**PERANGKAT LUNAK BANTU PENGOLAHAN DATA PARKIR**

Diajukan untuk memenuhi salah satu tugas mata kuliah Praktikum Rekayasa Perangkat Lunak Lanjut



Disusun oleh :

**Muhamad bionanda 1127050100**

**Moh iqbal 1211705110**

**JURURUSAN TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UIN SUNAN GUNUNG DJATI**

**BANDUNG**

**2014**

DAFTAR PERUBAHAN

|  |  |
| --- | --- |
| Revisi | Deskripsi |
| A |  |
| B |  |
| C |  |
| D |  |
| E |  |
| F |  |
| G |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INDEX  TGL | - | A | B | C | D | E | F | G |
| Ditulis oleh |  |  |  |  |  |  |  |  |
| Diperiksa oleh |  |  |  |  |  |  |  |  |
| Disetujui oleh |  |  |  |  |  |  |  |  |

Daftar Halaman Perubahan

|  |  |  |  |
| --- | --- | --- | --- |
| Halaman | Revisi | Halaman | Revisi |
|  |  |  |  |

**KATA PENGANTAR**

Puji dan Syukur saya panjatkan ke Hadirat Allah SWT, karena berkat limpahan Rahmat dan Karunia-nya sehingga saya dapat menyusun laporan ini dengan baik dan tepat pada waktunya, dalam laporan ini saya membahas mengenai perangkat lunak bantu pengolahan data parkir.

Saya menyadari, isi maupun cara penyampaian laporan ini masih jauh dari sempurna. Untuk itu saya mengharapkan saran dan kritik yang bersifat membangun dari para pembaca sehingga saya bisa mengembangkan laporan ini menjadi lebih baik

Terima kasih kepada semua pihak yang telah berperan serta dalam penyusunan laporan ini dari awal sampai akhir semoga Allah SWT senantiasa meridhai segala usaha kita, amin.

Bandung, 27 Desember 2013

Penulis

**DAFTAR ISI**

KATA PENGANTAR i

DAFTAR ISI ii

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Identifikasi Masalah 1

1.3 Maksud dan Tujuan Penelitian 2

1.4 Batasan Masalah 2

1.5 Metodologi Penelitian 2

1.6 Teknik Pengumpulan Data 3

BAB II LANDASAN TEORI

2.1 Parkir 4

2.2 UML 5

2.3 Waterfall 17

BAB III ANALISIS DAN PERANCANGAN

3.1 Use Case 20

3.2 Skenario Use Case 20

3.3 Sequence Diagram 23

3.4 Collaboration Diagram 24

3.5 Class Diagram 26

3.6 Activity Diagram 26

3.7 Component Diagram 27

3.8 Deployment Diagram 27

BAB IV IMPLEMENTASI PROGRAM

4.1 Interface Program 28

BAB V

5.1 Saran 31

5.2 Kesimpulan 31

DAFTAR PUSTAKA 12

**BAB I**

**PENDAHULUAN**

**1.1 LATAR BELAKANG**

Sistem parkir yang berlaku saat ini masih menggunakan karcis parkir sebagai bukti parkir kendaraan dan pembayaran biaya parkir kendaraan dilakukan secara tunai. Sistem parkir yang demikian memiliki kelemahan antara lain, kurangnya tingkat keamanan dan dapat menimbulkan praktik korupsi pada petugas parkir. Dari permasalahan tersebut, maka timbul ide untuk membuat suatu sistem komputerisasi parkir yang dapat membantu pengaturan lahan parkir. Sistem komputerisasi parkir sudah banyak berkembang namun sistem parkir yang digunakan pada saat ini sebagian besar masih menggunakan sistem parkir yang bersifat manual, dalam pengertian bahwa jika suatu kendaraan akan memasuki areal parkir, maka petugas pada pintu masuk akan mencatat nomor Pelat kendaraan tersebut pada karcis parkir

. `Kemudian karcis parkir tersebut akan diberikan kepada pemilik kendaraan apabila telah membayar biaya parkir secara tunai, lalu kendaraan tersebut sudah dapat memasuki lahan parkir. Pada saat kendaraan akan keluar dari area parkir, maka pengemudi kendaraan harus memberikan karcis Parkir tersebut kepada petugas pada pintu keluar sebagai bukti bahwa nomor pelat kendaraan yang tertulis sesuai dengan nomor pelat kendaraan yang tertulis pada karcis parkir sesuai dengan nomor pelat kendaraan tersebut. Bila sesuai, maka kendaraan tersebut baru boleh keluar dari area parkir. Sistem parkir manual tersebut kurang aman dan efisien.

Belakangan berkembang sistem parkir terkomputerisasi yang sudah mampu mengotomatisasi beberapa tahap proses di atas dengan bantuan komputer dan perangkat keras lainnya, namun sistem yang ada saat ini belum mengotomatisasi pada proses input pelat kendaraan dan pembiayaan parkir.

**1.2 Identifikasi Masalah**

Dengan melihat latar belakang di atas, maka rumusan masalahnya adalah :

1. Bagaimana membuat sebuah sistem yang bisa memberikan lokasi parkir beserta nomor area parkir secara otomatis ?
2. Bagaimana membuat sistem yang bisa melakukan komunikasi antar *client*?

1 **Model Proses Pengembangan**

* 1. **Maksud dan Tujuan Penelitian**

Maksud dan tujuan dari pelaksanaan penelitian ini yaitu salah satunya untuk memenuhi syarat dalam penyelesaian tugas awal mata kuliah Rekayasa Perangkat Lunak jurusan Teknik Informatika di Universitas Sunan Gunung Djati Bandung, adapun tujuan pembuatan sistem ini adalah :

1. Membuat sistem dengan basis data *client server*.
2. Memberikan kenyamanan dalam penentuan lokasi dan nomor parkir yang bersifat otomatis.
3. Membuat sebuah sistem yang terintegrasi, sehingga mengurangi kesalahan dalam pengolahan data parkir.
4. Dengan adanya sistem otomatisasi tempat parkir, pengunjung parkir lebih nyaman dalam memarkir kendaraanya karena tidak lagi dibingungkan dengan pemilihan lokasi parkir.
   1. **Batasan Masalah**  
      Dengan melihat rumusan masalah di atas, maka sistem ini terbatas pada :
5. Penentuan lokasi parkir disesuaikan pada kapasitas parkir tiap lahan dan nomor area parkir otomatis akan diberikan pada nomor area parkir yang kosong atau tidak terpakai.
   1. **Metodologi Penelitian**

Metodologi penelitian merupakan suatu proses yang digunakan untuk memecahkan suatu masalah secara logis, dimana memerlukan data-data untuk mendukung terlaksananya suatu penelitian, metodologi ini akan dijelaskan dalam bentuk pengumpulan data dan metode pengembangan sistem.

**1.6 Teknik Pengumpulan Data**

Metode yang digunakan dalam pengumpulan data untuk membuat laporan tugas awal ini antara lain :

1. Riset Lapangan

Riset yang dilakukan dengan cara mendatangi tempat kerja praktek dan pengumpulan datanya dilakukan langsung melalui responden.

1. Observasi

Dengan cara melakukan pengamatan langsung dan memperhatikan serta meneliti proses pekerjaannya.

1. Wawancara

Melakukan tanya jawab langsung kepada pihak yang berkepentingan.

**BAB II**

**LANDASAN TEORI**

**2.1 PARKIR**

Parkir adalah tempat pemberhentian kendaraan dalam jangka waktu pendek atau lama, sesuai dengan kebutuhan pengendara. Parkir merupakan salah satu unsur prasarana transportasi yang tidak terpisahkan dari sistem jaringan transportasi, sehingga pengaturan parkir akan mempengaruhi kinerja suatu jaringan, terutama jaringan jalan raya.

|  |
| --- |
|  |
|  |

Tidak semua pengembang pusat bisnis mampu menyediakan lahan parkir yang mencukupi, sehingga badan jalan yang berada di sekitarnya digunakan untuk lahan parkir. Apabila badan jalan tersebut dilalui lalu lintas dalam jumlah yang cukup besar maka bisa dipastikan bahwa parkir di badan jalan akan menimbulkan permasalahan lalu lintas (kecepatan menurun dan waktu tempuh meningkat).

Timbulnya permasalahan parkir di kota-kota besar menuntut para ahli transportasi untuk betul-betul memahami parkir. Konsep dan karakteristik parkir, analisis kebutuhan parkirr, perencanaan geometrik lahan parkir, dan kebijakan parkir merupakan materi bisa diimplementasikan untuk menangani permasalahan parkir.

**CARA DAN JENIS PARKIR**

**Menurut Penempatannya**

1). Parkir di tepi jalan (*on-street parking*)**.**Yakni parkir dengan menggunakan badan jalan sebagai tempat parkir

*Kerugian :*

* Mengganggu lalu lintas
* Mengurangi kapasitas jalan karena adanya pengurangan lebar lajur lalu lintas
* Meningkatkan kemungkinan terjadinya kecelakaan

*Keuntungan :*

* Murah tanpa investasi tambahan
* Bagi pengguna tempat parkir bisa lebih dekat dan mudah

*Posisi parkir :*

* Sejajar dengan sumbu jalan
* Tegak lurus sumbu jalan
* Membuat sudut dengan sumbu jalan

|  |
| --- |
| <http://3.bp.blogspot.com/-07U9SN6pAK8/UYIC6nlL68I/AAAAAAAAAuc/rYaeG1taQm4/s1600/ruang+parkir.jpg> |
| **Gambar Ruang Parkir Bersudut** Sumber : Menuju lalu lintas dan angkutan jalan yang tertib, DLLAJ, 1995 |

2). Parkir di luar badan jalan (*off-street parking*)**.**Yakni parkir kendaraan di luar badan jalan bisa di halaman gedung perkantoran, supermarket, atau pada taman parkir.

*Keuntungan :*

* Tidak mengganggu lalu lintas
* Faktor keamanan lebih tinggi

*Kerugian :*

* Perlu biaya investasi awal yang besar.
* Bagi pengguna dirasakan kurang praktis, apalagi jika kepentingannya hanya sebentar saja.
  1. **UML**

**UML (Unified Modeling Language)** adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (Object-Oriented). UMLtidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti JAVA, C++, Visual Basic, atau bahkan dihubungkan secara langsung ke dalam sebuah object-oriented database.

**BAGIAN-BAGIAN UML**

Bagian-bagian utama dari UML adalah view, diagram, model element, dan general mechanism.

**1. View**

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. View bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis view dalam UML antara lain : use case view, logical view, component view, concurrency view, dan deployment view.

**2. Use case View**

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan external actors. Actor yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. View ini digambarkan dalam use case diagrams dan kadang-kadang dengan activity diagrams. View ini digunakan terutama untuk pelanggan, perancang (designer), pengembang (developer), dan penguji sistem (tester).

**3. Logical View**

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (class, object, dan relationship ) dan kolaborasi dinamis yang terjadi ketika object mengirim pesan ke object lain dalam suatu fungsi tertentu. View ini digambarkan dalam class diagrams untuk struktur statis dan dalam state, sequence, collaboration, dan activity diagram untuk model dinamisnya. View ini digunakan untuk perancang (designer) dan pengembang (developer).

**4. Component View**

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari code module diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. View ini digambarkan dalam component view dan digunakan untuk pengembang (developer).

**5. Concurrency View**

Membagi sistem ke dalam proses dan prosesor. View ini digambarkan dalam diagram dinamis (state, sequence, collaboration, dan activity diagrams) dan diagram implementasi (component dan deployment diagrams) serta digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

**6. Deployment View**

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (nodes) dan bagaimana hubungannya dengan yang lain. View ini digambarkan dalam deployment diagrams dan digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

**DIAGRAM**

1.         Use Case Diagram

2.         Activity Diagram

3.         Sequence Diagram

4.         Communication Diagram (Collaboration diagram in versi 1.x)

5.         Class Diagram

6.         State Machine Diagram (Statechart diagram in versi 1.x)

7.         Component Diagram

8.         Deployment Diagram

9.         Composite Structure Diagram

10.     Interaction Overview Diagram

11.     Object Diagram

12.     Package Diagram

13.     Timing Diagram

Berikut ini saya akan menjelaskan tentang diagram-diagram tersebut :

**1.        Use Case Diagram**

Use case adalah abstraksi dari interaksi antara system dan actor. Use case bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah system dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah system dipakai. Use case merupakan konstruksi untuk mendeskripsikan bagaimana system akan terlihat di mata user. Sedangkan use case diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.

[](http://3.bp.blogspot.com/-eeW6fkN929w/UGCOIItXTsI/AAAAAAAAAHU/mzVVcOW-xFg/s1600/use+case+diagram.jpg)

**Gambar Use Case Diagram**

Diagram Use Case berguna dalam tiga hal :

      Menjelaskan fasilitas yang ada (requirements)

-          Use Case baru selalu menghasilkan fasilitas baru ketika sistem di analisa, dan design menjadi lebih jelas.

      Komunikas dengan klien

-          Penggunaan notasi dan simbol dalam diagram Use Case membuat pengembang lebih mudah berkomunikasi dengan klien-kliennya.

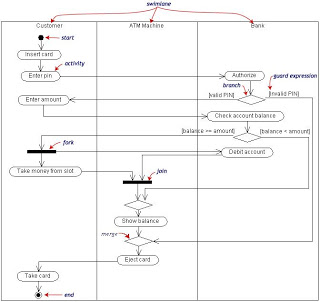
      Membuat test dari kasus-kasus secara umum

-          Kumpulan dari kejadian-kejadian untuk Use Case bisa dilakukan test kasus layak untuk kejadian-kejadian tersebut.

**2.        Activity Diagram**

Pada dasarnya diagram Activity sering digunakan oleh *flowchart*. Diagram ini berhubungan dengan diagram Statechart. Diagram Statechart berfokus pada *obyek yang dalam suatu proses* (atau proses menjadi suatu obyek), diagram Activity berfokus pada *aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal*. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Sebagai contoh, perhatikan proses yang terjadi. “Pengambilan uang dari bank melalui ATM.” Ada tiga aktifitas kelas (orang, dan lainnya) yang terkait yaitu : Customer, ATM, and Bank. Proses berawal dari lingkaran start hitam pada bagian atas dan berakhir di pusat lingkaran stop hitam/putih pada bagian bawah. Aktivitas digambarkan dalam bentuk kotak persegi. Lihat gambar di bawah ini, agar lebih jelas :

Contoh Diagram Activity ‘Pengambilan Uang melalui ATM’.

[](http://2.bp.blogspot.com/-2vmhVjcRytA/UGCOecfKPBI/AAAAAAAAAHc/VKsx55aQT0I/s1600/activity+diagram.jpg)

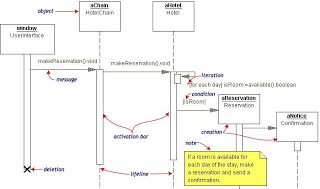
**Gambar**

Diagram Activity dapat dibagi menjadi beberapa jalur kelompok yang menunjukkan obyek mana yang bertanggung jawab untuk suatu aktifitas. Peralihan tunggal (*single transition*) timbul dari setiap adanya *activity* (aktifitas), yang saling menghubungi pada aktifitas berikutnya. Sebuah *transition* (transisi) dapat membuat cabang ke dua atau lebih percabangan *exclusive transition* (transisi eksklusif). Label *Guard Expression* (ada didalam [ ]) yang menerangkan output (keluaran) dari percabangan. Percabangan akan menghasilkan bentuk menyerupai bentuk intan. T*ransition* bisa bercabang menjadi beberapa aktifitas paralel yang disebut **Fork.** *Fork* beserta *join* (gabungan dari hasil output *fork*) dalam diagram berbentuk *solid bar* (batang penuh).

**3.        Sequence Diagram**

Diagram Class dan diagram Object merupakan suatu gambaran *model statis*. Namun ada juga yang bersifat *dinamis*, seperti **Diagram Interaction**. Diagram sequence merupakan salah satu diagram Interaction yang menjelaskan bagaimana suatu operasi itu dilakukan; *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Di bawah ini adalah diagram Sequence untuk pembuatan Hotel Reservation. Obyek yang mengawali urutan *message* adalah ‘aReservation Window’.

Contoh Diagram Sequence ‘Pemesanan kamar di Hotel’.

[](http://3.bp.blogspot.com/-9ndi0EW98Fw/UGCO48W6YvI/AAAAAAAAAHk/AtRstO0ZJmM/s1600/sequence+diagram.jpg)

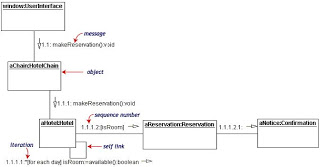
**Gambar**

‘Reservation window’ mengirim pesan makeReservation() ke ‘HotelChain’. Kemudian ‘HotelChain’ mengirim pesan yang sama ke ‘Hotel’. Bila ‘Hotel’ punya kamar kosong, maka dibuat ‘Reservation’ dan ‘Confirmation’. *Lifeline* adalah garis dot (putus-putus) vertikal pada gambar, menerangkan waktu terjadinya suatu obyek. Setiap panah yang ada adalah pemanggilan suatu pesan. Panah berasal dari pengirim ke bagian paling atas dari batang kegiatan (*activation bar*) dari suatu pesan pada *lifeline* penerima. *Activation bar* menerangkan lamanya suatu pesan diproses. Pada gambar diagram , terlihat bahwa ‘Hotel’ telah melakukan pemanggilan diri sendiri untuk pemeriksaan jika ada kamar kosong. Bila benar, maka ‘Hotel’membuat ‘Reservation’ dan ‘Confirmation’. Pemanggilan diri sendiri disebut dengan *iterasi*. *Expression* yeng dikurung dengan “[ ]”, adalah *condition* (keadaan kondisi). Pada diagram dapat dibuat *note* (catatan). Pada gambar, terlihat seperti selembar kertas yang berisikan teks. *Note* bisa diletakan dimana saja pada diagram UML.

**4.    Communication Diagram (Collaboration diagram in versi 1.x)**

Collaboration diagram menggambarkan interaksi antar objek seperti sequence diagram, tetapi lebih menekankan pada peran masing-masing objek. Setiap message memiliki sequence number, di mana message dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama. Diagram Collaboration juga merupakan *diagram interaction*. Diagram membawa informasi yang sama dengan diagram Sequence, tetapi lebih memusatkan atau memfokuskan pada kegiatan obyek dari waktu pesan itu dikirimkan.

Contoh Diagram Collaboration ‘Pemesanan kamar di Hotel’.

[](http://1.bp.blogspot.com/-U7ZSKCq9Pfg/UGCPWiiaosI/AAAAAAAAAHs/GPiWiDbQ8do/s1600/communication+diagram.jpg)

**Gambar**

Kotak kegiatan obyek diberi label dengan nama kelas atau obyek (atau keduanya). Nama kelas dibatasi dengan *colons / titik dua* ( : ). Setiap pesan pada diagram Collaboration mempunyai angka yang terurut. Pesan yang tingkatannya tertinggi adalah angka 1. Pesan yang berada pada tingkat yang sama memiliki *prefix* yang sama, namun *suffix* berbeda bergantung pada posisinya; hanya untuk angka 1, 2, dan seterusnya.

**5.    Class Diagram**

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment , pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan stereotype)

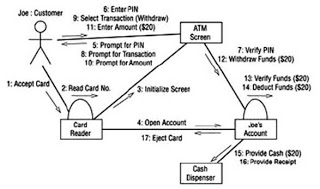
2. Atribut

3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

* Private, tidak dapat dipanggil dari luar class yang bersangkutan
* Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
* Public, dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah interface, yaitu class abstrak yang hanya memiliki metoda. Interface tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah class. Dengan demikian interface mendukung resolusi metoda pada saat run-time. Sesuai dengan perkembangan class model, class dapat dikelompokkan menjadi package. Kita juga dapat membuat diagram yang terdiri atas package.

[](http://4.bp.blogspot.com/-owMB5JmpwZ4/UGCPmaRJYFI/AAAAAAAAAH0/pM5b6Vin1RY/s1600/class+diagram.jpg)

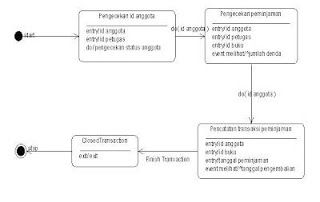
**Gambar Class Diagram**

**Hubungan Antar Class**

1. Asosiasi, yaitu hubungan statis antar class . Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain. Panah navigability menunjukkan arah query antar class.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar class . Class dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda class asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan ( message ) yang di-passing dari satu class kepada class lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram yang akan dijelaskan kemudian.

**6.    State Machine Diagram (Statechart diagram in versi 1.x)**

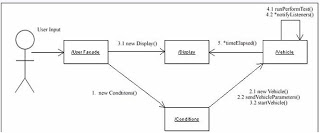
*Statechart diagram* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram* ). Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

[](http://4.bp.blogspot.com/-ktzgzLlwktA/UGCPyk3ASfI/AAAAAAAAAH8/KQ8_ZprGvjU/s1600/state+machine+diagram.jpg)

**Gambar State Machine Diagram (Statechart diagram in versi 1.x)**

**7.    Component Diagram**

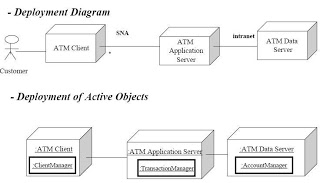
*Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan ( *dependency* ) di antaranya. Komponen piranti lunak adalah modul berisi *code* , baik berisi *source code* maupun *binary code* , baik *library* maupun *executable* , baik yang muncul pada *compile time, link time* , maupun *run time* . Umumnya komponen terbentuk dari beberapa *class* dan/atau *package* , tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface* , yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

[](http://4.bp.blogspot.com/-yHXFkTweLLA/UGCP8a_swwI/AAAAAAAAAIE/ZB16RMZ_tts/s1600/component+diagram.jpg)

**Gambar Component Diagram**

**8.    Deployment Diagram**

*Deployment/physical diagram* menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server, *workstation* , atau piranti keras lain yang digunakan untuk men- *deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

[](http://4.bp.blogspot.com/-PSbEcR3BUV8/UGCQFlsXbSI/AAAAAAAAAIM/Gg60azenv88/s1600/deployment+diagram.jpg)

**Gambar Deployment Diagram**

**9.    Composite Structure Diagram**

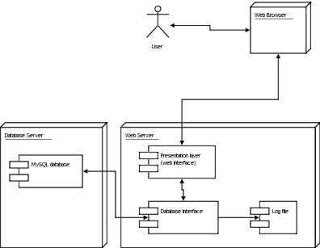
Diagram struktur komposit adalah diagram yang menunjukkan struktur internal classifier, termasuk poin interaksinya ke bagian lain dari sistem. Hal ini menunjukkan konfigurasi dan hubungan bagian, yang bersama-sama melakukan perilaku classifie. Diagram struktur komposit merupakan jenis diagram struktur statis dalam Unified Modeling Language (UML), yang menggambarkan struktur internal kelas dan kolaborasi.

Struktur komposit dapat digunakan untuk menjelaskan :

- Struktur dari bagian-bagian yang saling berkaitan

- Run-time struktur yang saling berhubungan

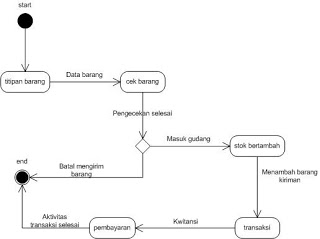
Contoh : Deskripsi dari bagian-bagian mesin yang saling berhubungan untuk melakukan fungsi mesin.

[](http://4.bp.blogspot.com/-fAXF9We4lDU/UGCQO0KCNqI/AAAAAAAAAIU/ucc_sGDzybo/s1600/composite+structure+diagram.jpg)

**Gambar Composite Structure Diagram**

**10.    Interaction Overview Diagram**

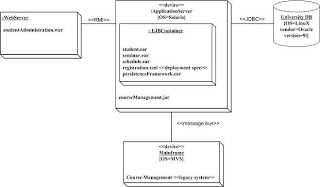
*Interaction Overview* Diagram adalah pencangkokan secara bersama antara activity diagram dengan sequence diagram. *Interaction Overview* Diagram dapat dianggap sebagai *activity* diagram dimana semua aktivitas diganti dengan sedikit *sequence* diagram, atau bisa juga dianggap sebagai *sequence* diagram yang dirincikan dengan notasi *activity* diagram yang digunakan untuk menunjukkan aliran pengawasan.

[](http://4.bp.blogspot.com/-zuI4AQneVJ8/UGCQfV794dI/AAAAAAAAAIc/gc2rKwwGkrk/s1600/interaction+overview+diagram.jpg)

**Gambar Interaction Overview Diagram**

**11.    Object Diagram**

*Object* diagram merupakan sebuah gambaran tentang objek-objek dalam sebuah sistem pada satu titik waktu. Karena lebih menonjolkan perintah-perintah 29 daripada *class*, *object* diagram lebih sering disebut sebagai sebuah diagram perintah.

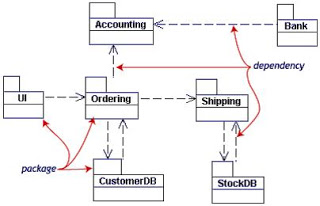
[](http://3.bp.blogspot.com/-BEbpRJCXzks/UGCQp7Ou6TI/AAAAAAAAAIk/NDXrb3mUpAo/s1600/object+diagram.jpg)

**Gambar Object Diagram**

**12.    Package Diagram**

Diagram objek melengkapi notasi grafik untuk pemodelan objek, kelas dan relasinya dengan yang lain. Diagram objek bermanfaat untuk pemodelan abstrak dan membuat perancangan program. Untuk mengatur pengorganisasian diagram Class yang *kompleks*, dapat dilakukan pengelompokan kelas-kelas berupa *packag*e (paket-paket). *Package* adalah kumpulan elemen-elemen logika UML. Gambar di bawah ini mengenai model bisnis dengan pengelompokan kelas-kelas dalam bentuk paket-paket :

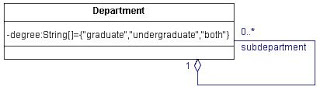
Contoh Diagram Package.

[](http://1.bp.blogspot.com/-KWgxyFvSsA8/UGCQ_FVfS0I/AAAAAAAAAIs/hXFDGFMLpqw/s1600/package+diagram+1.jpg)

**Gambar Package Diagram**

Ada jenis khusus dari diagram Class yaitu diagram Object. Kegunaannya untuk penjelasan yang sedikit dengan relasi yang sulit, khususnya relasi rekursif. Lihat gambar dibawah, diagram Class kecil menunjukkan bahwa ‘department’ dapat mengandung banyak ‘department’ yang lain.

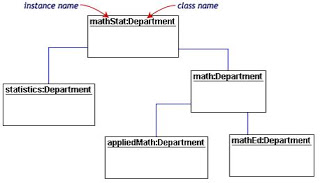
Class yang relasinya rekursif.

[](http://2.bp.blogspot.com/-xc3jfUDgBTs/UGCRL4WQaZI/AAAAAAAAAI0/8gGBeQLFecI/s1600/package+diagram+2.jpg)

**Gambar**

Setiap tingkatan pada diagram berpengaruh pada *single instance* (bagian tunggal). Nama bagian digarisbawahi dalam diagram UML. Untuk *Class name* (nama kelas) maupun *instance name* (nama bagian) bisa mengambil dari diagram Object selama arti diagram tersebut masih jelas.

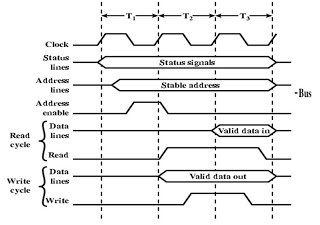
*Instance name* memiliki huruf yang digarisbawahi.

[](http://1.bp.blogspot.com/-4LqwUpi4QVY/UGCRXJSOawI/AAAAAAAAAI8/BjpodeJ5ttU/s1600/package+diagram+3.jpg)

**Gambar**

**13.    Timing Diagram**

*Timing* Diagram adalah bentuk lain dari *interaction* diagram, dimana fokus utamanya lebih ke waktu. Timing diagram sangat berdaya guna dalam menunjukkan faktor pembatas waktu diantara perubahan *state* pada objek yang berbeda.

[](http://1.bp.blogspot.com/-O3RSxfrVA7w/UGCRnoYIHaI/AAAAAAAAAJE/lFInjtQ4UnQ/s1600/timing+diagram.jpg)

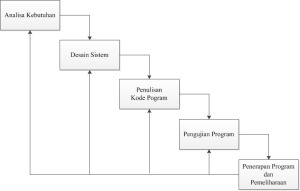
**Gambar Timing Diagram**

**Tujuan Penggunaan UML**

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahas pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
3. Memberikan model yang siap pakai, bahsa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah (blue print) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bias diketahui informasi secara detail tentang coding program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk diagram (reserve enginering).

**2.2 Water Fall**

Metode *waterfall* merupakan metode yang sering digunakan oleh penganalisa sistem pada umumnya. Inti dari metode *waterfall* adalah pengerjaan dari suatu sistem dilakukan secara berurutan atau secara linear. Jadi jika langkah ke-1 belum dikerjakan, maka langkah 2 tidak dapat dikerjakan. Jika langkah ke-2 belum dikerjakan maka langkah ke-3 juga tidak dapat dikerjakan, begitu seterusnya. Secara otomatis langkah ke-3 akan bisa dilakukan jika langkah ke-1 dan ke-2 sudah dilakukan.

[](http://agusdar.files.wordpress.com/2013/04/untitled.jpg)

Secara garis besar metode *waterfall* mempunyai langkah-langkah sebagai berikut : Analisa, Desain, Penulisan, Pengujian dan Penerapan serta Pemeliharaan. (Kadir, 2003)

**A. Tahapan Metode *Waterfall***

**1. Analisa Kebutuhan**     Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau studi literatur. Sistem analis akan menggali informasi sebanyak-banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user requirment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen ini lah yang akan menjadi acuan sistem analis untuk menerjemahkan ke dalam bahasa pemrogram.

**2. Desain Sistem**     Tahapan dimana dilakukan penuangan pikiran dan perancangan sistem terhadap solusi dari permasalahan yang ada dengan menggunakan perangkat pemodelan sistem seperti diagram alir data (*data flow diagram*), diagram hubungan entitas (*entity relationship diagram*) serta struktur dan bahasan data.

**3. Penulisan Kode Program**     Penulisan kode program atau *coding* merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan meterjemahkan transaksi yang diminta oleh *user.* Tahapan ini lah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.

**4. Pengujian Program**      Tahapan akhir dimana sistem yang baru diuji kemampuan dan keefektifannya sehingga didapatkan kekurangan dan kelemahan sistem yang kemudian dilakukan pengkajian ulang dan perbaikan terhadap aplikasi menjadi lebih baik dan sempurna.

**5. Penerapan Program dan Pemeliharaan**     Perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional.

**B. Keunggulan dan Kelemahan Metode *Waterfall***

Metode pengembangan *waterfall* mempunyai keunggulan dalam membangun dan mengembangkan suatu sistem, antara lain:

1. Kualitas dari sistem yang dihasilkan akan baik. Ini dikarenakan oleh pelaksanaannya secara bertahap. Sehingga tidak terfokus pada tahapan tertentu.
2. Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya. Jadi setiap fase atau tahapan akan mempunyai dokumen tertentu.

Dalam proses membangun dan mengembangkan suatu sistem, metode *waterfall* mempunyai beberapa kelemahan, antara lain:

1. Diperlukan majemen yang baik, karena proses pengembangan tidak dapat dilakukan secara berulang sebelum terjadinya suatu produk.*.*
2. Kesalahan kecil akan menjadi masalah besar jika tidak diketahui sejak awal pengembangan.
3. Pelanggan sulit menyatakan kebutuhan secara eksplisit sehingga tidak dapat mengakomodasi ketidakpastian pada saat awal pengembangan.

**BAB III**

**ANALISIS DAN PERANCANGAN**

* 1. Use Case



3.2 Skenario Usecase

1. Nama Use Case Login

Actor : Operator

Tujuan : Masuk kedalam system

Deskripsi : Proses login merupakan proses autentikasi untuk menggunakan sistem

Kondisi sebelum :Menjalankan sistem, lalu menginputkan username dan password.

Kondisi setelah :Tampil menu utama dari sistem.

1. Nama use case Catat masuk Parkir

Actor : Admin

Tujuan : Memasukan data pelanggan

Deskripsi : - Admin melakukan validasi data pada setiap pelanggan yang datang.

* + Admin memberikan nomor id untuk menandai setiap pelanggan baru dan kendaraan yang baru masuk

Kondisi sebelum : Masuk ke menu utama lalu memilih form data pelanggan

Kondisi setelah : Data pelanggan yang telah di inputkan akan masuk ke database pelanggan

1. Nama Use Case : Hapus data pelanggan

Actor : Admin

Tujuan : Menghapus data pelanggan

Deskripsi : Admin menghapus data pelanggan dengan berbagai pertimbangan dan alasan.

Kondisi sebelum: Telah menginput data pelanggan

Kondisi setelah : Data pelanggan yang dipilih telah dihapus

1. Nama Use Case : Ubah data pelanggan

Actor : Admin

Tujuan : Merubah data konsumen

Deskripsi : Admin merubah data konsumen, misalnya perubahan No plat, id , dll.

Kondisi sebelum : Telah menginputkan data pelanggan.

Kondisi setelah : Data pelanggan telah diubah.

1. Nama Use Case Cari data pelanggan

Actor : Admin

Tujuan : Mencari data pelanggan

Deskripsi : Admin mencari data pelanggan yang telah ada.

Kondisi sebelum : Telah menginputkan data pelanggan

Kondisi setelah : Data pelanggan berhasil dicari

1. Nama use case : Kendaraan keluar

Actor : Admin

Tujuan : Mengetahui detail transaksi

Deskripsi :

- Admin melakukan transaksi dengan pelanggan.

-Admin menampilkan harga sesuai dengan transaksi yang dilakukan, pelanggan melakukan pembayaran,

-Admin menginput data transaksi ke dalam database. .

Kondisi sebelum : Data pelanggan harus telah diinputkan

Kondisi setelah : Data transaksi yang telah diinputkan akan masuk kedalam database transaksi lalu akan mencetak struck transaksi yang telah terjadi dan diberikan kepada pelanggan.

1. Nama use case cetak data transaksi

Actor : Admin dan Pelanggan

Tujuan : Mencetak bukti transaksi

Deskripsi : Admin mencetak struck transaksi.

Kondisi sebelum : Telah menginput data transaksi.

Kondisi setelah : struck telah di cetak dan diberikan kepada pelanggan.

1. Nama use case laporan

Actor : Admin dan Owner

Tujuan :Melaporkan data yang telah masuk.

Deskripsi : Admin memberikan laporan kepada owner berupa data pelanggan dan data transaksi yang telah terjadi.

Kondisi sebelum : Telah menginput data pelanggan dan data transaksi

Kondisi setelah : Admin telah menyerahkan laporan data pelanggan dan data transaksi kepada owner.

* 1. Sequence Diagram

1. Sequence Login



1. Sequence Kendaraan masuk



1. Sequence kendaraan keluar

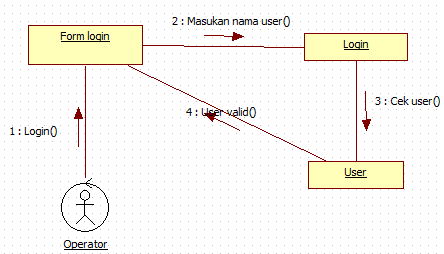


1. Sequence Laporan

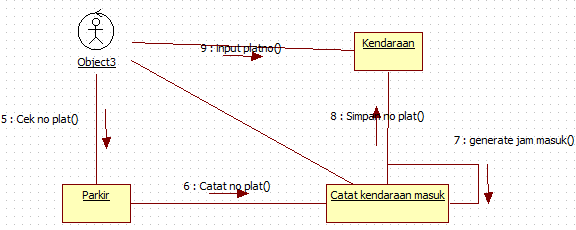
****

* 1. Collaboration Diagram

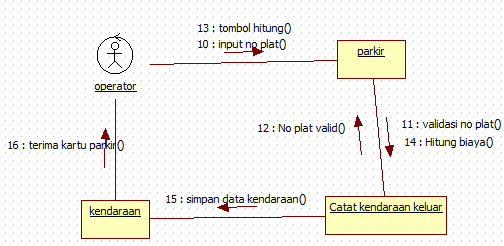
1. Collaboration Login



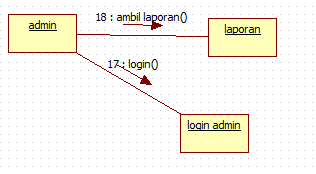
1. Collaboration kendaraan masuk



1. Collaboration Kendaraan keluar



1. Collaboration Laporan



* 1. Class Diagram

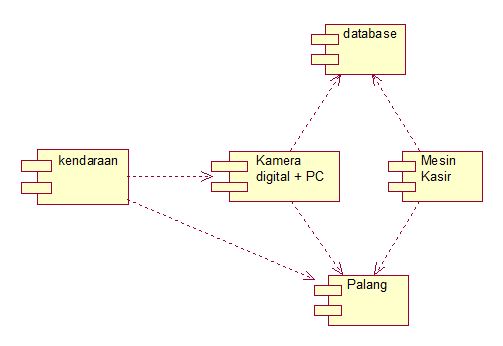


* 1. Activity Diagram

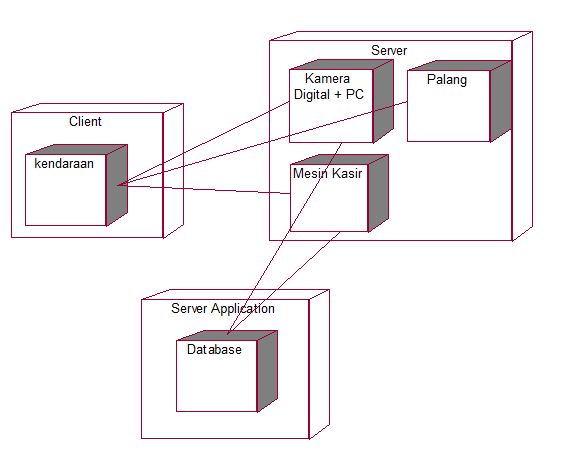
1. Activity Diagram

****

* 1. Component Diagram



* 1. Deployment Diagram

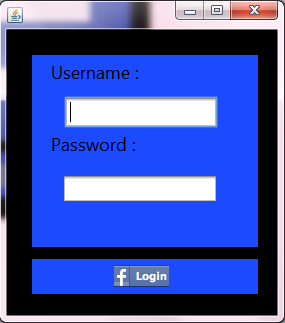


**BAB IV**

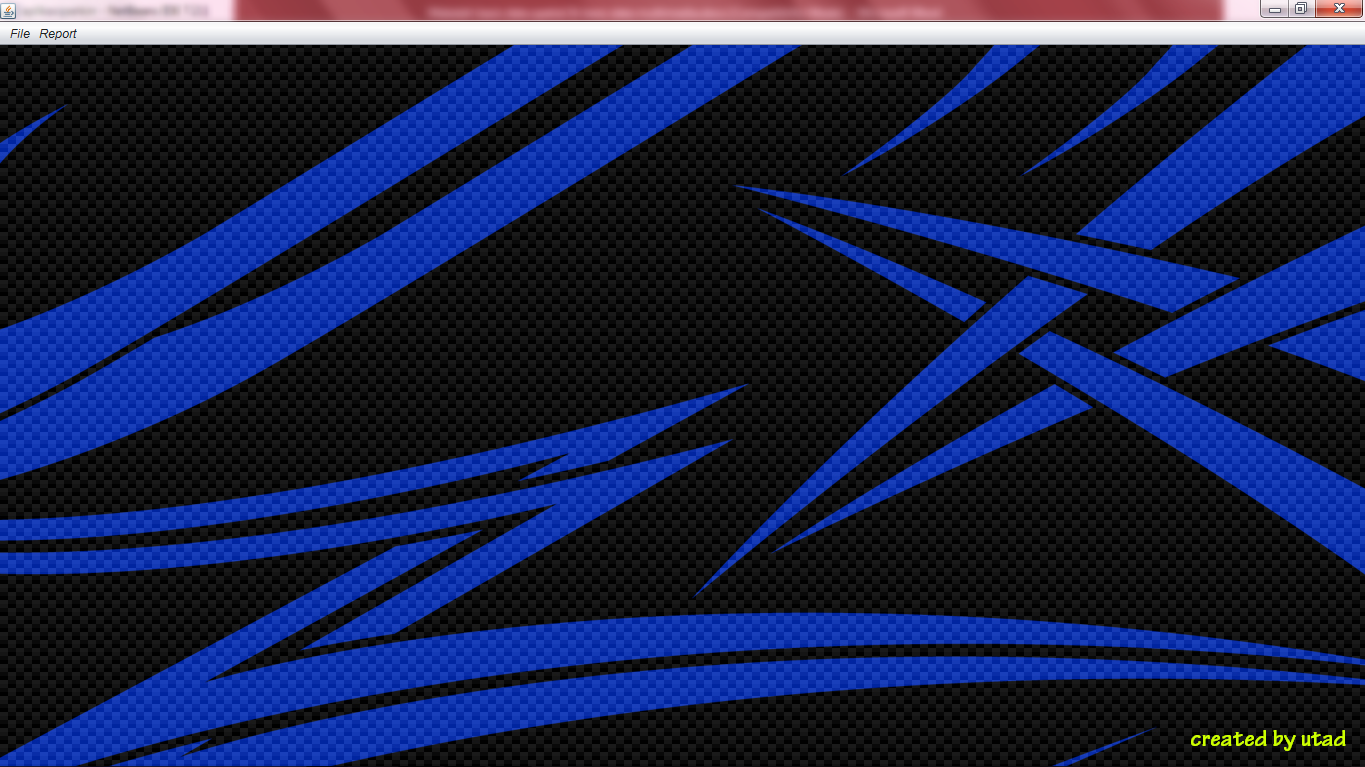
**IMPLEMENTASI PROGRAM**

* 1. **Interface program**

Login



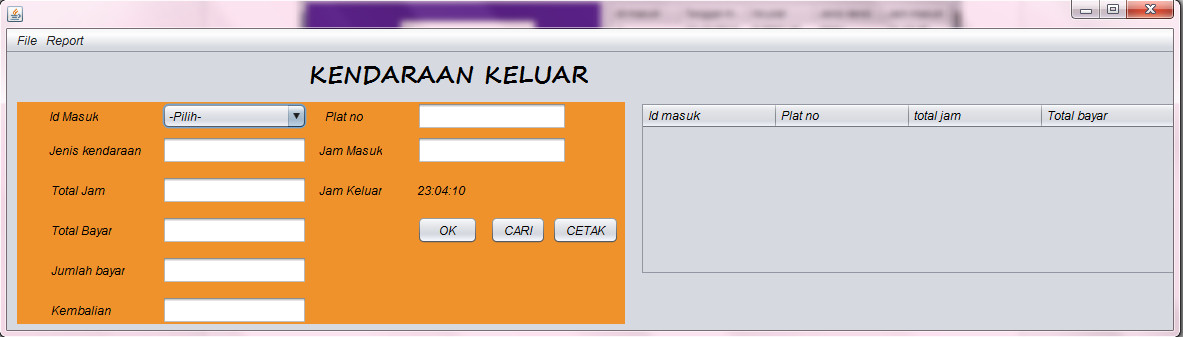
Menu utama



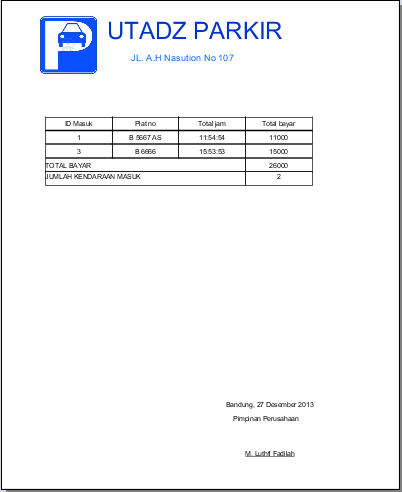
Kendaraan masuk



Kendaraan keluar



Report



Struk



**BAB V  
PENUTUP**

* 1. **Kesimpulan**

Adapun aplikasi parkir ini adalah merupakan modifikasi dari aplikasi parkir yang sudah ada pada umumnya. Pada aplikasi parkir ini lebih menggunakan lokasi parkir yang ada dengan menempatkan kendaraan – kendaraan sesuai dengan kebutuhan parkir. Sehingga dapat mengurangi pemborosan lahan parkir. Aplikasi ini dibuat berdasarkan hasil observasi yang dilakukan sehubungan dengan ketidakteraturan dan pemborosan pada lahan parkir yang terdapat pada beberapa pusat perbelanjaan.

**.**

* 1. **Saran**

Dalam aplikasi alokasi parkir ini terdapat kekurangan dari aspek keamanan, sehingga diharapkan dapat lebih di kembangkan lagi untuk memenuhi kriteria aplikasi yang baik dan bermanfaat bagi pengguna

**DAFTAR PUSTAKA**

* <http://noviastutik.blogspot.com/2012/09/diagram-diagram-dalam-uml-unified_24.html>
* <http://agusdar.wordpress.com/2013/04/13/metode-pengembangan-sistem-waterfall/>
* <http://www.ilmuprogrammer.com/2013/07/download-aplikasi-parkir-kendaraan.html>
* Jaworski, Jamie. 1998. Java 1.2 Unleashed. Macmillan computer Publishing.
* JDBC Guide: Getting Started. 1997.Sun Microsystem, Inc.