BAB 2
LANDASAN TEORI

2.1 Alat Musik

Bunyi musik juga dapat ditampilkan melalui permainan alat-alat musik, seperti gitar, piano, *recorder*, pianika. Sama seperti ketika bernyayi, ketika bermain alat musik, seseorang sebetulnya sedang membunyikan alunan nada-nada yang apabila dimainkan dengan baik akan terdengar indah. Berdasarkan pengelompokanya alat musik dibagi menjadi tiga bagian, diantaranya adalah: berdasar atas cara memainkan alat musik, berdasar atas sumber bunyi alat musik, berdasar atas fungsi alat musik [1].

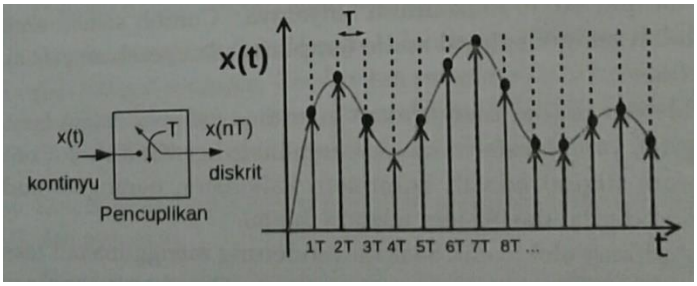
Berdasarkan sumber bunyi yang dihasilkan alat musik dibagi menjadi lima [1] yaitu sebagai berikut:

1. *Aerophone* yaitu alat musik yang sumber bunyinya berasal dari udara yang bergetar dengan cara ditiup atau di pompa. Contoh: flute, seruling, rekorder, tuba, melodeon, klarinet, saksopon dan *horn* yang cara memainkanya ditiup. Akordin dengan dipompa
2. *Idiophone* yaitu alat musik yang sumber bunyinya berasal dari batang logam atau kayu yang dipukul atau sumber bunyinya berasal dari alat itu sendiri. Contoh: bellira, calung, angklung, kulintang, perangkat gamelan.
3. *Elektrophone* yaitu alat musik yang sumber bunyinya berasal dari rangkain elektronika yang terdapat di dalam alat tersebut. Perkembangan teknologi saat ini membuat alat ini bias mengeluarkan suara dari berbagai jenis alat musik. Contoh: *keyboard*, Organ elektrik, Gitar elektrik, bass elektrik dan lain-lain.
4. *Membranophone* yaitu alat musik yang sumber bunyinya bersumber pada getaran pada selaput tipis yang terbuat dari kulit atau plastic yang berbunyi dengan cara dipukul. Contoh: tamborin, rebana, bedug, drum set, ketipung, bongo, konga, timpani dan lain-lain.

5. *Cordophone* yaitu alat musik yang sumber bunyinya berasal dari rangkaian dawai yang bergetar Karena dipetik, digesek maupun ditekan.
    1) Di petik: gitar, harpa mandolin, ukulele, banjo, siter, kecapi bas dan lain-lain.
    2) Di gesek: biola, viola, double bass, rebab
    3) Di tekan: piano akustik

2.2 Sinyal Analog dan Digital

Sinyal dapat dikelompokan menjadi sinyal kontinyu dan sinyal disktrit. Amplitudo sinyal kontinyu terdefinisi untuk semua waktu *t*, sedangkan amplitude sinyal disktrit terdefinisi untuk nilai *t* tertentu. Sinyal disktrit diperoleh dengan mencuplik (mengambil sampel) sinyal kontinyu di waktu-waktu tertentu. Pencuplikan dilakukan secara periodik dengan periode *T* (Gambar 2.1). Pencuplikan sinyal kontinyu x(*t*) dilakukan dengan memutus sambungkan saklar dengan periode *T*. untuk menghasilkan sinyal diskrit x(n*T*).



Gambar 2.1 Alur Proses MFCC

Sinyal kontinyu dapat dinyatakan dengan x(t) dan sinyal diskrit dinyatakan dengan x(*n*T) atau disingkat x[n], dengan n bilangan bulat dan T periode pencuplikan. Amplitudo sinyal diskrit x(*n*T) hanya terdefinisi di T detik, 2T, 3T dan seterusnya.

Sinyal analog adalah sinyal kontinyu yang mempunyai amplitudo kontinyu dalam jangkauan tertentu. Dengan kata lain, sinyal analog ini mempunyai karakteristik waktu dan amplitudo kontinyu.

Sinyal digital adalah sinyal diskrit yang mempunyai jangkauan amplitudo diskrit (terbatas). Dengan kata lain, sinyal digital ini mempunyai karakteristik waktu dan amplitudo yang diskrit. Waktu diskrit diperoleh dengan proses yang disebut kuantisasi [5].

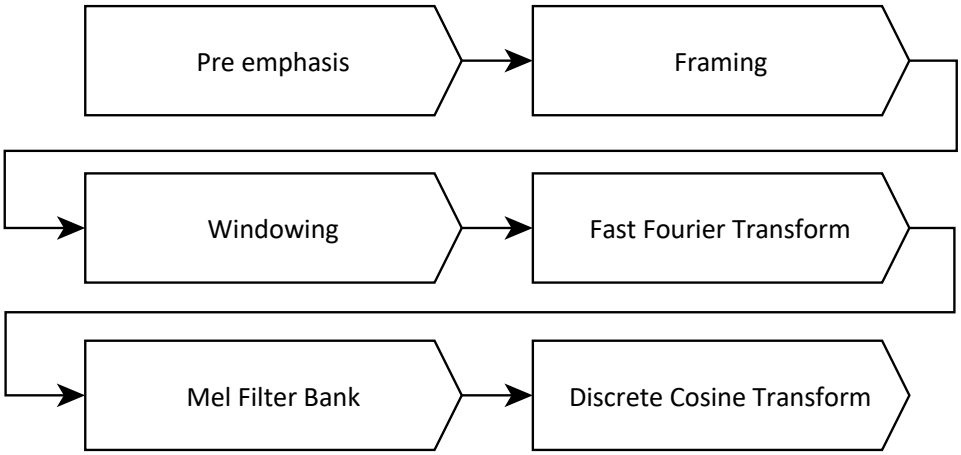
2.3 Konversi sinyal Analog ke Digital

ADC (*Analog-to-Digital Converter)* mengkonversikan tegangan masukan analog menjadi kode digital dalam bentuk biner n bit berdasarkan tegangan referensinya. Konverter analog ke digital (ADC) menghasilkan kode digital n bit sesuai amplitudo tegangan masukan analog relatif terhadap tegangan referensinya. Proses konversi analog ke digital dilakukan dalam 3 tahap, yaitu pencuplikan (*sampling*), kuantisasi dan pengkodean.

1. Pencuplikan (*Sampling*) yaitu mengubah sinyal waktu kontinyu menjadi sinyal waktu diskrit.
2. Kuantisasi yaitu mengubah amplitudo sinyal kontinyu menjadi amplitudo diskrit, hasil proses pencuplikan dan kuantisasi adalah sinyal digital (waktu diskrit, amplitudo diskrit) dari sinyal analog (waktu kontinyu, amplitudo kontinyu).
3. Pengkodean yaitu mengkodekan sinyal digital dalam representasi binernya (digital) [5].

2.4 Mel Frequency Cepstral Coefficients

MFCC metode yang digunakan dalam mengekstraksi ciri dari suara, hasil dari ekstraksi merupakan nilai vektor yang dapat dijadikan sebagai parameter pada algoritma klasifikasi. Dalam melakukan ekstraksi ciri menggunakan metode MFCC terdiri dari beberapa tahap *Pre-Emphasis, Framing, Windowing, FFT, Mel Filter Bank, DCT* [6] [7]. Adapun gambar alur dari proses MFCC sebagai berikut:



Gambar 2.2 Alur Proses MFCC

2.4.1 Pre-Emphasis

Pada langkah ini akan dilakukan proses sinyal melalui filter yang menekankan frekuensi yang lebih tinggi. [5]. Pada penelitian sebelumnya α = 0.95 [5] Adapun persamaan dari *pre-emphasis* adalah sebagai berikut [5]:

$$Sa[n] = s[n] - \alpha . s[n-1] \quad \textbf{(2.1)}$$

Dimana:

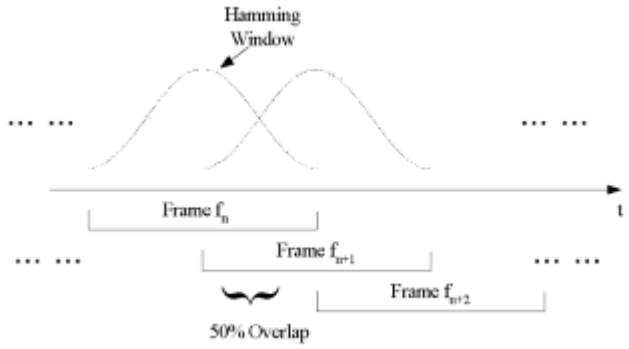
$Sa[n]$ = Hasil dari *pre-emphasis*

$s[n]$ = Sampel sebelum dilakukan *pre-emphasis*

α = 0.95

2.4.2 Framing

Proses segmentasi sampel suara yang diperoleh dari *analog to digital conversion* (ADC) menjadi *frame* yang lebih kecil dengan panjang dalam kisaran 20 sampai 40 ms. Tujuan dari *framing* adalah untuk menghindari diskontinuitas sinyal, Karena diskontinuitas sinyal dapat menyebabkan ekstraksi parameter yang salah selama analisis. Untuk menghindari diskontinuitas sinyal pada dua frame berturut-turut, setiap dua frame saling tumpang tindih (*overlapping*) [5]. Pada penelitian sebelumnya telah dilakukan *framing* 20ms, *overlaping* 10ms [9]. Proses dari *framing* dapat dilihat pada Gambar 2.3 [9]:



Gambar 2.3 Gambar proses *framing*

Adapun persamaan dari *framing* adalah sebagai berikut [10]:

$$Ns = sample\ rate\ \times\ durasi\ framing \quad \textbf{(2.2)}$$

Dimana:

Ns = Jumlah dari *sample point*

$durasi\ framing$ = 20ms (0.020)

$$No = sample\ rate\ \times\ durasi\ overlapping \quad \textbf{(2.3)}$$

Dimana:

No = Jumlah dari *overlapping*

$durasi\ overlapping$ = 10ms (0.010)

$$Nf = \frac{sample\ rate}{Ns} \quad \textbf{(2.4)}$$

Dimana:

Ns = jumlah *sample point*

Nf = Jumlah *frame rate*

2.4.3 Windowing

Pada langkah selanjutnya dalam pengolahan MFCC adalah melakukan *windowing*, setiap individu *frame* akan meminimalkan diskontinuitas sinyal pada awal dan akhir frame. Konsep ini adalah untuk meminimalkan penyimpangan spektral dengan menggunakan *window* dengan melancipkan sinyal menjadi nol

pada awal dan akhir setiap *frame* [11]. Adapun persamaan dari *windowing* adalah sebagai berikut [11]:

Persamaan (2.5) adalah proses perhitungan *hamming window*, setelah didapatkan hasil *hamming window* maka selanjutnya menghitung *windowing* dengan persamaan (2.6).

$$Wham[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{\mathrm{Ns}-1}\right) \quad , \qquad 0 \le n \le Ns-1 \tag{2.5}$$

Dimana:

$Wham[n]$ = Hasil dari *Hamming Window*

Ns = Jumlah *sample point*

$$Sb[n] = Sa[n] \times Wham[n], \quad 0 \le n \le Ns-1 \tag{2.6}$$

Dimana:

$Wham[n]$ = Nilai dari *Hamming windowing*

$Sa[n]$ = Sampel setelah dilakukan *pre-emphasis*

$Sb[n]$ = Hasil dari *windowing*

2.4.4 Fast Fourier Transform

Pada langkah selanjutnya adalah FFT, mengubah setiap frame sampel N dari domain waktu menjadi domain frekuensi. FFT adalah algoritma cepat untuk mengimplementasikan *Discrete Fourier Transform (DFT)* [11]. Adapun persamaan FFT sebagai berikut [11]:

$$F[k] = \sum_{n=0}^{Ns-1}\left(\mathrm{Sb[n]}\cos\frac{2\pi nk}{Ns}\right) - j\sum_{n=0}^{Ns-1}\left(\mathrm{Sb[n]}\sin\frac{2\pi nk}{Ns}\right), k = 0,1,2,\dots,Ns-1 \tag{2.7}$$

Dimana:

$F[k]$ = Hasil dari FFT

$Sb[n]$ = Sampel setelah dilakukan *windowing*

Persamaan (2.8) digunakan untuk mendapatkan hasil dari FFT.

$$Sc[n] = \left|[R^2 + I^2]^{1/2}\right| \tag{2.8}$$

2.4.5 Mel Filter Bank

Penelitian *psychophysical* telah menunjukkan bahwa persepsi manusia tentang frekuensi sinyal suara tidak mengikuti skala linier. Jadi untuk setiap nada dengan frekuensi aktual f diukur dalam Hz, nada subyektif diukur pada skala yang disebut skala 'mel'. Skala frekuensi mel adalah jarak frekuensi linear di bawah 1000 Hz dan jarak logaritmik di atas 1000 Hz. [11]. Pada penelitian sebelumnya jumlah *Filter Bank* 30, *Low mel frequency* 130Hz, *High mel frequency* 6800Hz [12]. Adapun persamaan *mel filter bank* sebagai berikut [6]:

$$mel(f) = 2595\, log_{10}\left(1 + \frac{f}{700}\right) \quad \textbf{(2.9)}$$

Dimana:

$mel(f)$ = Hasil *mel filter*

$$mel^{-1}(f) = 700 \times \left(10^{\frac{f}{2595}} - 1\right) \quad \textbf{(2.10)}$$

Dimana:

$mel^{-1}(f)$ = Hasil *inverse mel filter*

$$f_{b(i)} = \left(\frac{Ns}{Fs}\right) \times\ mel^{-1}\left(mel(Nlow) + i\frac{mel(Nhigh) - mel(Nlow)}{Nfbank - 1}\right) \quad \textbf{(2.11)}$$

Dimana:

$f_{b(i)}$ = Hasil *boundary point filter*

$mel(f)$ = Hasil *mel filter*

$mel^{-1}(f)$ = Hasil *inverse mel filter*

$Nlow$ = Jumlah *mel* terendah 130Hz

$Nhigh$ = Jumlah *mel* tertinggi 6800Hz

$$h(i,k) = \begin{cases} 0, & untuk\ k < f_{b(i-1)} \\ \dfrac{k - f_{b(i-1)}}{f_{b(i)} - f_{b(i-1)}}, & untuk\ f_{b(i-1)} \le k < f_{b(i)} \\ \dfrac{f_{b(i+1)} - k}{f_{b(i+1)} - f_{b(i-1)}}, & untuk\ f_{(i)} \le k < f_{b(i+1)} \\ 0, & untuk\ k > f_{b(i+1)} \end{cases} \quad \textbf{(2.12)}$$

Dimana:

$f_{b(i)}$ = Hasil *boundary point filter*

$h(k,i)$ = Koefisien dari *mel filter bank*

$$Sd[i] = \sum_{k=0}^{Ns-1} |Sc[i]|^2 \times h(i,k)\ , i = 1,2 \dots Nfbank \quad \textbf{(2.13)}$$

Dimana:

$Sc[k]$ = Sampel setelah dilakukan FFT

$Sd[k]$ = Hasil *Mel Filter Bank*

$h(k,i)$ = Koefisien dari *mel filter bank*

$Nfbank$ = Jumlah *Mel Filter Bank* 30

2.4.6 Discrete Cosine Transform

Ini adalah proses untuk mengubah *log Mel spectrum* menjadi domain waktu menggunakan *Discrete Cosine Transform (DCT)*. Hasil konversi tersebut disebut *Mel Frequency Cepstrum Coefficient*. Himpunan koefisien disebut vektor akustik. Oleh karena itu, setiap ucapan masukan ditransformasikan menjadi urutan vektor akustik [5]. Adapun persamaan DCT sebagai berikut [6]:

$$Se[n] = \sqrt{\frac{2}{Nbank}} \sum_{k=0}^{Nbank-1} \log(Sd[k+1]) \times \cos\left[n\left(\frac{2k-1}{2}\right)\frac{\pi}{Nfbank}\right], n = 0,1,2 \dots Nk-1 \quad \textbf{(2.14)}$$

Dimana:

$Se[n]$ = Hasil dari DCT

$Sd[k]$ = Sampel setelah dilakukan *Mel Filter Bank*

Nk = Jumlah koefisien DCT

$Nfbank$ = Jumlah *filter bank*

2.5 Learning Vector Quantization

LVQ adalah algoritma perhitungan saraf dengan kecerdasan komputasi, dengan menggunakan pembelajaran kompetitif untuk melakukan klasifikasi. LVQ mempelajari vektor fitur dari ekstraksi ciri yang dihasilkan oleh metode MFCC.

2.5.1 Training LVQ

Pada penelititan sebelumnya telah dilakukan pengujian LVQ dengan akurasi 98.57% [2], dengan nilai paramter epoh 90, *learning rate* (a) 0.007 dan penurunan *learning rate* (penurunan a) 0.977 [2], maka pada penelitian ini akan dilakukan dengan parameter yang sama. Adapun algoritma pada LVQ antara lain [2]:

1. Inisilaisasi parameter:

   $a, penurunan\ a, epoh$

   Dimana:

   a = *Learning Rate*

   $penurunan\ a$ = penurunan *Learning Rate*

   $epoh$= Maksimum epoh
2. Lakukan perhitungan langkah 3 dan 4 selama perulangan lebih kecil dari maksimum epoh
3. Perhitungan X_i terhadap W_i dengan persamaan *Euclidean Distance* sebagai berikut:

$$C_{xw} = \sqrt{\sum_{i=1}^{n}(X_i - W_i)^2 + \cdots + (X_n - W_n)^2} \quad \textbf{(2.15)}$$

   Dimana:

   X_i = Nilai pada Data Latih

   W_i = Nilai Bobot Awal
4. Setelah didapatkan nilai C_{xw} dari nilai $X_i$terhadap nilai W_i, lalu dapatkan nilai bobot baru dengan persamaan berikut:

Jika $T = C_{xw}$ maka $W_i(baru) = W_i(lama) + \alpha(X_i - W_i(lama))$ **(2.16)**

Jika $T \neq C_{xw}$ maka $W_i(baru) = W_i(lama) - \alpha(X_i - W_i(lama))$ **(2.17)**

Dimana:

T = Target dari kelas C_{xw} dengan nilai yang terkecil

Tahap mencari bobot baru, setelah di dapatkan nilai X_i terhadap W_i, Maka selanjutnya dilakukan perubahan α dengan persamaan sebagai berikut:

$\alpha = a - penurunan\ a \times a$ **(2.18)**

5. Langkah 3 dan 4 dikerjakan pada semua X_i terhadap W_i, sampai maksium epoh.
6. Setelah melakukan perhitungan hingga maksimum epoh, maka didapatkan data bobot baru yang akan dijadikan sebagai data bobot untuk pengujian metode LVQ.

2.5.2 Testing LVQ

Setelah didapatkan pembelajaran LVQ dengan data bobot baru, maka dilakukan pengujian mencari jarak *Eulicdean Distance* dengan persamaan (2.15), Dari hasil pengujian LVQ dengan perhitungan data bobot baru diperoleh nilai C_{xw}terkecil, hasil dari bobot dengan nilai terkecil merupakan hasil pengenalan.

2.6 Pengujian *Black Box*

Pengujian *black box*, dapat juga disebut pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak. Artinya, teknik pengujian *black box* memungkinkan memperoleh serangkaian kondisi masukan yang akan sepenuhnya menjalankan semua persyaratan fungsional untuk sebuah program. Pengujian *black box* bukanlah alternatif teknik *white-box*. Sebaliknya, ini adalah pendekatan pelengkap yang cenderung menemukan kelas kesalahan yang berbeda dari pada metode *whitebox*. Pengujian *black box* mencoba menemukan kesalahan dalam kategori berikut: (1) fungsi yang salah atau hilang, (2) kesalahan antarmuka, (3) kesalahan dalam struktur data atau akses database eksternal, (4) kesalahan perilaku atau kinerja, dan (5) Inisialisasi dan kesalahan terminasi [13].

2.7 Pengujian *K-fold Cross Validation*

Cross-Validation adalah metode statistik untuk mengevaluasi dan membandingkan algoritma pembelajaran dengan membagi data menjadi dua segmen, satu digunakan untuk belajar atau melatih model dan yang lainnya digunakan untuk memvalidasi model. Pada validasi silang yang khas, set pelatihan dan validasi harus disilangkan secara berturut-turut sehingga masing-masing titik data memiliki kesempatan untuk divalidasi. Bentuk dasar cross-validation adalah k-fold cross-validation. [14].

2.8 Pemograman Berorientasi Objek

Pemograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemograman berorientasi objek, dan pengujian berorientasi objek [15].

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemograman, dan pengujian perangkat lunak. Berikut adaah beberapa konsep dasar yang harus dipahami pemograman berorientasi objek [15]:

1. Kelas (*Class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*Object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, strutur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*Method*)

   Operasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada terstruktur. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (*Attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*Abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapulasi (*Encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja.

7. Pewarisan (*Inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

Antarmuka atau interface sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. Reusability

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi antar objek

    Komunikasi antar-objek dilakukan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.

12. Polimorfisme (*Polymorphism*)

    Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda [15].

2.9 Unified Modeling Language

UML (*Unifed Modeling Language)* adalah salah satu standar Bahasa yang banyak digunakan di dunia industry untuk mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek. Ada beberapa diagram yang digunakan proses pembuatan perangkat lunak berorientasi objek diantaranya, *use case diagram, activity diagram, class diagram dan sequence diagram* [15].

2.9.1 Use Case Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut actor dan *use case.*

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibangun.
2. *Use Case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor [15].

2.9.2 Class Diagram

Diagram kelas atau *Class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas [15].

2.9.3 Squence Diagram

Diagram Sekuen atau *Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Penomoran pesan berdasarkan urutan

interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu [15].

2.9.4 Activity Diagram

Diagram Aktivitas atau *Activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak [15].

2.10 Tools Pembangunan Perangkat Lunak

Adapun beberapa perangkat lunak pendukung dalam pembuatan aplikasi pengenalan suara ini diantaranya:

2.10.1 Android

Android adalah kumpulan perangkat lunak *open source* yang dibuat untuk beragam perangkat dengan berbagai faktor bentuk. Tujuan utama Android adalah membuat platform perangkat lunak terbuka yang tersedia bagi operator, OEM, dan pengembang untuk membuat gagasan inovatif mereka menjadi kenyataan dan mengenalkan produk dunia nyata yang sukses yang meningkatkan pengalaman mobile bagi pengguna [16]. Adapun versi rilis dalam pengembangan android sebagai berikut [17]:

Tabel 2.1 Rilis Versi Android

Nama Kode	Versi	API level
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2
Cupcake	1.5	API level 3, NDK 1
(Tidak ada nama kode)	1.1	API level 2
(Tidak ada nama kode)	1.0	API level 1

2.10.2 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - Integrated Development Environment (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Selain merupakan editor kode IntelliJ dan alat

pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya:

1. Sistem pembuatan berbasis *Gradle* yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat penguji dan kerangka kerja yang ekstensif
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

1. Modul aplikasi Android
2. Modul perpustakaan
3. Modul *Google App Engine*

Secara *default*, Android Studio menampilkan file proyek Anda dalam tampilan proyek Android seperti yang ditunjukkan dalam gambar 1. Tampilan ini diatur menurut modul untuk memberi akses cepat ke file sumber kunci proyek Anda [18].

Semua file versi terlihat di bagian atas di bawah *Gradle Scripts* dan masing-masing modul aplikasi berisi folder berikut:

1. manifests: Berisi file AndroidManifest.xml.
2. java: Berisi file kode sumber Java, termasuk kode pengujian JUnit.

3. res: Berisi semua sumber daya bukan kode, seperti tata letak XML, string UI, dan gambar bitmap.