BEHAVIOUR DAN KOMUNIKASI ANTAR SESAMA AGEN PADA SISTEM MANAJEMEN KELAS BERBASIS MULTI-AGEN DENGAN MENGGUNAKAN JADE

SKRIPSI

OLEH:

MARTIUS

04 04 03 0598

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

JUNI 2008

BEHAVIOUR DAN KOMUNIKASI ANTAR SESAMA AGEN PADA SISTEM MANAJEMEN KELAS BERBASIS MULTI-AGEN DENGAN MENGGUNAKAN JADE

SKRIPSI

OLEH:

MARTIUS

04 04 03 0598

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN UNTUK MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

JUNI 2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

***BEHAVIOUR* DAN KOMUNIKASI ANTAR SESAMA AGEN PADA SISTEM MANAJEMEN KELAS BERBASIS MULTI-AGEN DENGAN MENGGUNAKAN JADE**

yang dibuat untuk melengkapi sebagian persyaratan menjadi sarjana teknik pada Program Studi Teknik Komputer Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 25 Juni 2008

Martius

NPM 04 04 03 0598

PENGESAHAN

Skripsi dengan judul:

***BEHAVIOUR* DAN KOMUNIKASI ANTAR SESAMA AGEN PADA SISTEM MANAJEMEN KELAS BERBASIS MULTI-AGEN DENGAN MENGGUNAKAN JADE**

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Komputer Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam sidang ujian skripsi.

Depok, 25 Juni 2008

Dosen Pembimbing,

F Astha Ekadiyanto ST, M.Sc.

NIP 132 166 489

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada :

F. Astha Ekadiyanto ST, M.Sc.

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Disamping itu, penulis tidak lupa pula mengucapkan terima kasih kepada:

Prof. Dr.-Ing. Dr. h.c (UKM) Axel Hunger

yang telah memberikan ijin kepada penulis untuk mengerjakan skripsi ini di Laboratorium Multimedia, Mercator Office, Universitas Indonesia.

Martius

NPM 04 04 03 0598

Departemen Teknik Elektro

Dosen Pembimbing

F. Astha Ekadiyanto ST, M.Sc.

BEHAVIOUR DAN KOMUNIKASI ANTAR SESAMA AGEN PADA SISTEM MANAJEMEN KELAS BERBASIS MULTI-AGEN DENGAN MENGGUNAKAN JADE

ABSTRAK

Proaktif, otonom, dan mampu berkomunikasi adalah hal yang mengkategorikan sekelompok komponen untuk menjadi agen. Ia mempunyai inisiatif untuk mengerjakan tugas yang diberikan ke padanya, dengan atau tanpa ransangan dari pengguna, mandiri dalam mengerjakan tugas, dan mampu berkomunikasi dengan agen-agen yang lain dalam mencapai tujuan.

JADE adalah *platform* perangkat lunak yang menyediakan berbagai fungsionalitas layanan untuk Pemograman Berorientasi Agen. JADE juga mempunyai dokumentasi yang baik dan telah dikembangkan sekitar sepuluh tahun. Ketidak terikatannya kepada *platform* tertentu dan kemampuannya dalam menyederhanakan aplikasi terdistribusi, memberikan JADE masa depan yang lebih cerah sebagai *middleware* aplikasi multi agen.

Dalam Sistem Manajemen Kelas berbasis agen, setiap kelas dikelola oleh satu agen dan di setiap departemen tedapat satu agen yang menjembatani pengguna (dosen) dengan sistem untuk melakukan polling data absensi, pembatalan kegiatan perkuliahan, dan pencarian kelas kosong. Agen Server berperan sebagai perantara antara sistem dan database. Semua agen bersifat independen dalam melakukan tugasnya. Selain itu, agen bersifat proaktif dan mampu berkomunikasi dengan agen yang lain untuk mencapai tujuan. Dengan kemampuan yang dimiliki setiap agen (diimplementasikan dengan *behaviour*), tanggung jawab server untuk melayani seluruh *platform* dapat dikurangi. Sebagai aplikasi yang berjalan independen, agen harus mampu bekerja pada sumber daya komputasi yang minim. Selain itu, lamanya waktu yang dibutuhkan agen untuk berkomunikasi atau menemukan agen yang lain, juga harus diperhatikan agar sistem ini dapat bekerja dengan optimal. Dari percobaan yang telah dilakukan, diketahui bahwa Agen Kelas atau Agen Departemen membutuhkan memori 35000 KByte – 40000 Kbyte, dan untuk menemukan Agent Server, Agen Kelas membutuhkan waktu 30 milidetik – 61 milidetik. Selain itu, ditemukan bahwa Ontology pada JADE 3.5 tidak kompatibel dengan Java Hastable

Kata Kunci: Agen, JADE, Java, Sistem Manajemen Kelas, Komunikasi, *Behaviour*

Martius

NPM 04 04 03 0598

Electrical Engineering Department

Counsellor

F. Astha Ekadiyanto ST, M.Sc.

BEHAVIOUR AND COMMUNICATION BETWEEN AGENTS IN MULTI-AGENT BASED CLASS MANAGEMENT SYSTEM BY USING JADE

ABSTRACT

Being proactive, autonomous and able to communicate are things that characterize collection of components to become an agent. It can take an initiative to perform a given tasks with minimum or without explicit stimulus from user, carry out this tasks independently, and communicate with others in order to achieve its goal.

JADE is a software platform that provides basic middleware layer functionalities for Agent Oriented Programming (AOP). It has good documentation (API), and has been developed for about ten years. Its independence for specific platform and ability in simplifying distributed application, gives JADE brighter future as a middleware to develop multi-agent application.

This thesis explores behaviour and communication between agents, by using class room management as a study case. In Multi Agent Based Class Management System, there is one Class Agent in every class room that manages course activity. Department Agent lives in a computer in Department Building, as bridge between users (lecturers) and this system. Through Department Agent, lecturer can poll data of attendance list of the courses, cancel course schedules, and search an empty class room which can meet schedule and capacity required by lecturer. Server agent works as interface between this system and database. In this system, every agent is independent in doing its jobs. It will become proactive and communicate with others in achieving its goal. With ability that it has (implemented with *behaviour*), agents can reduce responsibility of server over this platform. As application running independently, agent has to have ability to be run in machine with limited computational resources. Besides of that, more attention has to be paid on time efficiency in communication and in finding others agents, to optimize this system. From testing of this application, found that it takes 35000 KByte – 40000 KByte of memory for Class Agent or Department Agent to live, and 30ms – 61ms for Class Agent to find Server Agent. During developtmnent of this aplication, found that Ontology in Jade version 3.5 is not compatible with Java Hastable.

Keyword: Agent, JADE, Java, Class Management System, Communication, Behaviour

DAFTAR ISI

PERNYATAAN KEASLIAN SKRIPSI ..... **i**
PENGESAHAN ..... **ii**
UCAPAN TERIMA KASIH ..... **iii**
ABSTRAK ..... **iv**
ABSTRACT ..... **v**
DAFTAR ISI ..... **vi**
DAFTAR GAMBAR ..... **ix**
DAFTAR TABEL ..... **xi**
DAFTAR LAMPIRAN ..... **xii**
DAFTAR SINGKATAN ..... **xiii**
BAB I PENDAHULUAN **1**
1.1 LATAR BELAKANG 1
1.2 TUJUAN PENULISAN 2
1.3 BATASAN MASALAH 2
1.4 SISTEMATIKA PENULISAN 3
BAB II JAVA, AGEN DAN JADE **5**
2.1 JAVA 5
2.1.1 Sejarah Java 5
2.1.2 Platform 5
2.1.3 Java Virtual Machine 7
2.1.4 Pustaka Kelas 7
2.1.5 Versi-versi Java 7
2.2 AGEN 9
2.2.1 Definisi Agen 9
2.2.2 Karakteristik Agen 10
2.3 JADE 11
2.3.1 Sejarah JADE 11
2.3.2 JADE dan Paradigma Agen 12
2.3.3 Arsitektur JADE 15

2.3.4 RMA 17
2.4 ECLIPSE 17
2.5 EJIP 18
BAB III PERANCANGAN AGEN KELAS DAN AGEN DEPARTEMEN PADA SISTEM MANAJEMEN KELAS 19
3.1 SKENARIO MANAJEMEN KELAS 19
3.2 DIAGRAM USE CASE DAN DIAGRAM AGEN UNTUK RANCANGAN SISTEM MANAJEMEN KELAS 20
3.3 PERANCANGAN AGEN KELAS 22
3.3.1 Aktivitas Agen Kelas 22
3.3.1.1 Siklus Hidup Agen Kelas 23
3.3.1.2 Pengaturan Perkuliahan 26
3.3.1.3 Absensi 26
3.3.2 Diagram *Class* untuk Agen Kelas 28
3.4 PERANCANGAN AGEN DEPARTEMEN 29
3.4.1 Aktivitas Agen Departemen 29
3.4.1.1 Siklus Hidup Agen Departemen 29
3.4.1.2 *Polling* data absen 30
3.4.1.3 Pembatalan Jadwal Perkuliahan 31
3.4.1.4 Pencarian Kelas Kosong dan Pendaftaran Jadwal Kuliah Pengganti 31
3.5 Diagram Class untuk Agen Departemen 33
3.6 KOMUNIKASI ANTAR AGEN 34
3.6.1 Struktur Data 35
3.6.2 *Template* Pesan 37
3.7 PENEMPATAN AGEN-AGEN 38
3.8 JARINGAN KOMPUTER UNTUK PENGUJIAN 39
BAB IV PENGUJIAN DAN ANALISA PROGRAM 40
4.1 INTERAKSI AGEN DENGAN PENGGUNA 40
4.2 INTERAKSI SESAMA AGEN 44
4.2.1 Komunikasi Antar Agen pada *Shift* Pertama 44
4.2.2 Komunikasi Antar Agen untuk *Polling* Data 47

4.2.3 Komunikasi Antar Agen untuk Pembatalan Jadwal Perkuliahan 48
4.2.4 Komunikasi Untuk Pencarian Kelas Kosong dan Pendaftaran Jadwal Kuliah..... 49
4.2.5 Komunikasi Antar Agen untuk Pengiriman Data Absensi yang Telah Diisi..... 51
4.3 BEHAVIOUR AGEN KELAS DAN AGEN DEPARTEMEN 52
4.4 PENGGUNAAN MEMORI 54
BAB V KESIMPULAN 58
DAFTAR ACUAN 59
DAFTAR PUSTAKA 61
LAMPIRAN A 62
LAMPIRAN B 65
LAMPIRAN C 67

DAFTAR GAMBAR

Gambar 2.1 Diagram Platform Java [2] 6
Gambar 2.2 *Remote Monitoring Agent* 18
Gambar 3.1 Diagram use case untuk Sistem Manajemen Kelas..... 21
Gambar 3.2 Agen diagram untuk Sistem Manajemen Kelas 22
Gambar 3.3 Diagram s*equence* aktifitas Agen Kelas ketika baru dihidupkan..... 24
Gambar 3.4 Diagram keadaan untuk pengaturan kegiatan perkuliahan oleh Agen Kelas dengan menggunakan FSMBehaviour..... 24
Gambar 3.5 Siklus hidup Agen Kelas..... 25
Gambar 3.6 Diagram *sequence* pengaturan suatu kegiatan perkuliahan pada Agen Kelas 26
Gambar 3.7 Diagram Alir Absensi 27
Gambar 3.8 Diagram class umtuk Agen Kelas..... 28
Gambar 3.9 Siklus hidup Agen Departemen 29
Gambar 3.10 Diagram alir *polling* data absen 30
Gambar 3.11 Diagram alir pembatalan jadwal perkuliahan 31
Gambar 3.12 Diagram *sequence* pencarian kelas kosong oleh Agen Departemen..... 33
Gambar 3.13 Diagram *sequence* untuk reservasi kelas oleh Agen Departemen 33
Gambar 3.14 Diagram *class* Agen Departemen..... 34
Gambar 3.15 Struktur data jadwal perkuliahan untuk satu kelas selama satu hari..... 36
Gambar 3.16 Struktur data pembatalan jadwal kuliah..... 37
Gambar 3.17 Diagram deployment untuk Sistem Manajemen Kelas..... 38
Gambar 3.18 Jaringan komputer untuk pengujian Sistem Manjemen Kelas..... 39
Gambar 4.1 GUI Agen Server 40
Gambar 4.2 GUI Agen Kelas..... 41
Gambar 4.3 GUI Agen Departemen 41
Gambar 4.4 GUI tempat memasukan NIP/NPM 42
Gambar 4.5 GUI untuk tawaran perubahan lama waktu absen..... 42
Gambar 4.6 GUI untuk pemilihan lama waktu pengisian absen..... 42
Gambar 4.7 Pembatalan Kelas..... 43

Gambar 4.8 Pencarian kelas kosong untuk jadwal kuliah yang telah dibatalkan 44
Gambar 4.9 Komunikasi agen-agen pada *sift* pertama, ketika kuliah baru dimulai ... 45
Gambar 4.10 Komunikasi antar agen pada saat Agen Departemen melakukan *polling* data absensi 48
Gambar 4.11 Komunikasi antar agen pada saat Agen Departemen melakukan pembatalan jadwal perkuliahan 48
Gambar 4.12 Komunikasi antar Agen untuk pencarian kelas jadwal kosong dan registrasi jadwal kuliah ke Agen Kelas yang mengelola kelas tersebut 50
Gambar 4.13 Komunikasi antar agen yang terjadi pada saat pengiriman data absensi yang telah diisi dari Agen Kelas ke Agen Server 52
Gambar 4.14 *Behaviour-behaviour* pada Agen Kelas, ketika sedang menangani perkuliahan 53
Gambar 4.15 *Behaviour-behaviour* pada Agen Departemen 53

DAFTAR TABEL

Tabel 1Spesifikasi Komputer untuk Pengujian..... 54
Tabel 2 Peningkatan penggunaan memori pada komputer tempat Agen Kelas hidup 55
Tabel 3 Peningkatan penggunaan memori pada komputer tempat Agen Departemen hidup..... 56
Tabel 4 Waktu yang dibutuhkan Agen Kelas untuk Mencari Agen Server..... 57

DAFTAR LAMPIRAN

LAMPIRAN A 62
LAMPIRAN B 65
LAMPIRAN C 67

DAFTAR SINGKATAN

AID	*Agent Identifier*
AMS	*Agent Management System*
AOP	*Agent Oriented Programming*
API	*Application Programmable Interface*
AWT	*Applate Window Toolkit*
CT	*container table*
DF	*Directory Facilitator*
FIPA	*Fuondation for Intellegent Physical Agent*
FTUI	Fakultas Teknik Universitas Indonesia
GADT	*Global Agent Descriptor Table*
GUI	*Graphical programmable Interface*
IDE	*Integrated Development Environmen*
JADE	*Java Agent Development Framework*
JCP	*Java Comunity Process*
JDK	*Java Development Kit*
JDT	*Java Development Tools*
JNDI	*Java Naming and Directory Interface*
JPDA	*Java Platform Debuger Architecture*
J2EE	*Java 2 Platform, Enterprise Edition*
J2ME	*Java 2 Platform, Micro Edition*
J2SE	*Java 2 Platform, Standard Edition*
LPGL	*Library Gnu Public Licence*
NIP	Nomor Induk Pegawai
NPM	Nomor Pokok Mahasiswa
RMA	*Remote Monitoring Agent*

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Agen bukan sesuatu yang baru. Teknologi ini telah dikenal lebih dari satu dasawarsa terakhir. Namun, metode Pemrograman Berorientasi Agen atau *Agent-Oriented Programming* (AOP), merupakan suatu paradigma perangkat lunak yang relatif baru. Teknologi ini, telah menjadi bahan diskusi dan penelitian di antara komunitas ilmiah untuk beberapa tahun terakhir. Aplikasi yang dikembangkan berdasarkan teknologi agen dan AOP, telah banyak bermunculan, mulai dari sistem yang sederhana untuk kepentingan pribadi, hingga aplikasi yang komplek di bidang industri. Beberapa domain di bidang industri yang telah memanfaatkan teknologi agen antara lain adalah pengendalian proses, diagnosa-diagnosa sistem, perakitan, manajemen jaringan dan transportasi logistik [1]. Salah satu *middleware* untuk teknologi agen, yang saat ini paling berkembang dan mempunyai dokumentasi yang baik, adalah JADE (Java Agent Development Framework).

Fakultas Teknik Universitas Indonesia terdiri dari tujuh departemen. Kegiatan perkuliahan ketujuh departemen tersebut dilakukan di ruang perkuliahan yang terdapat di gedung kuliah bersama. Setiap mata kuliah hanya menempati satu ruang perkuliahan selama jam kuliah tersebut berlangsung. Untuk selebihnya, ruang tersebut dibiarkan kosong. Penentuan di ruang mana suatu mata kuliah akan berlangsung, dilakukan di awal semester. Permasalahan muncul ketika terjadi pembatalan suatu jadwal kuliah dan ingin dicarikan ruang kosong sebagai tempat melakukan kuliah pengganti. Dalam melakukan pencarian ini, perbandingan jumlah mahasiswa peserta kuliah dan daya tampung ruang, serta ketersediaan jadwal dosen, menjadi bahan pertimbangan.

Di samping masalah di atas, komponen lain yang harus diperhatikan agar terlaksananya kegiatan perkuliahan yang baik adalah absensi (daftar hadir).

Selama ini, kegiatan absensi dilakukan secara konvensional, dan memungkinkan mahasiswa yang datang di akhir jadwal perkuliahan, bisa mengisi absen, dan tercatat telah mengikuti perkuliahan dengan baik. Selama ini, dalam pengisian absen, mahasiswa dapat mewakilkan dirinya melalui mahasiswa lain yang juga mengikuti kuliah tersebut.

Selain dua hal di atas, efisiensi dan manajemen kelas juga perlu ditingkatkan. Ketika dosen belum datang ke ruang perkuliahan, sedangkan perkuliahan telah dimulai, kondisi ini dapat diberitahukan ke pihak departemen, agar menghubungi dosen yang bersangkutan dihubungi. Jika ada pembatalan jadwal suatu mata kuliah, maka akan langsung diumumkan di ruang perkuliahan dan ruang tersebut akan berstatus kosong, sehingga dapat digunakan untuk kegiatan perkuliahan yang lain.

Dengan melihat semua hal di atas, dilakukan sebuah pendekatan dengan menerapkan sistem berbasis multi agen (dalam hal ini menggunakan JADE), untuk mencarikan solusi permasalahan manajemen ruang perkuliahan di FTUI. Di samping itu, juga ingin diuji kemampuan JADE, dalam bekerja dan menangani permasalahan, terutama mengenai tingkah laku dan komunikasi antar agen.

1.2 TUJUAN PENULISAN

Skripsi ini ditulis dengan beberapa tujuan yaitu:

1. Membuat sebuah aplikasi manajemen ruang perkuliahan FTUI
2. Mengetahui unjuk kerja JADE untuk aplikasi manajemen ruang perkuliahan FTUI, terutama pada tingkah laku dan komunikasi antar agen yang terlibat dalam sistem tersebut.

1.3 BATASAN MASALAH

Pada skripsi ini, aplikasi yang dibuat adalah aplikasi manajemen ruang perkuliahan FTUI. Infrastruktur untuk aplikasi ini, terdiri dari *host-host* yang terdapat di ruang perkuliahan, *host* yang terdapat di masing-masing departemen, dan sebuah server tempat data-data kegiatan perkuliahan dan absensi berada. Untuk identifikasi pengguna, bagi dosen, digunakan NIP (Nomor Induk Pegawai) dan bagi mahasiswa digunakan NPM (Nomor Pokok Mahasiswa). Pengguna

mengetikan NIP/NPM pada GUI yang telah disediakan. Dengan menggunakan antar muka yang mengambil data berupa string, data tersebut diambil dan diolah oleh sistem. Tujuan antar muka ini, agar dikemudian hari, aplikasi ini dapat digabungkan dengan perangkat yang menggunakan teknologi yang lebih canggih untuk mengidentifikasi pengguna, misalkan *smart card reader*, *finger print scanner*, dll. Skripsi ini tidak membahas GUI yang menjadi antar muka antara sistem dengan pengguna (manusia) dan juga agen yang menjadi perantara (agen yang bertindak sebagai server) antara agen-agen dengan *data base*. Namun, hanya menggunakan GUI dan agen perantara dengan *data base* yang telah dibuat oleh *programmer* lain yang bekerja pada proyek manajemen ruang perkuliahan FTUI. Metode analisa dan perancangan yang digunakan, merujuk pada jurnal yang diajukan oleh Magid Nikraz [2].

1.4 SISTEMATIKA PENULISAN

Sistematika penulisan skripsi ini adalah:

1. **BAB I PENDAHULUAN**

   Bab ini berisi Latar Belakang, Tujuan Penulisan, Batasan Masalah, dan Sistematika Penulisan.

2. **BAB II JAVA, AGEN DAN JADE**

   Pada bab ini diperkenalkan tentang teknologi Java, agen dan JADE sebagai *platform* untuk agen yang digunakan dalam skripsi ini.

3. **BAB III PERANCANGAN AGEN KELAS DAN AGEN DEPARTEMEN PADA SISTEM MANAJEMEN KELAS**

   Pada bagian ini akan dibahas proses pembuatan Agen Kelas, yaitu agen yang berada di ruang perkuliahan, yang bertanggung jawab mengelola semua kegiatan di kelas tersebut, dan proses pembuatan agen departemen, yaitu agen yang terdapat di setiap gedung departemen yang ada di FTUI, yang menjebatani antara agen kelas dengan pihak dari departemen.

4. **BAB V PENGUJIAN DAN ANALISA PROGRAM**

   Pada bab ini akan dilakukan pengujian dan analisa dari program yang telah dibuat

5. BAB V PENUTUP

Pada bab ini berisi kesimpulan dari skripsi ini.

BAB II

JAVA, AGEN DAN JADE

2.1 JAVA

Pada bagian ini akan dibahas tentang teknologi Java, yang mencakup sejarah, platform, JVM (*Java Virtual Machine*), pustaka kelas, dan berbagai versi Java.

2.1.1 Sejarah Java

Java, bahasa pemograman berorientasi objek yang sekarang ini telah dikenal luas di seluruh dunia, berawal dari proyek penilitian yang dilakukan oleh Sun Microsystem pada tahun 1991, dengan nama *Green Project*. Proyek ini dilatarbelakangi oleh ketidakpuasan para insinyur Sun Microsystem atas kondisi C++ Sun dan C API (*Application Programming Interface*). Beberapa insinyur yang bekerja di proyek ini antara lain adalah Patrick Naughton, James Gosling dan Mike Sheridan. Tujuan yang ingin mereka capai adalah terciptanya sebuah teknologi untuk pemrograman peralatan cerdas (baik berupa komputer *desktop*, server, superkomputer, maupun peralatan dengan sumberdaya yang terbatas, seperti yang kita kenal sekarang sebagai telepon selular, PDA, dll). Pada awalnya, Gosling hanya berusaha untuk memodifikasi bahasa C++. Namun, dalam perjalanyannya, pekerjaannya ini lebih mengarah pada pembentukan bahasa yang benar-benar baru yang pada waktu itu disebut “Oak”. Ide untuk menggunakan nama “Java” mucul ketika para insinyur tersebut datang ke sebuah kedai kopi yang di sana terdapat kata “java”. Sejak saat itu, nama “Java” mulai digunakan untuk bahasa pemograman dan *platform* ini. Secara resmi java diumumkan oleh Sun Microsystem pada tahun 1995.[3][4][5]

2.1.2 Platform

Java Platform adalah sebuah istilah yang merujuk pada sekumpulan program-program dari Sun Microsystem yang memungkinkan seseorang untuk

membangun dan menjalankan program yang dibuat dengan menggunakan bahasa pemrograman Java. *Platform* ini tidak tergantung dengan prosessor dan sistem operasi tertentu. Untuk bekerja, ia membutuhkan mesin eksekusi yang disebut *virtual machine* dan *compiler* dengan standar pustaka tertentu.

Pada *platform* Java, terdapat beberapa edisi, yaitu [3][4]:

- Java ME (Micro Edition)

  Edisi *platform* java yang menyediakan beberapa set pustaka (yang dikenal dengan nama *profiles*) untuk dijalankan pada peralatan dengan sumber daya yang terbatas (sperti PDA, telepon seluler dll).

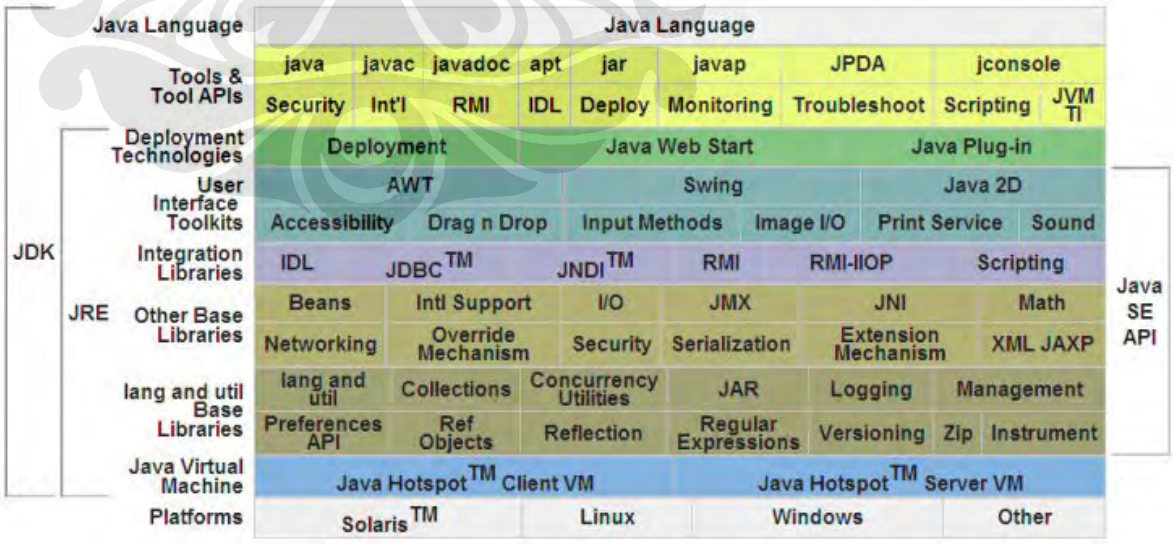
- Java SE (Standard Edition)

  Edisi *platform* java yang ditujukan untuk tujuan umum yang dijalankan pada komputer *desktop*, server, dan peralatan sejenis.

- Java EE (Enterprise Edition)

  Edisi *platform* java yang ditujukan untuk aplikasi-aplikasi *multi-tier client-server* pada perusahaan.

Komponen penting dalam *platform* Java adalah *compiler* bahasa Java, pustaka, dan *runtime environtment*, yaitu tempat dimana kode-kode Java yang berada dalam bentuk *byte*, dieksekusi berdasarkan aturan-aturan yang terdapat di spesifikasi *virtual machine*.



Gambar 2.1 Diagram Platform Java [2]

2.1.3 Java Virtual Machine

Java Virtual Machine adalah inti dari *platform* Java. Di dalamnya terdapat JIT (*Just In-Time*) *compiler* yang menerjemahkan *byte code* menjadi instruksi pada prosesor, pada saat program dijalankan dan menampung kode-kode asli program tersebut di memori selama proses eksekusi. Hal ini lah yang membuat Java tidak tergantung dengan jenis prosessor dan sistem operasi apapun.

2.1.4 Pustaka Kelas

Salah satu konsep yang diusung oleh Java adalah *reusable code* atau kemampuan untuk menggunakan kembali kode-kode yang telah dibuat sebelumnya untuk tujuan yang sama atau masih bersesuaian. Tujuan dari konsep ini adalah memudahkan *programmer* dalam bekerja dengan Java. Kode-kode tersebut disediakan dalam bentuk pustaka yang dapat dimuat secara dinamis dan dapat dipanggil oleh aplikasi ketika ia dijalankan.

Dalam *platform* Java, pustaka kelas mempunyai tiga tugas yaitu [3]:

- Menyediakan fungsi-fungsi yang telah teruji dengan baik untuk melakukan tugas-tugas umum.
- Menyediakan abstrak antar muka untuk tugas-tugas yang memiliki ketergantungan besar dengan prosessor atau sistem opersai tertentu
- Ketika terdapat *platform* yang melandasi suatu pekerjaan, di mana pekerjaan tersebut menggunakan *platform* java, namun *platform* tersebut tidak mendukung *feature-feature* yang diharapkan oleh suatu aplikasi Java, maka pustaka kelas akan mengatasi kekurangan komponen yang dibutuhkan.

2.1.5 Versi-versi Java

Semenjak diluncurkannya JDK (*Java Development Kit*) 1.0 pada tanggal 26 Januari 1996, Java telah melalui berbagai tahap perbaikan. Sejak J2SE 1.4 pengembangan java diatur oleh JCP (*Java Comunity Process*) yang menggunakan JSRs (*Java Specification Requests*) untuk mengajukan dan menjelaskan penambahan dan perubahan pada *platform* Java. Versi-vesi yang telah diluncurkan oleh Sun Microsystem sejak JDK 1.0 hingga sekarang adalah [3][4]:

- JDK 1.1

  JDK 1.1 dikeluarkan pada 19 Februari 1997. Penambahan dilakukan pada *extensive retooling* pada *AWT event model, inner class,* JavaBeans dan JDBC.

- J2SE 1.2

  J2SE 1.2 dikeluarkan pada 8 Desember 1998, dengan kode *playground*. Versi ini hinga versi J2SE 5 dikenal dengan nama Java 2 dengan nama versi "J2SE" (*Java 2 Platform, Standard Edition*) menggantikan JDK untuk membedakan *platform* dasar dari dari J2EE (*Java 2 Platform, Enterprise Edition*) dan J2ME (*Java 2 Platform, Micro Edition*). Penambahan yang dilakukan pada versi ini adalah *reflection framework* dan *collections framework*. Java IDL (untuk interoperabilitas dengan COBRA), integrasi Swing *graphical API* ke kelas-kelas inti. Disamping itu, juga diluncurkan *Java Plug-in* dan JVM dilengkapi dengan *JIT compiler* untuk pertama kalinya.

- J2SE 1.3

  J2SE 1.3 dikeluarkan pada 8 Mei 2000, dengan kode *Kestrel*. Perubahan dan penambahan yang dilakukan antara lain pada HotSpot JVM, JavaSound, JNDI (*Java Naming and Directory Interface*) dan JPDA (*Java Platform Debugger Arcitecture*)

- J2SE 1.4

  J2SE 1.4 diluncurkan pada 6 Februari 2002, dengan kode *Merlin*. Merupakan versi Java pertama yang diluncurkan di bawah pengembangan JSR 59. Perubahan yang dilakukan antara lain ekspresi-ekspresi reguler yang dimodelkan setelah Perl, *exception chaining*, prosessor yang telah terintegrasi dengan *XML parsed* dan XSLT, yaitu JAXP, dan Java Web Start.

- J2SE 5.0

  J2SE 5.0 diluncurkan pada 30 September 2004, dengan kode *Tiger*. Dikembangkan di bawah JSR 176, dengan penambahan beberapa *feature* penting pada bahasa.

- Java SE 6

  Java SE 6 diluncurkan pada 11 Desember 2006, dengan kode *Mustang*. Platform Java versi ini dilengkapi dengan *database manager* yang memfasilitasi penggunaan bahasa-bahasa script (seperti JavaScript yang digunakan mesin Rhino yang dimiliki oleh Mozila) dengan JVM dan mampu mendukung untuk bahasa Visual Basic. Peningkatan lainnya dilakukan pada GUI, terutama untuk mendukung WindowsVista. Selain itu, juga terdapat peningkatan pada antarmuka pada JPDA (*Java Platform Debuger Architecture*) dan JVM untuk pengecekan kesalahan dan pemantauan yang lebih baik.

2.2 AGEN

Agen menunjukan kemampuan yang menjanjikan untuk menjadi paradigma baru dalam pengembangan perangkat lunak. Dilihat dari sejarahnya, teknologi agen berasal dari penelitian di bidang kecerdasan buatan (*artificial intilligence*) [1]. Konsep-konsep tentang kecerdasan buatan tersebut, dibawa dan diterapkan ke dalam sistem terdistribusi. Setiap bagian yang menyusun sistem tersebut, menjadi otonom, memiliki kecerdasan untuk melakukan pengambilan keputusan tanpa harus mendapatkan perintah dari luar sistem, dan mampu berkomunikasi serta bernegosiasi antar sesamanya [7][8]. Hal inilah yang berkembang menjadi Pemrograman Berorientasi Agen (*Agent-Oriented Programming*, AOP)

2.2.1 Definisi Agen

Dari segi definisi, banyak orang yang memberikan definisi terhadap istilah agen itu sendiri. Namun, dari semua definisi tersebut, dapat ditarik suatu kesimpulan, bahwa agen adalah suatu komponen perangkat lunak khusus yang bersifat otonom yang menyediakan *interface* yang bersifat *interoperabel* ke berbagai sistem, dan dapat berlaku seperti agen pada manusia, yang bekerja pada *client*-nya, dalam rangka mencapai agenda yang diberikan padanya [1].

2.2.2 Karakteristik Agen

Agen, secara khusus mempunyai ciri-ciri sebagai berikut [1][6]:

- Otonom

  Agen dapat melakukan tugasnya tanpa adanya campur tangan langsung dari manusia atau agen lainnya, dan memiliki kendali terhadap aksi dan kondisi internalnya.

- Sosial

  Agen dapat berinteraksi dan bekerja sama dengan manusia atau agen-agen lain dalam rangka mencapai tujuannya.

- Reaktif

  Agen dapat memperhatikan lingkungannya dan merespon terhadap lingkungan tersebut berdasarkan waktu dan kondisi.

- Proaktif

  Agen tidak hanya bertindak karena merespon lingkungannya, namun juga dapat mengambil inisiatif untuk memunculkan perilaku (*behaviour*) yang ditujukan untuk mencapai tujuannya.

- Bergerak

  Agen memiliki kemampuan untuk berpindah dari satu *node* ke *node* yang berbeda di dalam sebuah jaringan komputer.

- Terpecaya

  Agen dapat dipercaya, bahwa ia tidak akan mengirimkan informasi yang salah.

- Berusaha Mencapai Tujuan

  Agen akan selalu berusaha untuk melakukan tugas, guna mencapai tujuan yang ditargetkan kepadanya.

- Rasional

  Agen selalu berusaha untuk melakukan apa yang ditugaskan padanya, dan tidak pernah mencoba untuk mencegah tujuanya untuk tercapai.

- Dapat Belajar

  Agen dapat belajar dan beradaptasi dengan lingkungannya dan tujuan yang diinginkan oleh penggunanya.

2.3 JADE

Pada bagian ini akan dihabas mengenai JADE, mulai dari sejarah, hubungannya dengan paradigma agen, arsitekturnya, serta RMA yang digunakan untuk memantau kegiatan agen-agen.

2.3.1 Sejarah JADE

JADE pertama kali dikembangkan oleh Telecom Italia di akhir tahun 1998, dalam rangka verifikasi spesifikasi-spesifikasi awal FIPA (*Fuondation for Intellegent Physical Agent*). Namun dalam perjalannya – dengan segala usaha dan dana yang diberikan pada proyek ini – diperoleh hasil yang melebihi apa yang sebelumnya diharapkan, dan menjadikan JADE sebagai *platform* untuk *middleware*. Visi dari proyek inipun berubah yaitu untuk menyediakan layanan-layanan yang mudah digunakan baik oleh para pengembang aplikasi maupun oleh orang-orang yang belum mengenal spesifikasi FIPA.

Pada tahun 2000, JADE mulai menjadi perangkat lunak yang *open source* dan didistribusikan oleh Telecom Italia dibawah LPGL (*Library Gnu Public Licence*). LPGL memberikan hak pada semua orang untuk menyalin JADE, akses hingga *source code*, melakukan perubahan pada *source code* dan juga melakukan penggabungan dengan perangkat lunak lain yang sama-sama berada di bawah LPGL. Sebagai imbalannya, semua pekerjaan yang dilakukan dengan JADE, harus dikembalikan ke komunitas dan akan berada di bawah LPGL.

Di tahun 2003, keterlibatan pihak industri dalam pengembangan JADE, semakin terlihat jelas. Hal ini ditandai dengan terbentuknya keja sama antara Telecom Italia Lab dan Motorola Inc. dalam membentuk Badan Pengatur JADE (*JADE Governing Board*). Organisasi ini ditujukan untuk mengembangkan dan mempromosikan JADE. Hingga sekarang, telah tercatat beberapa perusahaan lain yang bergabung dengan organisasi ini, diantaranya adalah France Telecom R&D, Whitestein Technologies AG dan Profactor GMBH.

Beberapa *extention* utama untuk inti JADE, disediakan oleh LEAP, yaitu sebuah proyek yang didanai Komisi Eropa. LEAP bekerja pada rentangan tahun 2000 hingga 2002, yang menghubungkan JADE dengan *Java Micro Edition* dan *Wireless Network Environtment*. Ketersediaan JADE *run-time* untuk *platform*

J2ME-CLCD dan –CDC serta penggunaannya dalam mengatasi permasalahan yang timbul pada telekomunikasi bergerak (*mobile telecomunication*) adalah beberapa fitur utama JADE .

2.3.2 JADE dan Paradigma Agen

Salah satu keutamaan JADE adalah penerapan dari abstraksinya yang dilakukan dengan bahasa pemrograman berorientasi objek yang telah dikenal luas, yaitu Java, yang mempunyai API yang baik. Dalam pengembangan tersebut, rancangan-rancangan JADE banyak dipengaruhi oleh abstraksi agen, diantaranya adalah [1]:

- Agen bersifat *autonomous* dan proaktif

  Agen mempunyai urutan eksekusi sendiri dan menggunakannya untuk mengatur siklus hidup dan secara otonomi dapat menentukan kapan melakukan tindakan tertentu.

- Agen dapat menolak suatu permintaan dan agen tidak berorientasi pada koneksi

  Komunikasi antar agen – yang berupa pengiriman pesan-pesan - bersifat tidak serempak (*asyncronous*). Dalam komunikasi ini, tidak terdapat suatu ikatan antara agen-agen yang berkomunikasi. Untuk mengenali agen penerima pesan, tidak diperlukan referensi objek agen tersebut. Proses identifikasi dilakukan dengan merujuk pada nama agen penerima yang terlebih dahulu ditempelkan pada pesan. Model komunikasi seperti ini memberikan beberapa keuntungan, diantaranya memungkinkan penerima untuk memilih pesan-pesan mana yang akan diambil dan mana yang akan ditolak serta pengendalian urutan eksekusi. Dalam komunikasi dengan tipe *multi-cast*, metode ini memberikan peluang untuk mengirimkan pesan ke banyak agen, tanpa terikat dengan referensi objek agen penerima.

- Sistem yang bersifat *Peer to Peer*

  Setiap agen mempunyai nama yang unik. Ia dapat berpindah-pindah baik di dalam maupun antar *platform* dengan merujuk pada layanan *white-pages* dan *yellow-pages*.

Dalam implementasinya JADE dirancang untuk menyediakan *programmer* berbagai fungsi yang mudah digunakan dan di-*edit*. Fungsi-fungsi tersebut antara lain adalah [1][12]:

- Sistem distributif

  Pada JADE, agen dijalankan pada sistem distributif, dengan urutan eksekusi yang berbeda-beda, dan mampu saling berkomunikasi.

- Mengikuti spesifikasi FIPA

  FIPA (*Foundation for Intellegent Physical Agent*) adalah sebuah lembaga internasional yang mengembangkan standar-standar terkait dengan teknologi agen. Dengan mengikuti standar-standar FIPA, memberikan JADE interoperabilitas dengan *platform* lain yang mengikuti standar FIPA.

- Pengiriman pesan-pesan yang tidak serempak secara efisien

  JADE berusaha memilihkan cara berkomunikasi terbaik dari beberapa cara yang ada, dan jika perlu dengan menghindari pengurutan objek-objek Java. Untuk komunikasi antar *platform*, pesan akan dirubah dari representasi Java internal JADE, menjadi sintak-sintak, kode-kode, dan protokol-protokol *transport* yang mengikuti aturan FIPA.

- Layanan *white-pages* dan *yellow-pages*

  Sistem yang terdistribusi dapat diimplementasikan untuk mewakili domain dan subdomain sebagai gambaran dari direktori-direktori yang terdistribusi juga.

- Pengaturan siklus hidup agen secara sederhana dan efektif.

  JADE menyediakan API dan *graphical tools* untuk mengatur siklus hidup agen, seperti menghidupkan, menunda, memulai lagi, membekukan, mencairkan, memindahkan, menggandakan dan mematikan agen. Proses pengaturan ini dapat berlangsung baik secara lokal, maupun dari jarak jauh.

- Dukungan mobilitas agen

  Agen dapat berpindah dari satu proses/mesin yang satu ke yang lainnya. Perindahan ini dibuat transparan terhadap komunikasi agen, sehingga agen

dapat menjaga proses interaksinya dengan agen yang lain selama proses perpindahan.

- Mekanisme berlangganan

  Agen-agen atau bahkan aplikasi-aplikasi luar dapat berlangganan dengan *paltform* JADE, untuk mendapatkan informasi tentang semua kejadian pada *platform,* termasuk kejadian yang berhubungan dengan siklus hidup dan pertukaran pesan.

- *Graphical tools*

  Graphical tools sangat membantu bagi *programmer* dalam pengembangan sistem yang terkait dengan berbagai macam pengurutan eksekusi dan proses yang rumit yang dijalankan di berbagai mesin. *Graphical tools* memungkinkan percakapan yang terjadi dapat diperiksa dan diemulasikan serta eksekusi agen dapat dikendalikan dari jauh.

- Dukungan untuk *Ontology* dan *Content Language*

  Programmer dapat memilih *ontology* dan *content language* yang akan digunakan dan dapat juga menambahkan *content language* yang baru. Semua proses pemeriksaan berdasarkan *ontology* dan *content language* yang dipilih dilakukan secara otomatis oleh *paltform.*

- Pustaka protokol-protokol interaksi

  Pustaka protokol-protokol interaksi memodelkan pola komunikasi tertentu untuk mencapai satu atau lebih tujuan. Protokol-protokol interaksi dapat direpresentasikan dan diimplementasikan sebagai sekelompok *finite state macine* yang terjadi secara bersamaan.

- Penggabungan dengan berbagai teknologi berbasis web

  Beberapa teknologi web yang dimaksudkan dalam hal ini adalah seperti JSP, servlets, aplets dan teknologi *web service*.

- Dukungan terhadap platform J2ME dan lingkungan nirkable

  JADE *run-time* tersedia untuk *platform* J2ME-CDC dan CLDC

- Antarmuka untuk peluncuran dan pengendalian *platform* dan komponen terdistribusinya yang dilakukan dari aplikasi eksternal.

- Kernel yang dapat diperluas

  Hal ini memungkinkan *programmer* untuk memperluas fungsionalitas dari *platform* JADE melalui penambahan layanan-layanan terdistribusi pada tingkatan kernel.

2.3.3 Arsitektur JADE

Platform JADE terbentuk dari berbagai *Container* yang terdapat pada satu komputer, atau mungkin tersebar pada jaringan komputer. *Container* inilah yang menjadi tempat hidup bagi agen-agen. Container menyediakan JADE *run-time* dan semua layanan-layanan yang dibutuhkan untuk penempatan dan pengeksekusian agen.

Terdapat suatu *container* khusus yang disebut "Main Container". *Container* ini adalah *container* pertama yang dihidupkan dan semua *container* lain berkewajiban untuk mendaftarkan dirinya ke *Main Container*. *Main Container* mempunyai tugas-tugas sebagai berikut [1]:

- Mengatur *container table* (CT) yang merupakan tempat pendaftaran referensi-referensi objek dan alamat-alamat *transport* dari semua *container* yang membangun *platform*
- Mengatur *Global Agent Descriptor Table* (GADT) yang merupakan tempat pendaftaran semua agen-agen yang berada di dalam *paltform*, termasuk status dan lokasinya.
- Tempat AMS dan DF. AMS dan DF adalah dua agen khusus dimana AMS menyediakan pengaturan agen dan layanan *white-pages*, sedangkan DF layanan *yellow-pages* dari *platform*

Untuk penamaan - secara langsung (*default*) –, selain *Main Container*, maka *container-container* yang lainnya akan diberi nama *Container-1*, *Container-2*, dan seterusnya. Namun demikian, *programmer* dapat merubah nama *container*, yaitu ketika *container* tersebut dihidupkan.

Secara umum, dalam beroperasi, suatu *platform* tidak melibatkan *main-container*, namun hanya lokal *cache* serta sejumlah *container* yang ditempati oleh agen-agen yang menjadi subjek dan objek dari sebuah operasi (seperti pengiriman dan penerimaan pesan). Dalam menemukan agen yang menjadi penerima dari

pesan yang akan dikirim, *container* akan mencarinya di LADT (*Local Agent Descriptor Table*). Jika tidak ditemukan, *container* yang bersangkutan akan menghubungi *main-container* di *platform* di mana ia berada, untuk mendapatkan referensi jauh (*remote reference*). Referensi jauh akan selalu disimpan secara lokal oleh *container* tersebut. Karena pada dasarnya sistem berbasis agen adalah sistem yang dinamis, kemungkinan besar referensi tersebut akan menjadi tidak valid setelah beberapa saat, yang mungkin disebabkan karena perpindahan atau pemusnahan agen-agen, atau juga dikarenakan kemunculan agen-agen yang baru. Jika referensi yang sudah tidak valid digunakan, *container* akan menerima *exception* yang memaksanya untuk me-*refresh cache* tempat *container* tersebut menyimpan referensi agen-agen lain, dan memperbaharui data tersebut sesuai dengan data yang ada di *main-container*. Pembaharuan isi *cache* menggunakan prinsip LRU (*Last Recently Used*) yang dirancang untuk menjaga komunikasi antar agen, yang cenderung beralangsung lebih dari satu kali.

Untuk membedakan suatu agen dengan agen yang lain, maka setiap agen diberikan satu identitas yang unik. Identitas ini adalah AID (*Agent Identifier*). AID terdiri dari data-data yang mengikuti struktur dan semantik yang didefenisikan oleh FIPA. Elemen-elemen dasar dari AID adalah nama agen dan alamatnya. Nama agen bersifat unik secara global, yang disusun dari nama lokal dan diikuti dengan nama *platform* (misal: untuk agen dengan nama “tius” yang terdapat pada *platform* “depok”, maka nama globalnya adalah “tius@depok”). Sedangkan untuk alamat agen, diambil dari alamat transport yang didapat dari *platform*, yang terkait dengan MTP (*Message Transport Protocol*), dimana pesan-pesan yang mengikuti spesifikasi FIPA dapat dikirim dan diterima. Alamat *transport* baru dapat ditambahkan ke AID, yang misalnya ditujukan untuk menerapkan MTP khusus untuk agen-agen tertentu.

Di dalam *Main-Container* terdapat dua agen khusus yang diinisialisasi dan dijalankan oleh JADE ketika *main-container* dihidupkan. Agen tersebut adalah:

- *Agent Management System* (AMS)

  AMS bertugas untuk mengawasi satu *platform*. Setiap agen yang berada pada paltform tersebut dan ingin melakukan interaksi guna mengakses *white-pages* dan untuk mengatur siklus hidupnya, harus mengakses agen

ini terlebih dahulu. Secara otomatis, semua agen yang berada dalam platform yang bersangkutan akan didaftarkan oleh JADE ke AMS pada awal kehidupan agen tersebut, guna mendapatkan AID yang valid.

- *Directory Fasilitator* (DF)

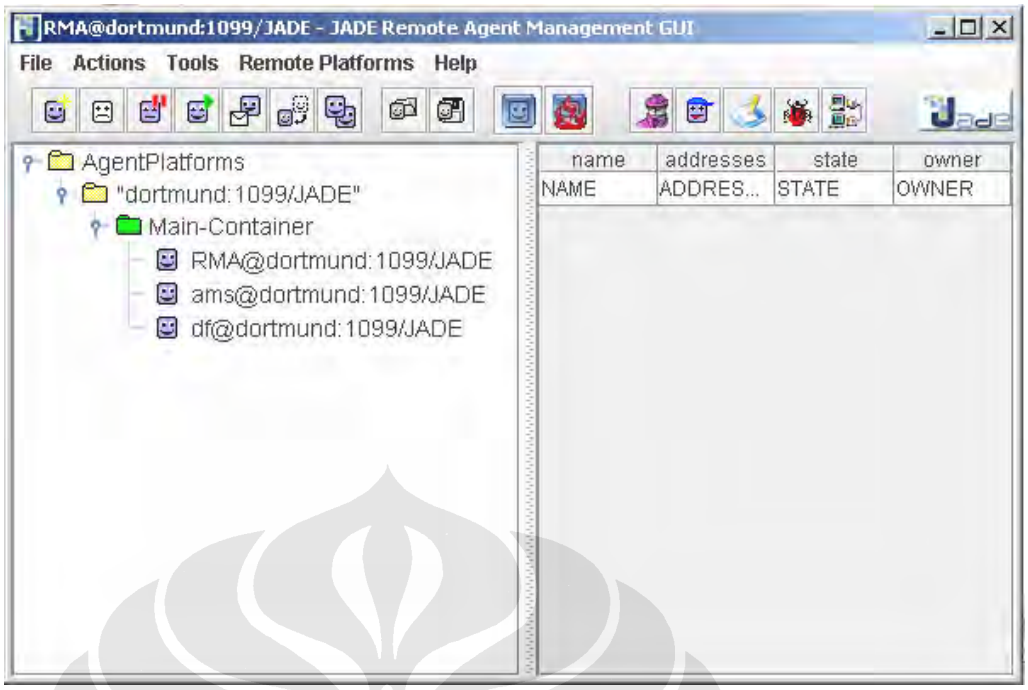
  Agen DF menerapkan layanan *yellow-pages* yang digunakan oleh agen-agen untuk mendaftarkan layanan yang dimilikinya atau untuk mencari layanan-layanan yang tersedia. Agen-agen lain dapat berlangganan dengan DF untuk mendapatkan informasi tentang layanan yang diingikannya yang didaftarkan di *yellow-pages*.

2.3.4 RMA

JADE RMA (*Remote Monitoring Agent*) adalah suatu alat yang mengimplementasikan sebuah *graphical platform management console*. Alat ini mengimplemenatsikan kelas `jade.tools.rma.rma` yang dalam pemakaiannya lebih sering dimunculkan dengan menggunakan GUI (yaitu dengan menambahkan peritntah `-gui`). RMA menyediakan fasilitas untuk memonitor dan mengatur sebuah *platform* JADE, yang terdiri dari beberapa *host* dan *container*. Salah satu layanan yang disediakan adalah pembentukan Agen Sniffer, yang berguna untuk memeriksa komunikasi berupa pengiriman pesan yang terjadi antar agen. Selain itu juga terdapat Agen *Instropector* yang berguna untuk melakukan pemeriksaan berbagai jenis *behaviour* sedang dimiliki oleh suatu agen, dan behaviour mana saja yang sedang aktif, serta yang mana saja yang sedang di-*block*. Tampilan GUI RMA ditunjukan pada Gambar 2.2.

2.4 ECLIPSE

Eclipse adalah suatu IDE (*Integrated Development Environment*) yang didalamnya terdapat JDT (*Java Development Tools*), yang dibuat dengan menggunakan Java [7]. Tujuan utama Eclipse adalah sebagai IDE untuk pengembangan aplikasi Java, namun bukan berarti aplikasi dengan bahasa pemrograman lain tidak dapat dikembangkan dengan menggunakan Eclipse.



Gambar 2.2 ***Remote Monitoring Agent***

2.5 EJIP

EJIP adalah *plug-in* yang ditambahkan pada Eclipse untuk menjalankan agen dengan *platform* JADE di Eclipse. EJIP menyediakan JVM instan untuk Eclipse dan JADE. Penggunaan pustaka SWT memungkinkan penyatuan yang baik antara tampilan EJIP dan sistem operasi pengguna serta meningkatkan unjuk kerja, dan memberikan kenyamanan dalam melakukan pemograman. Dengan menggunakan EJIP, perancangan aplikasi dengan menggunakan JADE, dapat dilakukan pada Eclipse dengan lebih baik [7].

BAB III
PERANCANGAN AGEN KELAS DAN AGEN DEPARTEMEN PADA SISTEM MANAJEMEN KELAS

3.1 SKENARIO MANAJEMEN KELAS

Kegiatan perkuliahan di Fakultas Teknik Universitas Indonesia, dilakukan di ruang perkuliahan yang terdapat di gedung kuliah bersama. Di dalam setiap ruang perkuliahan, akan ditempatkan satu host (komputer). Dengan ID yang unik yang dimiliki pengguna (NIP untuk dosen dan NPM untuk mahasiswa), setiap pengguna dapat melakukan pengisian absen. Untuk dosen, selain dapat melakukan pengisian absen, juga dapat melakukan kegiatan lain seperti, pengaturan lamanya waktu pengisian absen (pada Agen Kelas), pembatalan kegiatan perkuliahan dan pencarian kelas yang kosong untuk kegiatan kuliah pengganti.

Informasi NIP/NPM dimasukan oleh pengguna dalam format *String,* yang diketikan pada GUI. Sistem mempunyai antar muka yang menerima inputan *String* untuk diolah. Tujuannya untuk menjaga kompatibilitas jika suatu saat nanti pada sistem ini ditambahkan alat yang lebih canggih untuk identifikasi pengguna, misalkan berupa *smart card reader, finger print scanner,* dll.

Untuk mendukung jalannya sistem, diperlukan beberapa komputer. Komputer di kelas dan di gedung Departemen adalah komputer dengan kemampuan komputasi yang minim, namun masih dapat menjalankan JADE dan Java, dan sebuah database. Komputer yang terdapat di kelas merupakan komputer tempat hidup Agen Kelas yang bertanggung jawab mengelola semua kegiatan yang diadakan di kelas tersebut. Mahasiswa dan dosen yang kuliahnya diadakan di kelas tersebut, dapat melakukan absensi di komputer itu, selama waktu absensi belum habis. Jika dosen berhalangan mengajar, maka ia dapat membatalkan perkuliahannya, sebelum jadwal perkuliahan tersebut dimulai. Dosen yang bersangkutan juga dapat mencari kelas yang kosong sebagai tempat mengadakan kuliah pengganti.

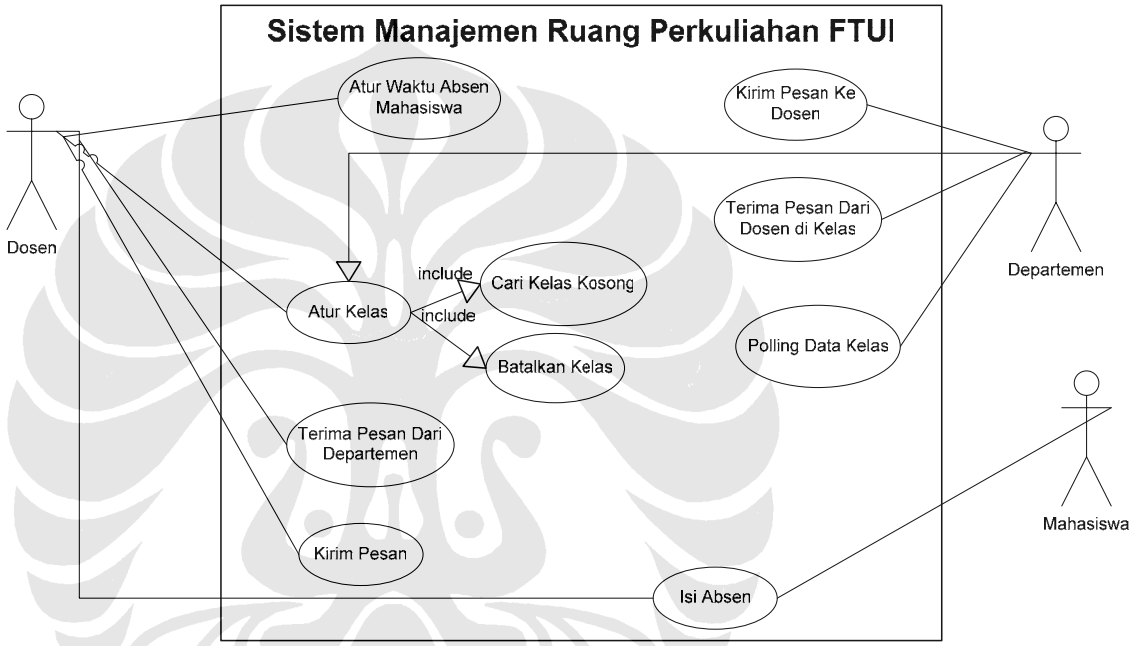
Setelah sampai di kelas, dosen dapat mengisi absensi, dan proses perkuliahan dapat dimulai. Jika dosen belum datang (ditandai dengan belum diisinya absensi dosen), maka Agen Kelas akan menunggu hingga 30 menit untuk membubarkan kelas, terhitung sejak kelas dimulai. Setiap 10 menit, selama masa menunggu tersebut, agen kelas akan mengirimkan kondisinya ke agen departemen. Tujuannya untuk memberitahukan bahwa dosen yang mengajar di kuliah tersebut belum berada di kelas. Pihak departemen juga dapat melakukan *pooling* untuk mengetahui kondisi kegiatan perkuliahan dari departemen yang bersangkutan.

3.2 DIAGRAM *USE CASE* DAN DIAGRAM AGEN UNTUK RANCANGAN SISTEM MANAJEMEN KELAS

Untuk memudahkan pembuatan dan pengembangan aplikasi Sistem Manajemen Kelas, diperlukan sebuah rancangan yang jelas. Rancangan ini, memberikan panduan dan batasan dalam pengembangan aplikasi. Metode yang digunakan untuk membuat rancangan Sistem Manajemen Kelas adalah metode yang disampaikan oleh Magid Nikraz, Giovani Caire dan Paris A Bahri [2], untuk menganalisa dan merancang sistem multi agen dengan menggunakan JADE. Pedekatan pertama yang dilakukan adalah penjabaran permasalahan ke dalam diagram *use case* serta diagram agen.

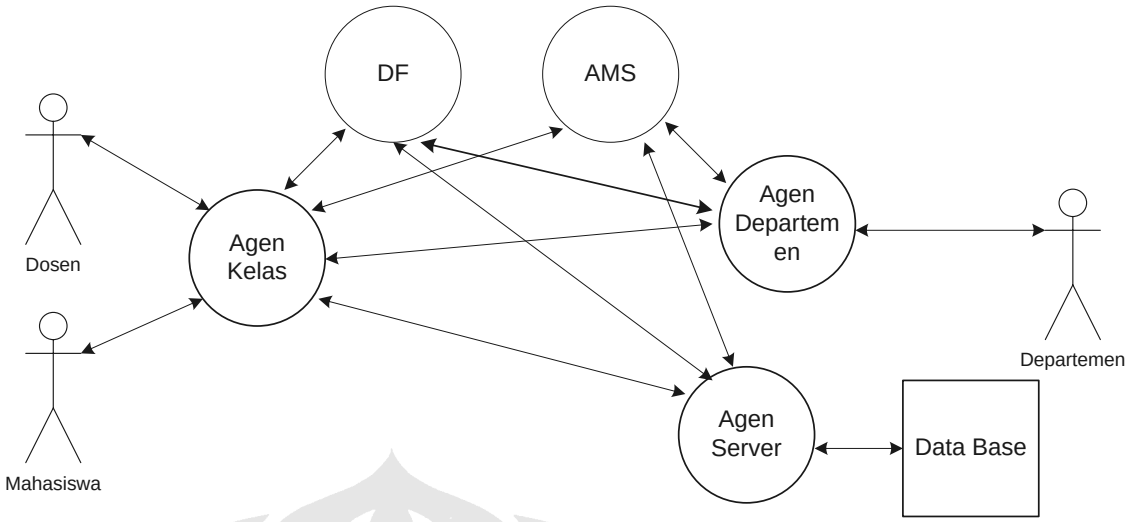
Diagram *use case* untuk Sistem Manajemen Kelas, ditunjukan pada Gambar 3.1. Pada diagram *use case* tersebut, dijelaskan beberapa fungsi yang dimiliki oleh pengguna sistem ini. Dosen dan mahasiswa dapat melakukan pengisian absen untuk setiap mata kuliah dimana ia terlibat di dalamnya. Dosen mempunyai wewenang untuk mengatur lamanya waktu absen (paling lama 30 menit terhitung sejak jadwal kuliah dimulai). Jika berhalangan mengajar, dosen dapat melakukan pembatalan kegiatan perkuliahan, dan dapat juga mencarikan kelas kosong sebagai tempat mengadakan kegiatan kuliah pengganti. Fungsi pencarian kelas kosong hanya dapat dilakukan pada komputer yang terdapat di departemen. Tujuannya adalah untuk menghindari pihak yang tidak berhak mengakses sistem, mencoba dengan paksa untuk memasuki sistem. Dosen juga dapat melakukan komunikasi berupa pengiriman pesan dengan pihak yang ada di

departemen. Untuk pihak yang ada di departemen (bisa petugas administratif atau dosen), sistem memberikan beberapa layanan, yaitu pembatalan kegiatan kuliah, pencarian kelas kosong untuk kegiatan kuliah pengganti, mengirim dan menerima pesan dari dosen yang berada di kelas serta melakukan *pooling* untuk mengetahui informasi perkuliahan yang sedang berlangsung (seperti apakah dosen untuk kuliah bersangkutan telah datang, dan berapa jumlah mahasiswa yang telah berada di kelas).



Gambar 3.1 Diagram use case untuk Sistem Manajemen Kelas

Dengan mempelajari dan menganalisa layanan-layanan yang diberikan oleh sistem ke pada pengguna, dapat dirancang sebuah sistem yang berbasiskan agen, yang diagram agennya ditunjukan pada Gambar 3.2 berikut.



Gambar 3.2 Agen diagram untuk Sistem Manajemen Kelas

Diagram agen diatas menunjukan bahwa terdapat tiga jenis agen pada sistem. Agen yang pertama adalah Agen Kelas, yang bertugas mengelola kegiatan perkuliahan di kelas. Pengguna yang akan berinteraksi dengan agen ini adalah dosen dan mahasiswa. Agen departemen bertugas menjembatani antara pengguna yang terdapat di departemen (dosen) dengan Agen Kelas. Agen ini memberikan layanan pembatalan kegiatan kuliah, pencarian kelas pengganti, dan layanan pengiriman pesan ke agen yang mengelola ruang perkuliahan, dimana perkuliahan dari departemen yang bersangkutan sedang berlangsung. AMS dan DF adalah agen-agen yang memberikan layanan *white-pages* dan *yellow-pages* ke seluruh *platform* [9][10][11]. Untuk Agen Server, seperti yang telah dijelaskan pada bab pertama, tidak akan dibahas dalam skripsi ini.

3.3 PERANCANGAN AGEN KELAS

Pada bagian ini akan dibahas perancangan Agen Kelas, yang mencakup semua kemampuan yang dipunyainya untuk melaksanakan tugas.

3.3.1 Aktivitas Agen Kelas

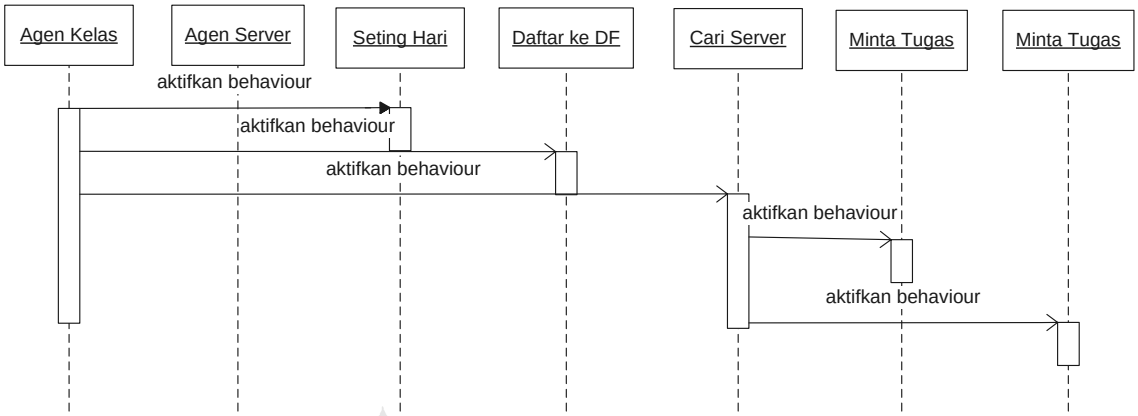
Dari penjabaran layanan-layanan yang akan diberikan oleh Agen Kelas, dapat dibuatkan beberapa diagram alir untuk menjelaskan algoritma-algoritma untuk menyelesaikan permasalahan yang dihadapi.

3.3.1.1 Siklus Hidup Agen Kelas

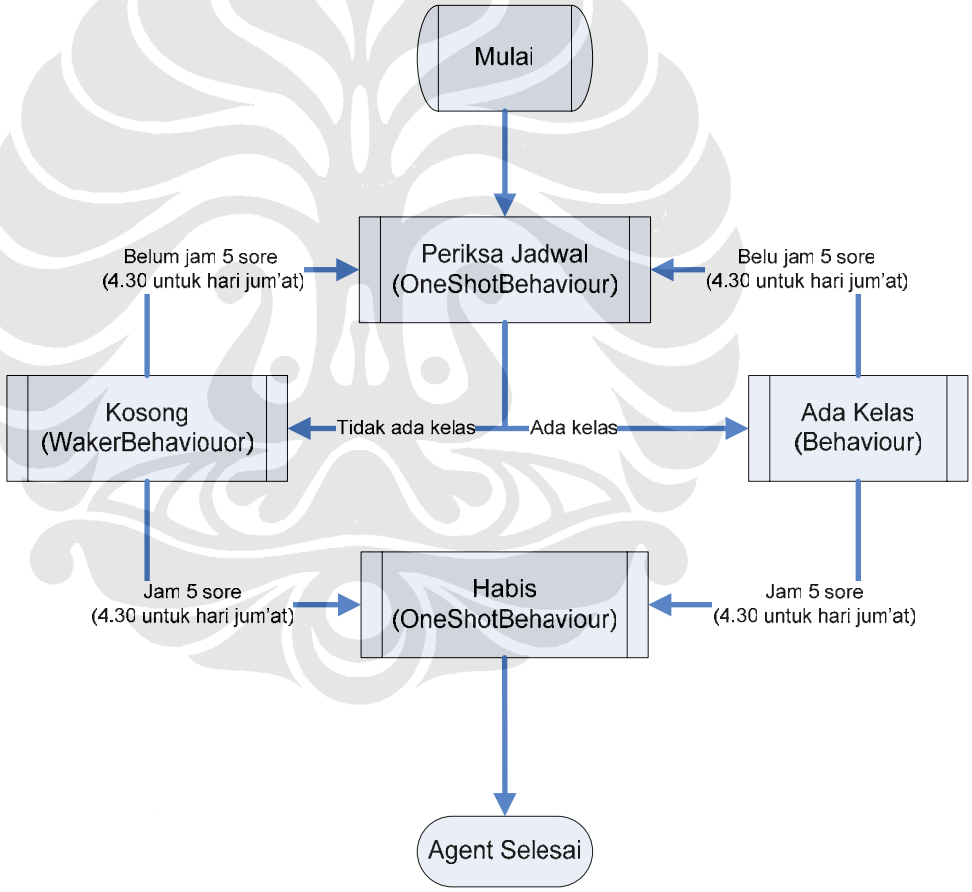
Ketika baru dihidupkan, Agen Kelas akan mendaftarkan dirinya ke layanan *yellow-pages* yang diimplementasika oleh Agen DF, untuk menawarkan layanan-layanan yang dimilikinya ke semua agen yang terdapat di sistem. Pada saat yang bersamaan, Agen Kelas akan melakukan pengaturan hari (dengan mengambil waktu komputer dimana ia berada). Tugas ini dijalankan dengan menggunakan *OneSotBehaviour*. Untuk mendapatkan data-data dari Agen Server tentang perkuliahan yang dilakukan di kelas yang dikelolanya pada hari yang bersangkutan, Agen Kelas harus menemukan Agen Server (dalam hal ini AID dari Agen Server) terlebih dahulu, kemudian mengirimkan pesan ke Agen Server untuk meminta data-data tersebut. Proses pencarian dilakukan dengan menggunakan *OneShotBehaviour*, sedangkan pengiriman pesan dilakukan dengan *TickerBehaviou*. Untuk menerima pesan dari Agen Server yang berisikan data-data perkuliahan, digunakan *CyclickBehaviour*, yang akan memeriksa apakah pesan telah diterima atau belum. Jika pesan belum diterima, maka *behaviour* ini akan mem-*block* dirinya sendiri, sehingga tidak menghabiskan sumber daya komputasi.

Setelah pesan yang berisikan data kegiatan perkuliahan untuk satu hari diterima oleh Agen Kelas, kegiatan perkuliahan di kelas tersebut siap untuk dimulai. Kegiatan perkuliahan diimplementasikan dengan menggunakan *FSMBehaviour*, yang dalam hal ini merupakan gabungan dari empat *Behaviour* lainnya, yaitu dua *OneSotBehaviour*, satu *WakerBehaviour*, dan satu *Behaviour*.

Gambar diagram *sequence* untuk aktifitas yang dilakukan oleh Agen Kelas ketika baru dihidupkan dapat dilihat pada Gambar 3.3. Sedangkan pengaturan kegiatan perkuliahan di kelas tersebut dilakukan dengan menggunakan algoritma yang ditunjukan pada diagram *state* yang ditampilkan pada Gambar 3.4.



Gambar 3.3 Diagram *sequence* aktifitas Agen Kelas ketika baru dihidupkan

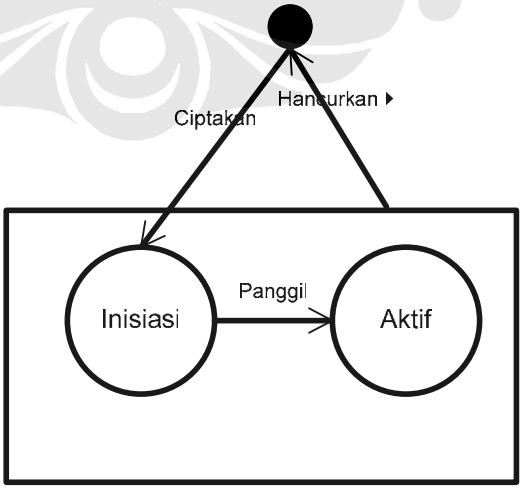


Gambar 3.4 Diagram keadaan untuk pengaturan kegiatan perkuliahan oleh Agen Kelas dengan menggunakan FSMBehaviour

Pada Gambar 3.4, hal pertama yang akan dilakukan oleh Agen Kelas yang menangani perkuliahan adalah memeriksa apakah ada kegiatan perkuliahan pada jam tersebut. *State* dimana proses pemeriksaan jadwal berlangsung disebut

"Periksa Jadwal". Tugas ini dilakukan dengan menggunakan *OneShotBehaviour*. Jika tidak ada jadwal perkuliahan, *state* akan berubah menjadi "Kosong", dimana pada behaviour ini Agen Kelas akan menunggu hingga jadwal perkuliahan berikutnya. Jika waktu menunggu telah habis, akan dilakukan pemeriksaan jam, apakah sekarang telah menunjukkan jam 05.00 sore untuk hari Senin hingga Kamis, atau 04.30 sore untuk hari Jum'at. Jika telah mencapai jam tersebut, berarti telah sampai pada akhir hidup Agen Kelas, *state* akan berubah menjadi "Habis" dan Agen Kelas akan dimatikan. Jika belum sampai pada jam tersebut, akan dilakukan pemeriksaan jadwal, apakah ada jadwal perkuliahan atau tidak. Jika terdapat jadwal perkuliahan, *state* akan berubah menjadi "Ada Kelas", Agen Kelas akan menangani kegiatan perkuliahan sesuai jadwal tersebut. Jika kuliah tersebut telah berakhir, akan dilakukan pemeriksaan jam, apakah telah menunjukan akhir siklus hidup agen. Jika jam menunjukan akhir hidup agen, *state* akan berubah menjadi "Habis", dan Agen Kelas akan dimatikan. Jika belum sampai ke akhir kelas, *state* akan berubah menjadi "Periksa Jadwal". Hal ini akan dilakukan terus menerus hingga akhir hidup Agen Kelas.

Secara sederhana, siklus hidup Agen Kelas terdiri dari beberapa bagian, yaitu ketika Agen Kelas tersebut dihidupkan, kemudian Agen Kelas akan menjalankan tugasnya, dan di sore hari (jam 17.00 untuk hari Senin-Kamis atau jam 16.30 untuk hari Jum'at) Agen Kelas akan dimatikan. Siklus hidup tersebut ditunjukkan pada Gambar 3.5.

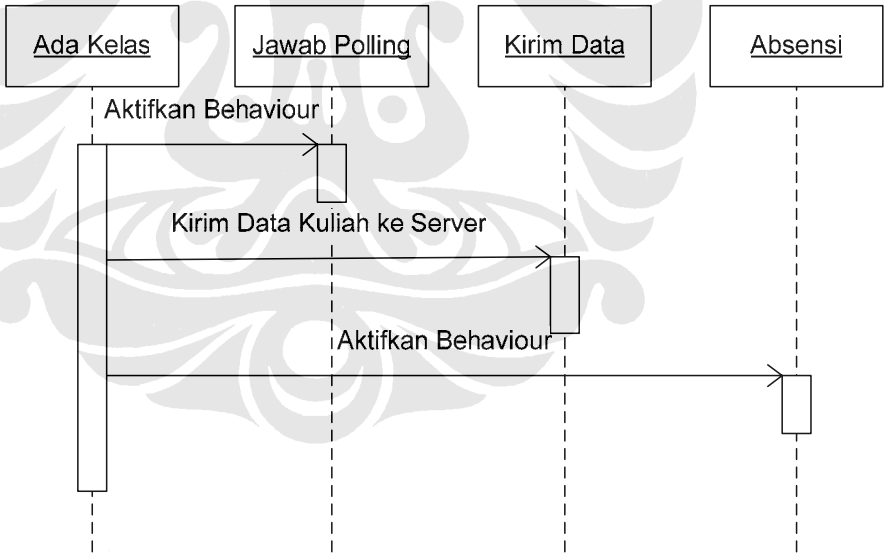


Gambar 3.5 Siklus hidup Agen Kelas

3.3.1.2 Pengaturan Perkuliahan

Pada saat perkuliahan dimulai, Agen Kelas akan mengaktifkan *OneShotBehaviour* untuk mengirim pesan mengenai data perkuliahan ke Agen Departemen, untuk memberitahukan bahwa salah satu kegiatan perkuliahan dari departemen tersebut, diadakan di kelas, yang dikelola oleh Agen Kelas ini. *CyclicBehaviour* untuk menjawab pesan untuk *polling* data dari departemen, juga diaktifkan. Jika pesan yang sesuai dengan *template* pesan untuk *CyclicBehaviour* ini diterima oleh Agen Kelas, maka Agen Kelas akan mengirimkan pesan berisikan informasi jumlah mahasiswa yang telah datang dan apakah dosen telah datang, ke Agen Departemen yang mengirimkan pesan *polling*. Jika kuliah berakhir, Agen Kelas akan menginformasikan hal tersebut ke Agen Departemen. Selain itu, data absensi mahasiswa dan dosen akan dikirim ke Agen Server.

Algoritma yang dijelaskan di atas, dapat digambarkan dengan menggunakan diagram *sequence* yang ditunjukan pada Gambar 3.6.



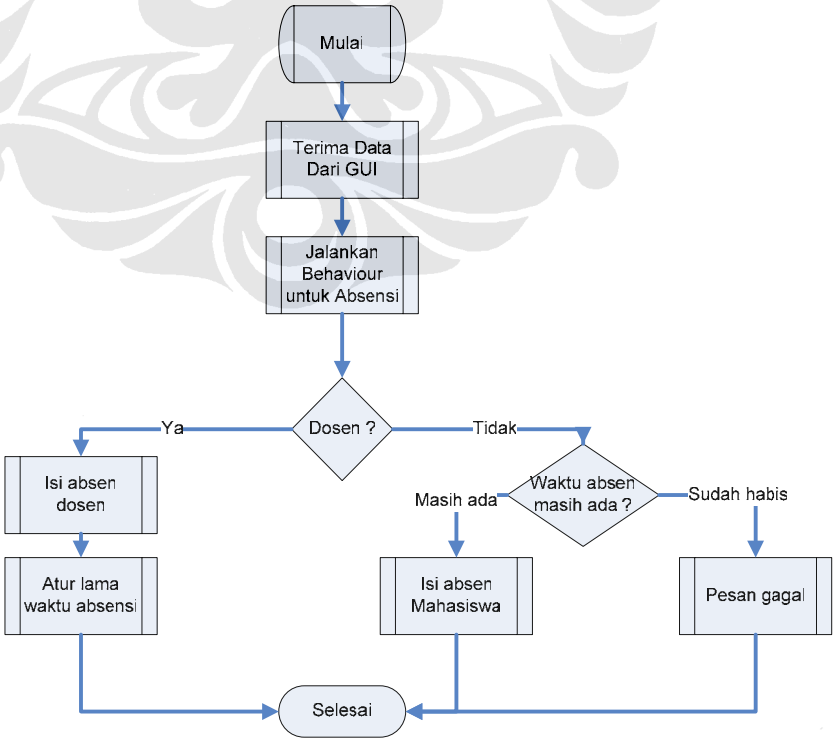
Gambar 3.6 Diagram *sequence* pengaturan suatu kegiatan perkuliahan pada Agen Kelas

3.3.1.3 Absensi

Jika suatu perkuliahan telah tiba waktunya untuk dimulai, Agen Kelas akan mengaktifkan layanan absensi untuk dosen dan mahasiswa peserta kuliah.

Diagram alir yang menjelaskan alur algoritma untuk menangani kegiatan pengisian absen, ditunjukkan pada Gambar 3.7.

Behaviour yang menangani aktifitas pengisian absen, diaktifkan untuk menerima inputan dari GUI yang berupa data NPN mahasiswa atau NIP dosen. Lama waktu abensi adalah 30 menit. Dosen dapat merubah lamanya waktu absen dengan kisaran antara 5 hingga 30 menit sejak awal jadwal perkuliahan. Mahasiswa hanya dapat melakukan absensi selama rentangan waktu tersebut. Jika dosen belum datang ke kelas, Agen Kelas akan mengirimkan ke Agen Departemen, data jumlah mahasiswa yang telah datang ke ruang kuliah. Kegiatan ini menggunakan *WakerBehaviour* dengan interval waktu dari satu pesan ke pesan berikutnya adalah 10 menit. Jika dosen melakukan absensi di kelas, *WakerBehaviour* ini akan dihilangkan dari Agen Kelas, namun jika setelah 30 menit, belum ada dosen yang melakukan absensi di ruang perkuliahan, maka kelas akan dibubarkan, dan *state* akan berubah menjadi "Kosong". Jika semua kegiatan tersebut telah dilalui, *Behaviour* "Ada Kelas" akan di *block* hingga waktu kuliah habis, atau sampai kuliah berakhir karena dosen memutuskan untuk mengakhiri perkuliahan lebih awal.

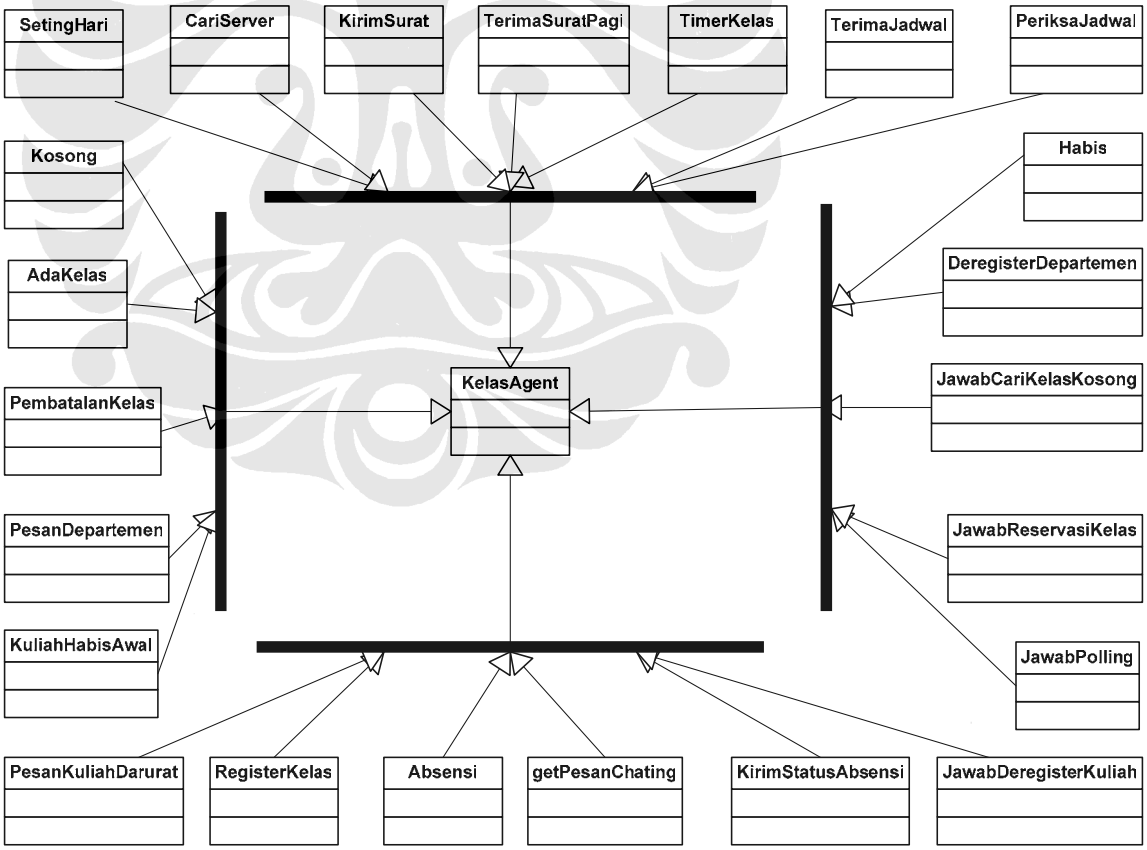


Gambar 3.7 Diagram Alir Absensi

3.3.2 Diagram *Class* untuk Agen Kelas

Untuk mempermudah dalam pemrograman, algoritma yang disampaikan dalam bentuk diagram alir, dijabarkan dengan menggunakan diagram *class*, untuk menentukan kelas-kelas apa saja yang akan dibuat. Karena keterbatasan ruang untuk menampilkan, hanya nama dari masing-masing kelas saja yang ditampilkan, sedangkan bagian *atribut* dan *method* sengaja tidak ditampilkan. Keterangan mengenai *atribut-atribut* dan *method-method* untuk masing-masing *class* terdapat di lampiran A Adapun diagram *class* untuk Agen Kelas ditunjukan pada pada Gambar 3.8.

Kelas yang ada di tengah diagram adalah `KelasAgent` untuk yang bertanggung jawab mengelola ruang perkuliahan. Kelas yang lainnya adalah *inner calss* yang berfungsi sebagai *Behaviours* yang memberikan kemampuan bagi KelasAgent untuk mengelola ruang perkuliahan.



Gambar 3.8 Diagram class umtuk Agen Kelas

3.4 PERANCANGAN AGEN DEPARTEMEN

Dari penjelasan mengenai Agen Departemen pada bagian sebelumnya, dapat ditentukan tiga buah tugas agen departemen yaitu:

- Memberikan layanan untuk *polling* data mahasiswa dan dosen dari departemen tersebut yang sedang melakukan perkuliahan di kelas bersangkutan.
- Melakukan pembatalan kegiatan perkuliahan
- Melakukan pencarian kelas yang kosong, dan melakukan pendaftaran perkuliahan di kelas kosong tersebut.

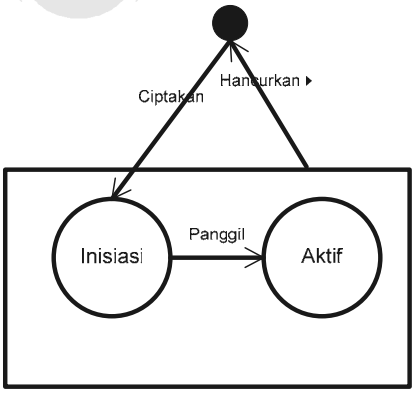
Dari ketiga poin di atas, dapat dibuatkan beberapa diagram tentang algoritma untuk menyelesaikan tugas tersebut.

3.4.1 Aktivitas Agen Departemen

Dari penjabaran layanan-layanan yang akan diberikan oleh Agen Departemen, dapat dibuatkan beberapa diagram alir untuk menjelaskan algoritma-algoritma untuk menyelesaikan permasalahan yang dihadapi.

3.4.1.1 Siklus Hidup Agen Departemen

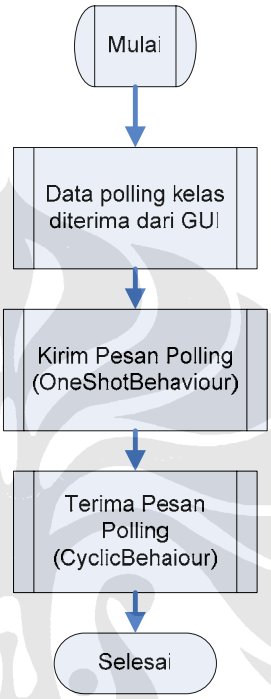
Agen Departemen memiliki siklus hidup yang sama dengan Agen Kelas. Pertama kali, Agen Departemen akan diinisialisasi, kemudian dipanggil dengan menggunakan referensi. Setelah Agen Departemen selesai mengerjakan tugasnya, Agen Departemen akan mati. Siklus hidup Agen Departemen ditunjukan pada Gambar 3.9.



Gambar 3.9 Siklus hidup Agen Departemen

3.4.1.2 *Polling* data absen

Pengguna sistem ini yang berada di setiap gedung Departemen, dapat melakukan polling data jumlah mahasiswa yang telah melakukan absensi dan apakah dosen yang mengajar di kelas tersebut telah datang atau belum. Diagram alir untuk melakukan polling adalah seperti ditunjukan pada Gambar 3.10.

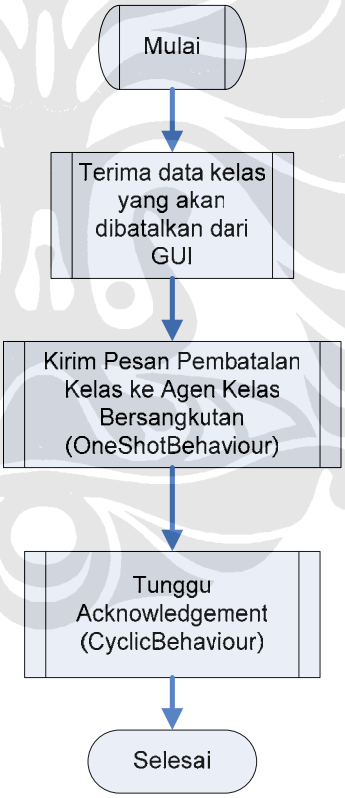


Gambar 3.10 Diagram alir *polling* data absen

Diagram alir di atas menunjukan alur program untuk menangani permintaan *polling* data kegiatan perkuliahan dari departemen bersangkutan di berbagai kelas yang dikelola oleh beberapa Agen Kelas. Agen Departemen mendapatkan permintaan *polling* daripengguna (dosen), yang disampaikan melalui GUI. Pesan polling data dikirimkan oleh Agen Departemen ke semua Agen Kelas yang terdaftar pada Agen Deaprtemen tersebut. Agen Departemen menggunakan *OneShotBehaviour* untuk melakukan tugas ini. Berikutnya, *CyclicBehaviour* untuk menerima pesan dari Agen Kelas, diaktifkan. Setiap pesan yang datang yang memenuhi *template* pesan untuk komunikasi ini, akan diterima oleh *CyclicBehaviour* ini, dan data tersebut langusung ditampilkan ke GUI.

3.4.1.3 Pembatalan Jadwal Perkuliahan

Untuk membatalakan jadwal perkuliahan, dosen memasukan data kuliah yang akan dibatalkan yang terdiri dari nama kuliah, kode kuliah, nomor ruang perkuliahan, tanggal perkuliahan, dan *shift* perkuliahan ke Agen Departemen, melalui GUI. Data tersebut akan dikirimkan ke Agen Kelas yang mengelola perkuliahan tersebut. Tugas ini dilakukan dengan menggunakan *OneShotBehaviour*. Jika pembatalan berhasil, maka akan dikirim *acknowledgmnt* ke Agen Departemen. *acknowledgmnt* ini akan diterima dengan menggunakan *CyclicBehaviour*. Jika pembatalan gagal, Agen Kelas tidak mengirimkan pesan apapun. Gambar diagram alir untuk menjelaskan algoritma guna menyelesaikan permasalahan di atas, ditunjukan pada Gambar 3.11.



Gambar 3.11 Diagram alir pembatalan jadwal perkuliahan

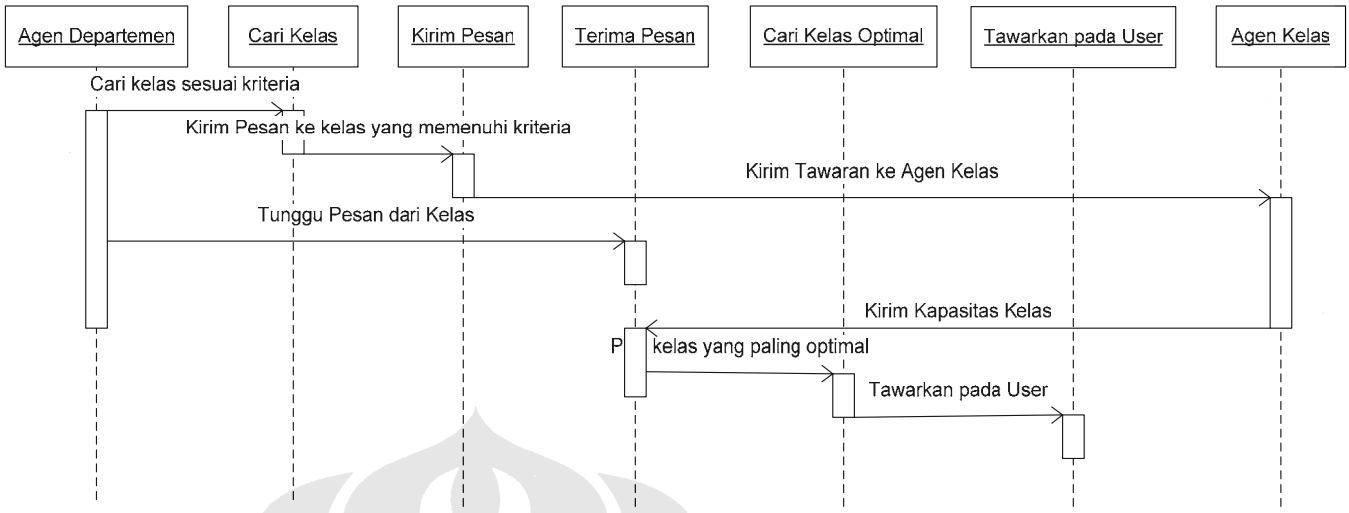
3.4.1.4 Pencarian Kelas Kosong dan Pendaftaran Jadwal Kuliah Pengganti

Untuk mencari kelas kosong pada waktu yang diinginkan, ada beberapa data yang harus dimasukan dosen, yaitu nama kuliah, kode kuliah, NIP dosen,

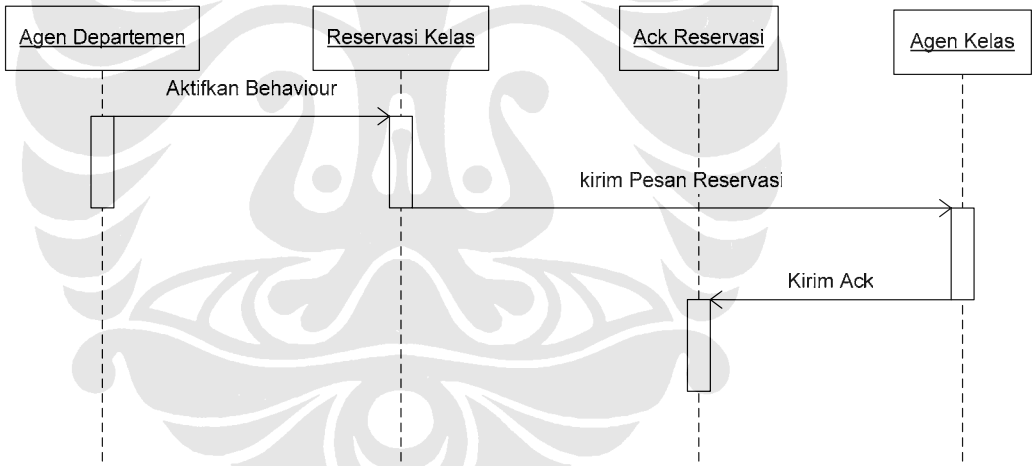
nomor pertemuan kelas yang akan dicarikan waktu penggantinya, jumlah peserta kuliah tersebut, dan tanggal kelas kosong yang diinginkan, *shift* kelas yang diinginkan. Agen Departemen akan mencari AID Agen-agen Kelas yang mempunyai kapasitas kelas sesuai permintaan, dan berada pada *state* "kosong", pada waktu yang diminta oleh dosen. Tugas ini dilakukan dengan menggunakan *OneShotBehaviour*. Setelah AID dari agen-agen kelas yang memenuhi permintaan dosen, ditemukan, dilakukan pengiriman pesan untuk menanyakan apakah Agen Kelas tersebut dapat menangani kegiatan perkuliahan yang ditawarkan oleh Agen Departemen. Pengiriman pesan ini dilakukan dengan menggunakan *OneShotBehaviour*. Pesan jawaban dari Agen-Agen Kelas akan diterima oleh Agen Departemen dengan menggunakan *CyclickBehaviour*. Tawaran dari Agen-agen Kelas akan diseleksi dan dipilih ukuran kelas yang paling optimal, artinya ruang perkuliahan dengan daya tampung sedikit lebih besar dari permintaan dosen.

Jika Agen Departemen telah menemukan kelas yang sesaui dengan permitnaan dosen, Agen Departemen akan menanyakan ke pada dosen, apakah akan melakukan pendaftaran jadwal kuliah di kelas tersebut. Jika tawaran diterima oleh dosen, akan dilakuakn reservasi kelas. Pesan reservasi dikirimkan oleh Agen Departemen dengan menggunakan *OneShotBehaviour*. *CyclicBehaviour* yang bertugas menanti jawaban apakah reservasi berhasil atau tidak dari Agen Kelas, diaktifkan. Jika pesan berisikan *acknowledgment* yang menyatakan reservasi berhasil, akan muncul *pop-up window* yang menginformasikan keberhasilan tersebut. Jika waktu untuk menunggu datangnya pesan balasan dari Agen Kelas telah habis, Agen Departemen akan menampilkan *pop-up window* yang memberitahukan kegagalan tersebut. Data-data yang telah dmasukan sebelumnya akan dihapus. Untuk tahap berikutnya, dosen dapat melakukan pencarian ulang.

Jika Dosen menolak untuk melakukan reservasi jadwal perkuliahan, data perkuliahan yang telah dimasukan sebelumnya akan dihapus, dan dosen dapat melakukan pencarian ulang atau memutuskan untuk mengakhiri pencarian. Diagram *sequence* untuk menjelaskan algoritma untuk menyelesaikan permasalahan diatas, ditunjukan pada Gambar 3.12 dan Gambar 3.13 berikut.



Gambar 3.12 Diagram *sequence* pencarian kelas kosong oleh Agen Departemen

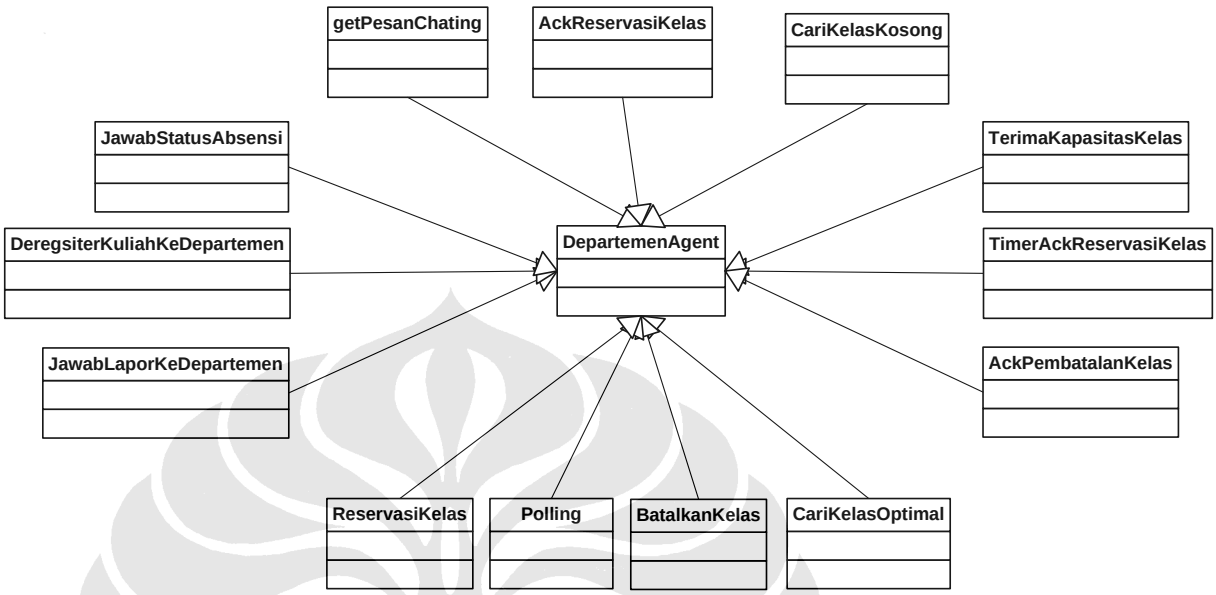


Gambar 3.13 Diagram *sequence* untuk reservasi kelas oleh Agen Departemen

3.5 Diagram *Class* untuk Agen Departemen

Untuk mempermudah dalam pemrograman, algoritma yang disampaikan dalam bentuk diagram alir, dijabarkan dengan menggunakan diagram *class*, untuk menentukan kelas-kelas apa saja yang akan dibuat. Karena keterbatasan ruang untuk menampilkan diagram *class* secara utuh, maka tampilan diagram *class* ini sengaja disederhanakan seperti yang terlihat pada Gambar 3.14, dengan

menghilangkan bagian *atribut* dan *method*. Keterangan mengenai *atribut-atribut* dan *method-method* untuk masing-masing *class* terdapat di lampiran B.



Gambar 3.14 Diagram *class* Agen Departemen

Kelas yang ada di tengah diagram adalah DepartemenAgen yang merupakan *blueprint* untuk Agen Departemen. Agen ini bertanggung jawab menjembatani antara pengguna di departemen dengan sistem, dan menyediakan layanan Agen Departemen, seperti yang telah disampaikan sebelumnya. Kelas yang lainnya adalah *inner calss* yang berfungsi sebagai *Behaviours* yang memberikan kemampuan bagi Agen Departemen dalam menjalankan tugasnya.

3.6 KOMUNIKASI ANTAR AGEN

Komunikasi antar agen dilakukan dengan berkirim pesan. Pesan yang dikirimkan dapat dimuati dengan objek, baik berupa string ataupun instan dari kelas yang lain. Metode mengirimkan pesan dengan objek yang ditumpangkan padanya, ada dua macam, yaitu dengan menggunakan ontology, dan dengan *serialization* dari Java[12][13]. Pada dari pengerjaan skripi ini, digunakan metode ontology. Namun karena keterbatasan Ontology yang tidak *compatible* dengan Hashtable, maka diputuskan untuk menggunakan *java serialization*.

3.6.1 Struktur Data

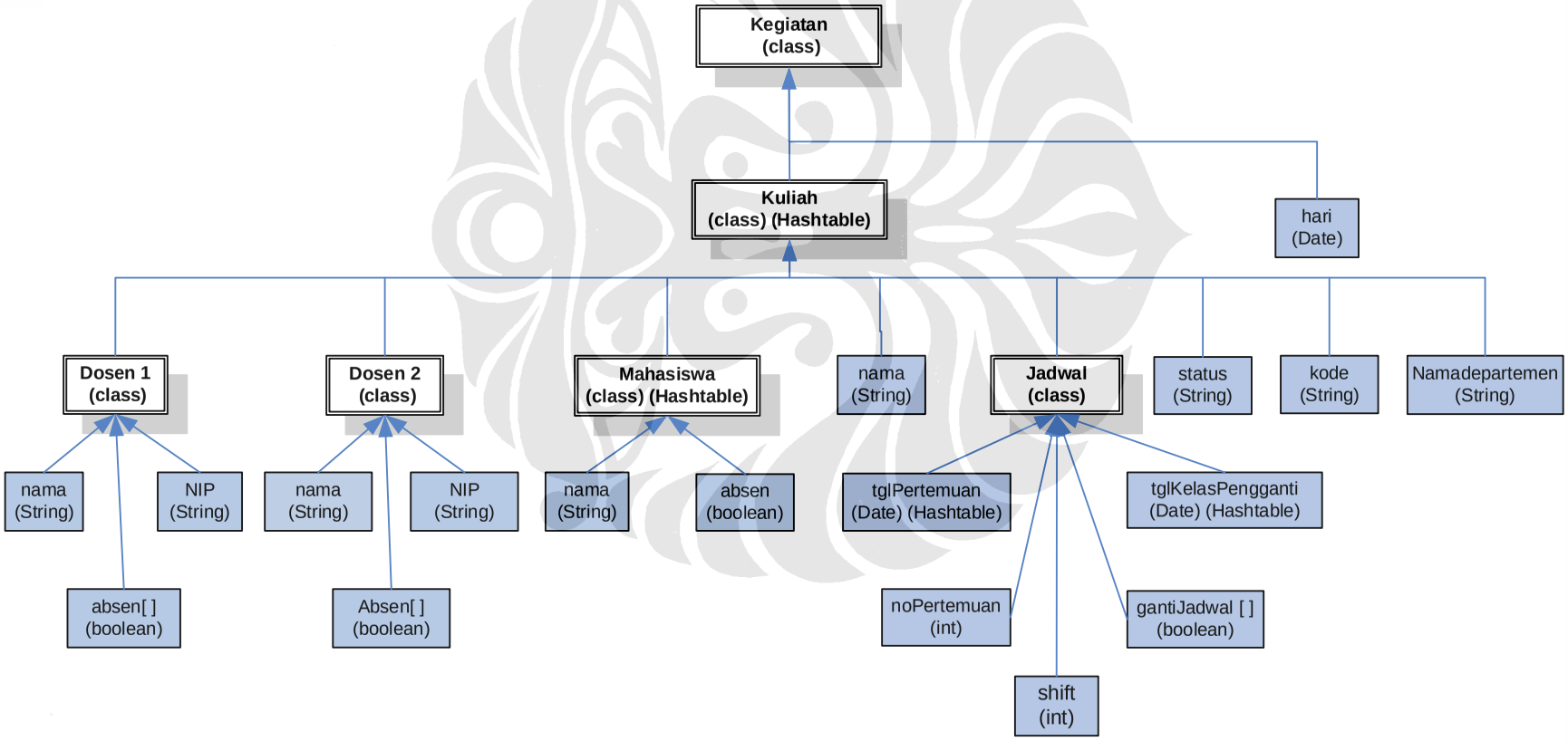
Struktur data jadwal kuliah untuk satu hari, yang dikirimkan di pagi hari oleh Agen Server ke Agen Kelas ditunjukan pada Gambar 3.15 (untuk keterangan lengkap setiap kelasnya, dapat dilihat pada Lampiran C).

Data tersebut terdiri dari satu *class* kegiatan yang terdiri dari dua data yaitu `kuliah` dengan tipe `Hashtable` dan hari dengan tipe `String`. `Hashtable` ini terdiri dari sembilan blok data objek dari kelas `Kuliah`, yang menunjukan sembilan *shift* perkuliahan yang ada dalam satu hari. Agen Kelas akan mengambil data kuliah dari `Hastable` ini berdasarkan *shift*, dimana nilai *shift* tersebut diambil dengan mengolah data waktu di komputer tempat Agen Kelas hidup pada saat itu.

Setiap kelas Kuliah terdiri dari dua objek `Dosen`, yang di dalamnya terdapat nama, NIP, dan absen dosen yang bersangkutan. Selain itu terdapat `Hashtable` mahasiswa yang berisikan objek dari kelas `Mahasiswa`. Setiap objek mahasiwa mengandung nama, NPM, dan daftar absen mahasiswa yang bersangkutan.

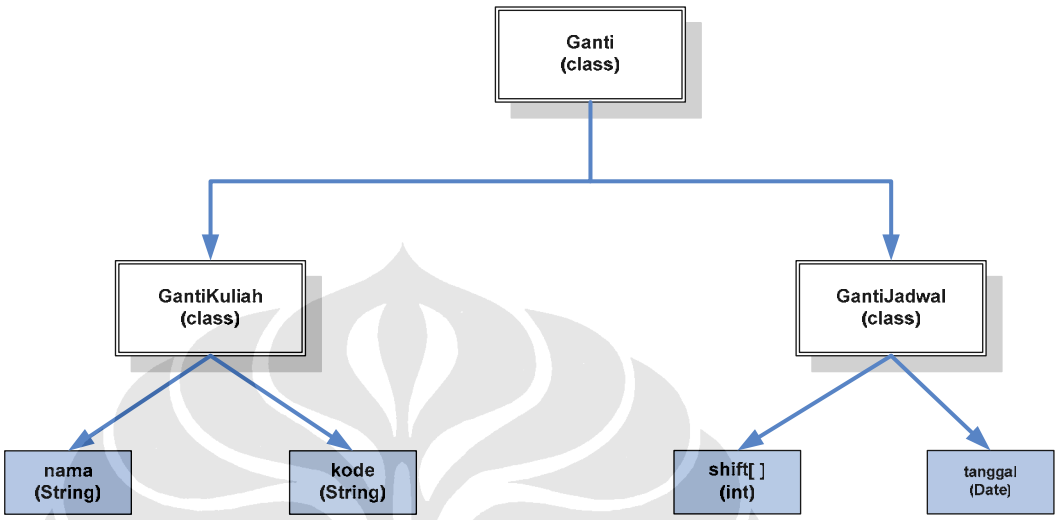
Objek lain yang terdapat di dalam kelas `Kuliah` adalah jadwal yang merupakan objek dari kelas `Jadwal`. Objek ini mengandung data tentang tanggal pertemuan, tanggal kelas pengganti, nomor pertemuan, keterangan pergantian jadwal, dan durasi perkuliahan.

Selain data di atas, juga terdapat tiga data lain dengan format `String`, yaitu nama kuliah, kode kuliah, dan nama departemen dimana kuliah ini berasal.



Gambar 3.15 Struktur data jadwal perkuliahan untuk satu kelas selama satu hari

Untuk pembatalan jadwal kuliah, data dikirim dalam bentuk objek yang dilampirkan ke pesan, dengan struktur yang ditunjukan pada Gambar 3.16:



Gambar 3.16 Struktur data pembatalan jadwal kuliah

Kelas `Ganti` terdiri dari dua objek dari kelas `GantiKuliah` dan `GantiJadwal`. Kelas `GantiKuliah` terdiri dari dua atribut, yaitu `namaKuliah` dan `kodeKuliah` yang akan diganti. Kedua data ini bertipe `String`. Kelas `GantiJadwal` terdiri dari dua atribut, yaitu shift kuliah yang akan diganti dan tanggal kegiatan kuliah yang akan diganti.

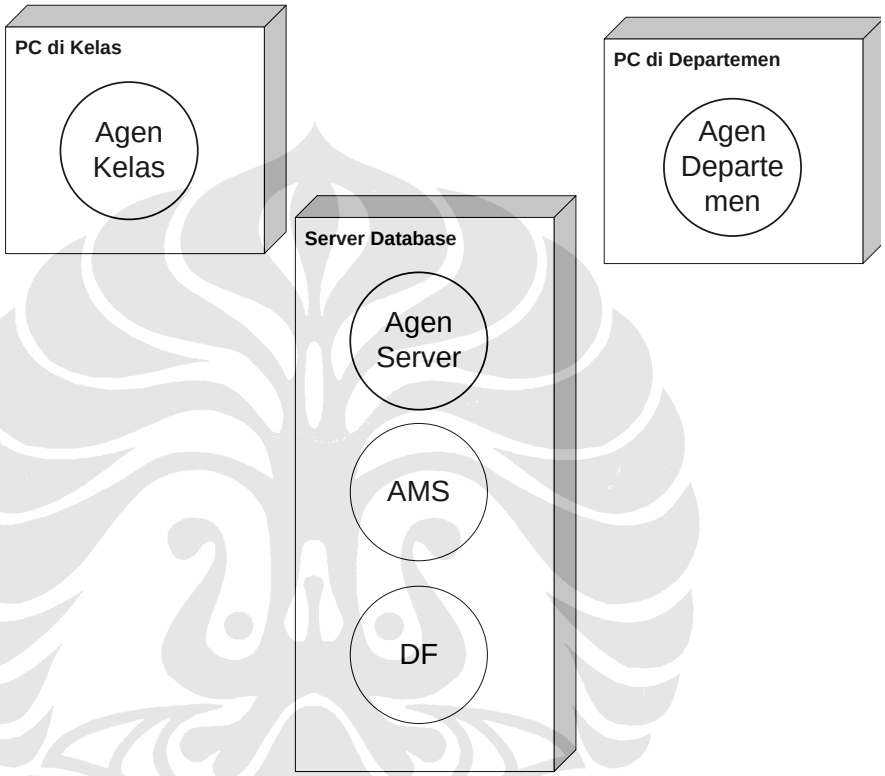
Selain struktur data seperti di atas, data lain yang dikirim adalah data dengan tipe `String` dan data dengan tipe `Hashtable`.

3.6.2 *Template* Pesan

Untuk menjamin bahwa pesan yang diterima oleh suatu agen, adalah pesan yang tepat, dan diproses dengan *behaviour* yang tepat pula, maka setiap *behaviour* yang menerima pesan, harus menyaring pesan yang masuk dengan menggunakan *template* pesan. Dalam skripsi ini, *template* pesan yang menggunakan berbagai parameter yaitu *ontology*, bahasa, *convesationID*, *performatif*, dan terkadang waktu pesan tersebut pertama kali dikirim, untuk memilih pesan yang akan diambil oelh satu *behaviour* pada agen.

3.7 PENEMPATAN AGEN-AGEN

Tahap selanjutnya adalah menentukan dimana agen tersebut akan ditempatkan. Gambaran dari penempatan agen dari sistem manajemen kelas FTUI, dapat dilihat pada Gambar 3.17.

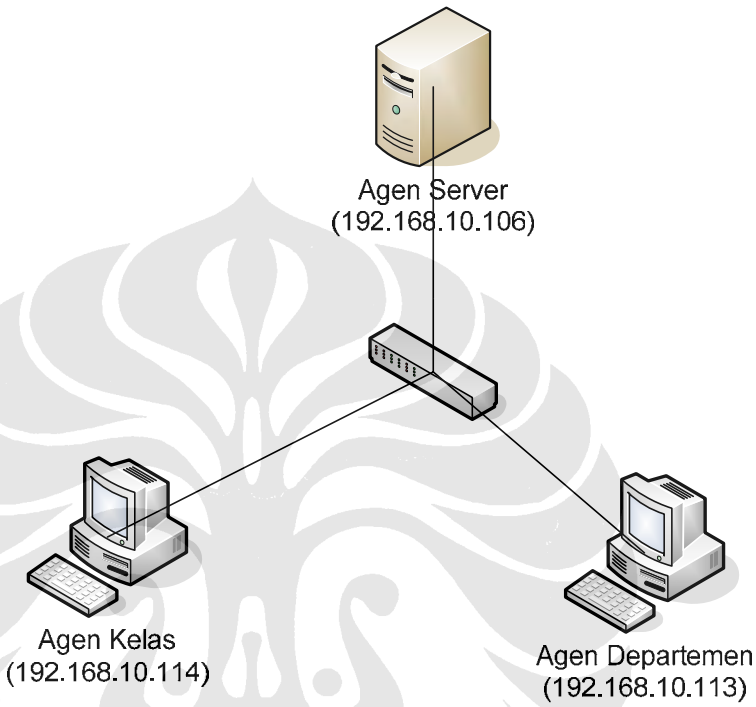


Gambar 3.17 Diagram deployment untuk Sistem Manajemen Kelas

Dari gambar di atas, dapat dilihat bahwa di setiap ruang perkuliahan terdapat Agen Kelas yang mengelola kegiatan perkuliahan di kelas tersebut. Agen departemen terdapat di setiap gedung departemen, yang memberikan layanan pada pengguna sistem yang berada di departemen untuk berinteraksi dengan sistem (seperti pembatalan kegiatan kuliah, pencarian ruang kuliah pengganti). Agen Server terdapat di sever, bertugas menjadi penghubung antara *database* dengan agen-agen yang lain. Pada komputer server, juga terdapat AMS dan DF.

3.8 JARINGAN KOMPUTER UNTUK PENGUJIAN

Untuk menjalankan dan menguji kinerja dari aplikasi ini, digunakan sebuah jaringan komputer sebagai tempat agen-agen dijalankan. Jaringan komputer yang digunakan ditunjukkan pada Gambar 3.18:



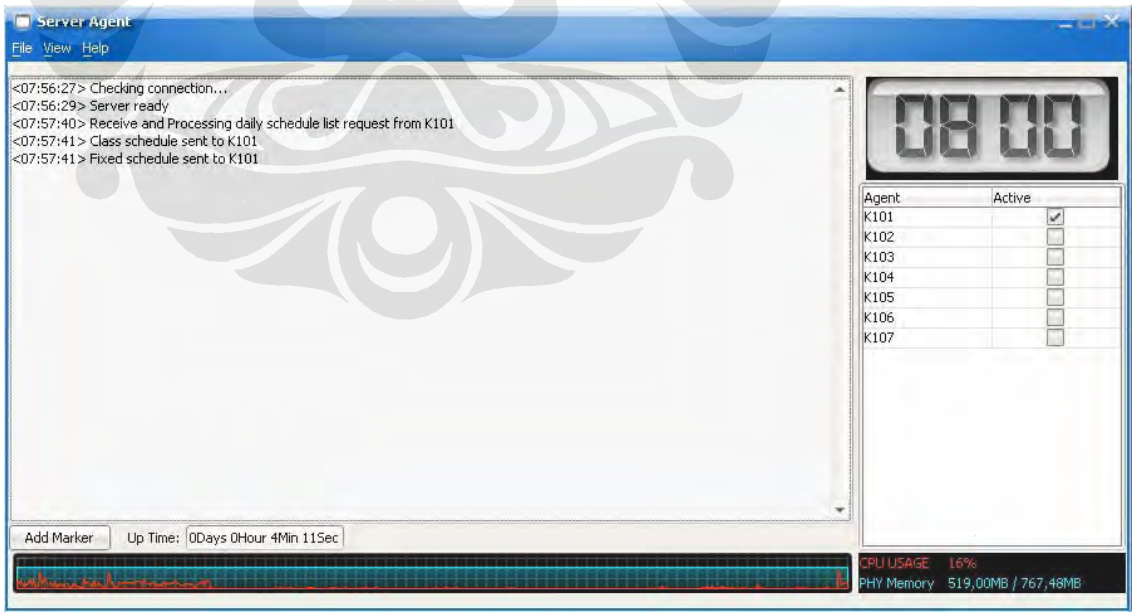
Gambar 3.18 Jaringan komputer untuk pengujian Sistem Manjemen Kelas

Agen-agen yang tergabung dalam sistem manajemen kelas FTUI ini, akan dijalankan pada jaringan lokal. Untuk pengujian dan pengambilan data pada skripsi ini, hanya tiga agen yang akan dijalankan, yaitu Agen Server, Agen Kelas (kelas K101), dan Agen Departemen (Departemen Elektro).

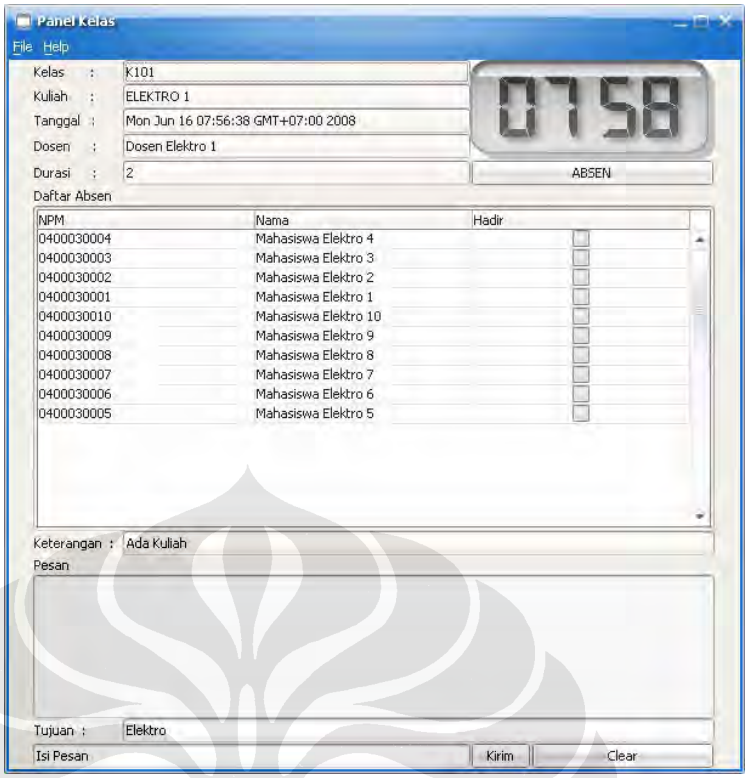
BAB IV
PENGUJIAN DAN ANALISA PROGRAM

4.1 INTERAKSI AGEN DENGAN PENGGUNA

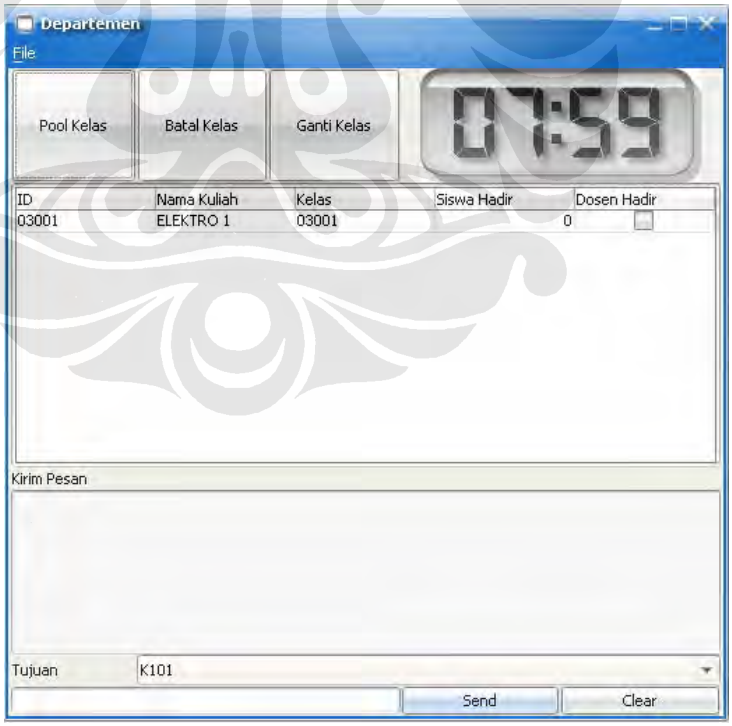
Dalam aplikasi Sistem Manajemen Kelas, seperti telah diterangkan sebelumnya, terdapat tiga jenis agen, yaitu Agen Server, Agen Kelas dan Agen Departemen. Tanpa menyimpang dari prinsip utama agen, yaitu menyelesaikan pekerjaan yang diberikan ke padanya dengan berkomunikasi antar sesama agen, pada sistem ini, terdapat interaksi antera agen-agen dengan pengguna. Pengguna berperan dalam memasukan data, sedangkan proses selanjutnya akan dilakukan oleh agen-agen. Seperti yang telah dijelaskan pada bagian batasan masalah, GUI bukanlah cakupan dari skripsi ini. GUI dalam hal ini adalah sebagai media untuk menunjukan interkasi yang ditawarkan oleh agen-agen terhadap pengguna. GUI dari ketiga agen yang telah disebutkan sebelumnya, ditunjukkan pada Gambar 4.1, Gambar 4.2 dan Gambar 4.3:



Gambar 4.1 GUI Agen Server

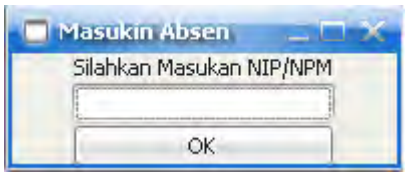


Gambar 4.2 GUI Agen Kelas



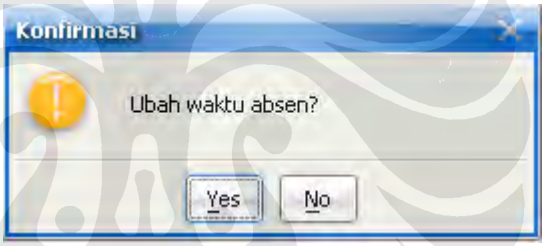
Gambar 4.3 GUI Agen Departemen

Dosen dan mahasiswa dapat mengisi absensi (daftar hadir), dengan menekan tombol “ABSEN” pada GUI Agen Kelas. *Pop-up window* untuk memasukan NIP/NPM diperlihatkan pada Gambar 4.4.

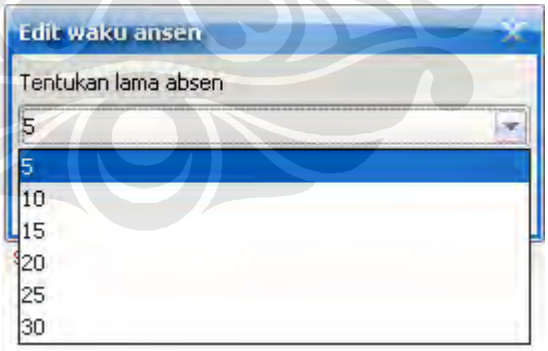


Gambar 4.4 GUI tempat memasukan NIP/NPM

Jika yang data yang dimasukan adalah NIP dosen, maka Agen Kelas akan menawarkan layanan pengaturan lamanya waktu absensi, yang berkisar antara 5 hingga 30 menit. Tampilan GUI untuk penawaran pengubahan lamanya waktu pengisian absen, ditunjukan pada Gambar 4.5 dan Gambar 4.6.



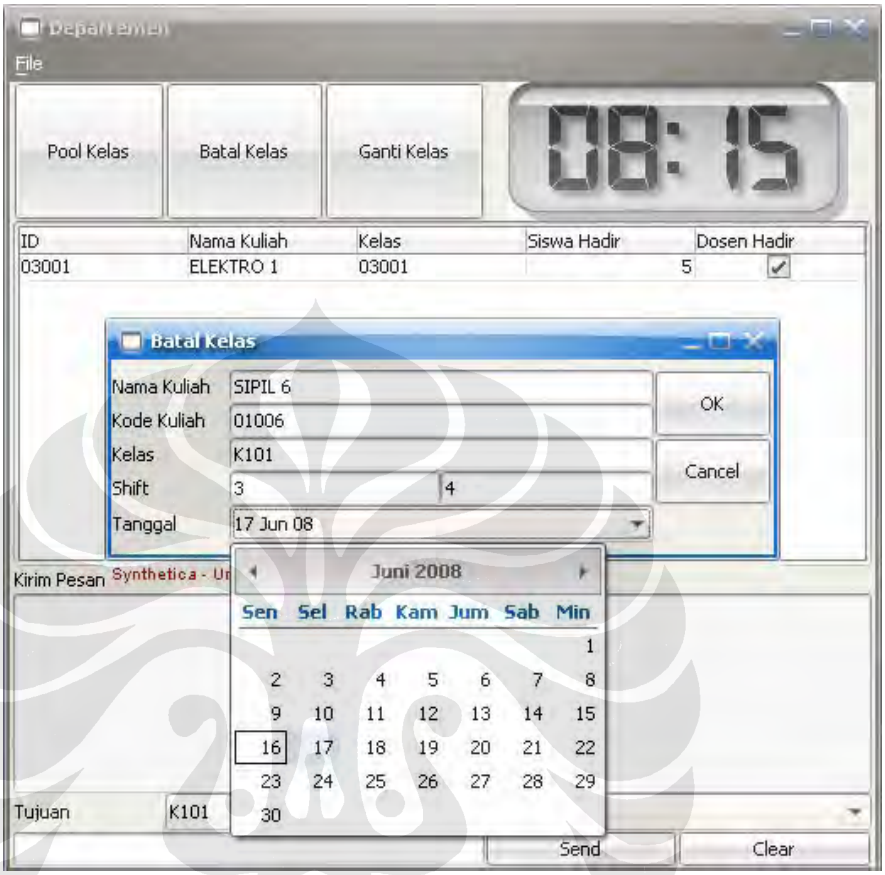
Gambar 4.5 GUI untuk tawaran perubahan lama waktu absen



Gambar 4.6 GUI untuk pemilihan lama waktu pengisian absen

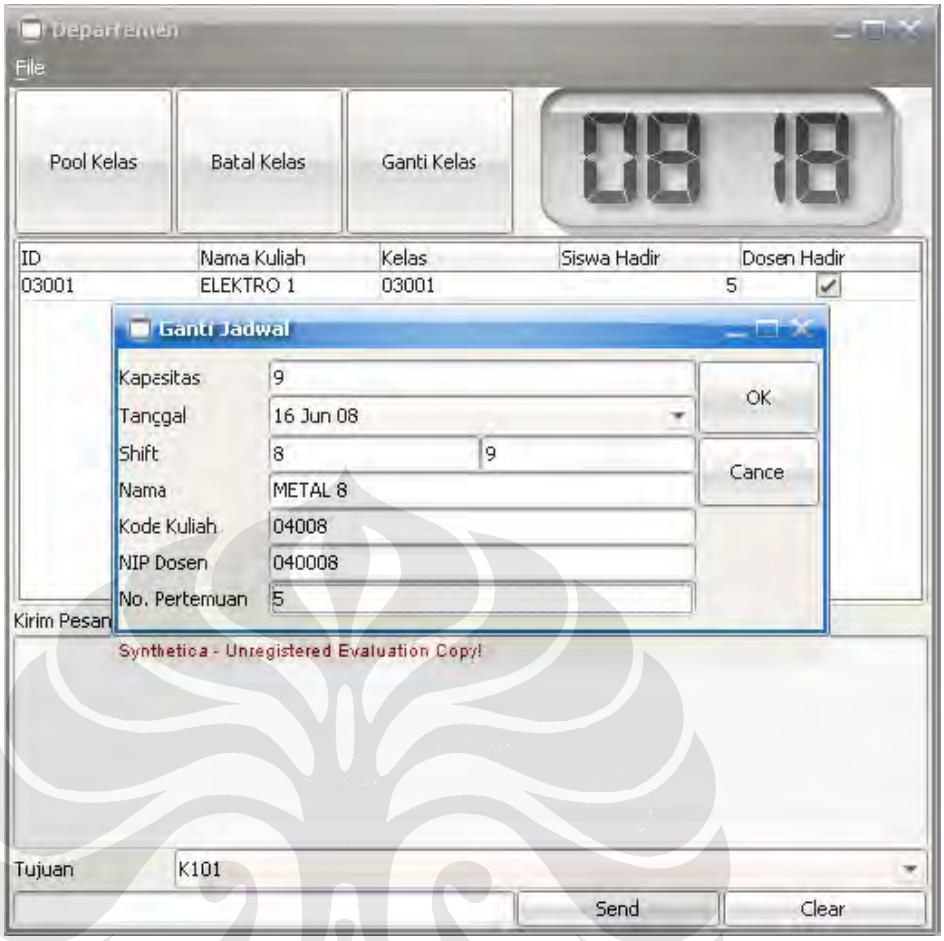
Untuk Agen Departemen, pengguna (dosen) dapat melakukan *polling* untuk mendapatkan data perkuliahan dari semua Agen Kelas yang sedang mengelola kuliah dari departemen yangtersebut. Pembatalan Kelas dilakuakan dari Agen Departemen, dengan menekan tombol “Batal Kelas”. Jika tombol ini

ditekan, agen akan meminta data kuliah yang akan dibatalkan. Tampilan untuk layanan tersebut ditunjukan paga Gambar 4.7.



Gambar 4.7 Pembatalan Kelas

Layanan lain yang diberikan oleh Agen Departemen adalah pencarian kelas kosong sebagai kelas pengganti untuk mata kuliah yang telah dibatalkan dan melakukan pendaftaran jadwal perkuliahan ke Agen Kelas yang mengelola kelas tersebut. GUI untuk pencarian kelas kosong diperlihatkan pada Gambar 4.8. Setelah menerima data kapasitas kelas yang diajukan oleh beberapa Agen Kelas, Agen Departemen akan mencarikan ruang kelas dengan kapasitas yang paling optimal untuk kegiatan perkuliahan tersebut.



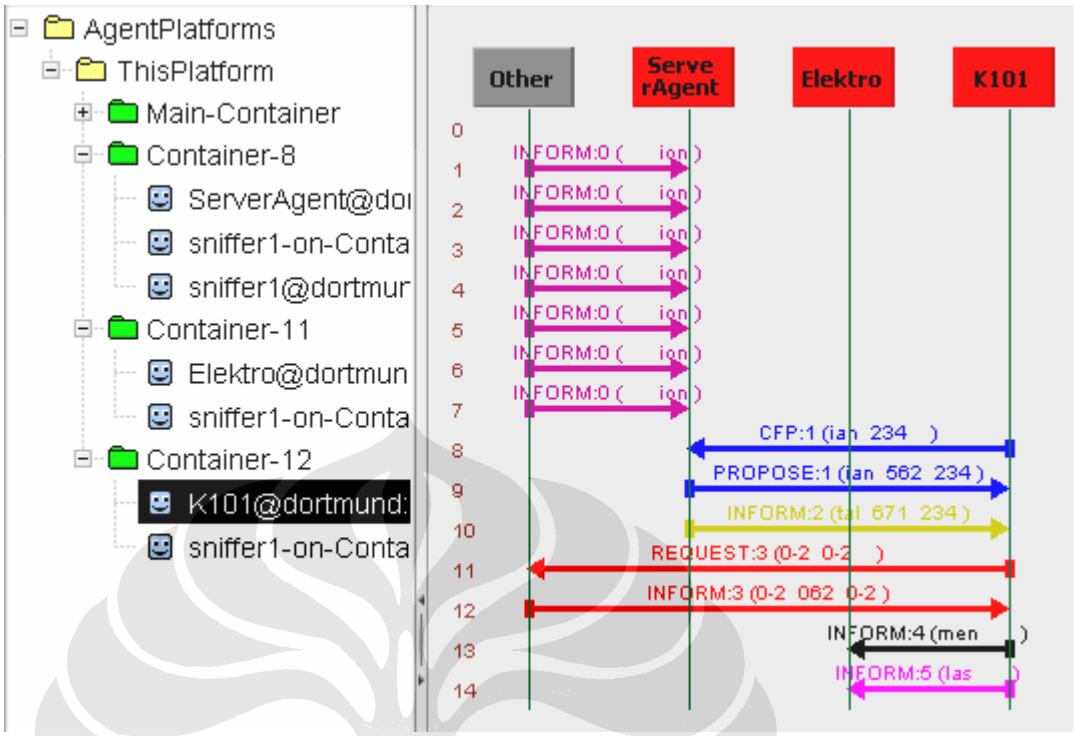
Gambar 4.8 Pencarian kelas kosong untuk jadwal kuliah yang telah dibatalkan

4.2 INTERAKSI SESAMA AGEN

Untuk menyelesaikan tugasnya, suatu agen akan berkomunikasi dengan agen yang lain. Komunikasi dilakukan dengan mengirimkan pesan. Untuk membedakan satu pesan dengan pesan yang lain, setiap satu jenis pesan mempunyai *template* yang unik. Untuk menganalisa komunikasi yang berlangsung antar sesama agen dalam menyelesaikan tugas yang diberikan kepadanya, digunakan Agen Sniffer, yang bertugas untuk memantau percakapan yang berlangsung antar sesama agen.

4.2.1 Komunikasi Antar Agen pada *Shift* Pertama

Ketika pertama kali dihidupkan, Agen Kelas membutuhkan Jadwal perkuliahan dari Agen Server. Dalam mengelola suatu perkuilahan, Agen Kelas juga membutuhkan komunikasi dengan Agen Departemen. Rangkaian komunikasi tersebut ditunjukan pada Gambar 4.9.



Gambar 4.9 Komunikasi agen-agen pada *sift* pertama, ketika kuliah baru dimulai

Komunikasi yang terjadi di atas dapat dijelaskan sebagai berikut:

- Pesan 1 menggunakan performatif INFORM, dikirim oleh AMS, yang isinya memberitahukan pada Agen Server, tentang kemunculan Agen Sniffer untuk Agen Server.
- Pesan 2 menggunakan performatif INFORM, dikirim oleh AMS, yang isinya memberitahukan pada Agen Server, tentang terbentuknya Container -11.
- Pesan 3 menggunakan performatif INFORM, dikirim oleh AMS, yang isinya memberitahukan pada Agen Server, tentang kemunculan Agen Departemen, yang hidup di Container -11
- Pesan 4 menggunakan performatif INFORM, dikirim oleh AMS, yang isinya memberitahukan pada Agen Server, tentang kemunculan Agen Sniffer yang hidup di Container -11, untuk mengawasi komunikasi Agen Departemen
- Pesan 5 menggunakan performatif INFORM, dikirim oleh AMS, yang isinya memberitahukan pada Agen Server, tentang terbentuknya Container -12.

- Pesan 6 menggunakan performatif INFORM, dikirim oleh AMS, yang isinya memberitahukan pada Agen Server, tentang kemunculan Agen Kelas, yang hidup di Container -12
- Pesan 7 menggunakan performatif INFORM, dikirim oleh AMS, yang isinya memberitahukan pada Agen Server, tentang kemunculan Agen Sniffer yang hidup di Container -12, untuk mengawasi komunikasi Agen Kelas.
- Pesan 8 menggunakan performatif CFP, dikirim oleh Agen Kelas, yang isinya meminta jadwal kuliah yang dilakukan di kelas yang dikelola oleh Agen Kelas bersangkutan.
- Pesan 9 menggunakan performatif PROPOSE, dikirim oleh Agen Server ke Agen Kelas. Pada pesan ini dilampirkan objek dari kelas Kegiatan, yang isinya adalah jadwal kegiatan perkuliahan selama satu hari, untuk dikelola oleh Agen Kelas. Isi pesan ini tidak dapat dibaca dengan menggunakan Agen Sniffer, karena objek tersebut dikirim dengan menggunakan *Java serialization*.
- Pesan 10 menggunakan performatif INFORM, dikirim oleh Agen Server ke Agen Kelas, yang isinya tentang keterangan jadwal kegiatan perkuliahan di kelas yang bersangkutan selama 1 minggu, keterangan kuliah yang dibatalkan, dan keterangan keterangan jadwal pengganti kuliah yang dibatalkan. Data yang dilampirkan pada pesan ini adalah data dengan format `Hashtable`, dengan menggunakan *java serialization*.
- Pesan 11 menggunakan performatif REQUEST, dikirim oleh Agen Kelas ke Agen DF, yang tujuannya untuk mendaftarkan layanan kelas tersebut ke Agen DF, sehingga agen lain dapat melihat layanan (kapasitas kelas) yang dimiliki Agen Kelas.
- Pesan 12 menggunakan performatif INFORM, dikirim oleh Agen DF, yang berisi informasi bahwa proses pendaftaran Agen Kelas telah berhasil
- Pesan 13 menggunakan performatif INFORM, dikirim oleh Agen Kelas ke Agen Departemen, yang isinya menginformasikan ke Agen Departemen, bahwa Agen Kelas ini sedang mengelola kegiatan perkuliahan dari departemen yang bersangkutan.

- Pesan 14 menggunakan performatif INFORM, dikirim oleh Agen Kelas ke Agen Departemen, yang berisi jumlah mahasiswa yang telah datang, dan apakah dosen yang mengajar telah masuk ke kelas.

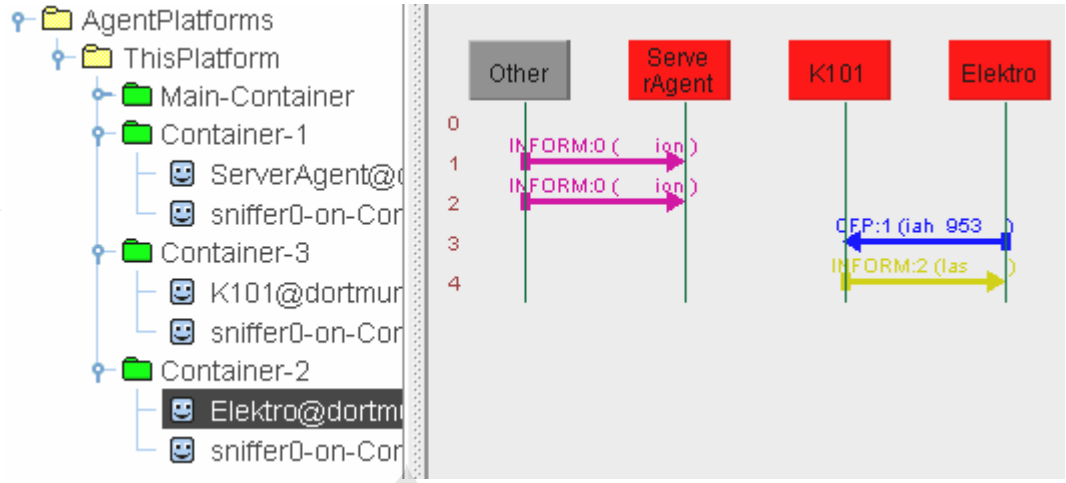
Dari komunikasi yang dilakukan oleh agen-agen di atas, terutama pada pesan ke sembilan dan ke sepuluh, dapat diketahui bahwa Agent Sniffer tidak dapat membaca objek yang dikirim dengan menggunakan *java serialization*. Kekurangan lain dari *java serialization* tidak *compatible* dengan lingkungan di luar Java. Namun, alasan digunakannya *Java serialization* adalah Hashtable dari Java adalah karena Ontology pada JADE tidak c*ompatible* dengan Hashtable di Java. Dalam aplikasi ini, banyak memanfaatkan Hashtable, karena kemudahan yang ditawarkan dalam pengambilan data yang tersimpan di dalamnya, yaitu dengan menggunakan *key*.

4.2.2 Komunikasi Antar Agen untuk *Polling* Data

Komunikasi lain yang terjadi antara agen, yang patut menjadi perhatian adalah komunikasi untuk *polling* data kegiatan perkuliahan yang sedang berlangsung, yang dilakukan oleh Agen Departemen. Alur pengiriman pesan yang terjadi, yang berhasil ditangkap oleh Agen Sniffer, diperlihatkan pada Gambar 4.10.

Komunikasi yang terjadi pada gambar di atas dapat dijelaskan sebagai berikut

- Pesan 1 dan 2, sama dengan penjelasan sebelumnya , adalah pesan yang dikirim oleh AMS ke Agen Server, untuk memberitahukan munculnya Agen Sniffer yang bertugas mengawasi Agen Kelas dan Agen Departemen.

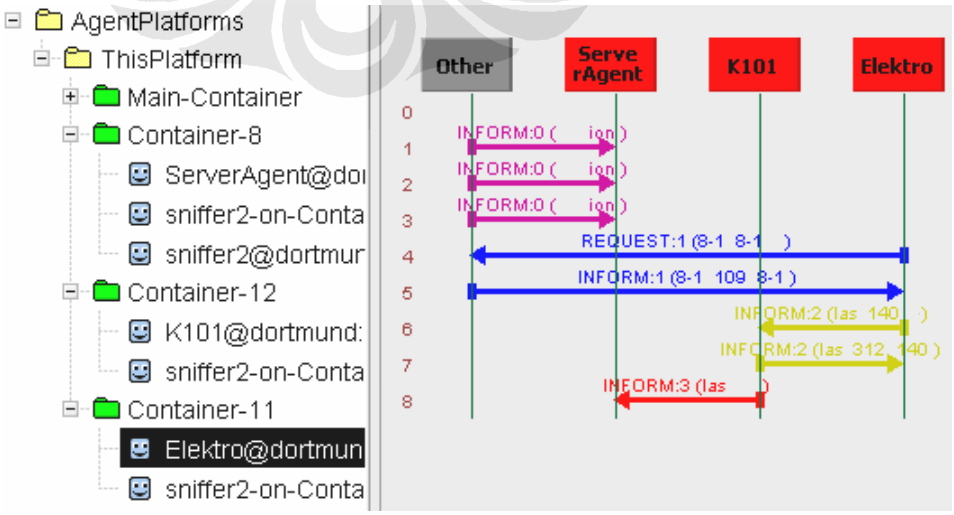


Gambar 4.10 Komunikasi antar agen pada saat Agen Departemen melakukan *polling* data absensi

- Pesan 3 menggunakanperformatif CFP, dikirim oleh Agen Departemen ke Agen Kelas, untuk meminta Agen Kelas mengirimkan data absensi perkuliahan yang sedang berlangsung ke Agen Departemen
- Pesan 4 menggunkan performatif INFORM, dikirim oleh Agen Kelas ke Agen Departemen, yang isinya berupa Hashatble tentang data absensi perkuliahan yang sedang dikelola oleh Agen Kelas.

4.2.3 Komunikasi Antar Agen untuk Pembatalan Jadwal Perkuliahan

Komunikasi lain yang diperhatikan adalah komunikasi antar agen untuk pembatalan kegiatan perkuliahan. Alur pengiriman pesan yang terjadi, yang berhasil ditangkap oleh Agent Sniffer ditunjukkan pada Gambar 4.11.



Gambar 4.11 Komunikasi antar agen pada saat Agen Departemen melakukan pembatalan jadwal perkuliahan

Komunikasi yang terjadi pada gambar di atas dapat dijelaskan sebagai berikut:

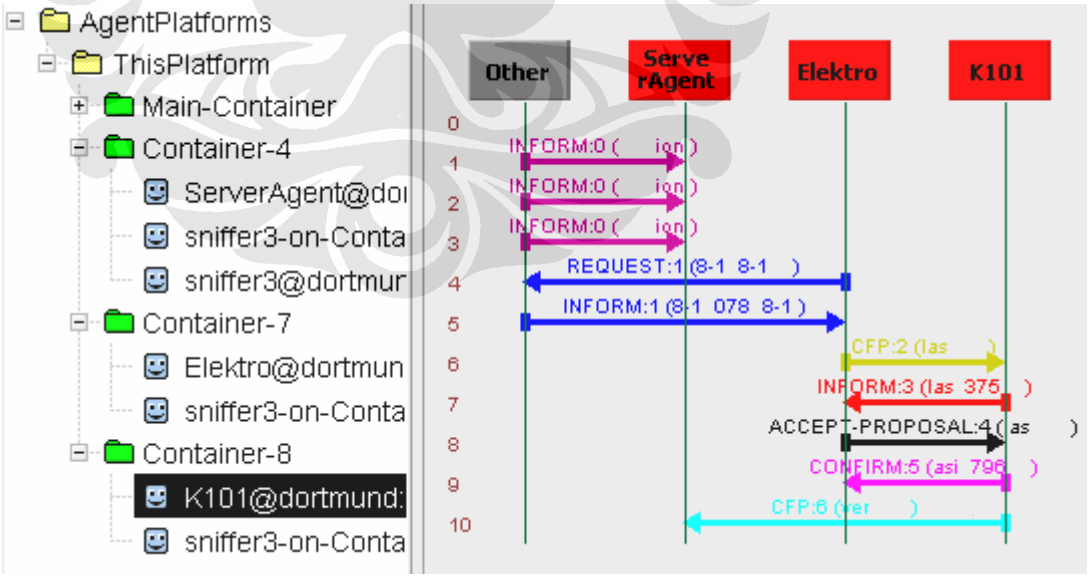
- Pesan 1, 2, dan 3 adalah pesan yang dikirim oleh AMS ke Agen Server, untuk memberitahukan munculnya Agen Sniffer untuk Agen Kelas dan Agen Departemen.
- Pesan 4 menggunakan performatif REQUEST, adalah pesan yang dikirim oleh Agen Departemen ke DF, yang tujuannya mencari AID Agen Kelas yang mengelola kuliah dari departemen yang dikelola oleh Agent Departemen.
- Pesan 5 menggunakan performatif INFORM, adalah pesan balasan yang dikirimkan oleh Agen DF ke Agen Departemen tentang AID Agen Kelas yang mengelola kuliah dari departemen bersangkutan.
- Pesan 6 menggunakan performatif INFORM, dikirimoleh Agen Departemen ke Agen Kelas, memberitahukan jadwal kuliah yang akan dibatalkan.
- Pesan 7 menggunakan performatif INFORM, dikirim oleh Agen Kelas ke Agen Departemen, menginformasikan apakah pembatalan kelas telah berhasil atau tidak
- Pesan 8 menggunakan performatif INFORM, dikirim oleh Agen Kelas ke Agen Server, menginformasikan data-data jadwal kelas yang akan dibatalkan.

4.2.4 Komunikasi Untuk Pencarian Kelas Kosong dan Pendaftaran Jadwal Kuliah

Agen Departemen dapat mencari kelas dengan kapasitas yang sesuai dan melakukan pendaftaran jadwal kuliah di ke Agen Kelas yang mengelola kelas tersebut. Dalam simulasi ini, hanya ditampilkan tiga agen, yaitu Agen Server, Agen Kelas, dan Agen Departemen. Alur pengiriman pesan yang terjadi, yang berhasil ditangkap oleh Agent Sniffer adalah seperti ditunjukan pada Gambar 4.12.

Komunikasi tersebut dapat dijelaskan sebagai berikut:

- Pesan 1, 2, dan 3 adalah pesan dengan performatif INFORM, dari AMS ke Agen Server, yang menginformasikan kemunculan Agen Sniffer untuk Agen Server, Kelas, dan Agen Departemen.
- Pesan 4 dengan performatif REQUEST, adalah pesan dari Agen Departemen ke Agen DF untuk mencari AID Agen Kelas yang mempunyai kapasitas kelas yang dapat menampung semua mahasiswa untuk kuliah yang akan diadakan.
- Pesan 5 dengan performatif INFORM, adalah pesan dari Agen DF ke Agen Departemen, mengenai AID dari Agen Kelas yang memiliki kapasitas yang cukukp untuk menyelengarakan kuliah pengganti.
- Pesan 6 dengan performatif CFP, dikirim oleh Agen Departemen ke Agen Kelas, yang isinya meminta kesedia Agen Kelas untuk mengirimkan kapasitas kelas yang dikelolanya.
- Pesan 7 dengan performatif INFORM, dikirim oleh Agen Kelas ke Agen Departemen, berisikan informasi kapasitas kelas yang dikelola Agen kelas tersebut.



Gambar 4.12 Komunikasi antar Agen untuk pencarian kelas jadwal kosong dan registrasi jadwal kuliah ke Agen Kelas yang mengelola kelas tersebut

- Pesan 8 dengan performatif ACCEPT-PROPOSAL, dikirim oleh Agen Departemen ke Agen Kelas yang dianggap memiliki kapasitas paling

optimal untuk melaksanakan kuliah pengganti. Pesan ini dikirim apabila dosen yang melakukan pencarian kelas kosong, setuju untuk mendaftarkan jadwal kuliahnya ke Agen Kelas tersebut.

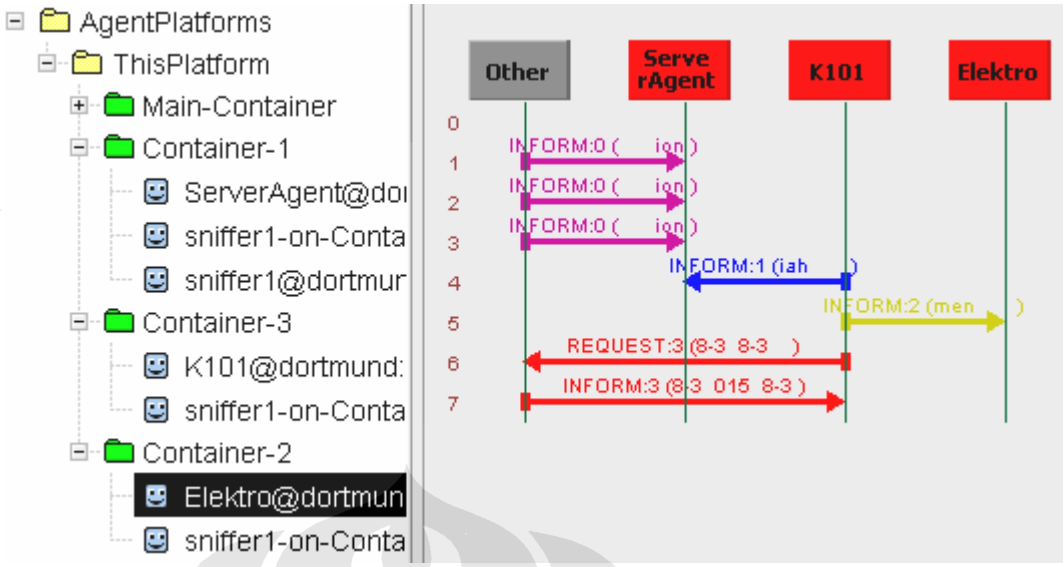
- Pesan 9 dengan performatif CONFIRM, dikirim oleh Agen Kelas ke Agen Departemen, yang isinya menginformasikan bahwa kelas telah berhasil direservasi.
- Pesan 10 dengan performatif CFP, dikirimkan oleh Agen Kelas ke Agen Server. Isi pesan ini adalah pemberitahuan jadwal perkuliahan untuk kelas pengganti yang akan dilaksanakan di kelas yang dikelola oleh Agen Kelas pengirim pesan.

4.2.5 Komunikasi Antar Agen untuk Pengiriman Data Absensi yang Telah Diisi

Tugas lain yang dilakukan oleh Agen Kelas adalah mengirimkan data absens peserta kuliah yang dilaksanakan di kelas yang dikelola oleh agen tersebut. Alur pengiriman pesan yang terjadi, yang berhasil ditangkap oleh Agen Sniffer adalah seperti ditunjukan pada Gambar 29.

Komunikasi tersebut dapat dijelaskan sebagai berikut:

- Pesan 1 dan 2 adalah pesan dari AMS ke Agen Server, tentang munculnya Agen Snifer.
- Pesan 3 dengan performatif INFORM, dikirim oleh Agen Kelas ke Agen Server. Di dalam pesan ini terdapat informasi absensi mahasiswa dan dosen, untuk kuliah yangbaru diselesaikan.
- Pesan 4 dengan performatif INFORM, adalah pesan yang dikirimkan oleh Agen Kelas ke Agen Departemen, untuk menyatakan kulah yang telah dikelolanya, telah selesai.
- Pesan 5 dengan performatif REQUEST, adalah pesan yang dikirim oleh Agen Kelas ke DF, untuk mencari AID Agen Departemen untuk kuliah berikutnya.



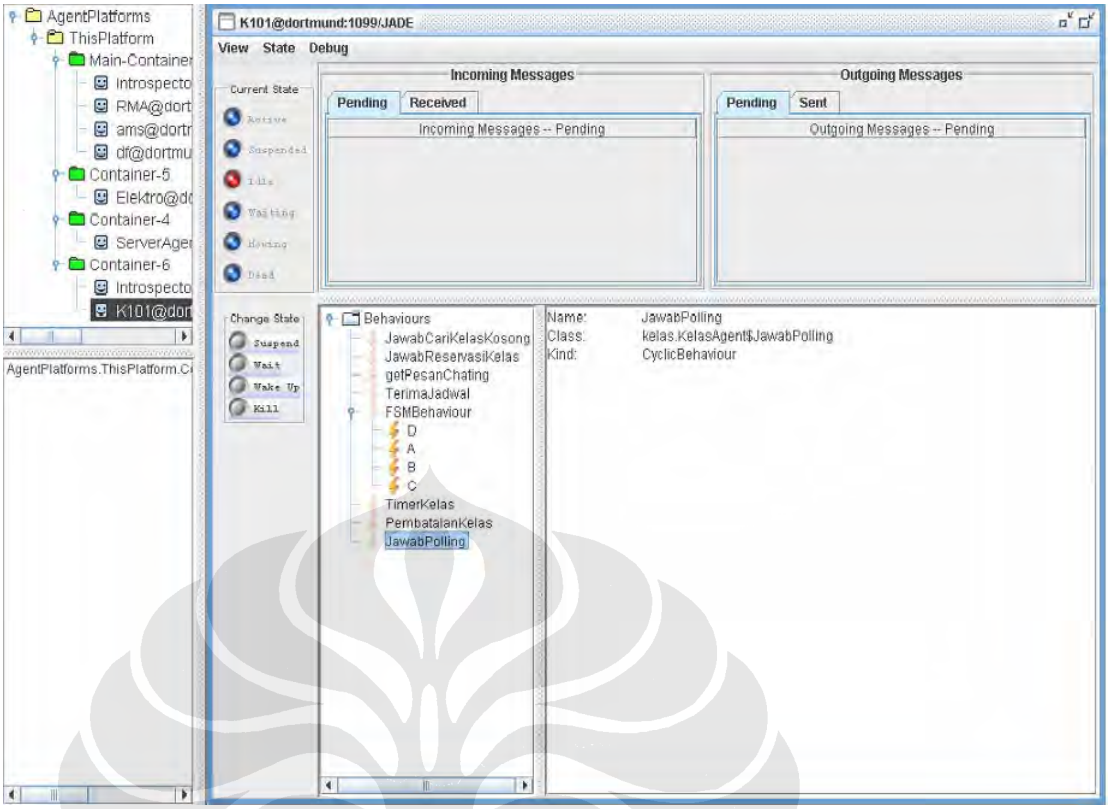
Gambar 4.13 Komunikasi antar agen yang terjadi pada saat pengiriman data absensi yang telah diisi dari Agen Kelas ke Agen Server

4.3 BEHAVIOUR AGEN KELAS DAN AGEN DEPARTEMEN

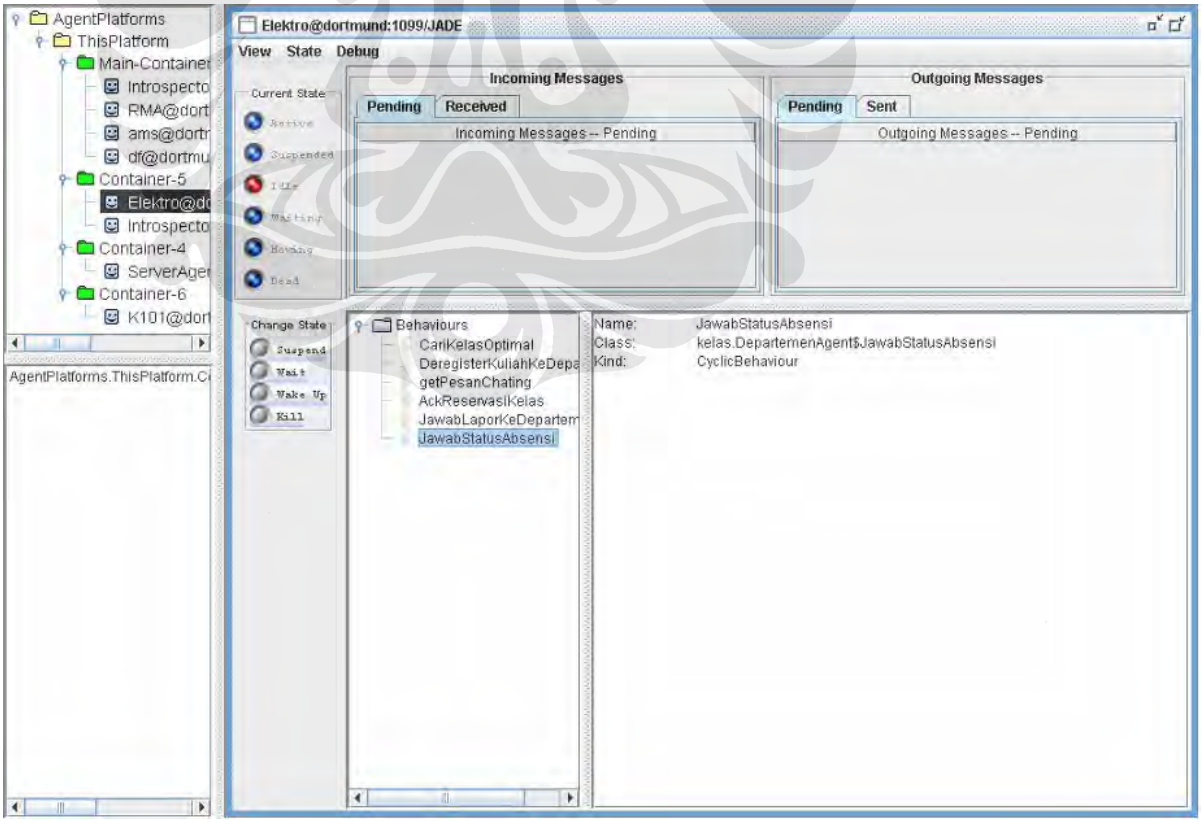
Behaviour adalah kemampuan yang dapat ditambahkan ke suatu agen untuk menyelesaikan tugas tertentu dan dapat juga dihilangkan ketika pekerjaan tersebut telah selesai. Dari gambar *behaviour* yang terdapat di Agen Kelas yang diambil oleh Agen Introspector, dapat dilihat *behaviour* apa saja yang sedang aktif dan *behaviouor* apa saja yang sedang di *block.* Pada saat data ini diambil, Agen Kelas sedang mengelola kegiatan perkuliahan, sehingga *behaviour* yang aktif adalah FSMBehaviour, yang terdiri dari empat *behaviour* A, B, C, dan D, yang merupakan berbagai *state* yang terdapat di dalam siklus pengaturan perkuliahan pada Agen Kelas.

Untuk Agen Departemen, semua *behaviour* sedang di-*block*. Behaviour ini akan aktif, jika ada pesan yang datang yang sesuai dengan *template* pesan untuk *behaviour* tersebut, dan atau karena ada tugas yang diberikan oleh pengguna, yaitu dosen.

Behaviour yang masih berguna, namun sudah menyelesaikan pekerjaannya, harus di block. Contoh *behaviour* yang memilki karakteristik ini adalah *CyclicBehaviour* yang biasanya digunakan untuk menanti pesan masuk. Jika *behaviour* ini tidak di-*block* dalam kondisi tidak ada tugas yang dikerjakan, maka ia akan terus melakukan *looping* dan akan menghabiskan sumber daya komputasi.



Gambar 4.14 *Behaviour-behaviour* pada Agen Kelas, ketika sedang menangani perkuliahan



Gambar 4.15 *Behaviour-behaviour* pada Agen Departemen

4.4 PENGGUNAAN MEMORI

Untuk mengetahui unjuk kerja dari aplikasi ini, dilakukan beberapa pengambilan data, yaitu jumlah penggunaan memori oleh Agen Kelas dan Agen Departemen, serta lamanya waktu yang dibutuhkan oleh Agen Kelas untuk menemukan Agen Server. Simulasi ini dilakukan pada komputer dengan spesifikasi yang ditunjuk kan pada Tabel 1.

Untuk Agen Kelas, dari simulasi yang telah dilakukan, didapatkan data seperti yang ditunjukkan pada Tabel 2.

Dari jumlah peningkatan penggunaan memori pada komputer tempat hidup Agen Kelas, dapat dikatakan bahwa Agen Kelas tidak membutuhkan momori yang besar menjalankannya, sehingga dapat dijalankan dengan komputer yang mempunyai sumber daya komputasi yang terbatas.

Tabel 1Spesifikasi Komputer untuk Pengujian

Parameter	Keterangan
Sistem Operasi	Microsoft Windows 2000 Professional
CPU	AMD Athlon XP (1666 MHz (12.5 x 133) 2000+)
Hard Disk	Maxtor 6E040L0 (40 GB, 7200 RPM, Ultra-ATA/133)
Mother Board	MSI KT4V (MS-6721)
Memory	768 MB, (DDR DDRAM)
Network Adapter	3Com EtherLink XL 10/100 PCI For Complete PC Management NIC (3C905C-TX)

Tabel 2 Peningkatan penggunaan memori pada komputer tempat Agen Kelas hidup

NO (n)	Memori Sebelum Agen Hidup (A) (Satuan = Kbyte)	Memori Setelah Agen Hidup (B) (Satuan = Kbyte)	Peningkatan Penggunaan Memori (X=B-A) (Satuan = Kbyte)	$(X-X')^2$
1	180928	219516	38588	50176
2	180992	219712	38720	8464
3	227584	265788	38204	369664
4	234880	272160	37280	2347024
5	267592	305360	37768	1089936
6	185024	226208	41184	5626384
7	366836	405508	38672	19600
8	253781	293532	39751	881721
9	201431	241634	40203	1934881
10	192371	230121	37750	1127844
Total			388120	13455694
				X' = 38812 Kbyte

Standar deviasi untuk jumlah peningkatan penggunaan memori pada saat Agen Kelas dihidupkan adalah:

$$S = \sqrt{\frac{\sum (X - X')^2}{n-1}} = 1222,73 KByte$$

Standar deviasi yang didapat, menunjukan rata-rata penyimpangan peningkatan penggunaan memori, yang nilainya tidak terlalu besar, sehingga dapat disimpulkan bahwa aplikasi ini telah mendekati unjuk kerja yang stabil untuk semua jenis komputer yang dicoba.

Untuk Agen Departemen, dari simulasi yang telah dilakukan, didapatkan data seperti ditunjukan pada Tabel 3.

Dari jumlah peningkatan penggunaan memori pada komputer tempat hidup Agen Departemen, dapat dikatakan bahwa Agen Departemen tidak membutuhkan momori yang besar untuk menjalankannya, sehigngga dapat dijalankan dengan komputer yang mempunyai sumber daya komputasi yang terbatas.

Tabel 3 Peningkatan penggunaan memori pada komputer tempat Agen Departemen hidup

NO (n)	Memori Sebelum Agen Hidup (A) (Satuan = Kbyte)	Memori Setelah Agen Hidup (B) (Satuan = Kbyte)	Peningkatan Penggunaan Memori (X=B-A) (Satuan = Kbyte)	(X-X')²
1	239440	276400	36960	1895027,56
2	239740	276724	36984	1829526,76
3	180836	220068	39232	801741,16
4	265437	303397	37960	141827,56
5	253641	292291	38650	98219,56
6	185000	220023	35023	10979944,96
7	234880	274880	40000	2766899,56
8	365837	406073	40236	3607720,36
9	200421	238430	38009	107321,76
10	200192	240504	40312	3902205,16
Total			383366	26130434,4
				X' = 38336,6 Kbyte

Standar deviasi untuk jumlah peningkatan penggunaan memori pada saat Agen Departemen dihidupkan adalah:

$$S = \sqrt{\frac{\sum (X - X')^2}{n-1}} = 1703,93 Kbyte$$

Sama seperti standar deviasi yang didapat pada pengujian Agen Kelas, nilai standar deviasi untuk Agen Departemen, juga menunjukan rata-rata perbedaan dalam pengingkatan jumlah penggunaan memori yang tidak terlalu besar dibandingkan dengan peningkatan penggunaan memori itu sendiri. Dari data ini, dapat dikatakan bahwa Agen Departemen telah hampir stabil dalam penggunaan sumber daya komputasi, walaupun dijalankan pada komputer yang berbeda.

Lamanya waktu yang dibutuhkan oleh Agen Kelas untuk mengakses Agen Server, ditunjukan pada Tabel 4.

Dari data waktu yang dibutuhkan untuk mengakses Agen Server, dapat diketahui bahwa dalam sistem ini dibutuhkan waktu yang tidak terlalu lama untuk berkomunikasi antara satu agen dengan yang lain.

Tabel 4 Waktu yang dibutuhkan Agen Kelas untuk Mencari Agen Server

NO (n)	Waktu Cari Server (X) (Satuan = millidetik)	$(X-X')^2$
1	32	29,16
2	31	40,96
3	31	40,96
4	15	501,76
5	62	605,16
6	47	92,16
7	43	31,36
8	41	12,96
9	32	29,16
10	40	6,76
Total	374	1390,4
		X' = 37,4

Standar deviasi untuk waktu yang dibutuhkan oleh Agen Kelas dalam mencari AID Agen Server adalah:

$$S = \sqrt{\frac{\sum (X - X')^2}{n-1}} = 12,43 \text{ milidetik}$$

Standar deviasi yang didapat dari data waktu yang dibutuhkan Agen kelas untuk mencari AID Agen Server, menunjukan komunikasi antara agen masih bersifat variatif. Artinya, masih banyak variabel lain yang dapat mempengaruhi waktu lamanya waktu komunikasi agen, seperti kondisi jaringan, dan kepadatan lalu-lintas data di jaringan.

BAB V
KESIMPULAN

Setelah merancang, memogram, dan melakukan pengujian hasil yang didapat dari data yang diperoleh dari program yang telah dijalankan, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Sistem Manajemen Kelas dengan berbasis agen, dapat membuat setiap *host* menjadi bersifat otonom.
2. Agen yang berbasis JADE, sesuai dengan paradigmanya, dapat menyelesaikan permasalahan secara independen dan atau dengan berkomunikasi dengan agen-agen yang lain.
3. *Ontology* pada JADE versi 3.5, tidak kompatibel dengan Hashtabel pada java. Untuk itu digunakan *Java serialization* dalam mengirimkan informasi berupa objek, antar sesama agen.
4. Agen berbasis JADE dapat hidup di komputer dengan kapasitas memori yang minim, dengan penggunaan memori antara 35000 KByte – 40000 KByte.
5. Proses pencarian Agen Server oleh Agen Kelas, membutuhkan waktu yang singkat, antara 30 milidetik sampai 61 milidetik.

DAFTAR ACUAN

[1] Bellifemine. Fabio, Caire. Giovanni, Greenwood. Dominic, "Developing Multi-Agent System with JADE", John Wiley and Sons, Ltd, West Sussex, England, 2007.

[2] Nikraz, Magid, et al. *A Methodology for the Analysis and Design of Multi-Agent System Using JADE*,Telecom Italia Lab, 2006

[3] (2008). *About the Java Technology*. Diakses 5 Juni 2008. http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

[4] Jon Byous (2003). Java technology: The Early Years. Diakses 11 Mei 2008, dari Sun Developer Network. http://java.sun.com/features/1998/05/birthday.html diakes 11 Mei 2008

[5] Feizabadi, Shahrooz (1996), *History of Java*. Diakses 5 Juni 2008 http://ei.cs.vt.edu/~wwwbtb/book/chap1/java_hist.html

[6] Tamma, Valentina, et al. Ontologies for Agents: Theory and Experiences. Birkhauser Verlag. Basel. 2005

[7] (2008). EJIP Eclipse Jade Integration. diakses 12 Mei 2008. http://b.hatena.ne.jp/entry/4074991

[8] Giorno, Paolo dan Brian Henderson, Seller. *Agent –Oriented Methodologies*. Idea Group Publishing. Hersey. 2005.

[9] Bergenti, Frederico, et al. *Methodologies and Software Engineering for Agent System*.Kluwer Academic Publisher. Boston. 2004

[10] Dix, Mehdi Dastani Jurgen dan Segrouchni, Amal El Fallah. *Programming Multi Agent System*. Springer. Melbourne. 2003

[11] Dinkloh, Martin dan Nismis, Jens. *A Tool for Integrated Design and Implementationof Conversation in MultyAgent System*. Universitat Karlsrure. Karlsrure.

[12] Bellifemine, Fabio et al. *Developing Multi-agent System wih JADE*. University of Parma. Parma.

[13] Sheremetov, Leonid et al.Optimization Algorithm for Dynamic Multi-agent Job Routing.

DAFTAR PUSTAKA

Bellifemine, Fabio, et al. *Developing Multi Agent System with JADE*. John Wiley & Sons, Ltd. New Jersey.2007

Bellifemine, Fabio, et al. *JADE Programmer's Guide*. Diakses 5 Juni 2008. http://jade.tilab.com/doc/programmersguide.pdf

Bellifemine, Fabio, et al. *JADE Administrator's Guide*. Diakses 5 Juni 2008. http://jade.tilab.com/doc/administratorsguide.pdf

Byous, Jojn (2003), *Java Technology: The Early Years*. Diakses 5 Juni 2008. http://java.sun.com/features/1998/05/birthday.html

H.M. Deitel, *Java How to Program, Sixth Edition,* New Jersey, Pearson Education, 2005

Rouff, Cristopher A, et al. *Agent Technology from a Formal Perspective*. Springer. USA. 2006.

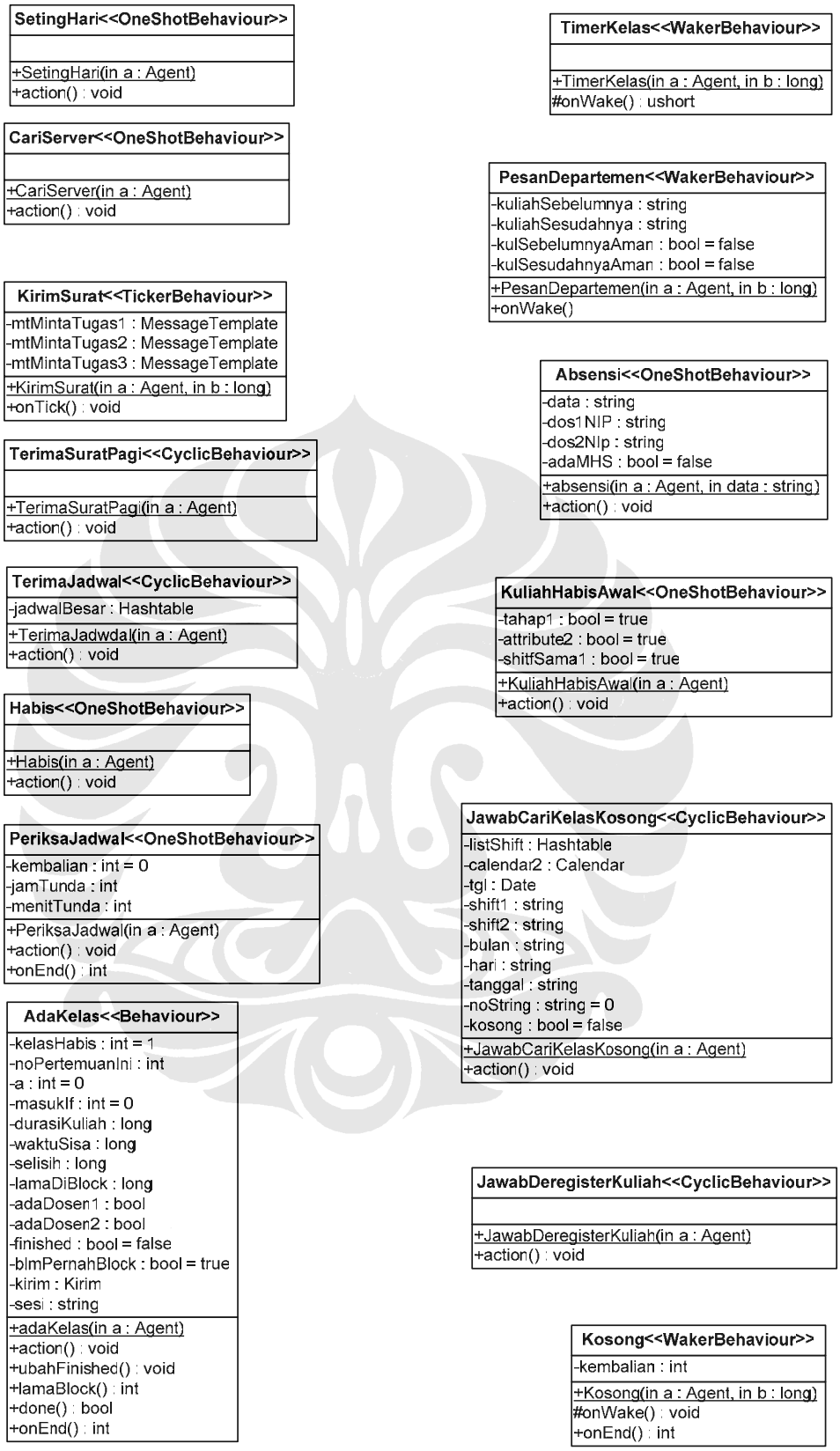
Unland, Rainer, et al. *Softwre Agent-Based Aplications, Platforms and Developments Kits*. Basel, Switzerland. 2005.

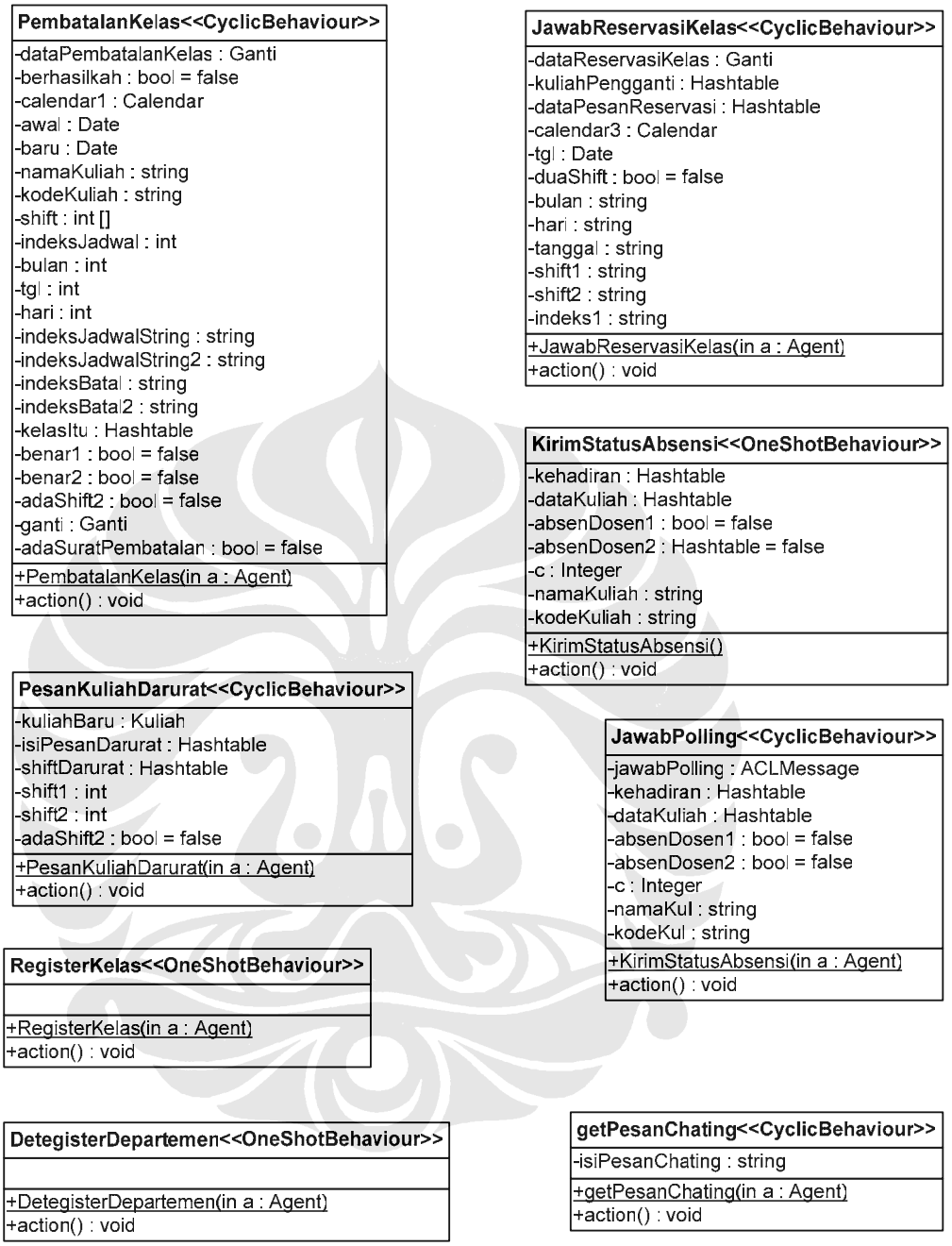
Lin, Fuhua Oscar. *Designing Distributed Learning Environment with Intelligent Software Agents*. Information Science Publishing. Hershey. 2005.

Dix, Mehdi Dastani Jurgen dan Segrouchni, Amal El Fallah. *Programming Multi Agent System*. Springer. Melbourne. 2003.

LAMPIRAN A

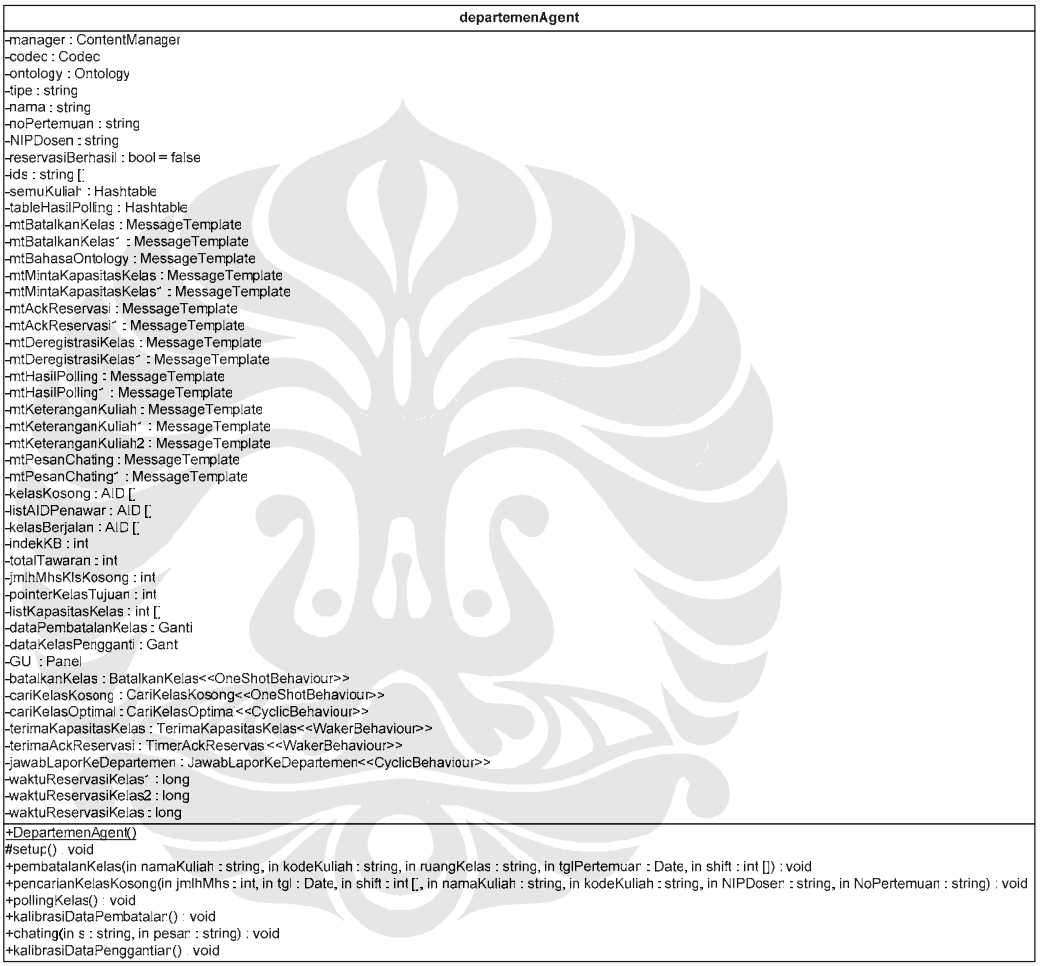
INFORMASI KELAS-KELAS YANG DIGUNAKAN UNTUK MEMBANGUN AGEN KELAS

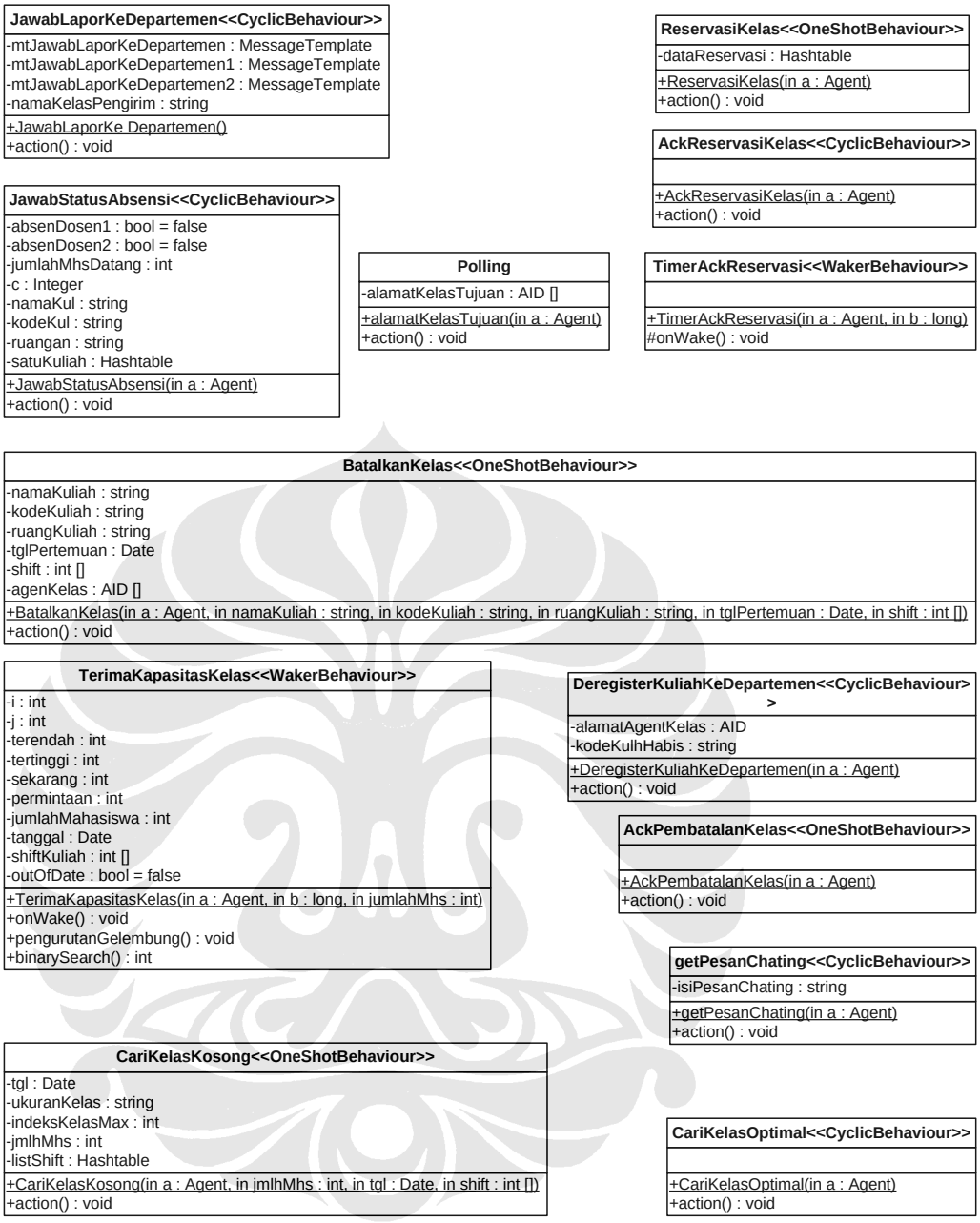

SetingHari<<OneShotBehaviour>>
+SetingHari(in a : Agent)
+action() : void
CariServer<<OneShotBehaviour>>
+CariServer(in a : Agent)
+action() : void
KirimSurat<<TickerBehaviour>>
-mtMintaTugas1 : MessageTemplate
-mtMintaTugas2 : MessageTemplate
-mtMintaTugas3 : MessageTemplate
+KirimSurat(in a : Agent, in b : long)
+onTick() : void
TerimaSuratPagi<<CyclicBehaviour>>
+TerimaSuratPagi(in a : Agent)
+action() : void
TerimaJadwal<<CyclicBehaviour>>
-jadwalBesar : Hashtable
+TerimaJadwdal(in a : Agent)
+action() : void
Habis<<OneShotBehaviour>>
+Habis(in a : Agent)
+action() : void
PeriksaJadwal<<OneShotBehaviour>>
-kembalian : int = 0
-jamTunda : int
-menitTunda : int
+PeriksaJadwal(in a : Agent)
+action() : void
+onEnd() : int
AdaKelas<<Behaviour>>
-kelasHabis : int = 1
-noPertemuanIni : int
-a : int = 0
-masukIf : int = 0
-durasiKuliah : long
-waktuSisa : long
-selisih : long
-lamaDiBlock : long
-adaDosen1 : bool
-adaDosen2 : bool
-finished : bool = false
-blmPernahBlock : bool = true
-kirim : Kirim
-sesi : string
+adaKelas(in a : Agent)
+action() : void
+ubahFinished() : void
+lamaBlock() : int
+done() : bool
+onEnd() : int
TimerKelas<<WakerBehaviour>>
+TimerKelas(in a : Agent, in b : long)
#onWake() : ushort
PesanDepartemen<<WakerBehaviour>>
-kuliahSebelumnya : string
-kuliahSesudahnya : string
-kulSebelumnyaAman : bool = false
-kulSesudahnyaAman : bool = false
+PesanDepartemen(in a : Agent, in b : long)
+onWake()
Absensi<<OneShotBehaviour>>
-data : string
-dos1NIP : string
-dos2NIp : string
-adaMHS : bool = false
+absensi(in a : Agent, in data : string)
+action() : void
KuliahHabisAwal<<OneShotBehaviour>>
-tahap1 : bool = true
-attribute2 : bool = true
-shitfSama1 : bool = true
+KuliahHabisAwal(in a : Agent)
+action() : void
JawabCariKelasKosong<<CyclicBehaviour>>
-listShift : Hashtable
-calendar2 : Calendar
-tgl : Date
-shift1 : string
-shift2 : string
-bulan : string
-hari : string
-tanggal : string
-noString : string = 0
-kosong : bool = false
+JawabCariKelasKosong(in a : Agent)
+action() : void
JawabDeregisterKuliah<<CyclicBehaviour>>
+JawabDeregisterKuliah(in a : Agent)
+action() : void
Kosong<<WakerBehaviour>>
-kembalian : int
+Kosong(in a : Agent, in b : long)
#onWake() : void
+onEnd() : int


PembatalanKelas<<CyclicBehaviour>>
-dataPembatalanKelas : Ganti
-berhasilkah : bool = false
-calendar1 : Calendar
-awal : Date
-baru : Date
-namaKuliah : string
-kodeKuliah : string
-shift : int []
-indeksJadwal : int
-bulan : int
-tgl : int
-hari : int
-indeksJadwalString : string
-indeksJadwalString2 : string
-indeksBatal : string
-indeksBatal2 : string
-kelasItu : Hashtable
-benar1 : bool = false
-benar2 : bool = false
-adaShift2 : bool = false
-ganti : Ganti
-adaSuratPembatalan : bool = false
+PembatalanKelas(in a : Agent)
+action() : void
JawabReservasiKelas<<CyclicBehaviour>>
-dataReservasiKelas : Ganti
-kuliahPengganti : Hashtable
-dataPesanReservasi : Hashtable
-calendar3 : Calendar
-tgl : Date
-duaShift : bool = false
-bulan : string
-hari : string
-tanggal : string
-shift1 : string
-shift2 : string
-indeks1 : string
+JawabReservasiKelas(in a : Agent)
+action() : void
KirimStatusAbsensi<<OneShotBehaviour>>
-kehadiran : Hashtable
-dataKuliah : Hashtable
-absenDosen1 : bool = false
-absenDosen2 : Hashtable = false
-c : Integer
-namaKuliah : string
-kodeKuliah : string
+KirimStatusAbsensi()
+action() : void
PesanKuliahDarurat<<CyclicBehaviour>>
-kuliahBaru : Kuliah
-isiPesanDarurat : Hashtable
-shiftDarurat : Hashtable
-shift1 : int
-shift2 : int
-adaShift2 : bool = false
+PesanKuliahDarurat(in a : Agent)
+action() : void
JawabPolling<<CyclicBehaviour>>
-jawabPolling : ACLMessage
-kehadiran : Hashtable
-dataKuliah : Hashtable
-absenDosen1 : bool = false
-absenDosen2 : bool = false
-c : Integer
-namaKul : string
-kodeKul : string
+KirimStatusAbsensi(in a : Agent)
+action() : void
RegisterKelas<<OneShotBehaviour>>
+RegisterKelas(in a : Agent)
+action() : void
DetegisterDepartemen<<OneShotBehaviour>>
+DetegisterDepartemen(in a : Agent)
+action() : void
getPesanChating<<CyclicBehaviour>>
-isiPesanChating : string
+getPesanChating(in a : Agent)
+action() : void

LAMPIRAN B

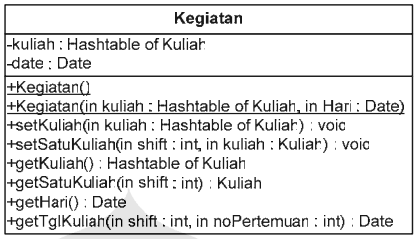
INFORMASI KELAS-KELAS YANG DIGUNAKAN UNTUK MEMBANGUN AGEN DEPARTEMEN

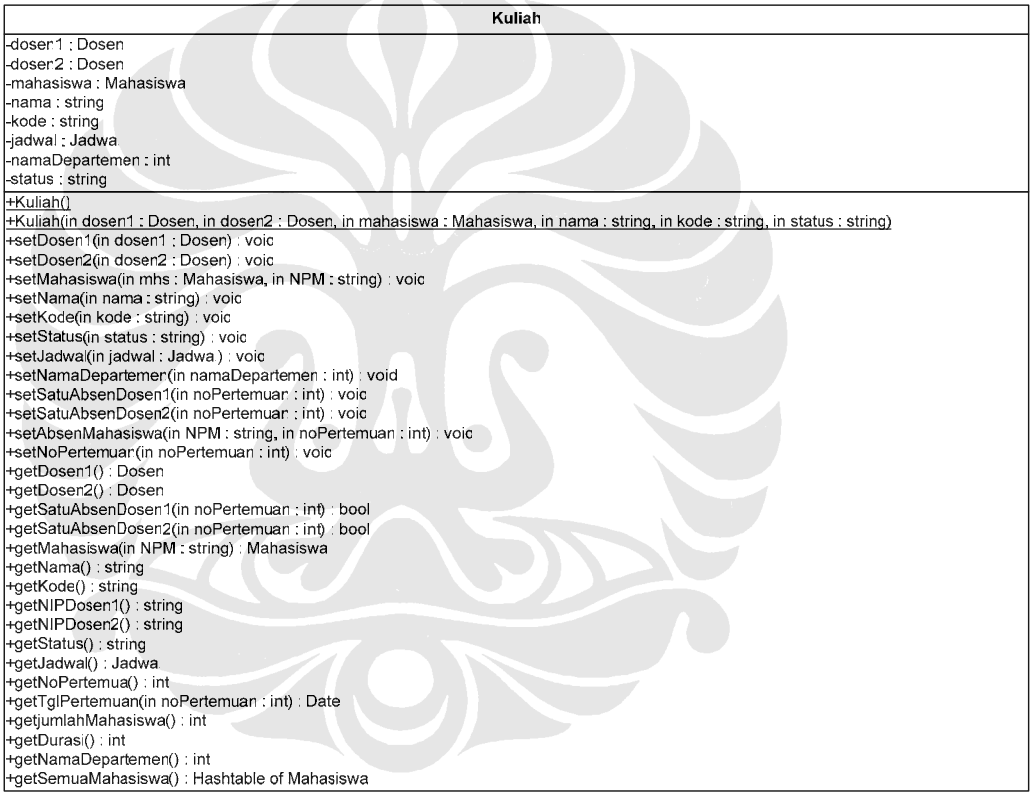


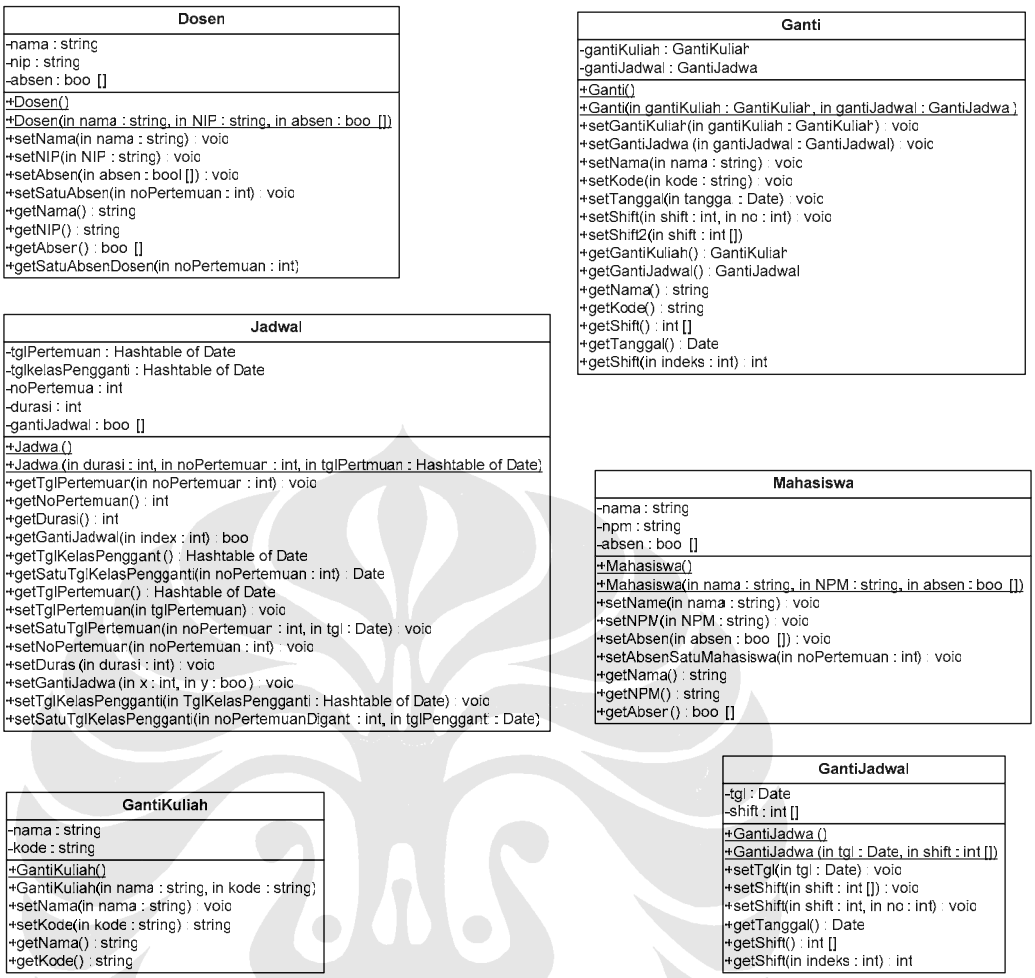

JawabLaporKeDepartemen<<CyclicBehaviour>>
-mtJawabLaporKeDepartemen : MessageTemplate
-mtJawabLaporKeDepartemen1 : MessageTemplate
-mtJawabLaporKeDepartemen2 : MessageTemplate
-namaKelasPengirim : string
+JawabLaporKe Departemen()
+action() : void
ReservasiKelas<<OneShotBehaviour>>
-dataReservasi : Hashtable
+ReservasiKelas(in a : Agent)
+action() : void
AckReservasiKelas<<CyclicBehaviour>>
+AckReservasiKelas(in a : Agent)
+action() : void
JawabStatusAbsensi<<CyclicBehaviour>>
-absenDosen1 : bool = false
-absenDosen2 : bool = false
-jumlahMhsDatang : int
-c : Integer
-namaKul : string
-kodeKul : string
-ruangan : string
-satuKuliah : Hashtable
+JawabStatusAbsensi(in a : Agent)
+action() : void
Polling
-alamatKelasTujuan : AID []
+alamatKelasTujuan(in a : Agent)
+action() : void
TimerAckReservasi<<WakerBehaviour>>
+TimerAckReservasi(in a : Agent, in b : long)
#onWake() : void
BatalkanKelas<<OneShotBehaviour>>
-namaKuliah : string
-kodeKuliah : string
-ruangKuliah : string
-tglPertemuan : Date
-shift : int []
-agenKelas : AID []
+BatalkanKelas(in a : Agent, in namaKuliah : string, in kodeKuliah : string, in ruangKuliah : string, in tglPertemuan : Date, in shift : int [])
+action() : void
TerimaKapasitasKelas<<WakerBehaviour>>
-i : int
-j : int
-terendah : int
-tertinggi : int
-sekarang : int
-permintaan : int
-jumlahMahasiswa : int
-tanggal : Date
-shiftKuliah : int []
-outOfDate : bool = false
+TerimaKapasitasKelas(in a : Agent, in b : long, in jumlahMhs : int)
+onWake() : void
+pengurutanGelembung() : void
+binarySearch() : int
DeregisterKuliahKeDepartemen<<CyclicBehaviour>>
-alamatAgentKelas : AID
-kodeKulhHabis : string
+DeregisterKuliahKeDepartemen(in a : Agent)
+action() : void
AckPembatalanKelas<<OneShotBehaviour>>
+AckPembatalanKelas(in a : Agent)
+action() : void
getPesanChating<<CyclicBehaviour>>
-isiPesanChating : string
+getPesanChating(in a : Agent)
+action() : void
CariKelasKosong<<OneShotBehaviour>>
-tgl : Date
-ukuranKelas : string
-indeksKelasMax : int
-jmlhMhs : int
-listShift : Hashtable
+CariKelasKosong(in a : Agent, in jmlhMhs : int, in tgl : Date, in shift : int [])
+action() : void
CariKelasOptimal<<CyclicBehaviour>>
+CariKelasOptimal(in a : Agent)
+action() : void

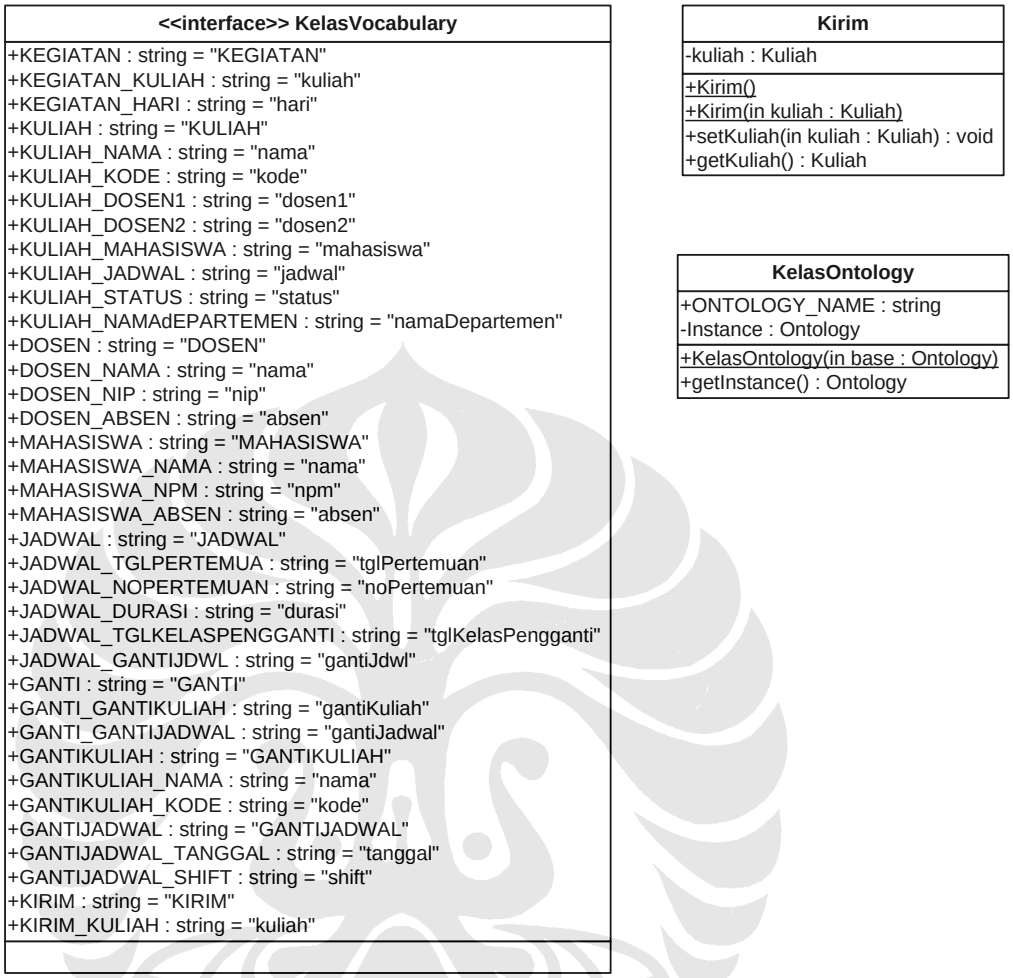
LAMPIRAN C

INFORMASI KELAS-KELAS YANG DIGUNAKAN UNTUK MEMBANGUN OBJEK JADWAL KULIAH






Dosen
-nama : string
-nip : string
-absen : boo []
+Dosen()
+Dosen(in nama : string, in NIP : string, in absen : boo [])
+setNama(in nama : string) : voic
+setNIP(in NIP : string) : voic
+setAbsen(in absen : bool []) : voic
+setSatuAbsen(in noPertemuan : int) : voic
+getNama() : string
+getNIP() : string
+getAbser() : boo []
+getSatuAbsenDosen(in noPertemuan : int)
Ganti
-gantiKuliah : GantiKuliah
-gantiJadwal : GantiJadwa
+Ganti()
+Ganti(in gantiKuliah : GantiKuliah, in gantiJadwal : GantiJadwa)
+setGantiKuliah(in gantiKuliah : GantiKuliah) : voic
+setGantiJadwa (in gantiJadwal : GantiJadwal) : voic
+setNama(in nama : string) : voic
+setKode(in kode : string) : voic
+setTanggal(in tangga : Date) : voic
+setShift(in shift : int, in no : int) : voic
+setShift2(in shift : int [])
+getGantiKuliah() : GantiKuliah
+getGantiJadwal() : GantiJadwal
+getNama() : string
+getKode() : string
+getShift() : int []
+getTanggal() : Date
+getShift(in indeks : int) : int
Jadwal
-tglPertemuan : Hashtable of Date
-tglkelasPengganti : Hashtable of Date
-noPertemua : int
-durasi : int
-gantiJadwal : boo []
+Jadwa ()
+Jadwa (in durasi : int, in noPertemuar : int, in tglPertmuan : Hashtable of Date)
+getTglPertemuar(in noPertemuar : int) : voic
+getNoPertemuan() : int
+getDurasi() : int
+getGantiJadwal(in index : int) : boo
+getTglKelasPenggant() : Hashtable of Date
+getSatuTglKelasPengganti(in noPertemuan : int) : Date
+getTglPertemuar() : Hashtable of Date
+setTglPertemuan(in tglPertemuan) : voic
+setSatuTglPertemuan(in noPertemuar : int, in tgl : Date) : voic
+setNoPertemuar(in noPertemuan : int) : voic
+setDuras(in durasi : int) : voic
+setGantiJadwa (in x : int, in y : boo) : voic
+setTglKelasPengganti(in TglKelasPengganti : Hashtable of Date) : voic
+setSatuTglKelasPengganti(in noPertemuanDigant : int, in tglPenggant : Date)
Mahasiswa
-nama : string
-npm : string
-absen : boo []
+Mahasiswa()
+Mahasiswa(in nama : string, in NPM : string, in absen : boo [])
+setName(in nama : string) : voic
+setNPM(in NPM : string) : voic
+setAbsen(in absen : boo []) : voic
+setAbsenSatuMahasiswa(in noPertemuan : int) : voic
+getNama() : string
+getNPM() : string
+getAbser() : boo []
GantiKuliah
-nama : string
-kode : string
+GantiKuliah()
+GantiKuliah(in nama : string, in kode : string)
+setNama(in nama : string) : voic
+setKode(in kode : string) : string
+getNama() : string
+getKode() : string
GantiJadwal
-tgl : Date
-shift : int []
+GantiJadwa ()
+GantiJadwa (in tgl : Date, in shift : int [])
+setTgl(in tgl : Date) : voic
+setShift(in shift : int []) : voic
+setShift(in shift : int, in no : int) : voic
+getTanggal() : Date
+getShift() : int []
+getShift(in indeks : int) : int


<<interface>> KelasVocabulary
+KEGIATAN : string = "KEGIATAN"
+KEGIATAN_KULIAH : string = "kuliah"
+KEGIATAN_HARI : string = "hari"
+KULIAH : string = "KULIAH"
+KULIAH_NAMA : string = "nama"
+KULIAH_KODE : string = "kode"
+KULIAH_DOSEN1 : string = "dosen1"
+KULIAH_DOSEN2 : string = "dosen2"
+KULIAH_MAHASISWA : string = "mahasiswa"
+KULIAH_JADWAL : string = "jadwal"
+KULIAH_STATUS : string = "status"
+KULIAH_NAMAdEPARTEMEN : string = "namaDepartemen"
+DOSEN : string = "DOSEN"
+DOSEN_NAMA : string = "nama"
+DOSEN_NIP : string = "nip"
+DOSEN_ABSEN : string = "absen"
+MAHASISWA : string = "MAHASISWA"
+MAHASISWA_NAMA : string = "nama"
+MAHASISWA_NPM : string = "npm"
+MAHASISWA_ABSEN : string = "absen"
+JADWAL : string = "JADWAL"
+JADWAL_TGLPERTEMUA : string = "tglPertemuan"
+JADWAL_NOPERTEMUAN : string = "noPertemuan"
+JADWAL_DURASI : string = "durasi"
+JADWAL_TGLKELASPENGGANTI : string = "tglKelasPengganti"
+JADWAL_GANTIJDWL : string = "gantiJdwl"
+GANTI : string = "GANTI"
+GANTI_GANTIKULIAH : string = "gantiKuliah"
+GANTI_GANTIJADWAL : string = "gantiJadwal"
+GANTIKULIAH : string = "GANTIKULIAH"
+GANTIKULIAH_NAMA : string = "nama"
+GANTIKULIAH_KODE : string = "kode"
+GANTIJADWAL : string = "GANTIJADWAL"
+GANTIJADWAL_TANGGAL : string = "tanggal"
+GANTIJADWAL_SHIFT : string = "shift"
+KIRIM : string = "KIRIM"
+KIRIM_KULIAH : string = "kuliah"
Kirim
-kuliah : Kuliah
+Kirim()
+Kirim(in kuliah : Kuliah)
+setKuliah(in kuliah : Kuliah) : void
+getKuliah() : Kuliah
KelasOntology
+ONTOLOGY_NAME : string
-Instance : Ontology
+KelasOntology(in base : Ontology)
+getInstance() : Ontology