BAB 2
LANDASAN TEORI

2.1 Pengenalan Huruf

Dalam kamus besar bahasa Indonesia, huruf didefinisikan sebagai tanda aksara dalam tata tulis yg merupakan anggota abjad yang melambangkan bunyi bahasa. Berdasarkan pengertian tersebut, dapat kita katakan kalau huruf adalah lambang dari bunyi. Misalnya bunyi be lambangnya atau hurufnya adalah b, bunyi el lambangnya adalah l, dan seterusnya.

2.1.1 Jenis-jenis Huruf

Huruf dapat diklasifikasikanberdasarkan bentuk dan bunyinya menjadi beberapa macam.

1. Huruf berdasarkan bentuk

   Berdasarkan bentuknya, huruf dibedakan menjadi 4 yaitu sebagai berikut.

   a. Huruf fonemis, yaitu huruf yang melambangkan satu bunyi seperti huruf latin.
   b. Huruf silabis, yaitu huruf yang melambangkan satu suku kata seperti huruf jepang atau aksara jawa.
   c. Huruf logograf atau idiograf, yaitu huruf yang melambangkan bunyi satu kata seperti huruf cina.
   d. Huruf piktograf, yaitu bunyi huruf yang dilambangkan dalam bentuk gambar atau lukisan peristiwa seperti relif manusia prasejarah.

2. Huruf berdasarkan bunyi

   a. Huruf vokal, yaitu bunyi yang tidak disertai hambatan pada alat bicara, hambatan hanya terdapat pada pita suara, tidak pada artikulator. Jadi udara yang keluar dari paru-paru melewati pita suara dan tidak ada artikulator atau alat ucap yang menghambat seperti bibir, gigi, ataupun lidah. Yang termasuk bunyi vokal adalah a, i, u, e, o, é.

b. Huruf konsonan, yaitu bunyi yang dibentuk dengan menghambat arus udara yang keluar dari paru-paru. Hambatan dapat terjadi pada sebagian alat bicara, seperti hambatan pada dua bibir pada bunyi B, hambatan pada ujung lidah dengan menyentuh belakang gigi depan atas pada bunyi T, dan sebagainya.
c. Huruf semivokal, yaitu bunyi huruf antara vokal dan konsonan. Dalam kamus besar bahasa indonesia, semivokal didifinisikan sebagai bunyi bahasa yangg mempunyai ciri vokal ataupun konsonan, bunyi ini mempunyai sedikit geseran dan tidak muncul sbg inti suku kata. Yang termasuk bunyi semivokal adalah bunyi y dan w.

2.1.2 Pengenalan Angka

Angka disebut juga digit. Angka tidak sama dengan bilangan, tetapi bilangan terdiri dari angka-angka. Misal: "456" adalah lambang bilangan untuk empat ratus lima puluh enam yang terdiri dari dua angka. Arti suatu angka dalam suatu lambang bilangan ditentukan oleh nilai tempatnya dalam lambang bilangan itu. Berikut contoh nilai tempat yang membedakan nilai angka dalam bilangan basis 10.

2.2 Handwriting Recognition

Pengenalan tulisan tangan (handwriting recognition) pertama kali dikenal pada awal tahun 1980-an dimana pada saat itu seorang bernama Dr. Charles Elbaum mendirikan sebuah perusahaan bernama "Nestor" dan membuat sebuah alat bernama "NestorWriter handwriting recognizer" dan mendapatkan nobel dari hasil karyanya ini. Penemuan ini menimbulkan euphoria tersendiri, banyak orang mengganggap bahwa penggunaan alat ini untuk komputasi, yang selanjutnya disebut pen computing dapat menggantikan penggunaan komputer dengan keyboard dan mouse konvensional dan akan semakin banyak orang yang akan akrab dengan komputer karena lebih banyak orang yang bisa menggunakan pena(pen) daripada yang bisa menggunakan keyboard [5].

2.2.1 Pengenalan *Gesture Touch*

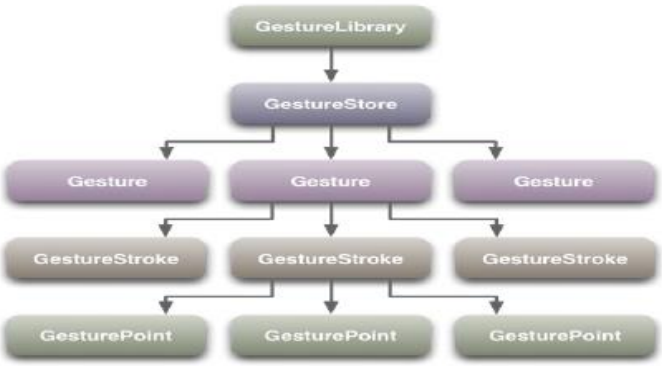
Gestures touch biasanya digunakan dari gerakan atau pergeseran jari pada layar sentuh sensitif pada Android, yang diproses menggunakan package android.gesture . Gerakan yang paling umum adalah tap, yang seperti klik mouse pada komputer. Gesture tertentu juga terdeteksi melalui antarmuka View.OnClickListener. Gerakan berikutnya adalah drag atau seret (seperti menyeret dengan mouse pada komputer), sebagai contoh ketika pertama kali menyalakan android umumnya melakukan slider untuk membuka kunci (key lock) pada Android dengan melakukan gerakan drag atau seret. contoh sederhananya ketika Gesture package akan melaporkan urutan gerakan jari pada layar sentuh secara penuh dan memungkinkan aplikasi untuk mendeteksi , misalnya, ketika membuat gerakan busur melengkung , ketika membuat gerakan searah jarum jam atau berlawanan dan jika gerakan itu lambat ataupun cepat . Aplikasi harus bisa mengkonversi titik koordinat X/Y dan waktu untuk mencari tahu apa gerakannya, gerakan – gerakan tersebut bisa manfaatkan pada aplikasi - aplikasi android,seperti pada game,widget,aplikasi menggambar dan sebagainya

2.2.2 Pengenalan *Gesture Builder*

Gesture Builder adalah sebuah tool yang digunakan untuk membuat suatu library dari gesture. Ketika aplikasi menjalankan sebuah gesture, aplikasi akan mencocokan gerakan tersebut ke GestureLibrary dan menentukan apakah gesture itu cocok dengan salah satu gesture yang sudah ada di library.

2.2.3 Pengenalan *Gesture Library*

Gesture Libary dibuat menggunakan tool yang di sediakan pengembang yaitu gesture builder, gesture library disimpan satu paket dengan aplikasi dan dipanggil dari raw resource dimana gesture library itu disimpan, dibawah ini adalah struktur dari gesture library.



Gambar 2. 1 Struktur Gesture Library

2.3 Metode Pembelajaran Interaktif

Metode pembelajaran interaktif adalah sebuah metode pembelajaran yang digunakan untuk menyajikan suatu bahasan atau sumber pembelajaran yang dilakukan oleh guru untuk menciptakan suasana interaktif yang edukatif, antara guru dengan siswa, siswa dengan siswa dan dengan sumber pembelajaran yang mendukung terjadinya proses belajar dan mengajar.

2.4 Definisi Aplikasi *Mobile*

Menurut *Buyens*, aplikasi mobile adalah aplikasi *mobile* berasal dari kata application dan mobile. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedangkan mobile dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain. Dengan menggunakan aplikasi mobile, Anda dapat dengan mudah melakukan berbagai macam aktifitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, browsing dan lainlain. Pemanfaatan aplikasi mobile untuk hiburan paling banyak digemari oleh hampir 70% pengguna telepon seluler, karena dengan memanfaatkan adanya fitur *game, music player*, sampai video *player* membuat kita menjadi semakin mudah menikmati kapan saja dan dimanapun.

2.5 Multimedia

Multimedia terdiri dari dua kata yaitu multi dan media, dimana multi berarti banyak, majemuk, dan beraneka ragam, sedangkan media berarti suatu alat perantara untuk penyampaian sesuatu. Multimedia juga merupakan kombinasi teks, grafik, suara, animasi dan video yang disampaikan dengan komputer atau peralatan manipulasi elektronik dan digital yang lain. Ketika pengguna diizinkan mengontrol apa dan kapan elemen-elemen tersebut akan dikirimkan, maka multimedia akan disebut multimedia interkatif.

Pengertian lainnya tentang multimedia, Tay Vaughan(2007:10) menyebutkan bahwa objek yang menjadi bagian multimedia yang terbagi dalam beberapa jenis yaitu :

1. Teks

   Teks merupakan dasar dari pengolahan kata dan informasi berbasis multimedia yang bertujuan untuk menyampaikan/menyajikan informasi kepada pengguna. Selain itu teks juga merupakan sarana yang efektif untuk mengemukakan pendapat yang berupa ide-ide serta instruksi kepada pengguna.
2. Gambar

   Secara umum gambar berarti still image seperti foto, gambar merupakan salah satu sarana yang sangat baik untuk menyajukan informasi karena pengguna (manusia) sangant berorientasi pada visual/penglihatan.
3. Animasi

   Adalah rangkaian gambar yang disusun secara berurutan atau dikenal dengan sebutan frame. Dalam satu frame terdiri dari beberapa atau banyak gambar, jika susunan gambar tersebut ditampilkan secara tersusun dan dengan waktu tertentu maka susunan gambar tersebut akan terlihat bergerak.

4. Audio

   Kelengkapan untuk lebih memperjelas penyampaian informasi. Suara juga dapat menjelaskan karakteristik suatu gambar seperti *sound effect*.

5. Video

   Adalah elemen multimedia yang paling kompleks karena penyampaian informasi yang lebih komunikatif dibandingkan dengan gambar biasa. Video berbeda dengan animasi dilihat dari penyajiannya yang utuh dalam kesatuan dari objek yang dimodifikasi sehingga terlihat saling mendukung penggambaran yang seakan-akan terlihat hidup.

6. *Interactive Link*

   Menggabungkan beberapa elemen multimedia unuk menciptakan informasi yang terpadu untuk pengaksesan informasi pada multimedia (linear dan non-linear). Sebagian multimedia adalah interaktif, dimana pengguna dapat menekan *mouse* atau objek pada layar seperti tombolatau teks yang menyebabkan sistem menjalankan perintah tertentu[6].

2.6 Multimedia Interaktif

Multimedia interaktif adalah suatu multimedia yang dilengkapi dengan alat pengontrol yang dapat di operasikan oleh pengguna, sehingga pengguna dapat memilih apa yang dikehendaki untuk proses selanjutnya. Ciri khasnya, multimedia ini dilengkapi dengan beberapa navigasi yang disebut juga dengan *graphical user interface (GUI),* baik berupa *icon* maupun *button, pop-up menu. scroll bar,* dan lainnya yang dapat di operasikan oleh *user* sarana *browsing* ke berbagai jendela informasi dengan bantuan sarana *hyperlink*. Penerapan multimedia interaktif ini dapat terdapat pada multimedia pembelajaran serta aplikasi *game*. Multimedia interaktif tidak memiliki durasi karena lama penayangan tergantung seberapa lama pengguna mem-*browsing* media ini.

2.7 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan

aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, 14 peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [7].

Fitur Android:

1. Framework aplikasi, memungkinkan daur ulang dan penggantian komponen.
2. Browser terintegrasi berbasis *engine Open Source WebKit* yang juga digunakan di *browser* IPhone dan Nokia S60v3.
3. Rancangan *handset. Platform* disesuaikan dengan kebutuhan VGA *(Video Graphics Adapter)* yang lebih besar, *library* grafik 2D dan 3D yang berdasarkan pada spesifikasi OpenGL ES 1.0 serta layout *smartphone* yang tradisional.
4. Konektivitas. Android mendukung berbagai teknologi konektivitas seperti GSM *(Global System for Mobile Communications)* / EDGE *(Enhanced Data rates for GSM Evolution),* CDMA *(Code Division Multiple Access),* EVDO *(Evolution-Data Optimized*), UMTS *(Universal Mobile Telecommunications System),* Bluetooth dan Wi-Fi *(Wireless Fidelity).*
5. Pesan. Android mendukung pengiriman pesan dalam bentuk SMS *(Short Message Service)* dan MMS *(Multimedia Messaging Service)*. Dukungan Java. *Software* yang ditulis dalam bahasa Java dapat dikompilasi dan akan dieksekusi pada mesin virtual Dalvik, yang merupakan implementasi dari VM *(Virtual Machine)* yang dirancang khusus untuk penggunaan perangkat bergerak.
6. Dukungan media. Android mendukung beberapa format audio/video seperti: H.263, H.264 (dalam kontainer 3GP atau MP4), MPEG-4 SP, AMR,

AMR-WB (dalam kontainer 3GP), AAC, HE-AAC (dalam kontainer MP4 atau 3GP), MP3, MIDI, OGG Vorbis, WAV, JPEG, PNG, GIF dan BMP.

7. Dukungan hardware tambahan. Android mendukung penggunaan kamera, layar sentuh, GPS *(Global Positioning System)*, pengukur kecepatan, magnetometer, akselerasi 2D bit blits (dengan orientasi *hardware*, *scaling*, konversi format piksel) dan akselerasi grafis 3D.

8. *Play store*. Mirip dengan *App Store* pada iPhone OS, Android *Play store* adalah sebuah katalog aplikasi yang dapat di-download dan diinstal pada telepon seluler secara online, tanpa menggunakan PC *(Personal Computer)*. Awalnya hanya aplikasi gratis saja yang didukung. Dan sejak tanggal 19 Februari 2009 aplikasi berbayar telah tersedia di Android *Play store* untuk Amerika Serikat.

9. *Multi-touch*. Android memiliki dukungan bawaan untuk *multi-touch* yang tersedia pada handset terbaru seperti HTC Hero. Pada awalnya fitur tersebut dinonaktifkan pada level kernel (mungkin untuk menghindari pelanggaran paten terhadap teknologi layar sentuh *Apple*). Sejak Google merilis *update* untuk Nexus One dan berencana juga untuk merilis *update* untuk Motorola Droid yang memungkinkan *multi-touch.*

10. Lingkungan pengembangan yang kaya, termasuk emulator, peralatan debugging, dan plugin untuk Eclipse IDE.

2.8 Android Studio

Android Studio adalah sebuah IDE untuk pengembangan aplikasi di platform Android. Saat ini usia Android Studio masih tergolong muda, baru versi 0.2.3 (masih *early access preview*). SDK sebelumnya di-*bundle* bersama dengan Eclipse, sementara Android Studio menggunakan IntelliJ IDEA Community Edition. Beberapa pendukung IntelliJ IDEA sering mengatakan bahwa Eclipse terlalu rumit bagi pemula. Android Studio menggunakan Gradle untuk manajemen proyeknya. Gradle adalah *build automation tool* yang dapat dikonfigurasi melalui DSL berbasis Groovy. Ini yang membedakan Gradle dari Ant atau Maven yang

memakai XML. Penggunaan DSL berbasis Groovy menyebabkan Gradle lebih fleksibel dan dapat diprogram dengan mudah.

Android Studio build system digunakan untuk membangun, test, menjalankan, dan membuat paket dari aplikasi yang dibangun. Build system tidak tergantung (*independent*) dari Android Studio, jadi developer dapat memanggilnya dalam Android Studio atau dengan menggunakan command line. Setelah menuliskan code pada aplikasi yang akan dibangun, developer dapat menggunakan fitur build system untuk [7] :

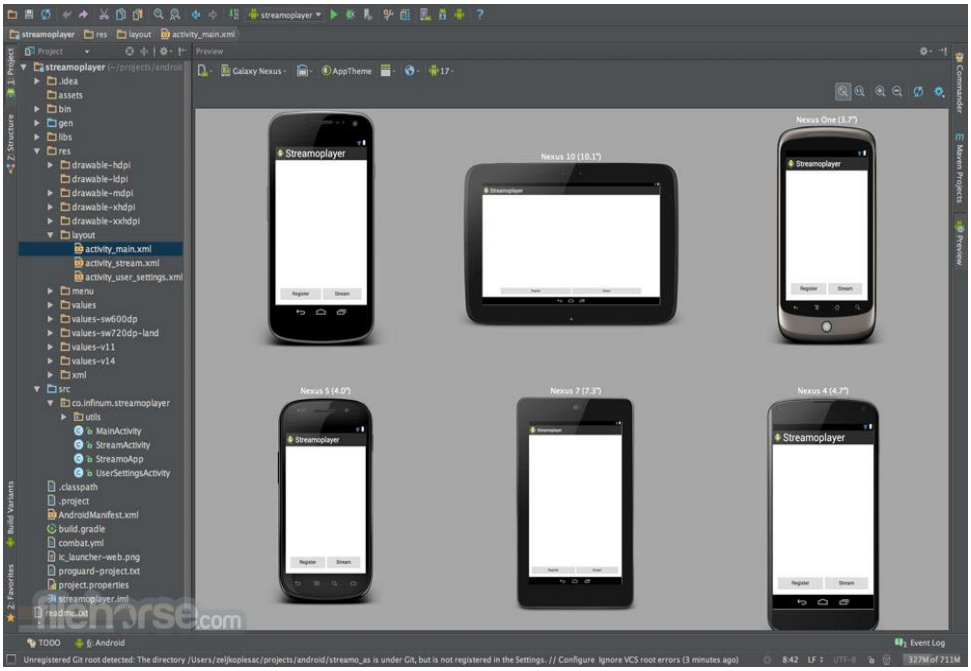
1. Kostumisasi, konfigurasi, dan meng-extends proses build
2. Membuat beberapa apk untuk aplikasi android dengan fitur yang berbeda menggunakan project yang sama
3. Menggunakan ulang code dan resources

Fleksibilitas dari system pengembangan Android Studio memungkinkan untuk mencapai keseluruhan hal ini tanpa harus memodifikasi file inti dari project. Android Studio ini adalah lingkungan pengembangan baru dan terintegrasi penuh, yang baru saja dirilis oleh Google untuk sistem operasi Android. Android Studio dirancang untuk menjadi peralatan baru dalam pengembangan aplikasi dan juga memberi alternative lain selain Eclipse yang saat ini menjadi IDE yang paling dipakai. Saat memulai proyek baru dengan Android Studio, struktur proyek akan muncul bersama dengan hampir semua berkas yang ada di dalam direktori SDK, peralihan ke sistem manajemen berbasis Gradle ini memberikan fleksibilitas yang lebih besar pada proses pembangunannya.

Android Studio developer dapat melihat perubahan visual apapun yang dilakukan pada aplikasi secara langsung.Dapat terlihat perbedaannya jika dipasang pada beberapa perangkat Android berbeda, termasuk konfigurasi dan resolusi secara bersamaan. Fitur lain di Android Studio adalah alat alat baru untuk *packing* dan memberi label code. Dengan begitu mengijinkan developer tetap menjadi yang teratas ketika berurusan dengan banyak code. Program ini juga menggunakan sistem *drag and drop* untuk memindahkan komponen melalui *interface user.*

Selain itu, lingkungan baru ini juga mendukung Google Cloud Messaging. Sebuah fitur yang mengijinkan untuk mengirim data dari server ke perangkat Android melalui cloud, cara terbaik untuk mengirim *Reminder* pada apps. Program ini juga membantu untuk melokalisasi aplikasi anda, memberi gambaran visual untuk tetap memprogram sambil mengontrol alur dari aplikasi. Kemudahan yang diberikan oleh Android Studio adalah sebagai berikut :

1. Lingkungan pengembangan yang mantap dan bersifat langsung.
2. Cara termudah untuk menguji performa pada perangkat dengan tipe lain.
3. Wizard dan template berisi elemen – elemen umum yang ada di semua pemrograman Android.
4. Editor dengan fitur lengkap dengan banyak peralatan ekstra untuk mempercepat pengembangan aplikasi anda. Layout editor yang memungkinkan untuk *drag and drop* komponen UI, pratinjau layout pada beberapa konfigurasi layar, dan banyak lagi.
5. Berbasis Gradle.
6. Android spesifik refactoring dan perbaikan yang cepat.
7. Alat Lint untuk menangkap kinerja, kegunaan, versi kompatibilitas dan masalah lainnya.
8. ProGruard dan app-signature.
9. Built-in dukungan untuk Google Cloud platform, sehingga mudah untuk mengintegrasikan Google Cloud Messaging dan App Engine sebagai komponen server-side.



Gambar 2. 2 Struktur Gesture Library

2.9 *Object Oriented Programming* (OOP)

OOP *(Object Oriented Programming)* atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan cara membangun aplikasi/sistem yang berorientasikan kepada objek. Objek sendiri adalah sesuatu yang memiliki satu atau lebih atribut *(Field)* dan macam aksi yang terdapat didalamnya *(Method)*, sebagai contoh terdapat objek yang bernama Bruno yang memiliki atribut berkaki 4, bermata 2 dan memiliki 1 ekor, selain itu Bruno juga memiliki macam aksi seperti tidur, makan dan berjalan. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut.

Untuk menyelesaikan suatu masalah dengan kolaborasi antar objekobjek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri. Pemrograman orientasi-objek menekankan konsep berikut.

1. Class

   Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek (*Class* juga dapat berarti sketsa/blueprint dari sebuah objek). Secara umum *Class* memiliki 2 anggota yaitu *Field* atribut yang dimiliki oleh objek dari sebuah *Class* dan *Method* aksi yang dapat dilakukan oleh objek dari sebuah *Class*. Sebagai contoh terdapat *Class of Horse* yang memiliki atribut berkaki 4, bermata 2 dan memiliki 1 ekor, selain itu aksi yang dimiliki oleh *Class* tersebut adalah tidur, makan dan berjalan.

2. Objek

   Adalah realisisi dari sebuah *Class* yang didalamnya terdapat atribut *(Field)* dan aksi *(Method)* yang menjadi suatu unit *(Class)* dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. Abstraksi

   Sebuah cara untuk membuat suatu *Class* yang memiliki keturunan(Inheritas) agar tidak dapat tercipta objek dari *Class* tersebut dengan tujuan menjadikan *Class* tersebut menjadi pola bagi *Class* turunannya. Sebagai contoh terdapat sebuah *Class* yang memiliki dua keturunan yang nantinya akan dibuat objek yang tercipta dari keturunannya yang tentunya dapat memiliki atribut dan aksi yang dimiliki oleh *Class* induk. Jika tidak dilakukan abstraksi pada *Class* induk maka dapat dibuat sebuah objek yang yang dapat memiliki atribut dan aksi yang tidak dimiliki oleh keturunan dari *Class* induk, seharusnya *Class* induk hanyalah sebuah bentuk umum/pola yang nantinya dapat digunakan oleh keturunannya tidak untuk dijadikan sebuah objek.

4. Enkapsulasi

Sebuah cara yang dilakukan untuk menjaga keamanan dan fleksibilitas dari sebuah *Class* yang dapat dilakukan dengan (membungkusnya) menggunakan *Access Identifier* yang dapat menentukan siapa saja yang dapat mengakses (baik dibaca maupun ditulis/diubah) anggota dari sebuah *Class*, Method *Getter/Setter* yang menjadikan sifat dari anggota tertentu pada sebuah *Class* menjadi dapat dibaca oleh *Class* lain namun hanya bisa ditulis/diubah oleh dirinya sendiridan *Property* yang merupakan cara yang lebih praktis dibandingkan dengan *Method Getter/Setter* dikarenakan *Property* membuat *method getter* dan *setter* menjadi satu tidak seperti *Method Getter/Setter* yang memisahkan antara *Method Getter* dan *Setter.*

5. Polimorfisme

   Polimorfisme atau dalam bahasa Yunani disebut Polymorphysm yang berarti banyak bentuk. Polimorfisme sendiri memiliki dua karakteristik yaitu :

   a. Sebuah *pointer* dari *Class* tertentu tidak hanya dapat mengacu pada objek yang tercipta dari dirinya sendiri pointer tersebut juga dapat menampung objek dari *Class* keturunannya (sebuah *pointer* dari suatu *Class* dapat mengacu kepada benyak jenis objek).
   b. Sebuah *method* yang terdapat pada sebuah *Class* dapat memiliki banyak implementasi, sebagai contoh terdapat sebuah *Class* yang menulis ulang *(override) method* yang diturunkan kepadanya.

6. Inheritas

   Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dari *Class* induk dengan jenis khusus/berbeda dari objek yang sudah ada objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

2.10 Unified Modelling Language (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak-artifak dari sistem, begitu diterapkan untuk menspesifikasi sistem, UML dapat digunakan untuk mengkomunikasi "apa" yang diperlukan dari sistem dan bagaimana sistem dapat direalisasikan.

Diagram UML diklasifikasikan menjadi dua kategori; struktur dan perilaku. Diagram struktur untuk memvisualisasikan, menspesifikasikan, membangun dan mendokumentasikan aspek statik dari sistem. Diagram struktur yang akan dibahas dalam penulisan ini adalah class diagram, sedangkan diagram perilaku untuk memvisualisasikan, menspesifikasikan, membangun dan mendokumentasikan aspek dinamis dari sistem. Diagram perilaku yang akan dibahas pada penulisan ini adalah use case diagram, sequence diagram dan *activity diagram*.

UML dapat membuat model semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. UML menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C [8].

2.10.1 *Use Case Diagram*

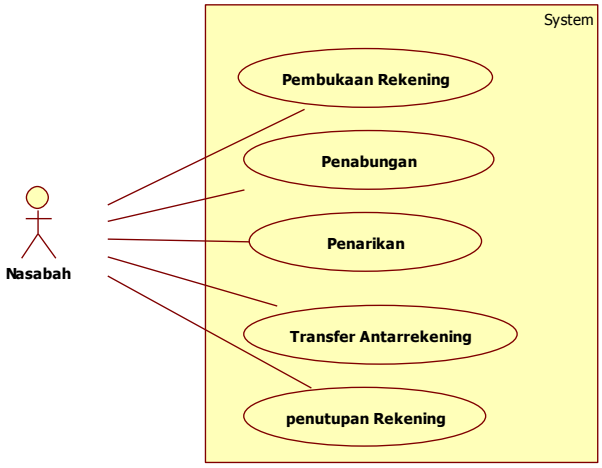
Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan use case diagram yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya. Selanjutnya, use case diagram tidak hanya sangat penting pada tahap analisis, tetapi juga penting untuk perancangan *(design),* untuk mencari (mencoba

menemukan) kelas-kelas yang terlibat dalam aplikasi, dan untuk melakukan pengujian *(testing).*

Elemen diagram *use-case* adalah :

1. Aktor adalah pemakai sistem, sesuatu atau seseorang yang berinteraksi dengan sistem.
2. Use case menspesifikasikan perilaku sistem atau bagian sistem dan merupakan deskripsi sekumpulan sekuen aksi termasuk varian-varian yang dilakukan sistem untuk memproduksi hasil atau nilai ke aktor.
3. Hubungan ketergantungan, generalisasi dan asosiasi merupakan partisipasi dari aktor dalam *use case*. Sebuah *use case* dapat meng-*include* fungsionalitas use case lain sebagai bagian dari proses dalam dirinya.

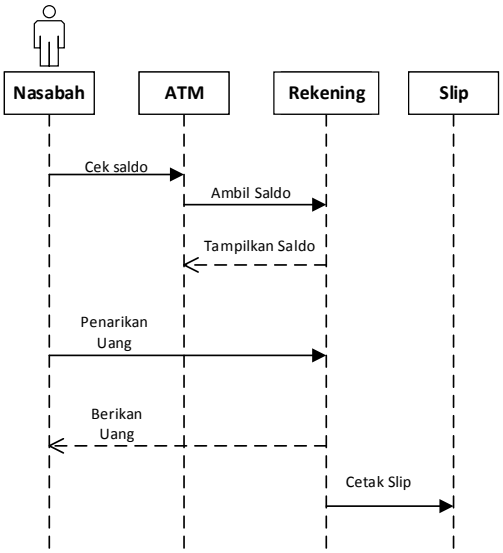
Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali use case yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah use case juga dapat meng-*extend use case* lain dengan behavior nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain. Berikut contoh *Use Case diagram* pada gambar 2.3



Gambar 2. 3 Contoh *Use Case Diagram*

2.10.2 Sequance *Diagram*

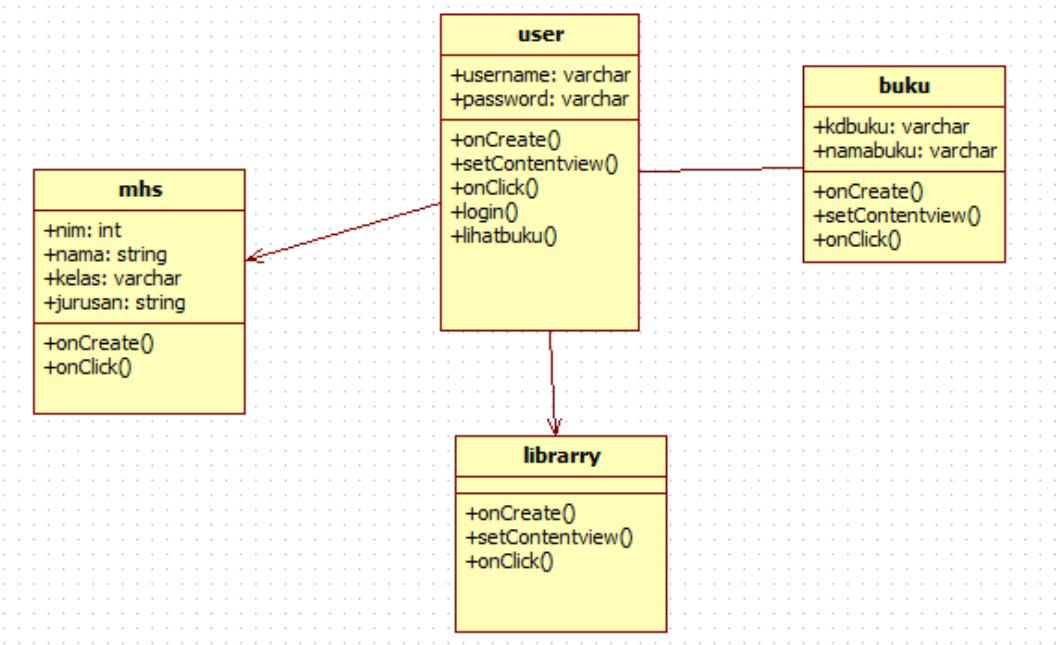
Sequence diagram menujukan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada basis keberurutan waktu dari pesan-pesan yang terjadi. *Sequence diagram* menguraikan pesan di mana interaksi berlangsung, waktu adalah suatu faktor penting yang ditandai dengan dimensi vertical sedangkan objek-ibjek yang terkait ditandai dengan dimensi horizontal. *Sequence diagram* biasa digunakan untuk menggambarkan rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan *output* tertentu. Berikut contoh *Sequence diagram* pada gambar 2.4 :



Gambar 2. 4 Contoh Sequence Diagram

2.10.3 *Class Diagram*

Kelas (*class*) merupakan kumpulan objek yang memiliki kesamaan dalam atribut, perilaku serta cara berhubungan dengan objek yang lain. Kelas membungkus (*encap-sulating*) objek-objek. Suatu kelas tunggal dapat digunakan untuk menciptakan sejumlah objek . Selain itu, suatu kelas juga dapat digunakan untuk menciptakan kelas-kelas lain yang mewarisi (*inherit*) sebagian atau seluruh data. Dengan membungkus objek-objek dalam stuktur kelas, dapat mengelompokkan objek-objek yang sama jenisnya, yang memiliki keadaan-keadaan/data-data serta perilaku-perilaku yang sama. Pada awal perkembangannya, kelas (*class)* didefinisikan sebagai tipe data abstrak (ADT[*Abstract Data Type*]) dalam bahasa pemrograman Simula 67 dan Smalltalk. Berikut contoh *class diagram* pada Gambar 2.5 :



Gambar 2. 5 Contoh Class Diagram

2.11. Bahasa Pemrograman

Pada 1991, sekelompok insinyur Sun Microsystem, Inc., dipimpin Patrick Naughton dan James Gosling merancang bahasa untuk perangkat konsumer seperti cable TV box. Karena perangkat tidak mempunyai banyak memori, bahasa harus berkuran kecil dan menghasilkan kode yang liat. Karena pabrikan-pabrikan berbeda memilih pemroses –pemroses berbeda maka bahasa harus netral arsitektur manapun. Proyek diberi nama “Green”.

Kebutuhan untuk kecil, liat dan netral platform mengantar tim mempelajari implementasi Pascal yang pernah dicoba. Niklaus Wirth, pencipta Pascal telah merancang bahasa kode antara (intermediate code) portable untuk mesin hipotesis. Kode antara ini kemudian dapat digunakan di sembarang mesin yang memiliki interpreter. Proyek Green menggunakan mesin maya untuk mengatasi isu netral terhadap arsitektur mesin. Karena orang-orang di proyek Green berbasis C++ bukan Pascal maka kebanyakan sintaks diambil dari C++ serta mengadopsi orientasi objek bukan prosedural. Mulanya bahasa yag

diciptakan diberi nama “Oak”, kemudian diganti “Java” karena telah ada bahasa yang bernama “Oak”.

Produk pertama proyek “Green” adalah “*7”, sebuah kendali jauh yang cerdas. Karena pasar belum tertarik dengan prosuk konsumer cerdas maka proyek Green harus menemukan pasar lain. Penerapan mengarah menjadi teknologi di web. Pada 1995, Netscape memutuskan membuat browser dilengkapi Java. Setelah itu diikuti IBM, Symantec, bahkan Microsoft.

Sebagai bahasa yang menampung hampir seluruh kemampuan terbaik bahasa pemrograman yang pernah dikembangkan umat manusia, maka bahasa ini memang menjadi tidak mudah, sedikit rumit dan kompleks. Namun, pantas sedikit bersusah agar mampu menguasai salah satu kakas terampuh umat manusia, yaitu bahasa pemrograman modern.

Bahasa Java tidak lagi hanya untuk pemanis di web sebagai applet yang membuat Duke berdansa. Java adalah kakas, tetap hanya perangkat, bagaimanapun tetap hanya orang hebat yang dapat memberi arti penting kakas seperti dikatakan James Gosling , tokoh terpenting di Java: “All along, the language was a tool, not the end.”

Bahasa Java telah disertai berton-ton pustaka kelas, namun tidak perlu mengetahui semuanya. Perlu diketahui sekumpulan yang diperlukan untuk menyelesaikan persoalan serius yang sedang dihadapi.

Bahasa pemrograman orientasi objek telah menjadi aliran utama (mainstream), Java berorientasi objek sejati melebihi C++. Segala sesuatu di Java kecuali sedikit tipe dasar (int, float, double, char) adalah objek [9].

Dibanding bahasa C++, Java lebih memberi kemudahan, antara lain:

1. Perancang Java menghilangkan keperluan dealokasi manual. Java dilengkapi garbage collector yang bertugas mendealokasi memori yang tidak diperlukan. Tidak ada lagi upaya pemrogram untuk dispose(). Pemrogram tidak dibebani urusan korupsi dan sampah memori.

2. Java menerapkan array sebenarnya, menghilangkan keperluan aritmatika pointer yang berbahaya dan berpeluan besar menyebabkan kesalahan.
3. Menghilangkan keniscayaan operasi penugasan (asignment) sebagai pengujian atas kesamaan di kalimat bersyarat. Di C/C++, pernyataan seperti

   (nPerson = 3) …

   sering menyebabkan masalah, memaksudkan pengujian namun yang terjadi adalah pemberian nilai. Pada kasus ini, kompilator tidak berhak memberi peringatan karena pernyataan itu merupakan pernyataan yang sah.
4. Menghilangkan pewarisan jamak (multiple inheritance) diganti fasilitas interface. Interface memberi banyak manfaat antara lain untuk pewarisan jamak tanpa dibebani kompleksitas yang muncul dari pengelolaan hirarki pewarisan jamak