PRAKTIKUM PARADIGMA PEMROGRAMAN

MODUL 8

Disusun oleh :

Nama : Fidelia Ping

NIM : 245410012

Kelas : Informatika 1

PROGRAM STUDI INFORMATIKA

PROGRAM SARJANA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

YOGYAKARTA

2025

MODUL 8
DASAR UNIFIED MODELING LANGUAGE (UML)

A. TUJUAN PRAKTIKUM

Mahasiswa dapat menggunakan UML untuk desain system

B. DASAR TEORI

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.

Bagian-bagian Dari UML

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

1. 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.

  Tujuan Penggunaan UML
  - Memodelkan suatu system (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi object
  - Menciptakan suatu Bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin

  Komponen – komponen yang terlibat dalam usecase diagram :

  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. Actor digambarkan dengan stick man (Gambar 8.1). Actor dapat digambarkan secara secara umum atau spesifik, dimana untuk membedakannya kita dapat menggunakan relationship.

Gambar 8.1 Notasi UML untuk actor

2. use case

   Use case ini merupakan bentuk fungsionalitas dari suatu sistem. Use case juga merupakan dialog antara actor dan sistem. Notasi use case pada UML dapat dilihat pada Gambar 8.2.

Gambar 8.2 Notasi UML untuk Use Case

Cara menentukan use case pada 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

Beberapa relasi yang terdapat pada use case diagram :

1. Association, menghubungkan link antar elemen
2. Generalization, disebut juga inheritance(pewarisan) artinya sebuah elemen merupakan spesialisasi dari elemen lain.
3. Dependency, elemen bergantung dalam beberapa cara pada elemen lain
4. Aggregation, merupakan bentuk asosiasi, dimana elemen dapat berisi elemen lain

Tipe relasi/stereotype antara lain:

1. <<include>> yaitu perilaku yang harus terpenuhi agar sebuah event dapat terjadi, pada kondisi ini use case menjadi bagian dari use case lainnya
2. <<extends>> yaitu perilaku yang hanya berjalan pada kondisi tertentu, misal : jam alarm
3. <<communicates>> biasanya ditambahkan untuk asosiasi yang menunjukkan asosiasinya yaitu communicates association.

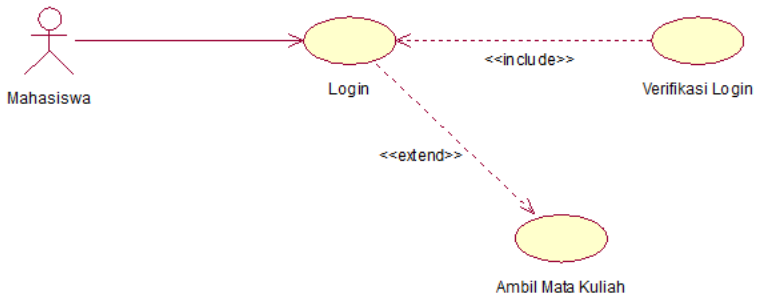
C. PEMBAHASAN LISTING

Praktik

Membuat Diagram Use Case

1. Klik pada use case view package di browser
2. Pilih New → Use Case Diagram, lalu diberi nama : Latihan
3. Double Click pada nama Latihan (pastikan Anda sudah berada di jendela Use Case Diagram).

   Pada praktik kali ini akan dibuat diagram seperti yang ada pada Gambar 8.3



Gambar 8.3 Hubungan Use Case dengan Use case lainnya.

Langkah-langkah yang dapat dilakukan adalah sebagai berikut.

Membuat actor pada Rational Rose

1. Klik gambar Actor  pada toolbar, lalu letakkan pada documentation window
2. Klik gambar Actor tersebut, dan namakan Mahasiswa.
3. Ulangi langkah 2 untuk menambah actor yang diperlukan pada diagram.

Membuat Use Case

1. Klik gambar Use Case  , pada toolbar, lalu letakkan pada documentation window
2. Klik gambar Use Case tersebut, dan namakan Login
3. Ulangi langkah 2 untuk menambah use case pada diagram

Membuat Relasi

1. Klik gambar  , untuk membuat relasi dependency dari actor ke use case.
2. Ulangi no. 1 untuk menambahkan dependency dari actor ke use case lainnya.

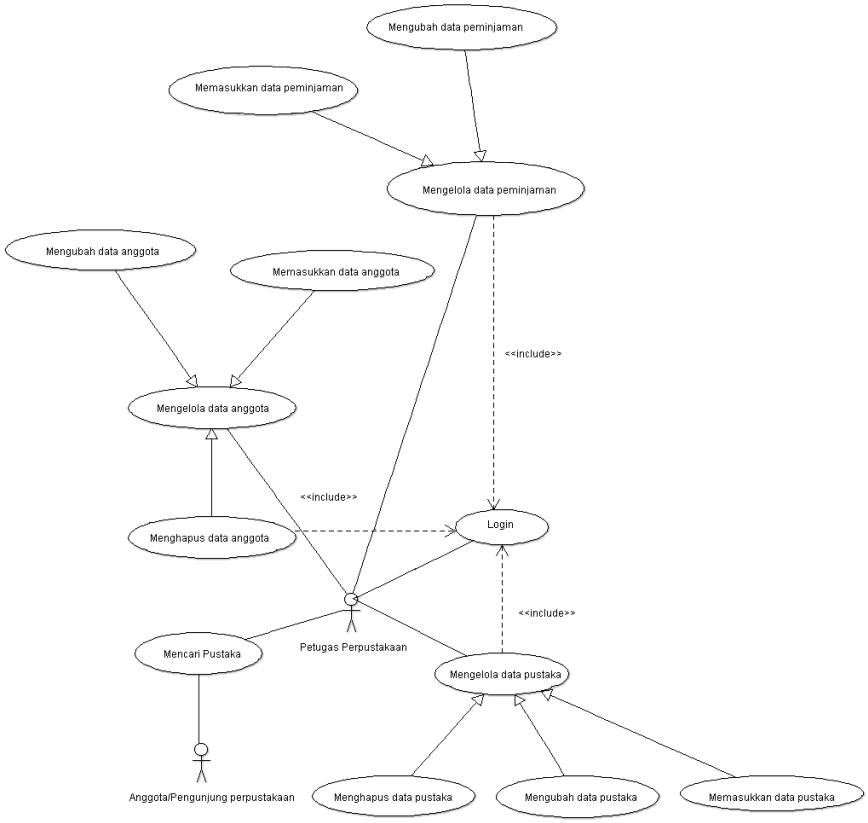
Membuat Relasi Include dan Extend

1. Klik gambar Use Case  kembali pada toolbar, lalu letakkan pada documentation window (Namakan Verifikasi Login)
2. Ulangi langkah 1 (Namakan Ambil Mata Kuliah).
3. Klik gambar  , untuk membuat hubungan dari use case ke use case.

4. Ulangi no. 3 untuk menambahkan dependency dari use case ke use case lainnya.
5. Double click pada relasi dari Use Case : Verifikasi Login ke Use Case : Login, lalu pada Stereotype, pilih <<include>>.
6. Ulangi langkah 5, untuk membuat relasi <<extend>>.dari Login ke Ambil Mata Kuliah, pada Stereotype, pilih <<extend>>.

LATIHAN

Gambarkan diagram usecase berikut menggunakan tools UML



Pembahasan :

- Petugas perpustakaan : orang yang dapat mengakses login, mengelola data peminjaman, mengelola data anggota, mencari Pustaka, dan mengelola data.
- Login : orang yang ingin mengakses data yang ada di haruskan login terlebih dahulu diantaranya jika user ingin melakukan; mengelola data Pustaka, mengelola data peminjaman, mengelola data anggota, dan petugas perpustakaan.
- Anggota/ Pengunjung perpustakaan : mencari Pustaka
- Mengelola data pustaka : di dalamnya terdapat memasukkan data Pustaka, mengubah data Pustaka, dan menghapus data Pustaka.

- Mengelola data anggota : di dalam nya terdapat memasukan data anggota, mengubah data anggota Pustaka, dan menghapus data Pustaka.
- Mengelola data peminjaman : di dalam nya terdapat memasukkan data peminjaman dan mengubah data peminjaman
- <<include>> : menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya

D. KESIMPULAN

UML adalah sebuah bahasa pemodelan yang serbaguna, digunakan untuk memahami, merancang, dan mendokumentasikan sistem dari berbagai aspek dan tidak terbatas pada pemodelan perangkat lunak saja.