BAB 2
LANDASAN TEORI

2.1. Al-Qur’an

Al-Quran berasal dari bahasa Arab yang berarti bacaan atau sesuatu yang dibaca berulang-ulang. Umat islam percaya bahwa Al-Qur'an merupakan penutup wahyu Allah yang diperuntukkan bagi manusia, dan bagian dari rukun iman, yang disampaikan kepada Nabi Muhammad Shallallahu ‘alaihi wa sallam, melalui perantaraan Malaikat Jibril. Maka setiap muslim (beragama islam) diwajibkan bisa untuk mempelajari serta mengamalkan Al-Quran, yang di dalamnya terkandung bagaimana menjalani hidup di dunia dan mendapatkan surga (selamat dunia akhirat). Di indonesia sendiri terdapat beberapa metode pembelajaran Al-Quran. Salah satunya dengan melatih menulis huruf – huruf Al-Quran yaitu huruf hijaiyah.

2.1.1. Huruf Hijaiyah

Huruf hijaiyah adalah huruf yang digunakan dalam Al-Qur’an, terdiri dari 28 huruf sebagai berikut huruf sebagai berikut :



Gambar 2. 1 Huruf Hijaiyah alif-sin[2



Gambar 2. 2 Huruf Hijaiyah shin-yaa[2]

2.1.2. Huruf Hijaiyah Sambung

Huruf hijaiyah sambung masing-masing mempunyai bentuk huruf sesuai posisinya yaitu di awal, di tengah dan di akhir :

1. Alif



Gambar 2. 3 Alif [5]

2. Baa



Gambar 2. 4 Baa [5]

3. Taa



Gambar 2. 5 Taa [5]

4. Tsaa



Gambar 2. 6 Tsaa [5]

5. Jim



Gambar 2. 7 Jim [5]

6. Haa



Gambar 2. 8 Haa [5]

7. Khaa



Gambar 2. 9 Khaa [5]

8. Dal



Gambar 2. 10 Dal [5]

9. Dzal



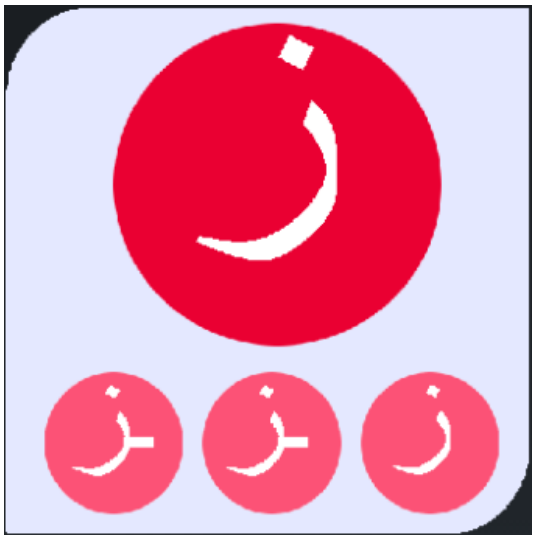
Gambar 2. 11 Dzal [5]

10. Raa



Gambar 2. 12 Raa [5]

11. Zay



Gambar 2. 13 Zay [5]

12. Sin



Gambar 2. 14 Sin [5]

13. Syin



Gambar 2. 15 Syin [5]

14. Shod



Gambar 2. 16 Shod [5]

15. Dhod



Gambar 2. 17 Dhod [5]

16. Tho



Gambar 2. 18 Tho [5]

17. Dzo



Gambar 2. 19 Dzo [5]

18. Ain



Gambar 2. 20 Ain [5]

19. Ghoin



Gambar 2. 21 Ghoin [5]

20. Faa



Gambar 2. 22 Faa [5]

21. Qof



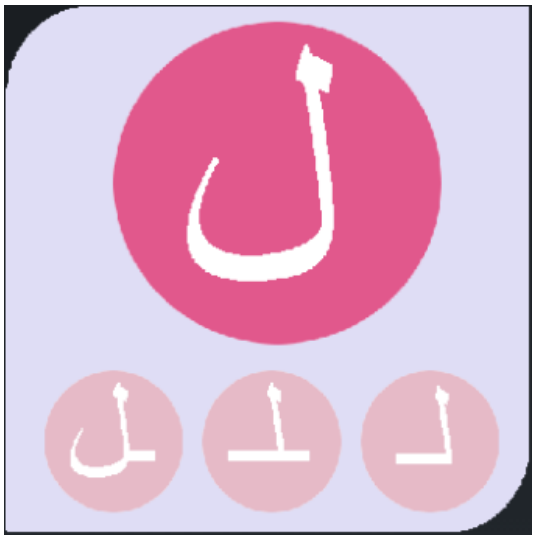
Gambar 2. 23 Qof [5]

22. Kaf



Gambar 2. 24 Kaf [5]

23. Lam



Gambar 2. 25 Lam [5]

24. Mim



Gambar 2. 26 Mim [5]

25. Nun



Gambar 2. 27 Nun [5]

26. Ha



Gambar 2. 28 Ha [5]

27. Waw



Gambar 2. 29 Waw [5]

28. Yaa



Gambar 2. 30 Yaa [5]

2.2. Metode Pembelajaran

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.

Metode praktik dilakukan setelah materi dipelajari atau guru memberikan demonstrasi. Metode *drill* digunakan ketika peserta didik diminta mengulang informasi pada topik-topik khusus sampai dapat menguasai topik-topik yang diajarkan. Metode praktik dan *drill* disebut juga metode praktik dan latihan. Metode tersebut diarahkan pada pengulangan (repitisi) untuk membantu peserta didik memiliki pemahaman yang lebih baik dan mudah mengingat kembali informasi yang sudah disampaikan [6].

2.3. *Handwriting Recognition*

Handwriting Recognition adalah sebuah sistem komputer yang dapat mengenali huruf yang berasal dari tulisan tangan. *Handwriting Recognition* itu sendiri merupakan aplikasi yang dikembangkan dari Optical Character Recognition (OCR) yang dapat mengenali suatu huruf kemudian langsung menterjemahkannya

dalam bentuk dokumen teks [7]. *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*.

2.3.1. Pengenalan *Gesture Touch*

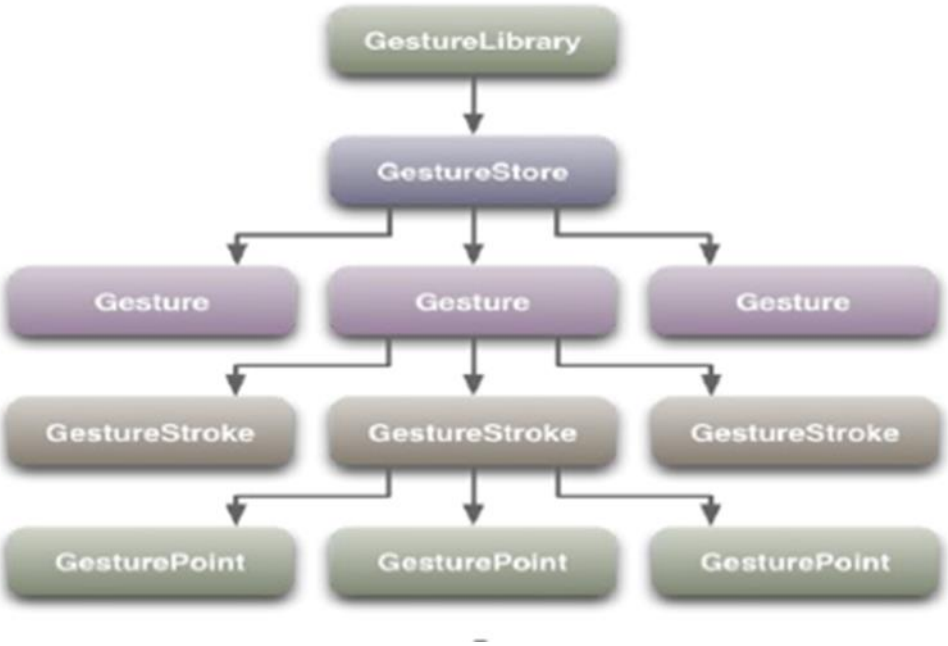
Gestures touch biasanya digunakan dari gerakan atau pergeseran jari pada layar sentuh sensitif pada Android, yang diproses menggunakan *package android.gesture*. Gerakan yang paling umum adalah tap, yang seperti klik *mouse* pada komputer. *Gesture* tertentu juga terdeteksi melalui antarmuka *View.OnClickListener*. 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.

2.3.2. 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*.

2.3.3. 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. 31 Struktur *Gesture Library* [8]

2.4. Definisi Aplikasi *Mobile*

Menurut *Buyens,* Aplikasi *mobile* adalah 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 lain-lain. 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, 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.

2.6. Pemrograman Berorientasi Objek

Perkembangan gaya pemrograman mencapai gaya pemrograman orientasi objek setelah era pemrograman terstruktur. Pemrograman berorientasi objek menggantikan pemrograman terstruktur karena mempunyai banyak keunggulan dalam menangani proyek yang luar biasa kompleks. Pemrograman menggunakan bahasa orientasi objek menawarkan fleksibilitas, kegunaulangan dan kemudahan perawatan [9].

Dengan pemrograman berorientasi objek, para pengembang menciptakan blok-blok kode, yang dinamakan objek [9]. Objek-objek ini kemudian digunakan oleh berbagai aplikasi. Suatu saat jika objek mebutuhkan perubahan, para pengembang hanya melakukan perubahan sekali saja dan kemudian ia dengan mudah akan mewariskannya ke objek-objek lain yang menjadi turunannya. Dengan kepraktisannya, perusahaan-perusahaan perangkat lunak saat ini berusaha mengadopsi teknologi baru ini dan mengintegrasikannya pada aplikasi-aplikasi yang sudah ada. Kenyataannya, kebanyakan aplikasi-aplikasi modern saat ini berorientasi objek.

Paradigma berorientasi objek adalah cara yang berbeda dalam memandang aplikasi-aplikasi [9]. Dengan pendekatan berorientasi objek, para pengembang membagi aplikasi-aplikasi besar menjadi objek-objek, yang mandiri satu terhadap

lainnya. Pengembang kemudian mengembangkan aplikasi dengan membuat objek-objek itu saling mengirim pesan (*message*) dan bekerja-sama. Diambil suatu contoh. Dengan pemrograman terstruktur, katakanlah ingin membuat *form* dengan *listbox* di dalamnya. Untuk hal ini, maka harus menulis kode-kode yang jumlahnya sangat banyak untuk membuat *form* dan *listbox* itu sendiri dan kemudian membuat kode-kode untuk tombol OK. Dengan pemrograman berorientasi objek, maka dengan sederhana hanya perlu menggunakan objek-objek yang telah ada sebelumnya: *form*, *listbox*, dan tombol OK. Maka, alih-alih membuat segalanya dari awal dengan pemrograman terstruktur, dengan pemrograman berorientasi objek tinggal menggunakan objek-objek yang sudah ada dan kemudian berfokus pada 'apa yang unik' untuk aplikasi tertentu yang sedang dikembangkan. "*Do not reinventing the wheel!*", kata banyak ahli. Maka, keunggulan utama mengembangkan objek-objek sekali saja dan menggunakannya lagi (*reusable*) berulang-ulang. Dengan pendekatan berorientasi objek, difokuskan pada informasi dan perilaku yang dimiliki suatu objek sehingga kemudian pengembang dapat mengembangkan sistem/perangkat lunak yang fleksibel dalam menghadapi perubahan-perubahan informasi dan/atau perilaku yang dituntut pengguna.

2.7. UML (*Unified Modelling Language*)

Unified Modelling Language (UML) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah untuk dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka yang lain [10].

Unified Modelling Language merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh *Booch, Object Modelling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE) [10]. Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan *design* ke dalam empat tahapan iteratif, yaitu identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode

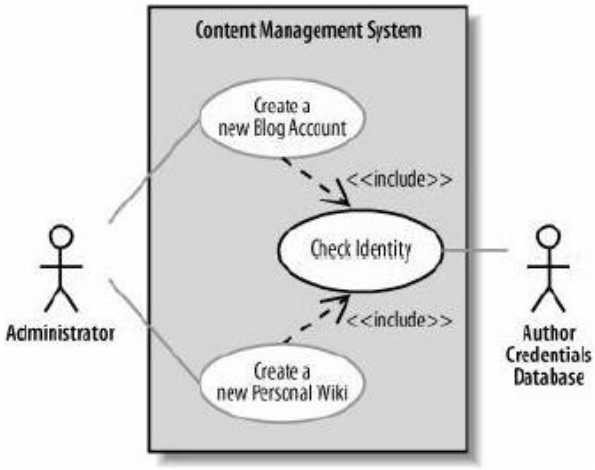
Booch adalah pada detail kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, design sistem, *design* objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep *OO*. Metode *OOSE* dari *Jacobson* lebih memberi penekanan pada *use case*. *OOSE* memiliki tiga tahapan yaitu membuat model requirement dan analisis, design dan implementasi, dan model pengujian (*test model*). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan *Unified Modelling Language*, metode *Booch*, *OMT* dan *OOSE* digabungkan dengan membuang elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga *Unified Modelling Language* lebih ekspresif dan seragam daripada metode lainnya.

2.7.1. *Use Case Diagram*

Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. Setiap skenario mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat biasa dikatakan *use case* adalah serangkaian skenario yang digabungkan bersama-sama oleh tujuan umum pengguna [10].

Berikut ini adalah contoh Use Case Diagram:



Gambar 2. 32 Use Case Diagram [11]

2.7.2. *Use Case Scenario*

Sebuah diagram yang menunjukkan kasus penggunaan dan aktor dapat menjadi titik awal yang bagus, tetapi tidak memberikan cukup detail untuk desainer sistem untuk benar-benar memahami persis bagaimana kekhawatiran sistem akan dipenuhi. *Use Case* Skenario adalah cara terbaik untuk mengungkapkan informasi penting ini dalam bentuk penggunaan description setiap kasus berbasis teks harus disertai dengan satu [11].

Berikut ini adalah contoh *Use Case Scenario*:

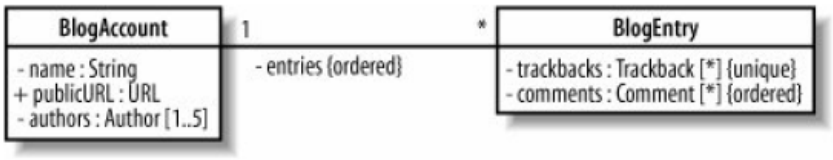
Use case name	**Create a new Blog Account**	
Related Requirements	Requirement A.1.	
Goal In Context	A new or existing author requests a new blog account from the Administrator.	
Preconditions	The system is limited to recognized authors and so the author needs to have appropriate proof of identity.	
Successful End Condition	A new blog account is created for the author.	
Failed End Condition	The application for a new blog account is rejected.	
Primary Actors	Administrator.	
	Secondary Actors	Author Credentials Database.
	Trigger	The Administrator asks the CMS to create a new blog account.
Main Flow	Step	Action
	1	The Administrator asks the system to create a new blog account.
	2	The Administrator selects an account type.
	3	The Administrator enters the author's details.
	4	The author's details are verified using the Author Credentials Database.
	5	The new blog account is created.
	6	A summary of the new blog account's details are emailed to the author.
Extensions	Step	Branching Action
	4.1	The Author Credentials Database does not verify the author's details.
	4.2	The author's new blog account application is rejected.

Gambar 2. 33 Use Case Scenario [11]

2.7.3. *Class Diagram*

Class Diagram adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang kita kembangkan. Diagram kelas memberikan gambaran/diagram statis tentang sistem/perangkat lunak dan relasi-relasi yang ada di dalamnya [12].

Berikut ini adalah contoh *Class Diagram*:



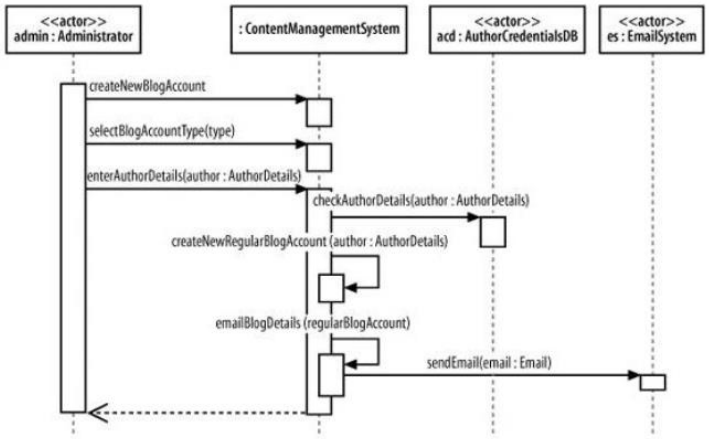
Gambar 2. 34 Class Diagram [11]

2.7.4. *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh objek dan *message* (pesan) yang diletakkan diantara objek-objek ini dalam *use case*.

Komponen utama sequence diagram terdiri atas objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertikal* [10].

Berikut ini adalah contoh *Sequence Diagram*:

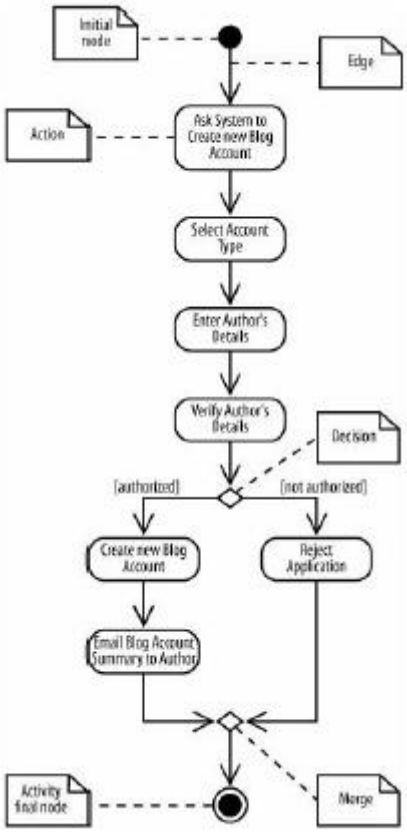


Gambar 2. 35 Sequence Diagram [11]

2.7.5. *Activity Diagram*

Activity Diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* mendukung perilaku paralel sedangkan *flowchart* tidak bisa.

Berikut ini adalah contoh *Activity Diagram*



Gambar 2. 36 Activity Diiagram [10]

2.8. Java

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” [13].

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.

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

- ✓ 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.
- ✓ Java menerapkan *array* sebenarnya, menghilangkan keperluan aritmatika pointer yang berbahaya dan berpeluan besar menyebabkan kesalahan.
- ✓ 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.
- ✓ 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.

2.9. Android

Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. 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, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. Sistem Operasi Android sendiri mencakup banyak ruang lingkup misalkan Android 1.5 (*Cup Cake*) SDK, Android 1.6 (*Donut*) SDK, Android 2.0/2.1 (*Eclair*) SDK, Android 2.2 (*Froyo – Frozen Yogurt*), Android 2.3 (*Gingerbread*), Android versi 3.0/3.1 (*Honeycomb*), Android versi 4.0 (*ICS: Ice Cream Sandwich*), Android versi 4.1 (*Jelly Bean*),

Android versi 4.4 (*KitKat*), Android versi 5.0 (*Lollipop*) dan Android versi 6.0 (*Marshmallow*).

2.10. Android Studio

Android Studio adalah sebuah IDE yang bisa digunakan untuk pengembangan aplikasi Android, dan dikembangkan oleh *Google* [14]. Android Studio merupakan pengembangkan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu IntelliJ IDEA. Android Studio direncanakan untuk menggantikan Eclipse ke depannya sebagai IDE resmi untuk pengembangan aplikasi Android. Sebagai pengembangan dari Eclipse, Android Studio mempunyai banyak fitur-fitur baru dibandingkan dengan Eclipse IDE. Berbeda dengan Eclipse yang menggunakan Ant, Android Studio menggunakan Gradle sebagai *build environment*.

2.11. Skala Guttman

Skala Guttman diberinama menurut ahli yang mengembangkannya, yaitu Louis Guttman [15]. Skala ini mempunyai beberapa ciri penting, yaitu :

a. Skala Guttman merupaka skala kumulatif. Jika seseorang mengiyakan pertanyaan atau pernyataan yang berbobot lebih berat, maka ia juga akan mengiyakan pertanyaan atau pernyataan yang kurang berbobot lainnya.
b. Skala Guttman ingin mengukur satu dimensi saja dari satu variabel yang multidimensi, sehingga skala ini termasuk mempunyai sifat unidimensional.

Cara membuat skala Guttman adalah sebagai berikut :

a) Susun sejumlah pertanyaan yang relevan dengan masalah yang ingin diselidiki.
b) Lakukan penelitian permulaan terhadap sejumlah responden yang dapat mewakili populasi yang akan diteliti. Sampel yang dipilih minimal besarnya 30.
c) Jawaban yang diperoleh kemudian dianalisis.
d) Susunlah jawaban pada suatu tabel Guttman.
e) Hitunglah koefisien reprodusibilitas dan koefisien skalabilitas

Rumus Koefisien Reprodusibilitas

$$Kr = 1 - \{e / n\}$$

Keterangan :

n : total kemungkinan jawaban, yaitu jumlah pertanyaan x jumlah responden

e : jumlah error

Kr : Koefisien Reprodusibilitas

Rumus tersebut akan digunakan untuk menghitung Koefisien reprodusibitilas jawaban kuesioner. Koefisien reprodusibilitas adalah suatu besaran yang mengukur derajat ketepatan alat ukur yang dibuat (daftar pertanyaan). Skalan Guttman menghendaki nilai koefisien reprodusibilitas > 0.90

Rumus Koefisien Skalabilitas

$$Ks = 1 - \{e / p\}$$

Keterangan :

e = jumlah error

p = jumlah kesalahan yang diharapkan

Ks = koefisien skalabilitas

Koefisien Skalabilitas digunakan untuk menghitung kemungkinan jumlah kesalahan yang diharapkan. Skala yang mempunyai Ks > 0,6 sudah dianggap baik.