BAB III
PELAKSANAAN KERJA MAGANG

A. Kedudukan dan Koordinasi

Dalam praktik kerja magang di PT Multipolar Technology Tbk, penulis berperan sebagai *smart card programmer* di bagian *Electronic Channel (E-Channel) Switching*. Praktik kerja magang ini dikoordinasi oleh Bapak Harda Elnanda selaku *E-Channel Switching Manager* PT Multipolar Technology Tbk. Posisi beliau sebagai *E-Channel Switching Manager* dapat dilihat pada Gambar 2.2.

B. Tugas yang Dilakukan

Selama dua bulan menjalani praktik kerja magang, terdapat beberapa tugas yang harus diselesaikan yaitu:

1. Membuat dan mengetes *applet* pada *Java Card* yang berfungsi untuk *read and write*.
2. Membuat aplikasi form Java yang terkoneksi dengan *smart card*.

Setiap hari penulis memberikan laporan kepada Pembimbing Lapangan sejauh mana *progress* dari tugas tersebut.

C. Uraian Pelaksanaan Kerja Magang

Pelaksanaan kerja magang diuraikan menjadi tiga bagian, yaitu proses pelaksanaan, kendala yang ditemukan, dan solusi atas kendala yang ditemukan.

1. Proses Pelaksanaan

Pada saat melaksanakan kerja magang ada beberapa hal yang dilakukan, mulai dari mempelajari konsep dasar *smart card* seperti *Application Protocol Data Unit* (APDU), mempelajari konsep *Java Card* seperti *package application identifier* (AID) dan *applet* AID, mempelajari penggunaan fitur-fitur dari Cyberflex Access *Software Development Kit* (SDK) seperti *Toolkit* dan *Program File Generator*, serta mempelajari struktur program *Java Card*. Setelah mengetahui bagaimana program

yang ada berjalan, penulis membuat perancangan program dan mengimplementasikannya dalam kode.

a. Konsep Dasar *Smart Card*

Smart card adalah sebuah kartu yang ter-*embed* dengan mikroprosesor dan *chip* memori atau hanya *chip* memori dengan *non-programmable logic*. Kartu mikroprosesor dapat menambah, menghapus, dan memanipulasi informasi dalam kartu, sementara kartu *chip* memori hanya bisa melakukan operasi yang telah ditentukan.

Smart card dapat membawa semua fungsi dan informasi yang diperlukan pada kartu sehingga tidak memerlukan akses ke *remote database* pada waktu transaksi.

Pada saat ini terdapat tiga kategori *smart card* yang berkembang dengan pesat yaitu:

- *Integrated Circuit (IC) Microprocessor Card* atau biasa disebut dengan *chip card*, menawarkan penyimpanan memori yang lebih besar dan keamanan data daripada *magnetic stripe card* tradisional. *Chip card* juga dapat memproses data di dalam kartu. Contoh: kartu kredit.
- *Integrated Circuit (IC) Memory Card*, dapat menampung 1-4 KB data, tetapi tidak mempunyai prosesor di dalam kartu untuk memanipulasi data. Oleh karena itu, *memory card* bergantung pada *card reader* atau *card accepting device* (CAD) untuk *processing* dan cocok untuk *fixed operation*. Contoh: kartu prabayar pada ponsel.
- *Optical Memory Card*, dapat menyimpan maksimal 4 MB data. Sekali di-*write*, data tidak bisa diubah atau dihapus. Jenis kartu ini ideal untuk menyimpan *record* seperti *medical files, driving records,* atau *travel histories*.

b. Konsep Dasar Teknologi Java Card

Java Card adalah sebuah teknologi perangkat lunak yang membuat aplikasi berbasis Java (*applet*) dapat berjalan secara aman pada *smart card*. *Smart card* sangat bermanfaat untuk keamanan karena dapat digunakan untuk menambah autentikasi dan akses keamanan ke sistem informasi yang memerlukan keamanan tingkat tinggi. Informasi yang tersimpan dalam *smart card* bersifat *portable*. Dengan teknologi *Java Card* kita bisa membawa informasi personal yang berharga dan sensitif seperti riwayat medis, nomor kartu kredit, atau *electronic cash balances* di dalam sebuah medium yang padat, tetapi sangat aman.

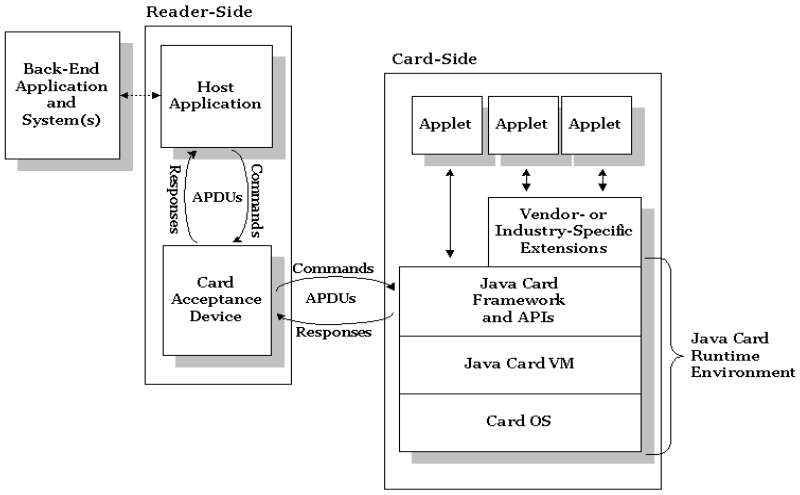
Java Card adalah *platform* Java terkecil yang ditargetkan untuk perangkat *embedded*. *Java Card* memberikan pengguna kemampuan untuk memprogram

perangkat dan membuatnya spesifik terhadap suatu aplikasi (*application specific*). Hal ini banyak digunakan pada kartu SIM (digunakan pada ponsel GSM) dan kartu ATM. *Java Card* pertama kali diperkenalkan pada tahun 1996 oleh divisi kartu Schlumberger yang kemudian bergabung dengan Gemplus untuk membentuk Gemalto. Produk *Java Card* didasarkan pada spesifikasi *Java Card platform* yang dikembangkan oleh Sun Microsystems (anak perusahaan dari Oracle Corporation). Banyak produk *Java Card* yang juga bergantung pada spesifikasi GlobalPlatform untuk pengelolaan aplikasi yang aman pada kartu (*download, installation, personalization, deletion*). Tujuan utama dari perancangan teknologi *Java Card* adalah portabilitas dan keamanan.

1) Elemen-Elemen Aplikasi *Java Card*

Sebuah aplikasi *Java Card* yang komplit terdiri dari *back-end application and systems, host (off-card) application, interface device (card reader),* dan *on-card applet, user credentials,* dan *software* yang mendukung. Semua elemen ini bersama-sama menyusun sebuah aplikasi *end-to-end* yang aman.

Sebuah aplikasi *Java Card* tipikalnya tidak berdiri sendiri. *Java Card* memiliki arsitektur yang terdiri dari *card-side, reader-side,* dan *back-end*, yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Arsitektur Aplikasi Java Card [1]

- ***Back-End Application and Systems***

  Aplikasi *back-end* memberikan layanan yang mendukung *in-card* Java *applet*. Sebagai contoh, sebuah aplikasi *back-end* dapat memberikan konektivitas kepada sistem keamanan yang menyediakan keamanan kuat, bersama dengan *in-card credentials*. Di dalam sebuah sistem pembayaran elektronik, aplikasi *back-end* dapat memberikan akses ke kartu kredit dan informasi pembayaran lainnya.

- ***Reader-Side Host Application***

  Aplikasi *host* berada pada *desktop* atau terminal seperti PC, terminal pembayaran elektronik, ponsel, atau *security subsystem*. Aplikasi *host* menangani komunikasi antara pengguna, *Java Card applet*, dan aplikasi *back-end* dari *provider*.

  Secara tradisional, aplikasi *reader-side* telah ditulis dalam bahasa C. Adopsi terbaru teknologi J2ME memungkinkan untuk mewujudkan aplikasi *host* dalam bahasa Java, misalnya dapat berjalan pada ponsel yang *support* MIDP dan *Security and Trust Services API*. *Smart card* vendor biasanya tidak hanya menyediakan sebuah *development kit*, tetapi juga API untuk mendukung aplikasi *reader-side* maupun *Java Card applet*. Contohnya adalah *OpenCard Framework* dan *Schlumberger Interoperability Java Native Interface* (IOP JNI).

- ***Reader-Side Card Acceptance Device***

  Card Acceptance Device (CAD) adalah perangkat antarmuka yang berada di antara aplikasi *host* dan perangkat *Java Card*. CAD menyediakan daya ke kartu. CAD dapat berupa *card reader* yang terpasang ke komputer *desktop*, atau dapat diintegrasikan ke dalam terminal seperti terminal pembayaran elektronik di restoran atau stasiun pengisian bahan bakar umum (SPBU). CAD meneruskan *command* APDU dari aplikasi *host* ke kartu, dan meneruskan *response* APDU dari kartu ke aplikasi *host*. Beberapa CAD memiliki *keyboard* untuk memasukkan PIN dan juga layar.

- ***Card-Side Applets and Environment***

  Java Card platform adalah sebuah *multiple-application environment*. Seperti yang diilustrasikan pada Gambar 3.1, satu atau lebih *Java Card*

applet dapat berada pada satu kartu bersama dengan *software* pendukung seperti sistem operasi kartu dan *Java Card Runtime Environment* (JCRE). JCRE terdiri dari *Java Card Virtual Machine* (JCVM), *Java Card Framework* dan API, dan beberapa *extension* API.

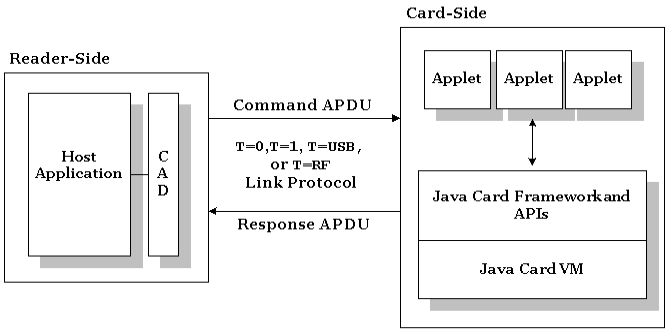
Semua *Java Card applet* meng-*extend applet base class* dan harus mengimplementasikan *install()* dan *process() method*. JCRE memanggil *install() method* ketika menginstal *applet* dan memanggil *process() method* setiap kali ada APDU masuk untuk *applet*.

Java Card applet diinstansiasi ketika di-*load* dan tetap hidup ketika listrik dimatikan. *Card applet* bertindak sebagai server dan pasif. Setelah kartu diberikan *power*, setiap *applet* tetap non-aktif sampai *applet* tersebut dipilih (*selected*). *Applet* dapat dipilih pada saat proses inisialisasi selesai dilakukan. *Applet* hanya aktif ketika APDU telah dikirim. Bagaimana *applet* menjadi aktif (*selected*) dijelaskan di bagian "*Life Cycle* dari *Java Card Applet*".

2) Berkomunikasi dengan *Java Card Applet* (Mengakses *Smart Card*) Menggunakan *Application Protocol Data Unit* (APDU)

Komunikasi antara aplikasi *host* dengan *Java Card applet* dapat menggunakan dua macam model komunikasi, yaitu *fundamental message-passing model* dan *Java Card Remote Method Invocation* (JCRMI). *Message-passing model* adalah basis untuk semua komunikasi *Java Card*. Pusatnya adalah *Application Protocol Data Unit* (APDU), suatu unit komunikasi berupa paket data logikal yang saling bertukar di antara *Card Acceptance Device* (CAD) dan *Java Card Framework*. *Java Card Framework* menerima *command* APDU yang dikirim dari CAD dan meneruskannya ke *applet* yang tepat. *Applet* memproses *command* APDU dan mengembalikan *response* APDU. Struktur APDU telah ditetapkan sesuai dengan standar internasional ISO/IEC 7816-3 dan 7816-4.

Komunikasi antara *reader* dengan *card* biasanya berdasarkan pada dua protokol, yaitu *byte-oriented* T=0 atau *block-oriented* T=1. Proses komunikasi dengan *Java Card applet* menggunakan APDU dapat dilihat pada Gambar 3.2.

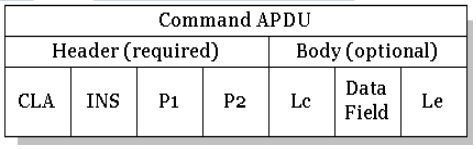


Gambar 3.2 Komunikasi dengan Java Card Applet Menggunakan APDU [1]

Terdapat dua jenis APDU yaitu *command* APDU dan *response* APDU.

- ***Command* APDU**

  Command APDU dikirim dari *reader* ke *smart card*. Struktur *command* APDU dapat dilihat pada gambar 3.3.



Gambar 3.3 Struktur Command APDU [1]

Command APDU mempunyai *required header* dan *optional body* yang terdiri dari:

- **CLA** (panjang 1 *byte*): untuk mengidentifikasi *class of instruction*. Berikut ini adalah nilai valid CLA yang ditentukan oleh ISO 7816-4:

CLA Value	***Instruction Class***
0x0n, 0x1n	*ISO 7816-4 card instructions, such as for file access and security operations*
0x20 to 0x7F	*Reserved*
0x8n or 0x9n	*ISO/IEC 7816-4 format you can use for your application-specific instructions, interpreting 'X' according to the standard*
0xAn	*Application- or vendor-specific instructions*
0xB0 to 0xCF	*ISO/IEC 7816-4 format you can use for application-specific instructions*

0xD0 to 0xFE	*Application- or vendor-specific instructions*
0xFF	*Reserved for protocol type selection*

Tabel 3.1 Nilai Valid CLA Sesuai ISO 7816-4

- **INS** (panjang 1 *byte*): untuk menentukan *instruction code* yang spesifik. Berikut ini adalah nilai valid INS yang ditentukan oleh ISO 7816-4:

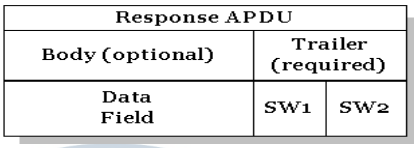
INS Value (Hexa)	***Command Description***
0E	*Erase Binary*
20	*Verify*
70	*Manage Channel*
82	*External Authenticate*
84	*Get Challenge*
88	*Internal Authenticate*
A4	*Select File*
B0	*Read Binary*
B2	*Read Record(s)*
C0	*Get Response*
C2	*Envelope*
CA	*Get Data*
D0	*Write Binary*
D2	*Write Record*
D6	*Update Binary*
DA	*Put Data*
DC	*Update Record*
E2	*Append Record*

Tabel 3.2 Nilai Valid INS Sesuai ISO 7816-4

- **P1** (panjang 1 *byte*): *instruction parameter 1*.
- **P2** (panjang 1 *byte*): *instruction parameter 2.*
- **Lc** (panjang 1 *byte*): jumlah *byte* dari data yang ada di dalam *data field command* APDU.
- ***Data field*** (panjang maksimal 256 *byte*): berisi data yang akan dikirim.
- **Le** (panjang 1 *byte*): jumlah *byte* maksimum dari respon data yang diharapkan.

➢ ***Response* APDU**

Response APDU dikirim dari *card* ke *reader*. Struktur dari *response* APDU dapat dilihat pada Gambar 3.4

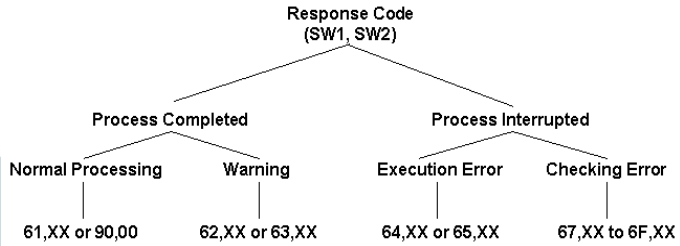


Gambar 3.4 Struktur Response APDU [1]

Sama seperti *command* APDU, *response* APDU juga mempunyai *optional* dan *required field* yang terdiri dari:

- ***Data field***: berisi respon data yang di-*return* oleh *applet* (panjang respon data ditentukan oleh Le yang ada di dalam *command* APDU).
- **SW1** (panjang 1 *byte*): *Status Word* 1.
- **SW2** (panjang 1 *byte*): *Status Word* 2.

Nilai-nilai dari *status word* ditentukan oleh ISO 7816-4 seperti pada Gambar 3.5.



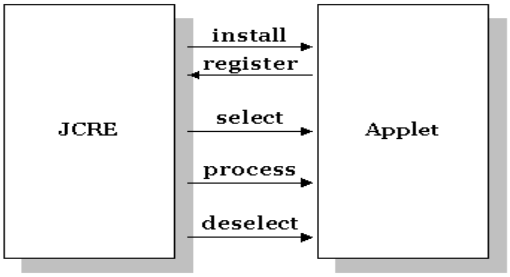
Gambar 3.5 Response Status Codes [1]

Setiap kali ada APDU masuk untuk *applet* yang terpilih (*selected*), JCRE memanggil *process() method* dari *applet*, mem-*passing* APDU yang masuk sebagai sebuah argumen. *Applet* harus mem-*parsing command* APDU, memproses data, menghasilkan *response* APDU, dan mengembalikan kontrol kepada JCRE.

3) *Life-Cycle* dari *Java Card Applet*

Setiap *applet* dalam kartu secara unik diidentifikasi oleh sebuah *Application Identifier* (AID). AID adalah sebuah urutan *bytes* dengan panjang antara 5-16 *bytes* yang telah ditentukan dalam ISO 7816-5.

Gambar 3.6 merangkum operasi dari *applet life-cycle method*. *Life-cycle* dari *applet* dimulai ketika applet di-*download* ke dalam kartu. JCRE memanggil *install() method* dan *applet* merespon dengan mengirim *register() method*. Setelah terinstal dan terdaftar (*installed and registered*), *applet* berada dalam kondisi awal *unselected state* (tidak terpilih).

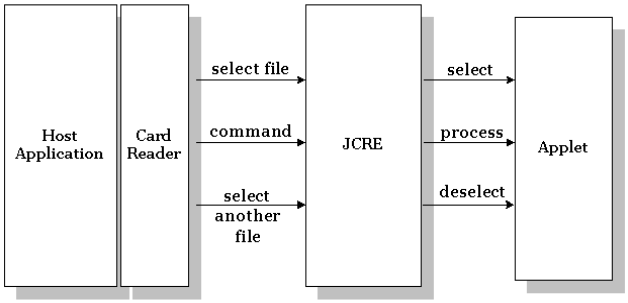


Gambar 3.6 Java Card Applet Life-Cycle Methods [1]

Ketika dalam keadaan tidak terpilih (*unselected*), *applet* tidak aktif. Sebuah *applet* akan dipilih (*selected*) untuk pengolahan APDU ketika aplikasi *host* meminta JCRE untuk memilih *applet* tertentu dalam kartu (dengan menginstruksikan *card reader* untuk mengirim *Select APDU* atau *Manage Channel APDU*). Untuk memberi tahu *applet* bahwa aplikasi *host* telah memilihnya, JCRE memanggil *select() method*. Biasanya *applet* melakukan inisialisasi yang tepat dalam persiapan untuk pengolahan APDU.

Setelah *selection* dilakukan, JCRE melewati perintah APDU yang masuk ke *applet* untuk diproses dengan memanggil *process() method*. JCRE menangkap setiap *exceptions* dimana *applet* gagal untuk menangkap (*catch*).

Applet deselection terjadi ketika aplikasi *host* memberi tahu JCRE untuk memilih *applet* lain. JCRE memberi tahu *applet* yang sedang aktif bahwa dia tidak lagi dipilih (*deselected*) dengan cara memanggil *deselect() method*, yang biasanya melakukan *clean-up logic* dan mengembalikan *applet* ke keadaan tidak aktif dan tidak terpilih (*inactive, unselected*). Proses lengkap dapat dilihat pada Gambar 3.7 di bawah.



Gambar 3.7 Menggunakan Java Card Applet Method [1]

4) *Application Identifier (AID)*

AID adalah sebuah nomor yang unik untuk menandakan sebuah aplikasi atau *applet* pada *smart card* dengan panjang antara 5-16 *bytes*. Penamaan AID sesuai dengan spesifikasi *smart card* yang telah ditentukan dalam ISO/IEC 7816-5. Format AID dapat dilihat pada Gambar 3.8.

Application identifier (AID)	
National registered application provider (RID)	Proprietary application identifier extension (PIX)
5 bytes	0 to 11 bytes

Gambar 3.8 Format AID [7]

AID terdiri dari dua elemen data yaitu *Registered Application Identifier* (RID) dengan panjang tetap 5 *bytes*, dan *Proprietary Application Identifier Extension* (PIX) dengan panjang antara 0-11 *bytes*. RID diberikan oleh badan registrasi nasional atau internasional sehingga dapat terdaftar secara nasional atau internasional. RID berisi kode negara, kategori aplikasi, dan nomor untuk penyedia aplikasi (*application provider*). Kode numerik ini menyebabkan hanya satu RID yang dialokasikan di dunia untuk mengidentifikasi aplikasi tertentu. Jika diperlukan penyedia aplikasi dapat menambah kode PIX, misalnya nomor serial dan versi untuk mengelola aplikasi yang digunakan.

Di dalam teknologi *Java Card*, setiap *applet* dikenali dan dipilih dengan AID yang unik, begitu pula dengan setiap *Java package*. Hal ini karena ketika sebuah *package* dimasukkan ke dalam kartu, maka *package* tersebut akan terhubung dengan *package-package* lain yang telah dimasukkan ke dalam kartu. *Package* AID dan *applet* AID harus mempunyai nilai RID yang sama dan nilai PIX yang berbeda (5 *byte* pertama harus sama, sisa *byte* berikutnya berbeda). Sebagai

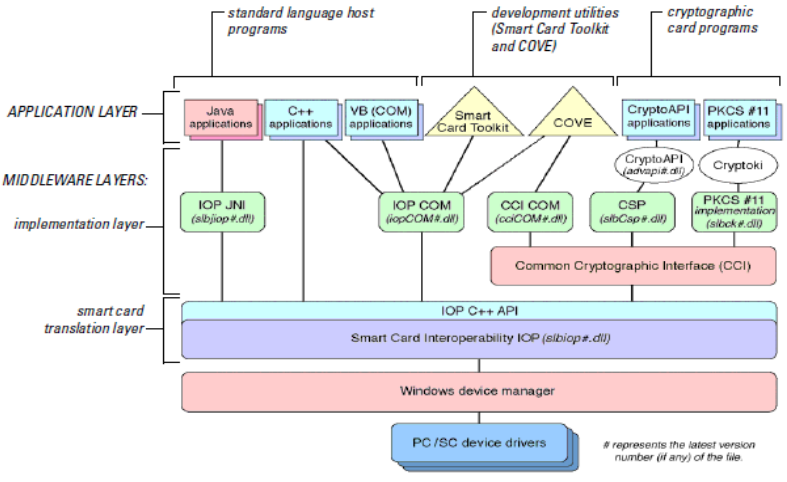
contoh: *package* AID = 11121314151617 (7 *byte*), *applet* AID = 111213141516 (6 *byte*). Keduanya memiliki nilai RID yang sama yaitu 1112131415 (5 *byte*).

c. Cyberflex Access *Software Development Kit* 4.5

Cyberflex Access *Software Development Kit* (SDK) memungkinkan *smart card developer* untuk membangun aplikasi PC berbasis *smart card* untuk *smart card* multi fungsi dari Schlumberger. Cyberflex Access SDK mempunyai *middleware implementation libraries* dan *suite of tools* yang menawarkan dukungan untuk:

- Mengkonfigurasi kartu untuk layanan kriptografi.
- Mengembangkan dan memuat Java *applet* secara dinamis.
- Mengintegrasikan *custom PC applications*.
- Menggunakan *smart card-enabled security applications* (*digital signature generation and verification, authentication, encryption/decryption*).

Kartu jenis Cyberflex Access 64K v2 merupakan salah satu alat yang digunakan untuk mengerjakan proyek kerja magang di PT Multipolar Technology Tbk. Kartu ini mempunyai kapasitas EEPROM sampai dengan 64 KB untuk menyimpan *applet* dan data. Cyberflex Access men-*support card program* (*applet*) yang ditulis sesuai dengan *Java Card* versi 2.1.1. Gambar 3.9 menunjukkan infrastruktur dan implementasi dari perangkat lunak Cyberflex Access SDK, dimana terdapat *interface layers* yang men-*support host card programs*.



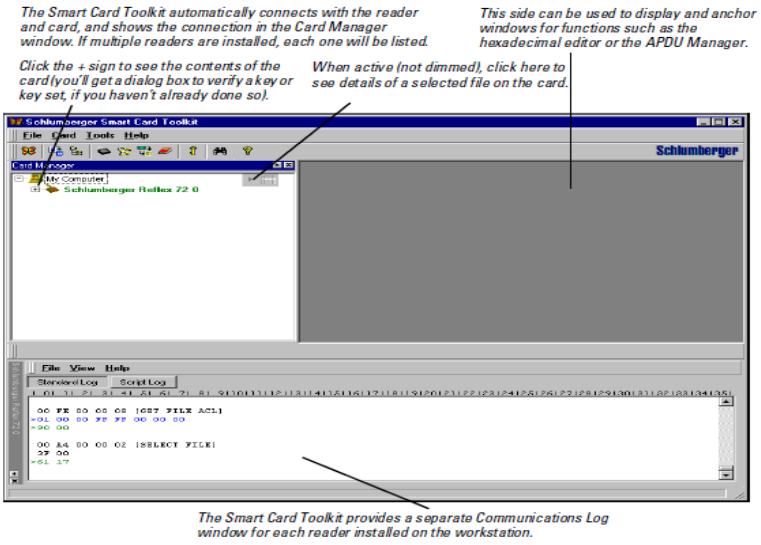
Gambar 3.9 Infrastruktur Perangkat Lunak Cyberflex Access SDK [2]

Cyberflex Access SDK men-*support* berbagai bahasa pemrograman dan standar untuk mengembangkan program *smart card*, yaitu:

- C dan C++
- Java
- COM (Component Object Model/seperti Visual Basic)
- PKCS #11
- Microsoft PC/SC

Untuk menggunakan Cyberflex Access SDK, *software* ini harus diinstal dulu di komputer. Setelah diinstal, kita dapat menggunakan berbagai konten yang disediakan di dalamnya yaitu:

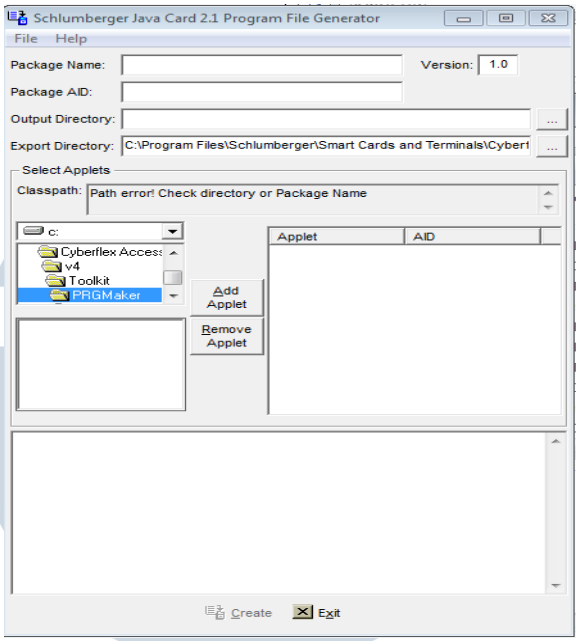
- ***Smart Card Toolkit Application***: seperangkat *tools* untuk membantu kita mempelajari kartu yang dipakai, memasukkan *applet* ke dalam kartu, mengetes program kartu, dan mengelola data yang ada di dalam kartu. Tampilan dari *Toolkit Application* dapat dilihat pada Gambar 3.10. Di dalam aplikasi *toolkit* ini terdapat beberapa fitur yang membantu kita untuk mengelola *applet* dan data yang ada di dalam kartu seperti *Card Manager, Make Card Applet, APDU Manager, Key Manager, Communications Log*, dan sebagainya.



Gambar 3.10 Smart Card Toolkit Application dari Cyberflex Access SDK [2]

- ***Program File Generator and MakeSolo utilities***: *Tools* untuk mengkonversi *Java Card class file* (*.class extension*) menjadi bentuk yang dipahami oleh Cyberflex

Access *card*, yaitu *file* dengan *.ijc extension*. Tampilan dari *Program File Generator* dapat dilihat pada Gambar 3.11.



Gambar 3.11 Program File Generator dari Cyberflex Access SDK

- ***Cyberflex Access Java library*:** *Java Card* 2.1.1 *class file* yang berinteraksi dengan sistem operasi kartu. *Library* ini mendukung program *Java Card* yang dikembangkan untuk Cyberflex Access *Java-enabled smart cards*.
- ***COVE application***: *Crptographic Object Viewer and Editor*, aplikasi untuk personalisasi kartu, menambah elemen yang dibutuhkan untuk melakukan operasi kriptografi pada kartu.
- ***C++ and COM libraries***: Implementasi Cyberflex Access SDK untuk bahasa C++ dan COM *environment*.
- ***Cyberflex Access SDK CSP***: *Cryptographic Service Provider* yang mendukung program kartu yang dikembangkan sesuai dengan Microsoft CryptoAPI versi 2.
- ***Cyberflex Access SDK PKCS #11 library***: *interface* yang mendukung program *smart card* yang sesuai dengan PKCS #11 versi 2.0.1.
- ***Reader and e-gate Drivers***: *Driver* yang men-*support Reflex Readers* dan *e-gate USB smart card*.
- ***Samples*:** Contoh-contoh *card program* yang dapat digunakan sebagai dasar untuk mulai menulis program *Java Card*.

- ***Documentation*:** Buku manual dalam format *PDF online*.

d. *Pengerjaan Proyek Magang*

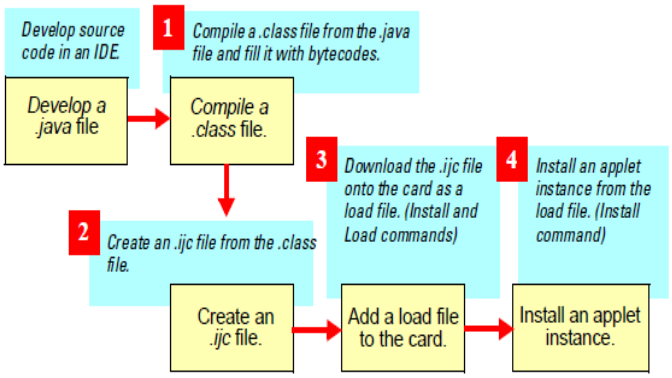
Ada dua proyek yang dikerjakan selama menjalani praktik kerja magang di PT Multipolar Technology Tbk, yaitu:

- Membuat *read and write string applet* pada *Java Card*.
- Membuat aplikasi form Java yang terkoneksi dengan *smart card*. Ada 2 aplikasi form yang dibuat:
  - Form *read and write string*.
  - Form penjumlahan 2 angka.

1) Proyek 1: *Read and Write String Applet* pada *Java Card* (*ReadWrite.java*)

Dalam proyek 1 ini aplikasi yang dibuat adalah sebuah *applet* yang berfungsi untuk menulis *string* ke dalam kartu (*write*) dan membacanya kembali (*read*) dengan menggunakan *Java Card platform*. Program *applet* ini diberi nama *ReadWrite.java*. Program ini dibuat dengan menggunakan Eclipse IDE versi 3.7.2 untuk Windows 7 32 bit dan *Java Card Development Kit* (JCDK) versi 2.2.2. Terdapat empat langkah yang harus dilakukan untuk menyelesaikan proyek 1 ini, seperti yang ditunjukkan dalam Gambar 3.12, yaitu:

1. Membuat *source code* (*.java*) dengan Eclipse IDE, kemudian di-*compile* menjadi *.class file*.
2. Menggunakan *Program File Generator* (komponen dari Cyberflex Access SDK) untuk mengkonversi *.class file* menjadi *.ijc file*.
3. *Download .ijc file* ke dalam kartu sebagai sebuah *program file* menggunakan *Toolkit Application* dari Cyberflex Access SDK, kemudian tentukan *package* AID-nya (*Package* AID = 11121314151617).
4. *Install applet instance* dari *program file* menggunakan *Toolkit Application*, kemudian tentukan *applet* AID (*Applet* AID = 111213141516).



Gambar 3.12 Langkah-Langkah Membuat Java Card Applet [3]

Setelah diinstal ke dalam kartu, *applet* dites menggunakan *APDU Manager*, salah satu fitur dari *Toolkit Application*. Untuk mengetahui apakah program *Java Card applet* ini berjalan dengan benar, *applet* diuji dengan mengirimkan *command* APDU dan *response* APDU. Proses pengujian dapat dilihat pada Gambar 3.13, 3.14, dan 3.15.

Berikut ini adalah format *command* APDU untuk operasi *Write String*:

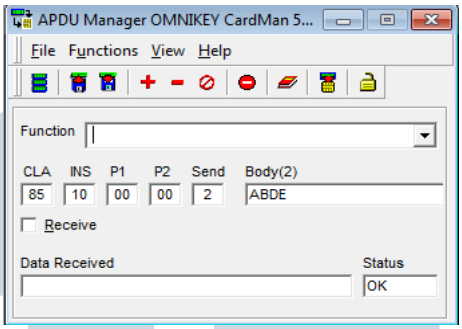
- CLA = 0x85
- INS = 0x10
- P1 = 0
- P2 = 0
- Lc = tergantung isi dari *data field* (berapa *byte string* yang dikirim)
- *Data field* = *string* yang mau dikirim
- Le = tidak diisi

Format *command* APDU untuk operasi *Read String*:

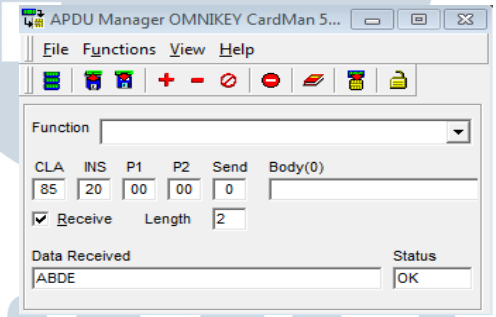
- CLA = 0x85
- INS = 0x20
- P1 = 0
- P2 = 0
- Lc = 0
- *Data field* = tidak diisi
- Le = diisi sesuai keinginan berapa *byte string* yang mau ditampilkan

Jika operasi berhasil dilakukan, maka *applet* akan mengirimkan *Response* APDU berupa:

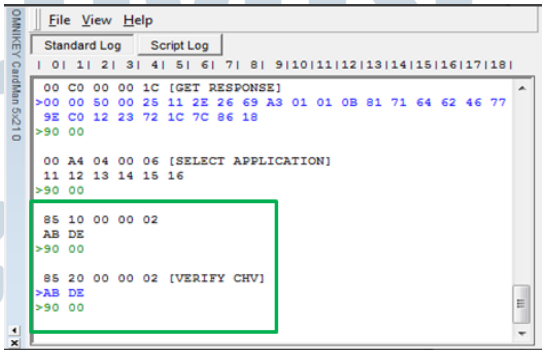
- *Data field*: menampilkan *string* yang dikirim (panjangnya disesuaikan dengan Le yang ada di dalam *command* APDU *Read String*).
- *Status word*: 90 00 (*Command successfully executed/OK*)



Gambar 3.13 Tampilan APDU Manager untuk Operasi Write String



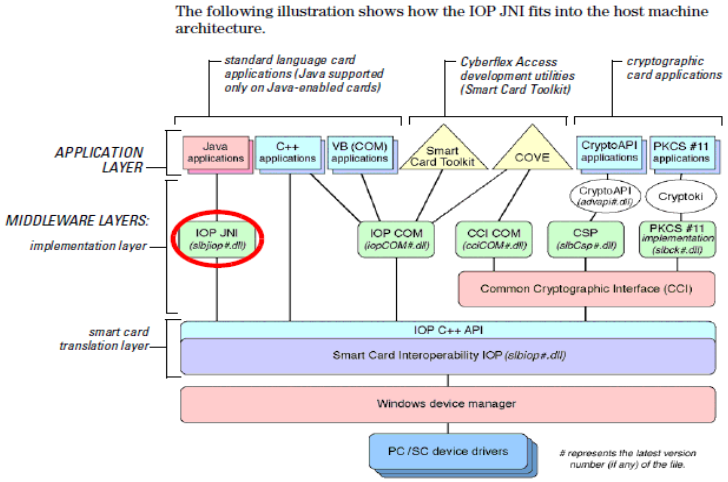
Gambar 3.14 Tampilan APDU Manager untuk Operasi Read String



Gambar 3.15 Tampilan Communication Log

2) Proyek 2: Aplikasi Form Java yang Terkoneksi dengan *Smart Card*

Dalam proyek 2 ini aplikasi yang dibuat adalah aplikasi form Java yang terkoneksi dengan *smart card*. Supaya bisa terkoneksi dengan *smart card* dan melakukan operasi di dalamnya, harus ada dua program aplikasi yang dibuat: program *Java Card applet* yang dimasukkan ke dalam kartu, serta aplikasi *host* berupa program *Java client* dengan tampilan form untuk *user interface*. Untuk menghubungkan antara aplikasi *Java Card*, *card reader*, dan aplikasi *host client*, sebuah *middleware* disediakan oleh Cyberflex Access SDK ketika pertama kali diinstal, yaitu *Interoperability Java Native Interface* (IOP JNI). Lokasi dari IOP JNI *middleware interfaces* dapat dilihat pada Gambar 3.16. Di dalam *middleware* ini terdapat *library*, *package*, kelas, dan metode yang dapat diimplementasikan untuk men-*support* aplikasi Java pada Schlumberger *smart card*. slbJIop.jar adalah IOP JNI *library* yang harus di-*add* ke dalam *Java Build Path Libraries* dari Eclipse IDE. Setelah *library* tersebut berhasil di-*add*, maka slb.iop *package* dapat diimpor ke dalam program sehingga kelas dan metode dari IOP JNI dapat diimplementasikan di dalam program *Java client*.



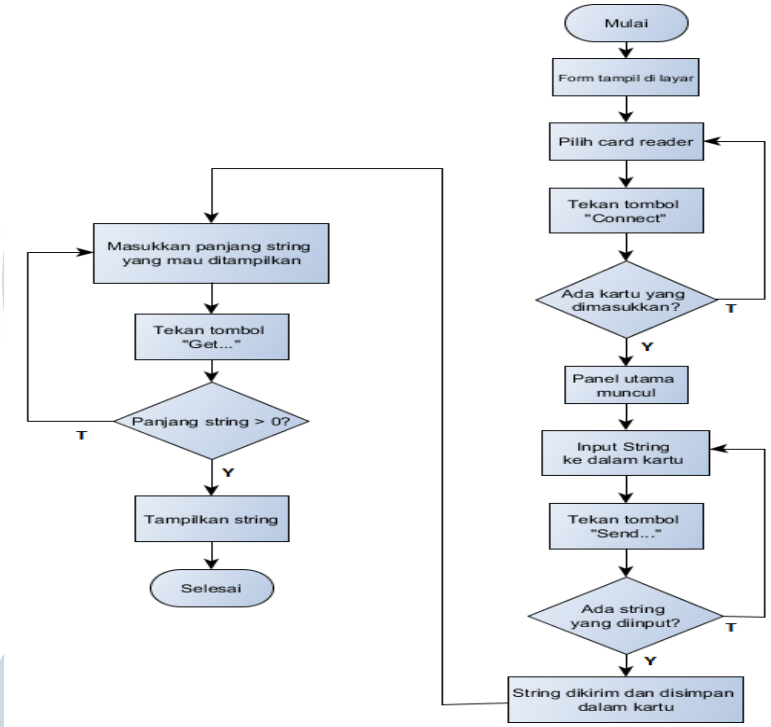
Gambar 3.16 Lokasi dari IOP JNI middleware interface [6]

Terdapat dua aplikasi *Java client* yang dibuat dalam bentuk form, yaitu: form *read and write string* serta form penjumlahan dua angka.

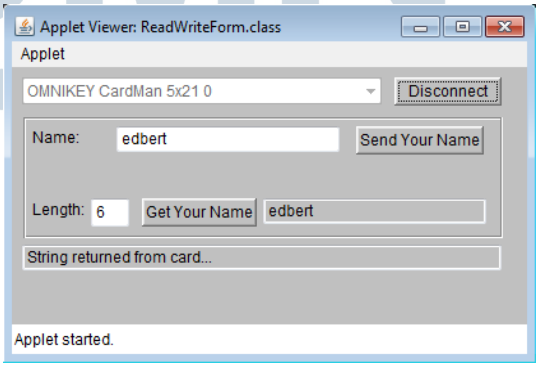
- **Aplikasi form *read and write string* (*ReadWriteForm.java*)**

  Program aplikasi ini diberi nama *ReadWriteForm.java*. Aplikasi ini berfungsi untuk menulis *string* ke dalam *card* (*write*) dan membacanya

kembali (*read*). Supaya aplikasi ini dapat beroperasi dengan *smart card*, aplikasi *Java Card applet* dari proyek 1 (*ReadWrite.java*) dimasukkan ke dalam kartu terlebih dahulu. Desain dari program aplikasi *ReadWriteForm* ditunjukkan melalui *flow chart* pada Gambar 3.17, sementara tampilannya dapat dilihat pada Gambar 3.18.



Gambar 3.17 Flow chart aplikasi form read and write string



Gambar 3.18 Tampilan dari aplikasi form read and write string

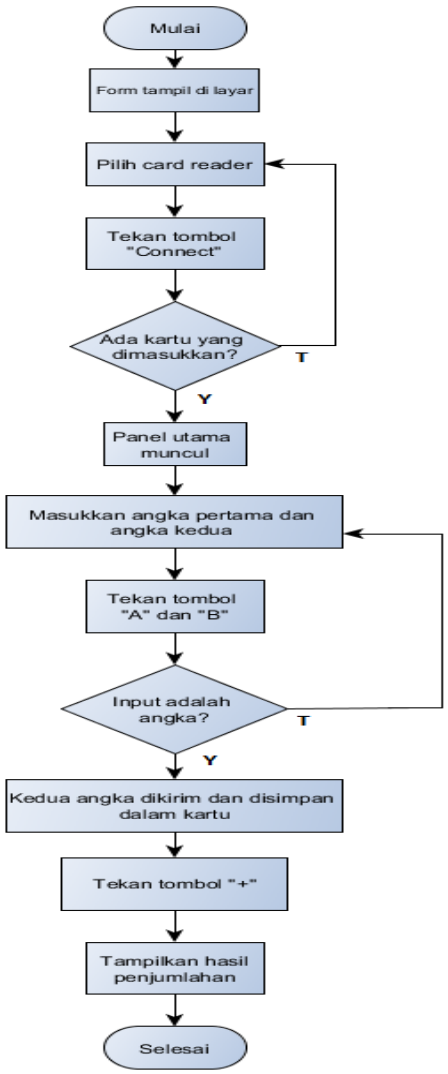
- **Aplikasi form penjumlahan dua angka (*SimpleMathForm.java*)**

  Program aplikasi ini diberi nama *SimpleMathForm.java*. Aplikasi ini berfungsi untuk meng-*input* dua angka ke dalam kartu dan mencetak hasil penjumlahannya. Sebelum aplikasi *SimpleMathForm* dapat berjalan, aplikasi *Java Card applet* yang bernama *SimpleMath.java* harus dibuat dan dimasukkan ke dalam kartu terlebih dahulu untuk melakukan operasi penjumlahan di dalam kartu. Langkah-langkah pembuatan *SimpleMath Java Card applet* sama seperti pada proyek 1 (*ReadWrite applet*), yang membedakan hanya *package* AID, *applet* AID, serta format *command* APDU.

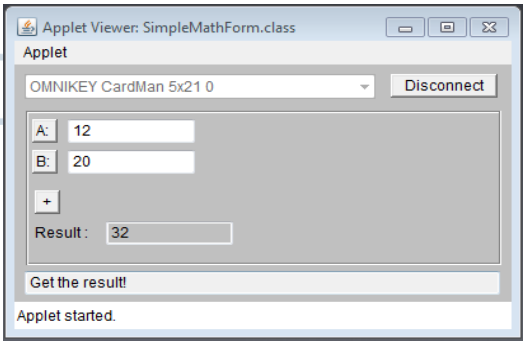
Package AID	F234123456100000
Applet AID	F234123456100001

Tabel 3.3 Package AID dan Applet AID dari SimpleMath Javacard Applet

Melalui aplikasi ini jika kita bisa menjumlahkan dua angka, maka proses penghitungan dan pencatatan debit, kredit, atau saldo bisa dilakukan untuk transaksi keuangan. Desain dari program aplikasi *SimpleMathForm* ditunjukkan melalui *flow chart* pada Gambar 3.19, sementara tampilannya dapat dilihat pada Gambar 3.20.



Gambar 3.19 Flow chart aplikasi form penjumlahan dua angka



Gambar 3.20 Tampilan dari aplikasi form penjumlahan dua angka

2. Kendala yang Ditemukan

Terdapat beberapa kendala yang ditemukan selama praktik kerja magang, di antaranya:

- Belum familiar dengan bahasa pemrograman Java dan *Java Card*.
- Pada saat proses pembuatan *Java Card applet*, konversi dari *.class file* menjadi *.ijc file* menggunakan *Program File Generator* sering *error* (hanya kompatibel dengan JRE versi 1.3 dan *Java Card* versi 2.1 atau 2.2)
- Ada salah satu metode dari slb.iop *package* yang *error*, yaitu *method SelectAID()*.

3. Solusi untuk Kendala yang Ditemukan

Dari beberapa kendala yang didapatkan selama menjalani kerja magang, penulis mendapatkan solusi berupa:

- Mempelajari lebih dalam mengenai konsep dan bahasa pemrograman Java dan *Java Card* dari berbagai sumber di internet.
- Supaya tidak terjadi *error* ketika men-*generate ijc file*, di dalam Eclipse IDE *compiler compliance level*-nya diatur menjadi 1.3 (di dalam Eclipse IDE masuk ke menu *Project* → *Properties* → *Java Compiler* → atur *Compiler compliance level*-nya menjadi 1.3 → *Apply* → *OK*).
- Mengakali metode dari slb.iop *package* yang *error* (*method SelectAID*), dengan cara mencoba metode yang lain, yaitu *method SendCardAPDU()*.