Modul Responsi

Rekayasa Perangkat Lunak

Dosen Pengampu:

Astria Hijriani, S.Kom. M.Kom.

Penyusun:

Tim Asisten Rekayasa Perangkat Lunak

Edisi 1 (2017)

Laboratorium Komputasi Dasar
Jurusan Ilmu Komputer
FMIPA Universitas Lampung

Deskripsi Mata Kuliah

Pembahasan mengacu pada siklus pengembangan perangkat lunak yang terdiri atas tahapan perencanaan, analisa, perancangan, pembuatan program, pengujian dan pemeliharaan. Bentuk perkuliahan dipadukan dengan proyek semester. Setiap kelompok peserta diharuskan membangun sebuah perangkat lunak, yang telah didefinisikan pada awal semester. Materi yang akan diberikan meliputi metodologi pengembangan perangkat lunak: computer aided software engineering (CASE) tools; Perencanaan proyek pengembangan perangkat lunak; analisis permasalahan dan kebutuhan pemakai; penyusunan spesifikasi perangkat lunak; Prinsip dasar perancangan perangkat lunak; teknik perancangan berorientasikan pada proses, data, obyek; permasalahan dalam penuliasan program; software quality assurance; ukuran mutu perangkat lunak; pengujian perangkat lunak; pemeliharaan perangkat lunak.

Tujuan Perkuliahan

Agar mahasiswa dapat memahami pengembangan perangkat lunak beserta tahapan dari pembuatannya.

Deskripsi Isi Perkuliahan

Materi pembelajaran dalam Mata Kuliah Rekayasa Perangkat Lunak Lanjutan terdiri dari beberapa pokok bahasan seperti: pengenalan dasar-dasar UML, Usecase Diagram, Activity Diagram, Sequence Diagram, Class Diagram, Flow Chart, dan desain dengan studi kasus.

Daftar Isi

Materi 1
Pengenalan dasar-dasar UML

Tujuan Instruksional	**:** Pengantar
Pokok bahasan ini menjelaskan tentang pengenalan awal UML serta pengertian dan dasar-dasar UML.	
Kompetensi yang Diharapkan	**:**
Praktikan mampu memahami pengertian dan dasar-dasar UML.	
Waktu Pertemuan	: 120 menit

1.1 Pengenalan UML

UML (*Unified Modeling Language*) merupakan pengganti dari metode analisis berorientasi object dan design berorientasi object (OOA&D) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an.

UML merupakan gabungan dari metode Booch, Rumbaugh (OMT) dan Jacobson. Tetapi UML ini akan mencakup lebih luas daripada OOA&D. Pada pertengahan pengembangan UML dilakukan standarisasi proses dengan OMG (Object Management Group) dengan harapan UML akan menjadi bahasa standar pemodelan pada masa yang akan datang.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat.

Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. Jika anda ingin berdiskusi tentang desain dengan seseorang, maka Anda hanya membutuhkan bahasa pemodelan bukan proses yang digunakan untuk mendapatkan desain.

UML merupakan bahasa standar untuk penulisan *blueprint software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak.

1.2 Sejarah Singkat UML

UML dimulai secara resmi pada oktober 1994, ketika Rumbaugh bergabung dengan Booch pada Relational Software Corporation. Proyek ini memfokuskan pada penyatuan metode Booch dan OMT. UML versi 0.8 merupakan metode penyatuan yang dirilis pada bulan Oktober 1995. Dalam waktu yang sama, Jacobson bergabung dengan Relational dan cakupan dari UML semakin luas sampai diluar perusahaan OOSE. Dokumentasi UML versi 0.9 akhirnya dirilis pada bulan Juni 1996. Meskipun pada tahun 1996 ini melihat dan menerima *feedback* dari

komunitas *Software Engineering* . Dalam waktu tersebut, menjadi lebih jelas bahwa beberapa organisasi perangkat lunak melihat UML sebagai strategi dari bisnisnya. Kemudian dibangunlah UML *Consortium* dengan beberapa organisasi yang akan menyumbangkan sumber dayanya untuk bekerja, mengembangkan, dan melengkapi UML.

Di sini beberapa *partner* yang berkontribusi pada UML 1.0, diantaranya *Digital Equipment Corporation, Hewlett-Packard, I-Logix, Intellicorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Relational, Texas Instruments dan Unisys*. Dari kolaborasi ini dihasilkan UML 1.0 yang merupakan bahasa pemodelan yang ditetapkan secara baik, *expressive*, kuat, dan cocok untuk lingkungan masalah yang luas. UML 1.0 ditawarkan menjadi standarisasi dari *Object Management Group* (OMG). Dan pada Januari 1997 dijadikan sebagai standar bahasa pemodelan.

Antara Januari–Juli 1997 gabungan *group* tersebut memperluas kontribusinya sebagai hasil respon dari OMG dengan memasukkan *Adersen Consulting, Ericsson, ObjectTimeLimeted, Platinum Technology, Ptech, Reich Technologies, Softeam, Sterling Software dan Taskon*. Revisi dari versi UML (versi 1.1) ditawarkan kepada OMG sebagai standarisasi pada bulan Juli 1997. Dan pada bulan September 1997, versi ini dierima oleh OMG Analysis dan Design Task Force (ADTF) dan OMG *ArchitectureBoard*. Dan Akhirnya pada Juli 1997 UML versi 1.1 menjadi standarisasi.

Pemeliharaan UML terus dipegang oleh OMG Revision Task Force (RTF) yang dipimpin oleh Cris Kobryn. RTP merilis editorial dari UML 1.2 pada Juni 1998. Dan pada tahun 1998 RTF juga merilis UML 1.3 disertai dengan user guide dan memberikan technical cleanup.

1.1 Pengertian UML

1.1.1 Pengertian Unified Modeling Language (UML)

UML adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan artifacts (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, artifact tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya [HAN98]. Selain itu UML adalah bahasa pemodelan yang menggunakan konsep orientasi object. UML dibuat oleh Grady Booch , James Rumbaugh , dan Ivar Jacobson di bawah bendera Rational Software Corp [HAN98]. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai perspektif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

1.1.2 Bagian-bagian Dari 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*.

- **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*).

- **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*).

- **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*).

- **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*).

- **Deployment view**
  Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya.

  View ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

2) Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya dialokasikan untuk view tertentu. Adapun jenis diagram antara lain :

- **Use Case Diagram**
  Menggambarkan sejumlah external actors dan hubungannya ke use case yang diberikan oleh sistem. Use case adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari use case symbol namun dapat juga dilakukan dalam activity diagrams.

  Use case digambarkan hanya yang dilihat dari luar oleh actor (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam sistem.

- **Class Diagram**
  Menggambarkan struktur statis class di dalam sistem. Class merepresentasikan sesuatu yang ditangani oleh sistem. Class dapat

berhubungan dengan yang lain melalui berbagai cara: associated (terhubung satu sama lain), dependent(satu class tergantung/menggunakan class yang lain), specialed (satu class merupakan spesialisasi dari class lainnya), atau package (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa class diagram.

- **State Diagram**
  Menggambarkan semua state (kondisi) yang dimiliki oleh suatu object dari suatu class dan keadaan yang menyebabkan state berubah. Kejadian dapat berupa object lain yang mengirim pesan.

  State class tidak digambarkan untuk semua class, hanya yang mempunyai sejumlah state yang terdefinisi dengan baik dan kondisi class berubah oleh state yang berbeda.

- **Sequence Diagram**
  Menggambarkan kolaborasi dinamis antara sejumlah object. Kegunaanya untuk menunjukkan rangkaian pesan yang dikirim antara object juga interaksi antara object, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

- **Collaboration Diagram**
  Menggambarkan kolaborasi dinamis seperti sequence diagrams. Dalam menunjukkan pertukaran pesan, collaboration diagrams menggambarkan object dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan sequence diagrams, tapi jika penekanannya pada konteks gunakan collaboration diagram.

- **Activity Diagram**
  Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi.

- **Component Diagram**
  Menggambarkan struktur fisik kode dari komponent. Komponent dapat berupa source code, komponent biner, atau executable component. Sebuah komponent berisi informasi tentang logic class

atau class yang diimplementasikan sehingga membuat pemetaan dari logical view ke component view.

- **Deployment Diagram**
  Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukkan hubungan komputer dengan perangkat (nodes) satu sama lain dan jenis hubungannya. Di dalam nodes, executeable component dan object yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh node tertentu dan ketergantungan komponen.

1.1.3 Gambaran dari UML

UML merupakan bahasa pemodelan yang memiliki pembendaharaan kata dan cara untuk mempresentasikan secara fokus pada konseptual dan fisik dari suatu sistem. Contoh untuk sistem software yang intensive membutuhkan bahasa yang menunjukkan pandangan yang berbeda dari arsitektur sistem, ini sama seperti menyusun/mengembangkan software development life cycle. Dengan UML akan memberitahukan kita bagaimana untuk membuat dan membaca bentuk model yang baik, tetapi UML tidak dapat memberitahukan model apa yang akan dibangun dan kapan akan membangun model tersebut. Ini merupakan aturan dalam software development process.

1.1.4 Area Penggunaan UML

UML digunakan paling efektif pada domain seperti :

- Sistem Informasi Perusahaan
- Sistem Perbankan dan Perekonomian
- Bidang Telekomunikasi
- Bidang Transportasi
- Bidang Penerbangan
- Bidang Perdagangan
- Bidang Pelayanan Elekronik
- Bidang Pengetahuan
- Bidang Pelayanan Berbasis Web Terdistribusi

Namun UML tidak terbatas untuk pemodelan software. Pada faktanya UML banyak untuk memodelkan sistem non software seperti:

- Aliran kerja pada sistem perundangan.
- Struktur dan kelakuan dari Sistem Kepedulian Kesehatan Pasien
- Desain hardware dll.

1.1.5 Tujuan Penggunaan UML

1) Memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep be rorientasi object.
2) Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

Materi 2
Use Case Diagram

Tujuan Instruksional **:** Pengantar
Pokok bahasan ini menjelaskan tentang komponen-komponen yang terlibat dalam use case diagram.

Kompetensi yang Diharapkan **:**

1. Praktikan mampu membuat sebuah skenario suatu sistem yang nantinya dapat diimplementasikan menjadi sebuah perangkat lunak.
2. Praktikan bisa memahami alur dari setiap tahap yang digunakan dalam perancangan perangkat lunak menggunakan UML.
3. Praktikan dapat memahami hubungan atara actor dengan use case diagram.
4. Praktikan mampu membuat use case diagram dari skenario yang telah ada.

Waktu Pertemuan : 120 menit

2.1 Kelakuan Sistem

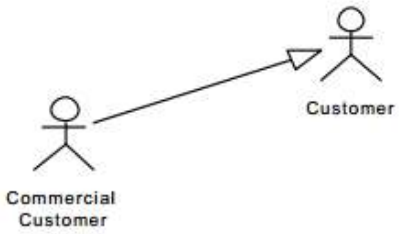
1) Kebutuhan sistem adalah fungsionalitas apa yang mesti disediakan oleh sistem, apakah didokumentasikan pada model use case yang menggambarkan fungsi sistem yang diharapkan (use case), yang mengelilinginya (actor) dan hubungan antara actor dengan use case (use case diagram).
2) Use case model dimulai pada tahap inception dengan mengidentifikasi actor dan use case utama pada sistem. Kemudian model ini diolah lebih matang di tahap elaboration untuk memperoleh lebih detail informasi yang ditambahkan pada use case.

2.2 Komponen – Komponen yang Terlibat dalam Use Case Diagram

2.2.1 Actor

Pada dasarnya actor bukanlah bagian dari use case diagram, namun untuk dapat terciptanya suatu use case diagram diperlukan beberapa actor dimana actor tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah actor mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima dan memberi informasi pada sistem, actor hanya berinteraksi dengan use case tetapi tidak memiliki kontrol atas use case. Actor digambarkan dengan stick man. Actor dapat digambarkan secara secara umum atau spesifik, dimana untukmembedaka nnya kita dapat menggunakan relationship

Contoh :



Gambar 1. Actor

Ada beberapa kemungkinan yang menyebabkan actor tersebut terkait dengan sistem antara lain:

- Yang berkepentingan terhadap sistem dimana adanya arus informasi baik yang diterimanya maupun yang dia inputkan ke sistem.
- Orang ataupun pihak yang akan mengelola sistem tersebut.
- External resource yang digunakan oleh sistem.
- Sistem lain yang berinteraksi dengan sistem yang akan dibuat.

2.2.2 Use Case

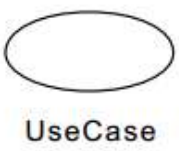
Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga customer atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.

Catatan:

Use case diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (user), sehingga pembuatan use case lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.

Cara menentukan Use Case dalam suatu sistem:

- Pola perilaku perangkat lunak aplikasi.
- Gambaran tugas dari sebuah actor.
- Sistem atau “benda” yang memberikan sesuatu yang bernilai kepada actor.
- Apa yang dikerjakan oleh suatu perangkat lunak (* bukan bagaimana cara mengerjakannya).



Gambar 2. Use Case

2.2.3 Relasi Dalam Use Case

Ada beberapa relasi yang terdapat pada use case diagram:

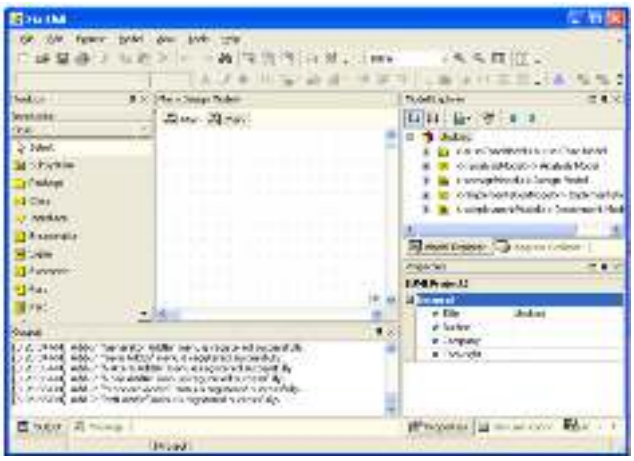
1) *Association*, menghubungkan link antar element.
2) *Generalization*, disebut juga inheritance (pewarisan), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.
3) *Dependency*, sebuah element bergantung dalam beberapa cara ke element lainnya.
4) *Aggregation*, bentuk assosiation dimana sebuah elemen berisi elemen lainnya.

Tipe relasi/ stereotype yang mungkin terjadi pada use case diagram:

1) **<<include>>** , yaitu kelakuan yang harus terpenuhi agar sebuah event dapat terjadi, dimana pada kondisi ini sebuah use case adalah bagian dari use case lainnya.
2) **<<extends>>**, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan alarm.
3) **<<communicates>>**, mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah communicates association . Ini merupakan pilihan selama asosiasi hanya tipe ralationship yang dibolehkan antara actor dan use case.

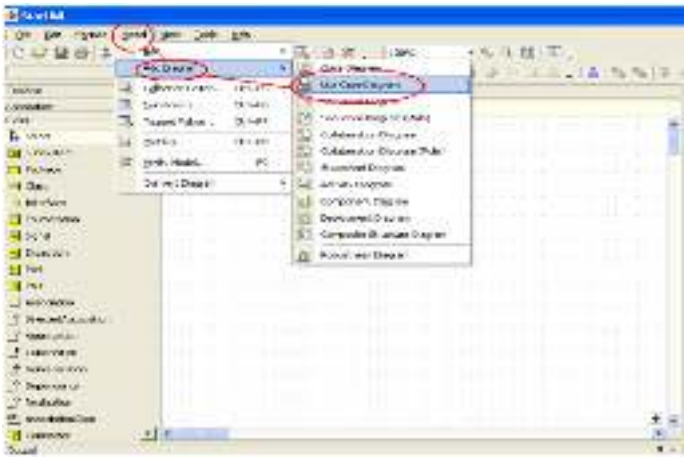
2.2.4 Cara Membuat Use Case Diagram dengan StarUML

1) Berikut tampilan StartUML pada Windows

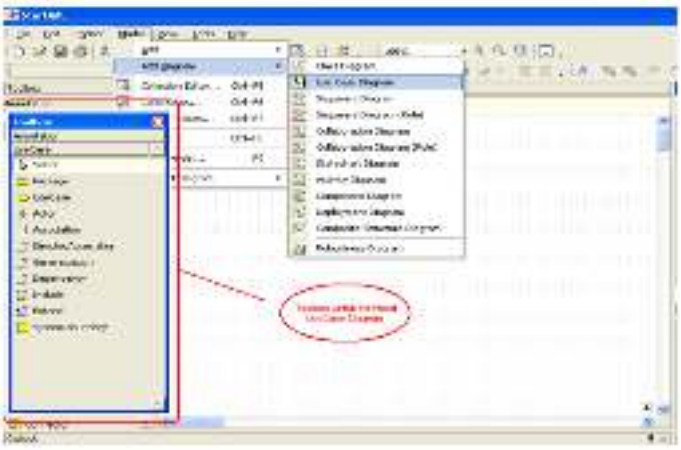


Gambar 3. Tampilan StartUML pada Windows

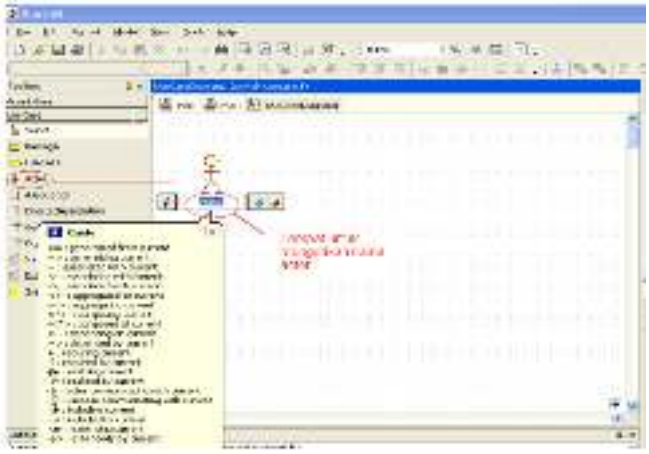
2) Pada tampilan awal pilih **Model** yang terletak pada tool bar, lalu **Add Diagram** dan pilih **Use Case Diagram**.



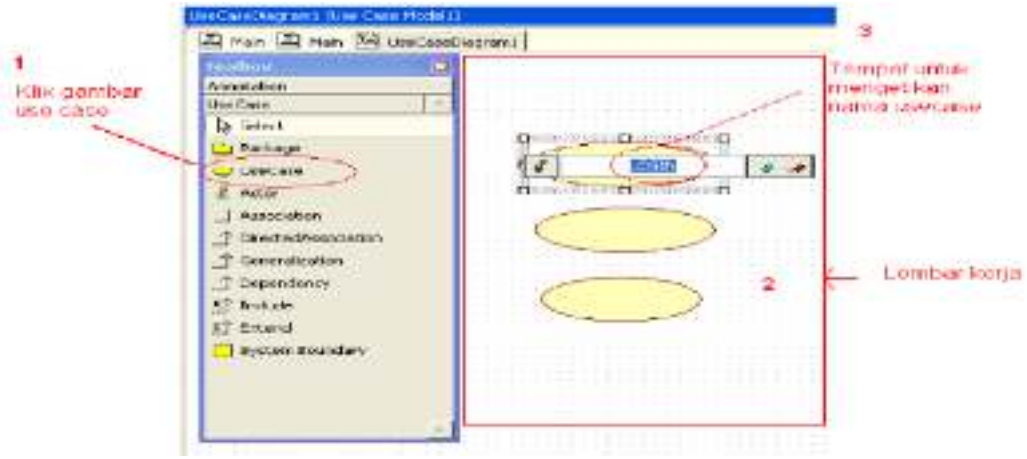
3) Maka tampilan toolbox pada sebelah kiri akan berubah.



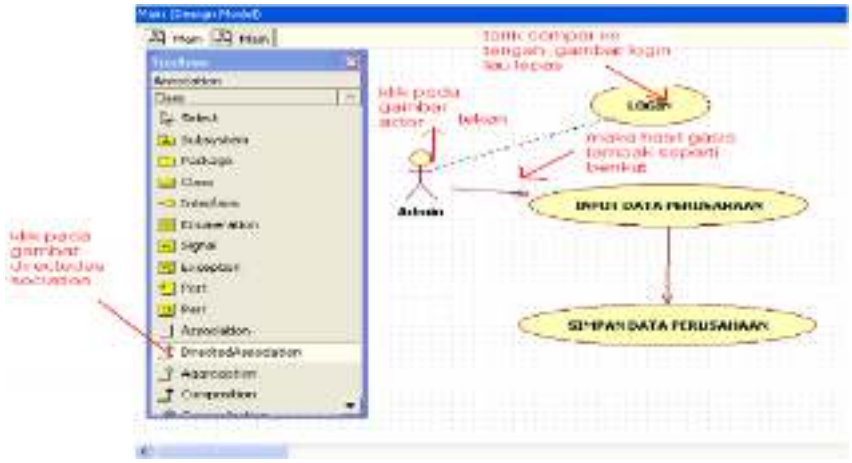
4) Klik pada gambar aktor dan taruh kursor pada samping toolbox. Maka akan muncul gambar orang yang disebut dengan actor dan beri nama actor.



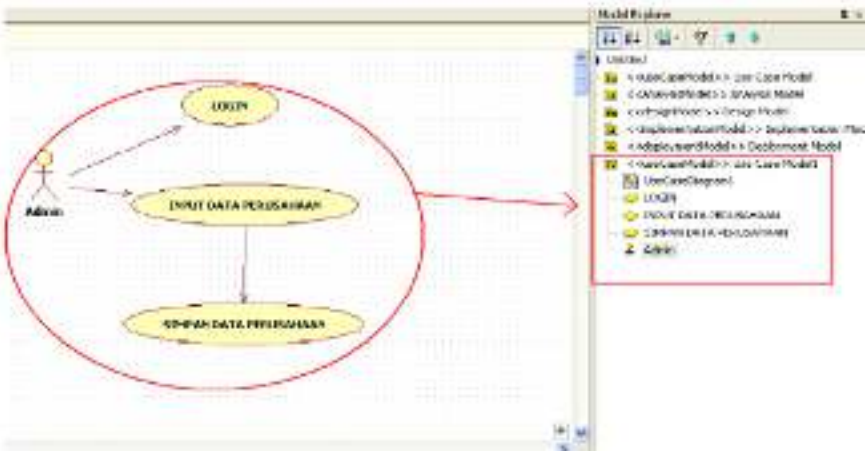
5) Selanjutnya pilih usecase pada menu toolbox, tekan tiga kali pada lembar kerja untuk membuat tiga use case dan beri nama pada setiap use case.



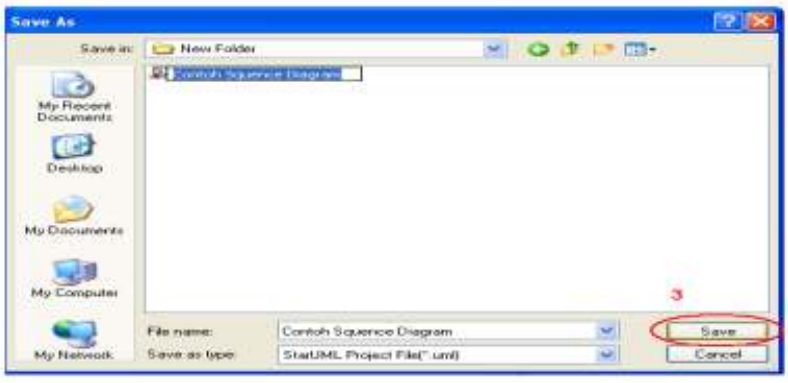
6) Untuk membuat garis hubung antara actor dan use case pilih directed association tekan kursor pada gambar actor lalu arahkan pada usecase dan lepas, maka garis akan terhubung.



7) Pastikan pada model explorer akan tersimpan nama dan use case diagram yang telah dibuat.



8) Jika telah selesai simpan file dengan cara pilih file -> save as.



TUGAS MODUL 2 !

Buatlah use case diagram dari kasus dibawah.

Sebuah bank mengoperasikan ATM dan mengelola banyak tabungan, setiap nasabah memiliki setidaknya satu rekening tabungan pada satu bank tertentu. Setiap tabungan dapat diakses melalui kartu debit. Proses utama sistem ATM berkomunikasi dengan pusat komputer dan didesain untuk menangani beberapa transaksi. Setiap transaksi menunjuk sebuah tabungan tertentu. Suatu transaksi akan menghasilkan satu dari dua hal berikut: transaksi diterima atau mengeluarkan pesan penolakan transaksi".

Untuk melakukan sebuah transaksi akan melalui dua tahap: pengecekan tabungan dan pemroses transaksi. Proses pengecekan tabungan akan menetapkan persetujuan untuk proses transaksi. Jika persetujuan ditolak, ATM akan mengeluarkan pesan penolakan, namun jika diterima, transaksi akan diproses de ngan menggunakan nomor rekening tabungan dan ATM membaca dari kartu debit.

Pengecekan tabungan dilakukan bersamaan pada saat ATM memvalidasi kartu debit dari bank yang bersangkutan. Jika kartu valid, password akan dicek dengan nasabah.

Materi 3
Activity Diagram

Tujuan Instruksional	**:** Pengantar
Pokok bahasan ini menjelaskan tentang elemen-elemen dalam Activity Diagram.	
Kompetensi yang Diharapkan	**:**
Praktikan dapat menggambarkan activity diagram.	
Waktu Pertemuan	: 120 menit

3.1 Activity Diagram

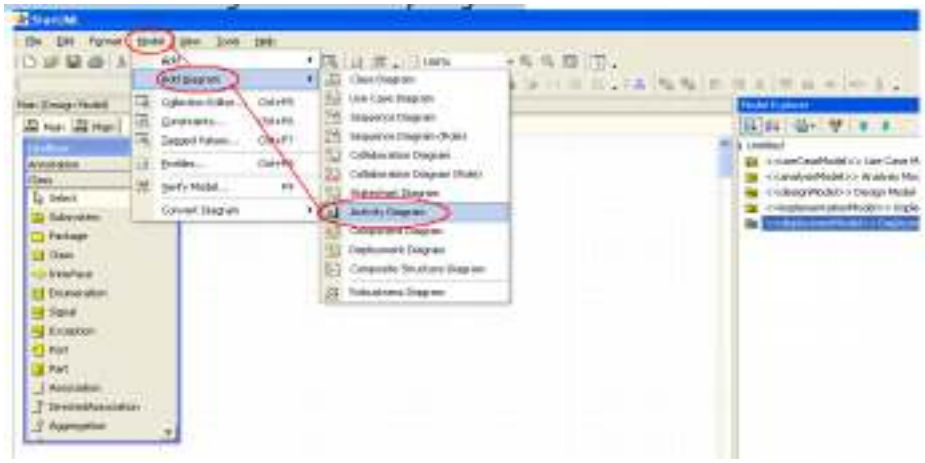
Activity diagram memodelkan workflow proses bisnis dan urutan aktivitas dalam sebuah proses. Diagram ini sangat mirip dengan flowchart karena memodelkan workflow dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Menguntungkan untuk membuat activity diagram pada awal pemodelan proses untuk membantu memahami keseluruhan proses. Activity diagram juga bermanfaat untuk menggambarkan parallel behaviour atau menggambarkan interaksi antara beberapa use case.

3.2 Elemen – Elemen Activity Diagram

1) *Status start* (mulai) dan end (akhir)
2) *Aktifitas* yang merepresentasikan sebuah langkah dalam workflow.
3) *Transition* menunjukkan terjadinya perubahan status aktivitas (Transitions show what state follows another).
4) *Keputusan* yang menunjukkan alternatif dalam workflow.
5) *Synchronization bars* yang menunjukkan subflow parallel. Synchronization bars dapat digunakan untuk menunjukkan concurent threads pada workflow prosesbisnis.
6) *Swimlanes* yang merepresentasikan role bisnis yang bertanggung jawab pada aktivitas yang berjalan.

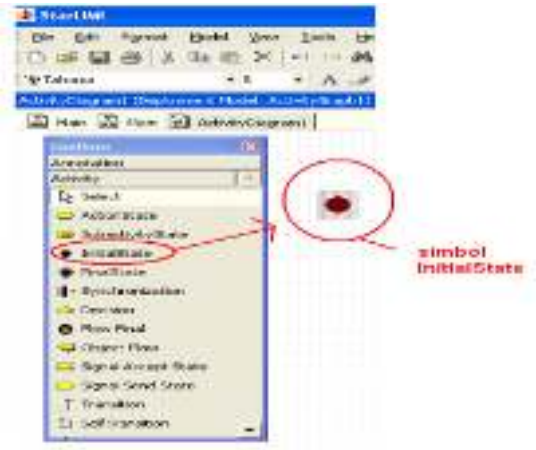
3.3 Cara Membuat Activity Diagram dengan StarUML

1) Pilih model -> add diagram -> activitiy diagram.

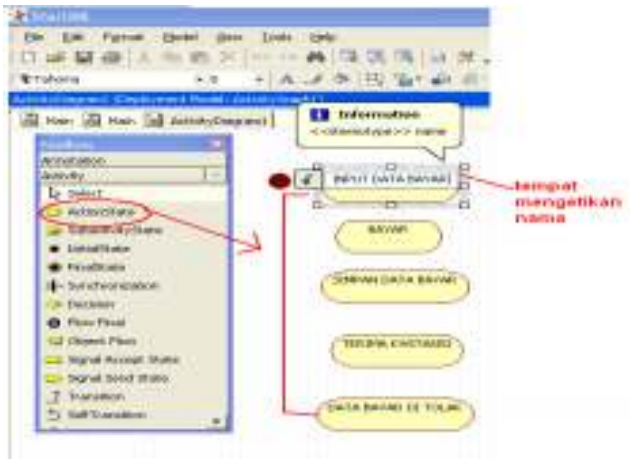


2) Selanjutnya akan muncul toolbox yang berisikan gambar atau simbol yang menjelaskan alur activity diagram.

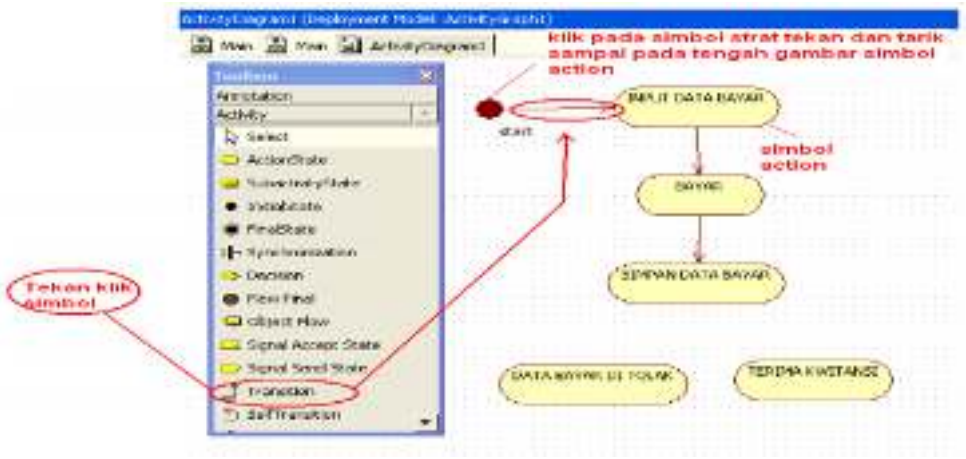
3) Untuk membuat activity diagram diawali dengan memasukkan simbol initial state yang menunjukkan awal dari sebuah alur activity.



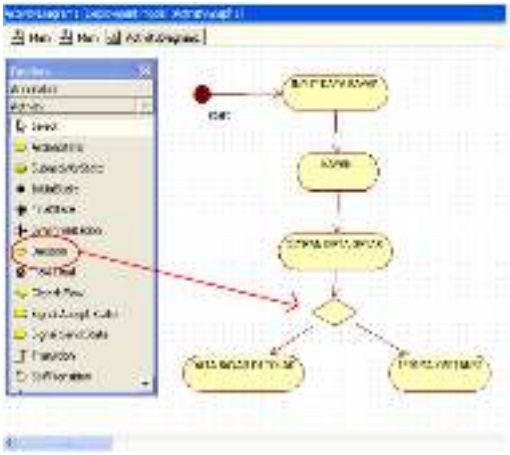
4) Setelah memasukkan simbol initial state pilih simbol action state, beri nama dengan cara klik dua kali pada simbol action.



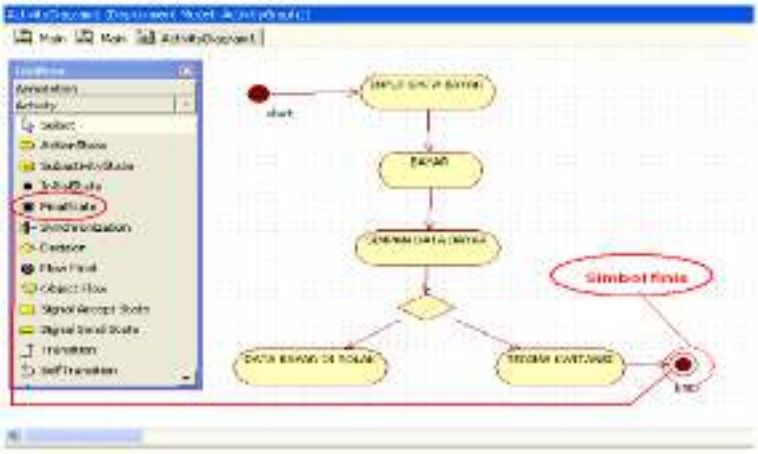
5) Selanjutnya untuk menghubungkan antara simbol, menggunakan garis transition yang terletak pada toolbox.



6) Setelah membuat garis pada activity diagram terdapat simbol decision yang menjelaskan terjadi dua hasil dari sebuah alur.



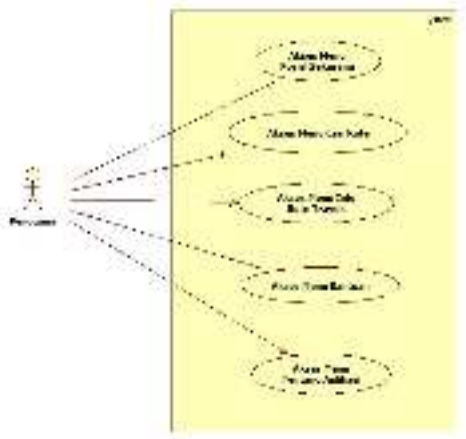
7) Terakhir setelah alur selesai dalam activity wajib menggunakan simbol finalstate yang menjelaskan alur diagram telah selesai.



8) Untuk cara penyimpanan pilih File → save as → dan tentukan tempat penyimpanan file.

TUGAS MODUL 3 !

Buatlah Activity Diagram dari Use Case Diagram Aplikasi Pencarian Rute Angkutan Umum dibawah.



Catatan :

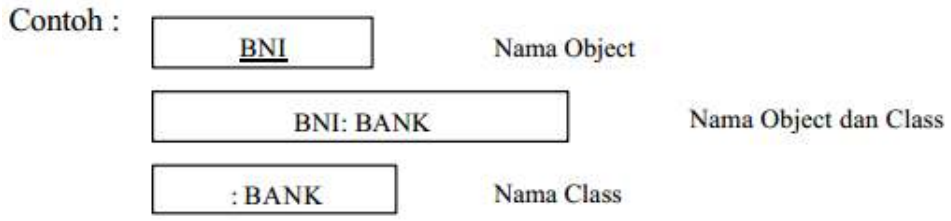
Berikan penjelasan (skenario) pada masing masing Activity Diagram.

Materi 4
Sequence Diagram

Tujuan Instruksional	**:** Pengantar
Pokok bahasan ini menjelaskan tentang Sequence Diagram.	
Kompetensi yang Diharapkan	**:**
Praktikan dapat menggambarkan sequence diagram	
Waktu Pertemuan	: 120 menit

4.1 Sequence Diagram

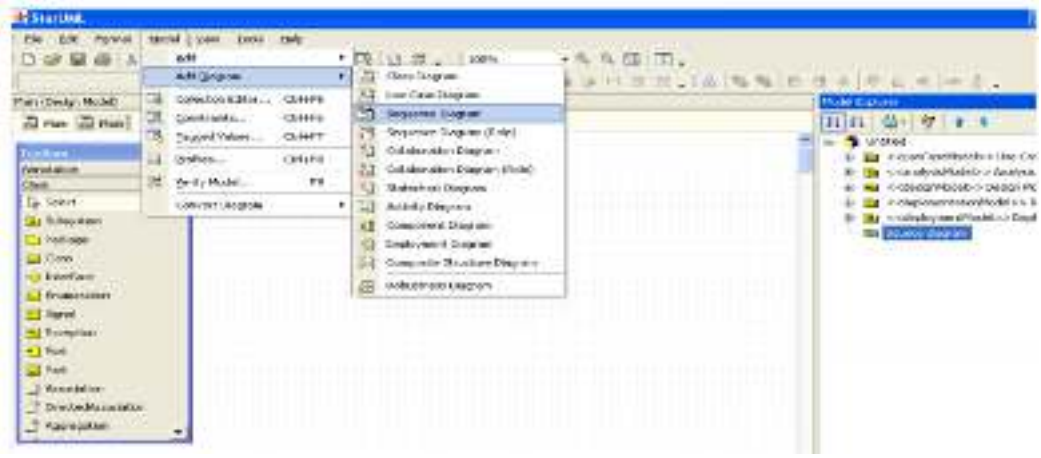
Sequence Diagram menggambarkan interaksi antara sejumlah object dalam urutan waktu. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara object juga interaksi antar object yang terjadi pada titik tertentu dalam eksekusi sistem. Dalam UML, object pada diagram sequence digambarkan dengan segi empat yang berisi nama dari object yang digarisbawahi. Pada object terdapat 3 cara untuk menamainya yaitu : nama object, nama object dan class serta nama class.



Gambar 3. Penamaan Objek

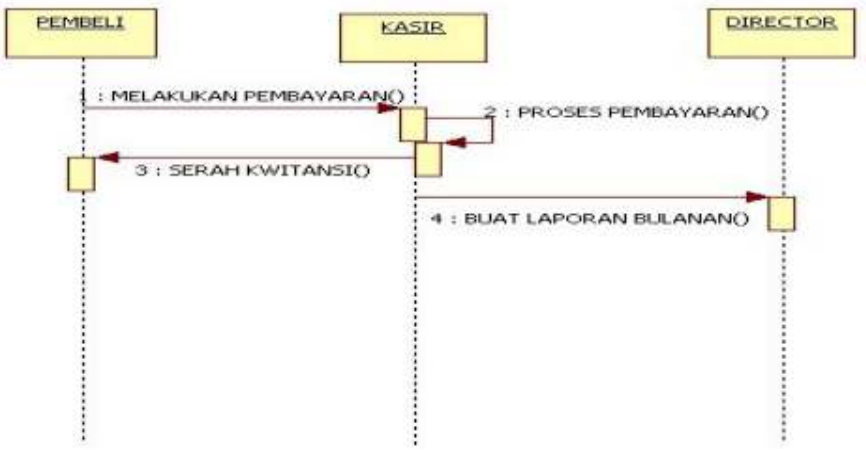
4.2 Cara Membuat Sequence Diagram dengan StarUML

1) Pertama pilih model -> add diagram -> Sequence Diagram.Pertama pilih model -> add diagram -> Sequence Diagram.

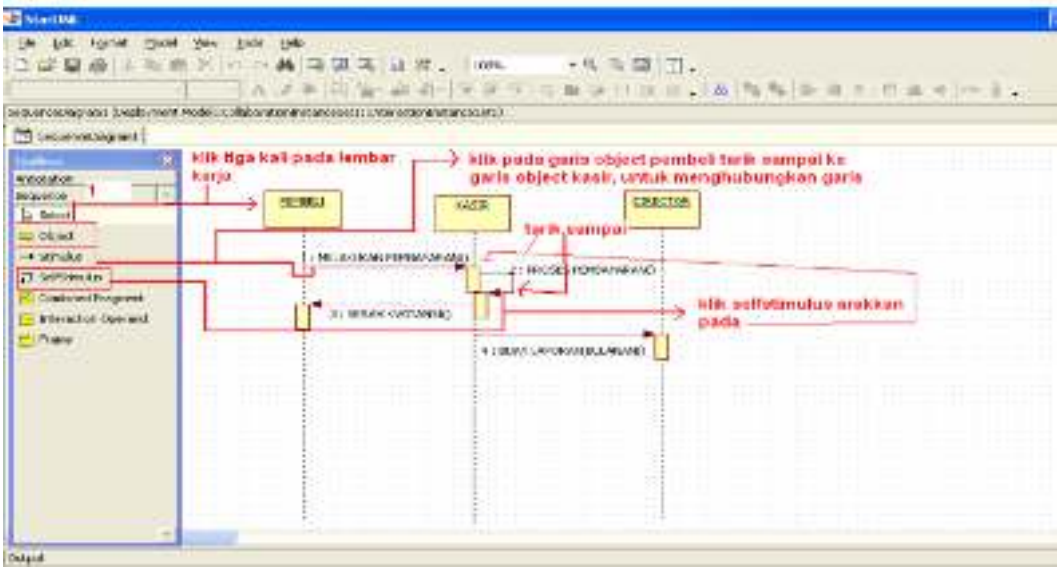


2) Selanjutnya pada toolbox sequence diagram terdapat simbol untuk membuat alur diagram.

3) Berikut adalah contoh dari sequence diagram.



4) Cara membuat alur diatas adalah sebagai berikut:
   - Pembeli dalam alur diagram di atas menggunakan simbol abject yang terletak pada toolbox.
   - Sedangkan untuk membuat garis yang menghubungkan antara object menggunakan stimulasi yang terletak pada toolbox.
   - Untuk memberikan nama pada garis klik dua kali pada garis maka akan muncul tempat untuk mengetik.
   - Dalam object kasir terdapat garis melengkung kebawah yang menunjukkan suatu proses yang disebut setstimulasion.



5) Untuk menyimpan pilih file → save as → dan pilih lokasi penyimpanan.

TUGAS MODUL 4 !

Buatlah Sequence Diagram berdasarkan Use Case Diagram dan Activity Diagram Aplikasi Pencarian Rute Angkutan Umum yang telah dikerjaan di Tugas Modul 3.

Materi 5

Class Diagram

Tujuan Instruksional **:** Pengantar
Pokok bahasan ini menjelaskan tentang Class Diagram.

Kompetensi yang Diharapkan **:**

1. Praktikan bisa memberikan atribut dan operasi pada masing-masing class yang didefenisikan.
2. Praktikan dapat menggambarkan relasi antar class dan tipe-tipenya.
3. Praktikan dapat menggambarkan Class diagram

Waktu Pertemuan : 120 menit

5.1 Definisi Object dan Class

Object adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasanbatasan dan pengertian yang tepat. Object bisa mewakili sesuatu yang nyata seperti komputer, mobil atau dapat berupa konsep seperti proses kimia, transaksi bank, permintaan pembelian, dll. Setiap object dalam sistem memiliki tiga karakteristik yaitu State (status), Behaviour (sifat) dan Indentity (identitas).

Cara mengidentifikasi object:

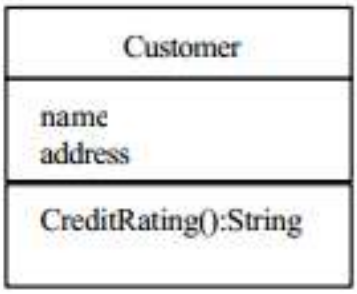
1) pengelompokan berdasarkan kata/frase benda pada skenario.
2) berdasarkan daftar kategori object, antara lain:
   - object fisik, contoh:pesawat telepon
   - spesifikasi/rancangan/deskripsi, contoh: deskripsi pesawat
   - tempat, contoh:gudang
   - transaksi, contoh: penjualan
   - butir yang terlibat pada transaksi, contoh: barang jualan
   - peran, contoh :pelanggan
   - wadah, contoh : pesawat terbang
   - piranti, contoh:PABX
   - kata benda abstrak, contoh: kecanduan
   - kejadian, contoh:pendaratan
   - turan atau kebijakan, contoh:aturan diskon
   - catalog atau rujukan, contoh: daftar pelanggan

Class adalah deskripsi sekelompok object dari property (atribut), sifat (operasi),relasi antar object dan sematik yang umum. Class merupakan template

untuk membentuk object. Setiap object merupakan contoh dari beberapa class dan object tidak dapat menjadi contoh lebih dari satu class.

Penamaan class menggunakan kata benda tunggal yang merupakan abstraksi yang terbaik.

Pada UML, class digambarkan dengan segi empat yang dibagi. Bagian atas merupakan nama dari class. Bagian yang tengah merupakan struktur dari class (atribut) dan bagian bawah merupakan sifat dari class (operasi).



Gambar 4. Class

TUGAS MODUL 5 !

Buatlah Class Diagram dari skenario pada modul 1 untuk studi kasus pada ATM, defenisikan candidate class, dimana candidate class secara kasar dapat diambil dari kata benda yang ada, atau sesuai dengan apa yang telah dijelaskan diatas.