BAB 2
TINJAUAN PUSTAKA

2.1 Citra

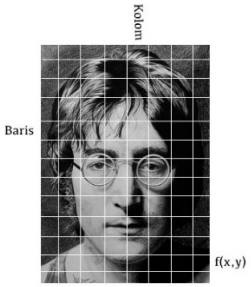
Citra adalah gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinu menjadi gambar diskrit melalui proses sampling [7]. citra dibedakan menjadi dua jenis yaitu citra analog dan citra digital. Citra merupakan keluaran dari suatu sistem perekam data yang bersifat optik, analog maupun digital. Perekaman data citra dibagi menjadi 2 yaitu Citra Analog dan Citra Digital.

2.1.1 Citra Analog

Citra analog adalah citra yang bersifat kontinu, seperti gambar pada monitor televisi, foto sinar-X, foto yang tercetak di kertas foto, lukisan, pemandangan alam, hasil CT scan, gambar-gambar yang terekam pada pita kaset, dan lain sebagainya [7].

2.1.2 Citra Digital

Citra digital adalah gambar dua dimensi yang bisa ditampilkan pada layar komputer sebagai himpunan/diskrit nilai digital yang disebut pixel/picture elements. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahata pada bidang dua dimensi [7]. Suatu citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dianamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x,y dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital [8]. Gambar 2.1 merupakan ilustrasi digitalisasi citra.



Gambar 2. 1 Illustrasi Digitalitas Citra

2.2 Pengolahan Citra Digital

Pengolahan citra digital artinya melakukan pemrosesan gambar berdimenasi dua melalui komputer digital. Pengolahan citra adalah isitlah umum untuk berbagai teknik yang keberadaannya untuk memanipulasi dan memodifikasi citra dengan berbagai cara [8]. Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah. Setiap foto dalam bentuk citra digital dapat diolah menggunakan perangkat lunak tertentu. Misalnya citra yang gelap dapat diolah supaya menjadi lebih terang, merubah format warna menjadi grayscale, resize citra dan lain sebagainya.

2.3 Pattern Recognition

Pattern Recognition (PR) atau pengenalan Pola, merupakan salah satu cabang AI (*artifical intelegence)* [8], yang mempelajari pembuatan sebuah sistem untuk dapat mengenali suatu pola tertentu. Pola dalam hal ini merujuk pada pengertian suatu komposisi stimulus penginderaan yang kompleks yang dapat dikenali oleh manusia sebagai pengamat sebagai suatu kelompok objek. *Pattern Recognition* merupakan proses pengenalan kembali terhadap pola yang pernah dikenal. Oleh karena itu, Ketika gambar yang sama pernah dilihat sebelmunya, maka proses tersebut akan mengenali sebagai sesuatu yang sebelumnya telah dialami.

2.3.1 Penerapan *Pattern Recognition*

Seperti yang telah disebutkan diatas bahwa AI merupakan salah satu cabang Ilmu Komputer. Tapi karena kompleksitas area AI maka dibuatkan sub-sub bagian yang dapat berdiri sendiri dan dapat berdiri sendiri dan saling bekerja sama dengan sub bagian lain atau dengan disiplin ilmu lain [8]. Berikut ini beberapa cabang ilmu sub bagian dari AI :

1. *Character Recognition*

   Salah satu era pengenalan pola yang secara umum menangani permasalahan otomatisasi dan informasi. Sistem OCR mempunyai *front end device* yang terdiri dari lensa scan, document, data transport dan sebuah detector.
2. *Speech Recognition*

Pengenalan pola yang secara umum telah banyak dikembangkan saat ini. Sistem ini mengijinkan kita untuk berkomunikasi antara manusia dengan memasukan data ke komputer. Meningkatkan efisiensi industri manufaktur, mengontrol mesin, dan sebagainya.

3. *Face Recognition*

Salah satu pengaplikasian sistem pengenalan pola yang ditunjukan kepada pengenalan pola wajah, *face recognition* adalah sebuah sistem yang mengenali image wajah manusia yang biasa digunakan untuk otomatisasi dan security sebuah industri.

4. *Machine Vision*

Sebuah sistem mesin yang menggunakan pengenalan pola sebagai dasar dari tahapannya. Mesin ini menangkap sebuah atau sekelompok objek dengan kamera dan selanjutnya dianalisa untuk di deskripsikan.

2.4 Aksara Latin (Abjad)

Pada umumnya tulisan semua Bahasa di Dunia ini ditulis dengan abjad atau secara *alphabet,* walaupun masih banyak lagi Bahasa yang menggunakan symbol-simbol tertentu dalam penulisannya. Dalam tugas akhir ini peneliti mengambil studi kasus yaitu aksara latin (abjad).

1. Alphabet

Menurut kamus Oxford *alphabet* adalah abjad atau huruf yang dimulai dari a sampai z yang biasa kita gunakan untuk menulis sebuah kata atau kalimat. Gamabr 2.2 merupakan kumpulan huruf *alphabet.*

Aa	Bb	Cc	Dd	Ee
Ff	Gg	Hh	Ii	Jj
Kk	Ll	Mm	Nn	Oo
Pp	Qq	Rr	Ss	Tt
Uu	Vv	Ww	Xx	Yy
Zz				

Gambar 2. 2 Alpabet

2.5 Grayscale

Citra grayscale menangani gradasi warna hitam dan putih, yang menghasilkan efek warna abu-abu, warna gambar dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih [8]. Berikut adalah rumus konversi citra berwarna (RGB) menjadi grayscale.

$$I = (0.2989 * R) + (0.5870 * G) + (0.1141 * B) \quad (2.1)$$

Keterangan :

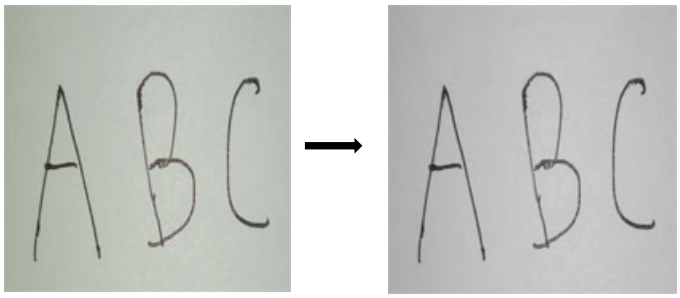
I = fungsi pencarian nilai skala keabu-abuan

R = komponen nilai merah (*Red)* dari suatu titik *pixel*

G = komponen nilai hijau (*Green*) dari suatu titik *pixel*

B = komponen nilai biru (*Blue*) dari suatu titik *pixel*

Persamaan di atas merupakan salah satu rumus yang digunakan untuk mengkonversikan citra berwarna menjadi grayscale. Persamaan 2.1 dipilih dalam penelitian ini karena mata manusia seacara alami sensitif terhadap cahaya merah dan hijau. Maka dari itu, warna-warna ini diberi bobot yang lebih tinggi untuk memastikan bahwa keseimbangan intesitas relatif dalam citra grayscale yang dihasilkan mirip dengan citra warna RGB [9]. Gambar 2. 2 merupakan contoh citra *RGB* yang diubah ke *grayscale.*



Gambar 2. 3 Hasil *Grayscale*

2.6 Sauvola threshold

Sauvola threshold merupakan metode *threshold* yang mampu mendeteksi citra tulisan tangan dengan sangat baik, dengan cara menghitung ambang menggunakan rentang nilai dinamis dari nilai standar deviasi citra *grayscale* [9]. *Sauvola* termasuk dalam *locak threshold* dimana nilai ambang ditentukan oleh nilai pixel tetangga, tujuan dilakukannya proses *threshold* adalah untuk menyederhanakan bentuk citra.

$$T(x,y) = m(m,y) * (1 + k * \left(\frac{s(x,y)}{R} - 1\right)) \quad (2.2)$$

$$m(x,y) = \frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} img(i,j)}{i*j} \quad (2.3)$$

$$s(x,y) = \frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} (img(i,j) - m(x,y))^2}{(i*j)-1} \quad (2.4)$$

Keterangan :

R : nilai maksimum dari standar deviasi (128 untuk citra grayscale)

k : kernel dengan nilai antara 0.2 – 0.5

m : fungsi yang menghasilkan nilai rata-rata dari sejumlah pixel citra.

s : fungsi yang menghasilkan deviasi dari sejumlah pixel citra

T : fungsi yang menghasilkan nilai threshold (ambang)

x : nilai koordinat lebar citra dalam rumus (i)

y : nilai koordinat tinggi citra dalam rumus (j)

Setelah nilai ambang T(x,y) sudah didaptkan, selanjutnya masukkan ke persamaan di bawah ini.

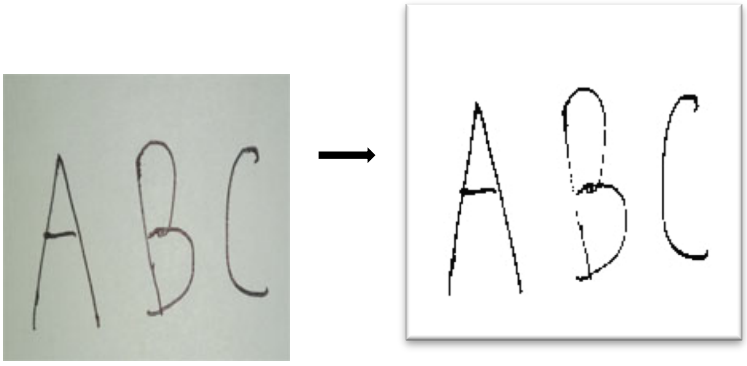
$$f(x,y) = \{^{0, img(x,y) < T(x,y)}_{255, img(x,y) \geq T(x,y)} \quad (2.5)$$

Keterangan :

f: fungsi yang menghasilkan nilai 0 atau 255

img: nilai grayscale citra

Gambar 2. 2 merupakan contoh citra *RGB* yang diubah ke *grayscale.*



Gambar 2. 4 Hasil Savuola Threshold

2.7 Segmentasi Citra

Segmentasi citra adalah teknik untuk membagi citra menjadi beberapa daerah atau region dimana setiap daerah memiliki kemiripan atribut [13]. Menurut Gonzales dan Wintz bahwa segmentasi adalah proses pembagian sebuah citra ke dalam sejumlah bagian atau objek. Segmentasi merupakan suatu bagian yang sangat penting dalam analisis citra secara otomatis, sebab pada prosedur ini obyek yang diinginkan akan disadap untuk proses selanjutnya, misalnya : pada pengenalan pola.

Algoritma segmentasi didasarkan pada 2 buah karakteristik nilai derajat kecerahan citra, yaitu: discontinuity dan similarity. Pada item pertama, citra dipisahkan/dibagi atas dasar perubahan yang mencolok dari derajad kecerahannya. Aplikasi yang umum adalah untuk deteksi titik, garis, area dan sisi citra. Pada kategori kedua, didasarkan

atas tresholding, region growing, dan region spiltting and merging. Prinsip segmentasi citra bisa diterapkan untuk citra yang statis maupun dinamis. Segmentasi citra adalah membagi suatu citra menjadi wilayah-wilayah yang homogen [14]. Beberapa teknik segmentasi antara lain meliputi : pengembangan, penandaan, komponen terhubung, segmentasi berbasis cluster, dan transformasi hough, fokus pada segmentasi citra pada penelitian ini adalah menggunakan pengambangan dan penandaan komponen terhubung.

2.8 Ekstrasi Ciri

Ekstrasi fitur adalah proses pengukuran terhadap data yang telah di normalisasi untuk membentuk sebuah fitur. Nilai fitur digunakan oleh pengklasifikasi untuk mengenali unit masukan dengan unit target keluaran dan memudahkan pengklasifikasian karena nilai ini mudah untuk dibedakan [11].

2.8.1 Directional Element Feature

Directional Element Feature (DEF) merupakan metode ekstrasi ciri yang telah banyak digunakan dan terbukti memberikan hasil yang baik dalam pengenalan tulisan tangan [4]. Metode DEF pada OCR yang melihat perbedaan kontur dan tanpa mengalami proses *skeletonizing* [4].

Berikut adalah tahapan-tahapan yang ada dalam ekstraksi DEF [4]:

1. *Resize*

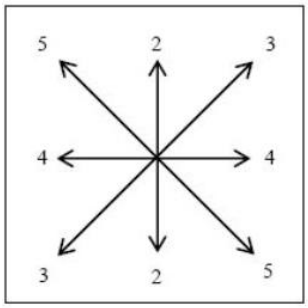
   Resize digunakan untuk merubah ukuran *pixel* sesuai dengan kebutuhan dalam melakukan proses ektraksi ciri. Hal ini dilakukan agar semua citra karakter dari hasil segmentasi mempunyai ukuran yang sama dan menghasilkan ciri yang baik pada proses *dot orientation*.

2. *Dot Orientation*

   Directional Element Feature adalah pencarian nilai *feature* berdasarkan label arah dari sebuah pixel. Pada metode ini setiap pixel f*oreground*(piksel karakter) pada citra digital akan diberikan label (*feature* arah) dengan cara

menelusuri tetangga dari masing-masing pixel *foreground* searah perputaran jarum jam dimana arah yang digunakan terdiri dari 4 arah.

Besar nilai arah yang digunakan bias dilihat pada gambar Contoh pembagi 3 zona pada citra biner dapat dilihat pada gambar 2.4.



Gambar 2. 5 Nilai Arah DEF

Untuk melakukan pelabelan arah pada masing-masing *pixel* dapat dilakukan dengan cara sebagai berikut [4]:

a. Lakukan pengecekan secara raster dari kiri ke kanan.
b. Apabila menemukan sebuah *pixel foreground* maka lakukan pengecekan dengan melihat tetangga dari *pixel* tersebut.
c. Perhatikan gambar 2.5 Misalkan 0 adalah sebuah *foreground,* maka nilai arah 0 didapatkan dengan melakukan pengecekan secara berurutan dari X1-X8. pengecekan berhenti ketika menemukan X yang merupakan *foreground*, maka ubahlah nilai 0 menjadi nilai arah berdasarkan aturan di bawah ini:
    a) Jika posisi X1 atau X5 maka nilainya 5
    b) Jika posisi X2 atau X6 maka nilainya 2
    c) Jika posisi X3 atau X7 maka nilainya 3
    d) Jika posisi X4 atau X8 maka nilainya 4

X1	X2	X3
X8	O	X4
X7	X6	X5

Gambar 2. 6 Nilai Arah DEF

3. *Vector Construction*

Tahapan ini untuk menghitung nilai dari setiap *pixel* ketetanggaan yang sebelumnya telah melalui proses *dot orientation,* awalnya citra akan dibagi menjadi N_1 *x* N_1 area yang saling overlap sebanyak $N_1/2$ *pixel* terhadap area yang bertetanggaan.

Selanjutnya area, setiap area dibagi kembali menjadi 4 *sub-area,* yaitu A, B, C dan D, yang eksklusif satu sama lainnya. Setelah diperoleh jumlah pixel di setiap *sub-area,* akan dikalikan masing-masing jumlah elemen ketetanggan disetiap *sub-area* dengan bobot(*w*) pada *sub-area* tersebut dan dijumlahkan dari setiap elemen di *sub-area* dengan rumus sebagai berikut[4]:

$$x_j = w_A x_j + w_B x_j + w_C x_j + w_D x_j \qquad (2.6)$$
$$j = 1,\ldots,4$$

Dimana j merupakan pixel ketetanggaan.

Selanjutnya vektor ciri DEF yang diperolh dengan menggabungkan jumlah elemen disetiap area menjadi satu kolom vektor *V*.

$$V = [X_1^1, X_2^1, X_3^1, X_4^1, \ldots, X_1^N, X_2^N, X_3^N, X_4^N] \qquad (2.7)$$

Dimana *N* merupakan jumlah area dari setiap karakter. Vektor *V* inilah yang nantinya jadi input untuk proses *learning* dan *testing.*

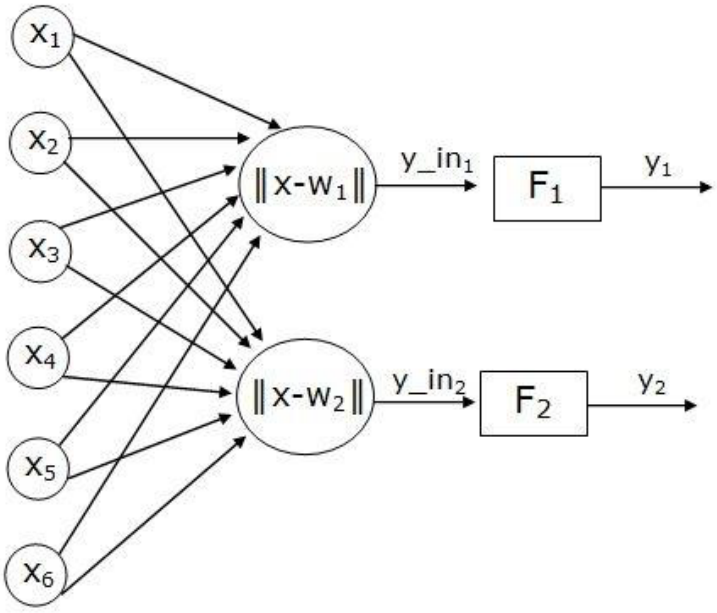
2.9 Teknik *Learning* (Belajar)

Teknik *learning* merupakan salah satu bagian dari ilmu kecerdasan buatan. Dalam learning kita tidak harus tahu anturan yang berlaku dalam sistem yang dibuat, melainkan aturan yang diharapkan bisa diharapkan secara otomatis ditemukan. Proses belajar dalam teknik learning menggunakan data-data masukan sebagai pengalaman y

ang baru untuk dapat meningkatkan performasi dari sistem. Program komputer yang sanggup belajar adalah program yang bisa meningkatkan performasinya melalui pengalaman [11]. Ada beberapa metode yang menggunakan teknik *learning* seperti *decision tree,* jaringan saraf tiruan, algoritma genetika dan lain-lain.

2.9.1 Learning Vector Quantization (LVQ)

Banyak metode yang bisa digunakan untuk pengenalan citra tulisan tangan, antara lain metode Artificial Neural Network, Backpropagation, Hough Transform, dan Support Vector Machine. Dari beberapa metode yang disebutkan metode Learning Vector Quantization masih belum banyak digunakan. Learning Vector Quantization(LVQ) merupakan jaringan syaraf dengan tipe arsitektur jaringan lapis-tunggal umpan-maju (Single Layer Feedforward) yang terdiri atas unit masukan dan unit keluaran. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor masukan [2]. Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitifini hanya tergantung pada jarak antara vektor vektor masukan. Jika 2 vektor masukan mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor masukan tersebut ke dalam kelas yang sama [2].



Gambar 2. 7 Arsitektur LVQ

$$Cj = \sqrt{\sum_{i=1}^{n}(X_i - W_j)^2} \quad (2.8)$$

Dimana:

C_j = nilai jarak Euclidean

X_i = nilai vector input

W_j = nilai vector bobot

Selain itu LVQ selalu memperbaharui bobotnya sesuai jumlah maksimal epoch yang ditetapkan. Epoch yaitu satu siklus pelatihan yang melibatkan semua pola. Persamaan 2 merupakan rumus bobot LVQ yang digunakan untuk memperbaharui nilai bobot:

$$\overrightarrow{W_m} \leftarrow \overrightarrow{W_m} + \alpha \,.\, (\overrightarrow{X} - \overrightarrow{W_m}) \quad (2.9)$$

Dimana:

W_m = bobot

α = *learning rate*

$\overrightarrow{X}$ = input

2.10 Bahasa pemograman

Bahasa pemograman diperlukan untuk menjalankan instruksi-instruksi apa yang harus dilakukan komputer. Komputer tidak bisa memahami bahasa manusia, sehingga diperlukan pengguna bahasa komputer di dalam program komputer. Penelitian ini menggunakan bahasa Java untuk mengembangkan aplikasi dan bahasa pemograman SQL untuk mengelola *database.*

2.10.1 Java

Bahasa Java dikembangkan oleh Sun Microsystem. Java awalnya dikenal dengan Oak, yang didesain pada tahun 1991 untuk chip-chip yang tertanam pada peralatan-peralatan elektronik. Pada tahun 1995, diberi nama baru Java, yang didesain ulang untuk mengembangkan aplikasi-aplikasi internet. Java telah menjadi sangat populer. Perkembangannya yang sangat cepat dan penerimaannya di kalangan pengguna dapat dijejak dari karakteristik perancangannya [17].

Java memiliki banyak fitur, bahasa pemrograman bertujuan-umum yang dapat digunakan untuk mengembangkan aplikasi-aplikasi tingkat tinggi. Saat ini, Java tidak lagi hanya digunakan untuk pemrograman Web, tetapi juga aplikasi-aplikasi *standalone* bebas *platform* pada *server*, *desktop*, dan *devices* bergerak (*mobile*) [17].

Java merupakan bahasa pemograman yang tangguh terbukti handal pada banyak aplikasi. Terdapat tiga edisi java [17], yaitu :

1. Java SE (*Standar Edition*)
   Digunakan untuk mengembangkan aplikasi-aplikasi pada sisi client atau *applet.*
2. Java EE (*Enterprise Edition*)
   Digunakan untuk mengembangkan aplikasi-aplikasi pada sisi *server,* seperti *Java Servlets* dan *Java server pages.*

3. Java ME (*Micro Edition*)
   digunakan untuk mengembangkan aplikasi-aplikasi untuk *device* bergerak, seperti telepon genggam.

Peneletian ini menggunakan Java SE untuk mendapatkan aplikasi pengenalan tulisan tangan.

2.11 UML (Unified Modeling Language)

Unified Modeling Language (UML) adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem komputer. Didalam pemodelan objek guna menyajikan sistem yang berorientasi pada objek dan pada orang lain, akan sangat sulit dilakukan jika pemodelan tersebut dilakukan dalam bentuk kode bahasa pemrograman [10].

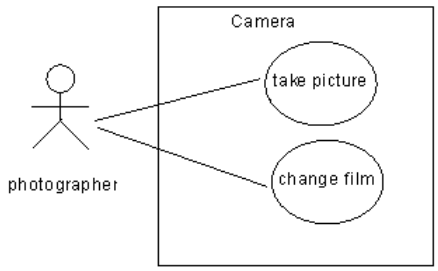
UML disebut sebagai bahasa pemodelan bukan metode. Bahasa pemodelan (sebagian besar grafik) merupakan notasi model yang digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. UML merupakan bahasa standar untuk penulisan *blueprint software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian. Alat-alat dari sistem perangkat lunak. UML biasanya disajikan dalam bentuk diagram atau gambar yang meliputi *class* beserta atribut dan operasinya, serta hubungan antara kelas. UML terdiri dari banyak diagram diantaranya *use case diagram , activity diagram, class diagram,* dan *squence diagram*.

2.11.1 Use Case Diagram

Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan *use case diagram* yang sesungguhnya merupkan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya.

Use Case Diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para akor dengan sistem [10]. Dalam hal ini, setiap objek yang berinteraksi dengan sistem merupakan aktor untuk sistem berprilaku kepada aktronya. Aktor dalam

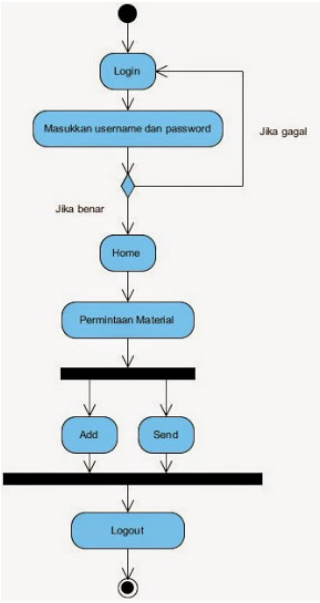
use case diagram digambarkan sebagai ikon yang berbentuk manusia, sementara *use case* digambarkan sebagai ikon yang berbentuk manusia, sementara *use case* digambarkan elips yang berisi nama *use case* yang bersangkutan. Untuk mempermudah pemahaman, aktor biasanya dituliskan sebagai kata benda, sementara *use case* biasanya dituliskan dengan kata kerja. Gambar 2.7 merupakan contoh *usecase diagram.*



Gambar 2.8 Contoh Penggunaan Use Case Diagram

2.11.2 Activity Diagram

Activity Diagram pada dasarnya menggambarkan skenario secara grafis. *Activity Diagram* cukup serupa dengan diagram alur (*flow chart*), yang membedakan hanya *swimlane* yang menunjukkan suatu *state* berada pada objek/kelas tertentu. Keunggulan dari *activity diagram* adalah lebih mudah dipahami dibanding skenario. Selain itu, dengan menggunakan *activity diagram,* kita dapat melihat dibagian manakah sistem dari suatu skenario akan berjalan [10].



Gambar 2.9 Contoh *Activity Diagram*

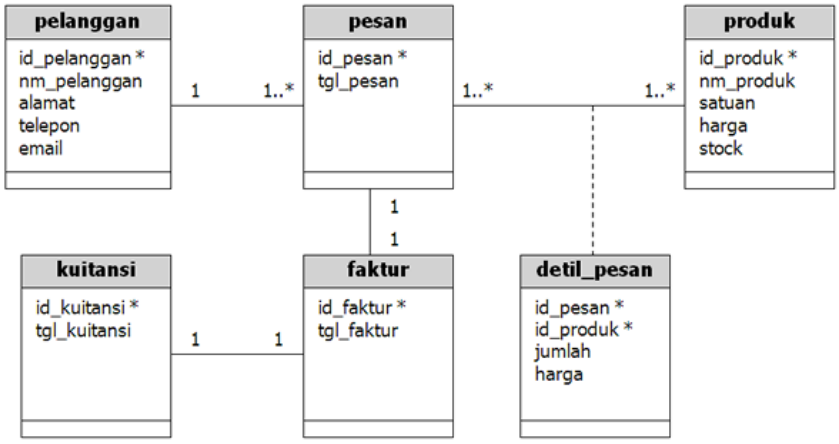
Dari gambar 2.8 di atas merupakan contoh *activity diagram* dengan alur :

1. Lingkaran hitam penuh menandakan memulai aktivitas
2. Proses memasukkan *username* dan *password*.
3. Belah ketupat menandakan sebuah kondisi apabila terpenuhi akan menuju *home* bila tidak akan kembali ke proses login.
4. Masuk ke proses fork untuk menjalankan aktivitas secara pararel kemudian menjalankan aktivitas di bawahnya
5. Masuk ke proses join untuk menggabungkan proses yang dipisahkan oleh fork
6. Melakukan aktivitas *logout*
7. Lingkaran hitam sebagian menandakan akhir dari suatu aktivitas

2.11.3 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem [10]. *Class Diagram* menggambarkan struktur dan deskripsi kelas, package, dan objek beserta

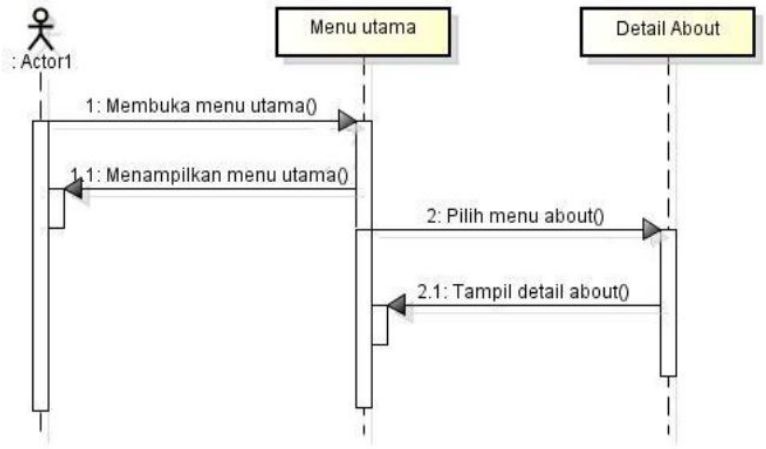
hubungan satu sama lain seperti *containment,* pewarisan, asosiasi dan sebagainya. Gambar 2.9 merupakan contoh *class diagram.*



Gambar 2. 10 Contoh *Class Diagram*

2.11.4 Sequence Diagram

Diagram seakan menggambarkan kelakuan/perilaku objek pada *use case,* dengan mengdeskripsikan waktu hidup objek dan message yang dikirimkan dan diteruma antar objek [10]. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Oleh karena itu untuk menggambarkan diagram skuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.



Gambar 2. 11 Contoh *Squence Diagram*

2.12 Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan-perbedaan antara hasil yang diterapkan dengan hasil yang terjadi [15].

Pengujian seharusnya meliputi tiga konsep berikut:

1. Demonstrasi validasi perangkat lunak pada masing-masing tahap disiklus pengembangan sistem.
2. Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.
3. Pemeriksaan perilaku sistem dengan mengeksekusi sistem pada data sampel pengujian.

Pada dasarnya pengujian diartikan sebagai aktivitas yang dapat atau hanya dilakukan setelah pengkodean (kode program selesai). Namun, pengujian seharusnya dilakukan dalam skala lebih luas. Pengujian dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan. Evaluasi terhadap spesifikasi dan perancangan juga merupakan teknik pengujian. Kategori pengujian dapat dikategorikan menjadi dua [15], yaitu :

1. Berdasarkan ketersediaan logik sistem, terdiri dari *Black box testing* dan *White box testing.*
2. Berdasarkan arah pengujian, terdiri dari pengujian *top down* dan Pengujian *bottom up.*

2.12.1 Pengujian *Black Box*

Konsep *black box* digunakan untuk mempresentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Di dalam *black box,* item-item yang diuji dianggap "gelap" karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari *black box* [10].

1. *Graph-based testing*
2. *Equivalence partitioning*
3. *Comparison testing*
4. *Orthogonal array testing*

Pada pengujian *black box,* akan dicoba beragam masukan dan memeriksa kelauran yang dihasilkan. Sehingga dapat mempelajari apa yang dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan. Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis skenario, dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain[10].

2.12.2 Pengujian Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* dan *reference value*). Tingkat akurasi diperoleh dengan persamaan sebagai berikut[10].

$$Akurasi = \frac{jumlah\ karakter\ sama}{jumlah\ seluruh\ karakter}\ X\ 100\% \qquad (2.10)$$