menggunakan metode *blackbox*. Kasus pengujian meliputi pengujian menambah modul, pengujian menghapus instalasi modul dan pengujian mengubah status modul.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Album foto digital Fotokita dibangun dengan menggunakan pemrograman berorientasi objek sehingga dapat dipecah menjadi elemen-elemen kecil sehingga bisa dibangun secara modular.
2. Modul yang dibangun harus memenuhi beberapa aturan tertentu agar bisa berfungsi pada sistem.
3. Daur modul pada Fotokita adalah *installed*, *enabled, disabled* dan *uninstalled*.

6.2 Saran

Berikut ini merupakan beberapa saran mengenai pengembangan lebih lanjut album foto digital Fotokita modular berdasarkan hasil rancangan, implementasi dan uji coba yang telah dilakukan.

1. Antarmuka pengguna Fotokita masih sangat sederhana sehingga diperlukan pengembangan lebih lanjut.
2. Optimasi logika untuk memuat *plugins*.

DAFTAR PUSTAKA

[1] A. Rouf, "PENGUJIAN PERANGKAT LUNAK DENGAN MENGGUNAKAN METODEWHITE BOX DAN BLACK BOX," *STIMIK HIMSYA JOURNAL,* vol. VIII, no. 1, pp. 3-6, 2012.

[2] K. Knoernschil, Java Application Architechture, Modularity Patterns with Examples Using OSGi, Indiana: Prentice Hall, 2013.

[3] "IBM Developer Works," IBM, [Online]. Available: http://www.ibm.com/developerworks/library/j-5things12/. [Diakses 31 May 2016].

[4] "Class ServiceLoader<S>," Oracle, [Online]. Available: http://docs.oracle.com/javase/6/docs/api/java/util/ServiceLoader.html. [Diakses 31 May 2016].

[5] "Wikipedia - Standard Widget Toolkit," [Online]. Available: https://en.wikipedia.org/wiki/Standard_Widget_Toolkit. [Diakses 8 June 2016].

[6] "Properties (Java Platform)," Oracle, [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/util/Properties.html. [Diakses 8 June 2016].

[7] "Java - The Properties Class," [Online]. Available: http://www.tutorialspoint.com/java/java_properties_class.htm. [Diakses 8 June 2016].

[8] M. A. Ridwan, Rancang Bangun Aplikasi Editor Album Foto Digital 'Fotokita' Berbasis Desktop, Surabaya: Institut Teknologi Sepuluh Nopember (ITS) Surabaya, 2016.

[9] F. Merdianto, Rancang Bangun Aplikasi Album Foto Digital 'Fotokita' dengan Penyimpanan Dinamis, Surabaya: Institut Teknologi Sepuluh Nopember (ITS) Surabaya, 2016.

BIODATA PENULIS

Penulis, A. Heynoum Dala Rifat lahir di Kota Makassar pada 2 Juli 1994 dan dibesarkan di Kota Makassar. Penulis adalah anak pertama dari tiga bersaudara. Penulis menempuh pendidikan formal di SD Islam Athirah Bukit Baruga (2000-2005), SMP Dian Harapan Makassar (2005-2008), SMA Negeri 2 Tinggimoncong (2008-2012) dan S1 Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya (2012-2016). Selain itu, penulis juga berkesempatan untuk melakukan pertukaran pelajar di Joseph-König-Gymnasium (2010-2011) dan Chung-Ang University (2015).

Selama perkuliahan, penulis aktif dalam beberapa kegiatan diantaranya yaitu pengurus Shorinji Kempo ITS, pengurus Ikatan Alumni SMA Negeri 2 Tinggimoncong, pengurus Anti-Corruption International chapter Indonesia.

Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Rekayasa Perangkat Lunak (RPL) dan menjadi administrator di Laboratorium Rekayasa Perangkat Lunak dengan ketertarikan penulis terdapat pada analisis perancangan sistem dan konstruksi perangkat lunak. Penulis dapat dihubungi melalui alamat surel **dalarifat@yahoo.com**