BAB 2
LANDASAN TEORI

2.1 Multimedia

Tay Vaughan(2006:3) menyebutkan definisi Multimedia diartikan sebagai kombinasi dari macam-macam objek media, yaitu teks, foto, seni grafis, animasi, audio, video yang dimanipulasi secara digital [4].

a. Teks

Teks merupakan dasar dari pengolahan kata dan informasi berbasis multimedia. Beberapa hal yang perlu diperhatikan adalah penggunaan hypertext, auto hypertext, text style, import text dan export text.

b. Image

Secara umum image atau grafik berarti still image (gambar tetap), dari kelas objek termasuk dalam kategori image adalah visible image, non-visible, abstrak.

c. Animasi

Animasi berarti gerakan-gerakan image atau video. Dalam authoring software, biasanya animasi mencakup kemampuan recording dan playback. Fasilitas yang dimiliki oleh software animasi mencakup integrated animation, tool, animation clip, impor animasi, recording, play back dan transition effect.

d. Audio

Penyajian audio merupakan cara lain untuk memperjelas pengertian suatu informasi. Contohnya, narasi merupakan kelengkapan dan penjelasan yang dilihat melalui video.

e. Full-motion dan live video

Full-motion dan live video berhubungan dengan penyimpanan sebagai video klip, sedangkan live video merupakan hasil pemrosesan yang diperoleh dari kamera.

William Ditto (2006) menyatakan definisi multimedia dalam ilmu pengetahuan mencakup beberapa aspek yang saling bersinergi, antara teks, grafik, gambar statis, animasi, film dan suara. Sejumlah penelitian membuktikan bahwa penggunaan multimedia dalam pembelajaran menunjang efektivitas dan efisiensi

proses pembelajaran. Penelitian tersebut antara lain yang dilakukan oleh Francis M. Dwyer. Hasil penelitian ini antara lain menyebutkan bahwa setelah lebih dari tiga hari pada umumnya manusia dapat mengingat pesan yang disampaikan melalui tulisan sebesar 10 %, pesan audio 10 %, visual 30 % dan apabila ditambah dengan melakukan multimedia, maka akan mencapai 80 %. Berdasarkan hasil penelitian ini maka multimedia interaktif (user melakukan) dapat dikatakan sebagai media yang mempunyai potensi yang sangat besar dalam membantu proses pembelajaran [4].

Multimedia interaktif adalah suatu multimedia yang dilengkapi dengan alat pengontrol yang dapat dioperasikan 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 dioperasikan oleh user untuk 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 penayangannya tergantung seberapa lama pengguna mem-browsing media ini [4].

2.2 Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media.

Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut suatu paket atau suite aplikasi (*Application Suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu

paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Seringkali, mereka memiliki kemapuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah [3].

2.3 Android

2.3.1 Pengertian

Android adalah sistem operasi yang berbasis *Linux* untuk telepon seluler seperti telepon pintar (*Smartphone*) dan komputer *tablet*. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti begerak.

Bebrapa pengertian lain dari *Android*, yaitu :

1. Merupakan *platform* terbuka (*Open Source*) bagi para programmer untuk membuat atau mengembangkan aplikasi.
2. Merupakan sistem operasi yang dibeli *Google Inc* dari *Android Inc*.
3. Bukan bahasa pemrograman, akan tetapi hanya menyediakan lingkungan hidup atau *runtime environment* yang disebut DVM (*Dalvik Virtual Machine*) yang telah dioptimasi untuk *device* dengan sistem memori yang kecil [2].

2.3.2 Versi Android

Banyak *smartphone* dan PC Tablet yang menggunakan sistem operasi android dengan versi yang berbeda. Semakin tinggi versi, fiturnya semakin canggih dan banyak. Beberapa uraian versi *android* seperti dibawah ini [2]:

1. *Android* versi 1.1

Pada 9 Maret 2009, *Google* merilis *Android* versi 1.1. *Android* versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan Email.

2. *Android* versi 1.5 (*Cupcake*)

Pada pertengahan Mei 2009, *Google* kembali merilis telepon seluler dengan menggunakan *Android* dan SDK (*Software Development Kit*) dengan versi 1.5

(*Cupcake*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke *Youtube* dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke *headset bluetooth*, animasi layar dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

3. *Android* versi 1.6 (*Donut*)

Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol *applet* VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus pada kamera, *camcorder* dan galeri yang diintegrasikan pada CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text-to-spech engine*. Kemampuan dial kontak teknologi *text to change speech* (tidak tersedia pada semua ponsel).

4. *Android* versi 2.0/2.1 (*Eclair*)

Pada 3 Desember 2009 kembali diluncurkan ponsel *Android* dengan versi 2.0/2.1 (*Eclair*), perubahan yang dilakukan adalah mengoptimalkan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML 5, daftar kontak yang baru, dukungan *flash* untuk kamera 3.2 MP, *digital zoom*, dan *Bluetooth* 2.1.

Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan kompetisi aplikasi *mobile* terbaik (*killer apps*-aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik.

Dengan semakin berkembangnya dan semakin bertambahnya jumlah *handset Android*, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi *Android*. Aplikasi terkenal yang diubah ke dalam sistem operasi *Android* adalah Shazam, *Backgrounds* dan *WeatherBug*.

Sistem operasi *Android* dalam situs internet juga dianggap penting untuk menciptakan aplikasi *Android* asli, contohnya oleh MySpace dan Facebook.

5. *Android* versi 2.2 (*Froyo* : *Frozen Yoghurt*)

Pada 20 Mei 2010, *Android* versi 2.2 (Froyo) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, integrasi dengan V8 JavaScript *engine* yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portabel dan kemampuan *auto update* dalam aplikasi Android Market.

6. *Android* versi 2.3 (*Gingerbread*)

Pada 6 Desember 2010, Android versi 2.3 (*Gingerbread*) diluncurkan. Perubahan-perubahan umum yang didapat dari versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antarmuka (*User Interface*) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (*reverb, equalization, headphone virtualization,* dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC) dan dukungan jumlah kamera yang lebih dari satu.

7. *Android* versi 3.1 (*Honeycomb*)

Android Honeycomb dirancang khusus untuk tablet. *Android* versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung multi prosesor dan juga akselerasi perangkat keras (*hardware*) untuk grafis. Tablet peratam yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

8. *Android* versi 4.0 (*Ice Cream Sandwich*)

Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Honeycomb* untuk *smartphone* dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara *offline* dan berbagi informasi dengan menggunakan NFC.

9. *Android* versi 4.1 (*Jelly Bean*)

Pada tanggal 27 Juni 2012 Google mengumumkan *Android* versi 4.1 (*Jelly Bean*) dalam konferensi Google I/O berdasarkan kernel Linux 3.0.31. *Jelly Bean*

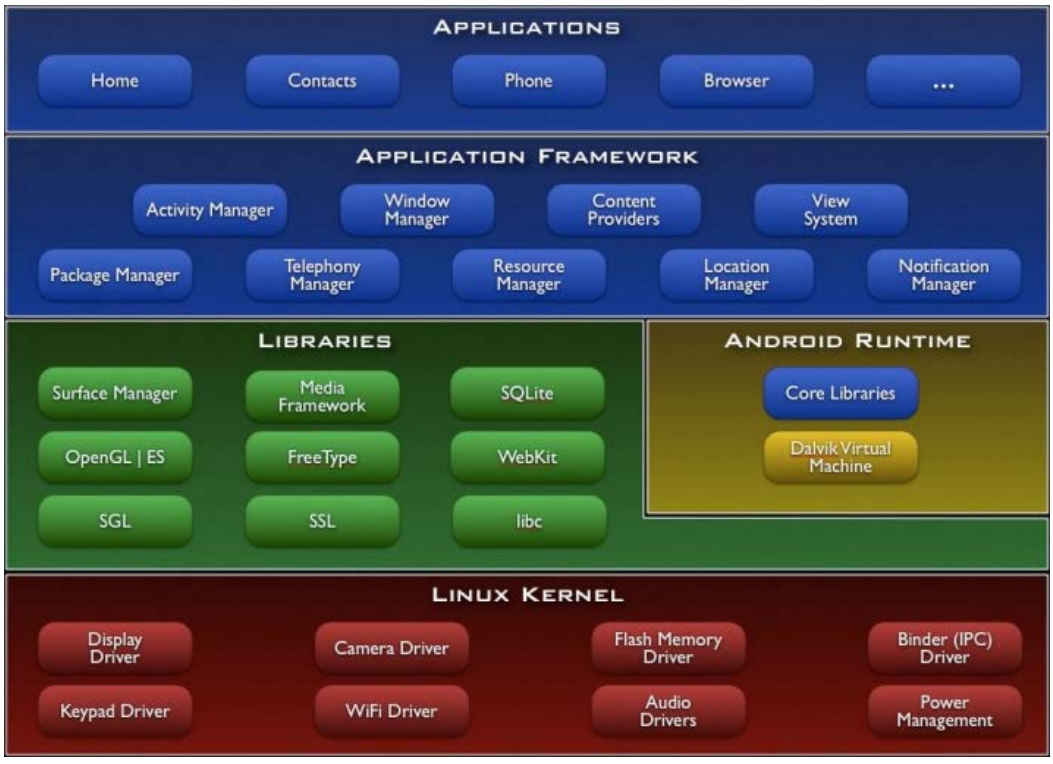
adalah pembaruan penting yang bertujuan untuk meningkatkan fungsi dan kinerja antarmuak pengguna (*User Interface*). Pembaruan ini diwujudkan dalam proyek *Butter*, perbaikan ini termasuk antisipasi sentuh, *triple buffering*, perpanjangan waktu vsync dan peningkatan *frame rate* hingga 60fps untuk menciptakan UI yang lebih halus. Perangkat pertama yang menggunakan sistem operasi ini adalah tablet Nexus 7, yang dirilis pada 13 Juli 2012.

10. *Android* versi 4.4 (KitKat)

Android versi 4.4 (KitKat) direncanakan akan dirilis pada bulan Oktober 2013. Setelah sebelumnya beredar rumor bahwa Android versi berikutnya setelah *Jelly Bean* diperkirakan akan diberi nomor 5.0 dan dinamai Key Lime Pie.

2.3.3 Anatomi Aplikasi Android

Dalam paket sistem operasi Android tediri dari beberapa unsur seperti tampak pada gambar 2.1. Secara sederhana arsitektur Android merupakan sebuah kernel Linux dan sekumpulan pustaka C / C++ dalam suatu *Framework* yang menyediakan dan mengatur alur proses aplikasi [2].



Gambar 2.1 Detail Anatomi Android

2.3.4 Komponen Aplikasi *Android*

Fitur penting android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada.

Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan [2].

2.3.5 Tipe Aplikasi *Android*

Terdapat tiga kategori aplikasi pada android [2]:

1. *Foreground Activity*

Aplikasi yang hanya dapat dijalankan jika tampil pada layar dan tetap efektif walaupun tidak terlihat. Aplikasi dengan tipe ini pasti mempertimbangkan siklus hidup *activity*, sehingga perpindahan antar *activity* dapat berlangsung dengan lancar.

2. *Background Service*

Aplikasi yang memiliki interaksi terbatas dengan user, selain dari pengaturan konfigurasi, semua dari prosesnya tidak tidak tampak pada layar. Contohnya aplikasi penyaringan panggilan atau sms *auto respon*.

3. *Intermittent Activity*

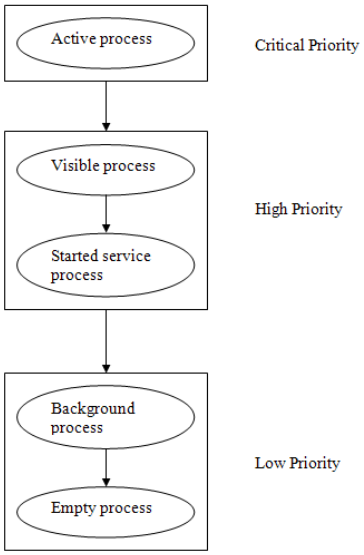
Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika diperlukan akan memberi tahu pengguna tentang kondisi tertentu. Contohnya pemutar musik. Untuk aplikasi yang kompleks akan sulit untuk menentukan kategori aplikasi tersebut apalagi aplikasi memiliki ciri-ciri dari semua kategori. Oleh karenanya

perlu pertimbangan bagaimana aplikasi tersebut digunakan dan menentukan kategori aplikasi yang sesuai.

2.3.6 Siklus Hidup Android

Siklus hidup aplikasi Android dikelola oleh sistem, berdasarkan kebutuhan pengguna, sumberdaya yang tersedia, dan sebagainya. Misalnya Pengguna ingin menjalankan *browser web*, pada akhirnya sistem yang akan menentukan menjalankan aplikasi. Sistem sangat berperan dalam menentukan apakah aplikasi dijalankan, dihentikan sementara, atau dihentikan sama sekali. Jika pengguna ketika itu sedang menjalankan sebuah *Activity*, maka sistem akan memberikan perioritas utama untuk aplikasi yang tersebut. Sebaliknya, jika suatu *Activity* tidak terlihat dan sistem membutuhkan sumber daya yang lebih, maka *Activity* yang prioritas rendah akan ditutup.

Android menjalankan setiap aplikasi dalam proses secara terpisah, yang masing-masing memiliki mesin *virtual* pengolah sendiri, dengan ini melindungi penggunaan memori pada aplikasi. Selain itu juga Android dapat mengontrol aplikasi mana yang layak menjadi prioritas utama. Karenanya Android sangat sensitive dengan siklus hidup aplikasi dan komponen-komponennya. Perlu adanya penanganan terhadap setiap kondisi agar aplikasi menjadi stabil. Gambar 2.2 menunjukkan prioritas dari aplikasi [2].



Gambar 2.2 Prioritas Aplikasi Berdasarkan *Activity*

2.3.7 Kelebihan *Android*

Sudah banyak platform untuk perangkat selular saat ini, termasuk didalamnya Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiM), dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan Android. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada *platform* lain, Android adalah yang pertama menggabungkan hal seperti berikut [2]:

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan Linux dan *open source*. Pembuat perangkat menyukai hal ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar royality. Sementara pengembang *software* menyukai karena Android dapat digunakan diperangkat manapun dan tanpa terikat oleh *vendor* manapun.
2. Arsitektur komponen dasar Android terinspirasi dari teknologi *internet Mashup*. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.
3. Banyak dukungan *service*, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, *database* SQL, *browser* dan penggunaan peta. Semua itu sudah tertanam pada Android sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja sistem menjadi lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.
5. Dukungan grafis dan suara terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh Flash menyatu dalam 3D menggunakan OpenGL memungkinkan membuat aplikasi maupun *game* yang berbeda.
6. *Portabilitas* aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan

menggunakan bahas pemrograman Java dan dieksekusi oleh mesin *virtual Dalvik*, sehingga kode program *portable* antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard*, layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

2.4 Perancangan Berorientasi Objek

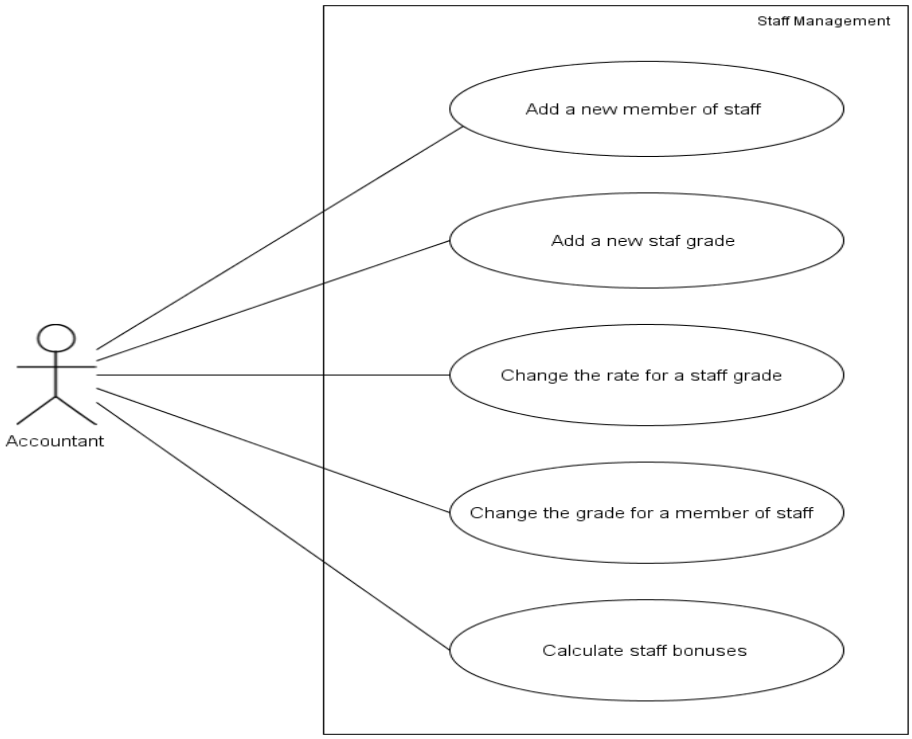
Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah mempermudah *programmer* didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata. Suatu perusahaan *software* yaitu *Rational Software*, telah membentuk konsarium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysist Design* (OOAD) [1].

2.4.1 Unified Modelling Language (UML)

UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifacts dari sistem *software*, untuk memodelkan bisnis, dan sistem *non-software* lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan *artifact* yang terdapat dalam sistem. *Artifact* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifact* dapat berupa model, deskripsi atau *software* [1].

2.4.1.1 Use Case Diagram

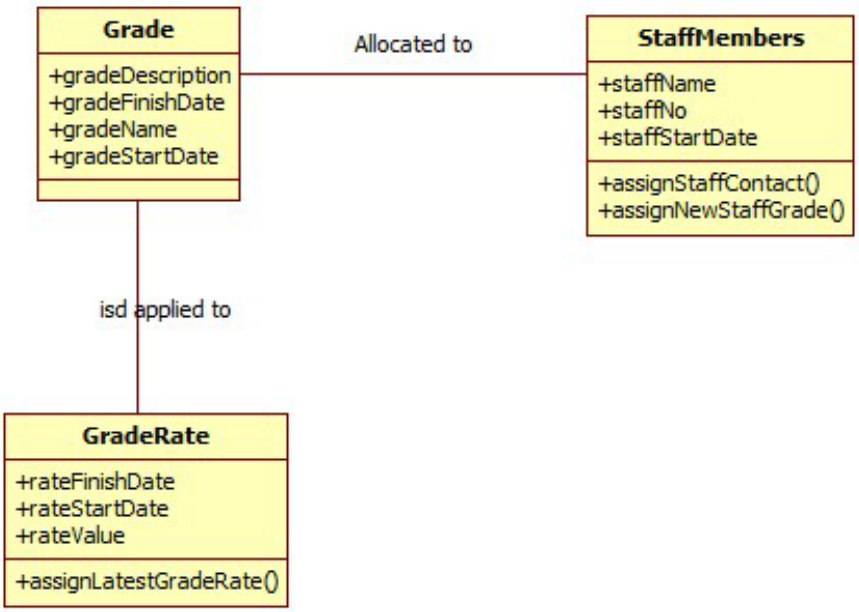
Use Case Diagram menjelakan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (Aktor). Diagram ini menunjukan fungsionalitas suatu sistem yang berinteraksi dengan dunia luar. *Use Case Diagram* dapat digunakan selama proses analisis untuk menagkap *requitment* sistem dan untuk memahami bagaimana sistem bekerja [1].



Gambar 2.3 *Use Case Diagram*

2.4.1.2 Class Diagram

Class Diagram menjelaskan dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class Diagram* memperlihatakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam model desain dari suatu sistem. Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan prilaku sistem. Selama tahap desain, *class diagram* berperan dalam menagkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat [1].



Gambar 2.4 *Class Diagram*

2.4.1.3 Behavior Diagram

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu [1]:

1. *Statechart Diagram*

Statechart Diagram berfungsi untuk memodelkan prilaku dinamis satu kelas satu objek.

2. *Activity Diagram*

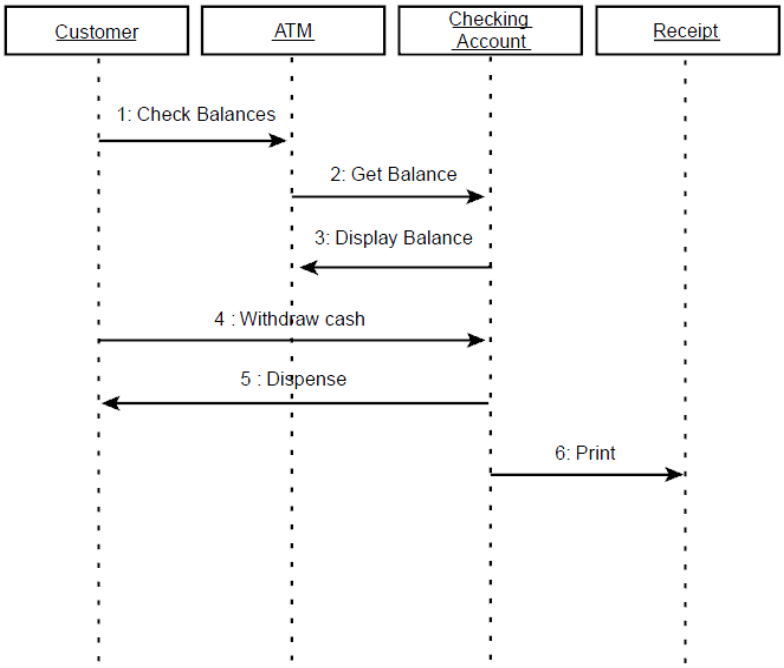
Activity Diagram memodelkan alur kerja (work flow) sebuah proses bisnis dan urutan aktifitas dalam suatu proses.

3. *Interaction Diagram*

Interaction Diagram dibagi menjadi dua model diagram yaitu :

1. ***Sequence Diagram***

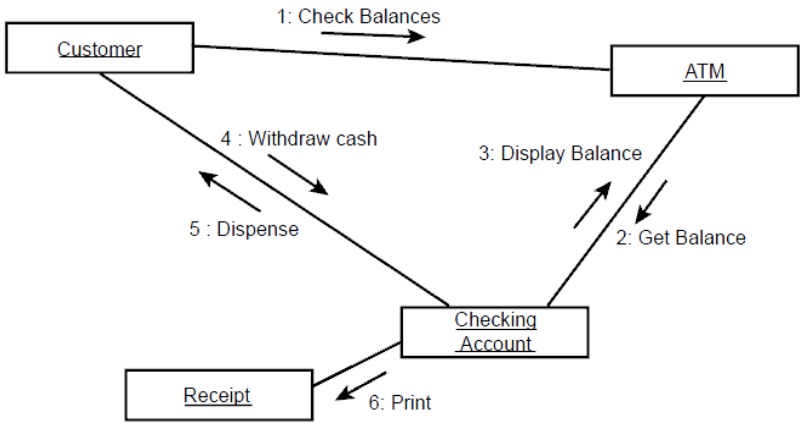
   Sequence Diagram menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus bersosialisasi dengan *use case*. Sequence diagram, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu dalam *use case*.



Gambar 2.5 *Sequence Diagram*

2. *Collaboration Diagram*

Collaboration Diagram melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (*relationship*) antar objek, sedangkan *sequence diagram* menekankan pada urutan kejadian. Dalam *collaboration diagram* terdapat beberapa objek, *link*, dan *message*.



Gambar 2.6 *Collaboration Diagram*

2.4.1.4 Implementation Diagram

Implementation diagram dibagi menjadi dua diagram, yaitu [1]:

1. *Component Diagram* menggambarkan alokasi semua kelas dan objek kedalam komponen-komponen dalam desain fisik sistem *software*. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen-komponen *software*, seperti *source code*, *binary code*, dan komponen tereksekusi (*execute components*)
2. *Deployment Diagram* memperlihatkan pemetaan *software* kepada *hardware*. Dimana akan berjalan (di *server/multitier*, *standalone* atau lainnya), dan menggambarkan model koneksi dan kemampuan jaringan dan hal lainya yang bersifat fisik.

2.5 Skala Likert (*Likert Scale*)

Skala likert adalah suatu skala psikometrik yang digunakan dalam kuesioner dan merupakan salah satu teknik yang dapat digunakan dalam evaluasi suatu program atau kebijakan perencanaan. Rensis Likert telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala Likert. Skala Likert ini merupakan skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena. Dan pada evaluasi, skala likert digunakan untuk (a) Menilai keberhasilan suatu kebijakan atau program (b) Menilai manfaat pelaksanaan suatu kebijakan atau program (c) Mengetahui kepuasan stakeholder terhadap pelaksanaan suatu kebijakan atau program [3].

Dalam skala Likert terdapat dua bentuk pernyataan yaitu pernyataan positif yang berfungsi untuk mengukur sikap positif, dan pernyataan negatif yang berfungsi untuk mengukur sikap negatif objek. Skor pernyataan positif dimulai dari 1 untuk sangat tidak setuju (ST), 2 untuk tidak setuju (T), 3 untuk tak memutuskan (N), 4 untuk setuju (S), dan 5 untuk sangat setuju (SS). Skor pernyataan negatif dimulai dari 1 untuk sangat setuju (SS), 2 untuk setuju (S), 3 untuk tak memutuskan (N), 4 untuk tidak setuju (T), dan 5 untuk sangat tidak setuju (ST). Skala Likert digunakan untuk mengukur **kesetujuan dan**

ketidaksetujuan seseorang terhadap sesuatu rencana program, pelaksanaan program ataupun tingkat keberhasilan suatu program.

Tabel 2.1 Bobot Penilaian Skala Likert

Arah Pertanyaan	Bobot Penilaian				
	Sangat Setuju (SS)	Setuju (S)	Tak Memutuskan (N)	Tidak Setuju (T)	Sangat Tidak Setuju (ST)
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

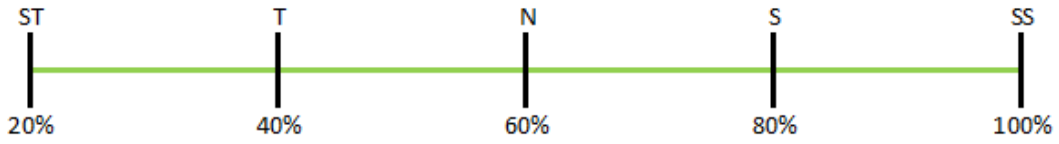
Berdasarkan data hasil kuesioner tersebut, dapat dicari persentase masing-masing jawaban dengan menggunakan rumus :

$$Y = \frac{X}{Skor\ ideal}\ x\ 100\%$$

Keterangan :

Y	=	Nilai persentase yang dicari
X	=	Jumlah nilai kategori jawaban dikalikan dengan frekuensi ($\sum$ = N . R)
N	=	Nilai dari setiap jawaban
R	=	Frekuensi
Skor ideal	=	Nilai tertinggi dikalikan dengan jumlah sampel

Setelah diperoleh nilai Y maka interpretasi nilai dapat dilihat sebagai berikut :



Gambar 2.7 Skala Penilaian Kuesioner

Setelah didapatkan jumlah yang pasti dari setiap kuesioner, selanjutnya penulis menarik kesimpulan dari masing-masing kuesioner yang disediakan.

2.5.1 Kelebihan Skala Likert

Beberapa Kelebihan Skala Likert dalam riset berupa survei, antara lain :

1. Mudah dibuat dan diterapkan.
2. Skala Likert lebih mudah membuatnya dibandingkan dengan skala yang lain seperti skala Thurstone.
3. Terdapat kebebasan dalam memasukan pertanyaan- pertanyaan, asalkan sesuai dengan konteks permasalahan yang diteliti.
4. Jawaban suatu item dapat berupa alternatif, sehingga informasi mengenai item tersebut diperjelas.
5. Reliabilitas pengukuran bisa diperoleh dengan jumlah item tersebut diperjelas.
6. Karena jangka responsi yang lebih besar membuat skala Likert dapat memberikan keterangan yang lebih jelas dan nyata tentang pendapatan atau sikap responden tentang isu yang dipertanyakan.

2.5.2 Kekurangan Skala Likert

Beberapa Kekurangan Skala Likert dalam riset berupa survei, antara lain :

1. Karena ukuran yang digunakan adalah ukuran ordinal, skala Likert hanya dapat mengurutkan individu dalam skala, tetapi tidak dapat membandingkan berapa kali satu individu lebih baik dari individu yang lain.
2. Kadangkala total skor dari individu tidak memberikan arti yang jelas, karena banyak pola respons terhadap beberapa item akan memberikan skor yang sama. Adanya kelemahan di atas sebenarnya dapat dipikirkan sebagai *error* dari respons yang terjadi.

2.6 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse [2]:

1. *Multi-platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.

2. *Multi-language*: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.
4. Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.6.1 Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP :

1. *Core platform*
2. OSGi
3. SWT (*Standard Widget Toolkit*)
4. JFace
5. Eclipse *Workbench*

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse

tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (C/C++ *Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, *plug-in* UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya [2].

2.6.2 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari Eclipse Platform dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise. Adapun versi Eclipse yang telah diluncurkan yaitu [2]:

1. Eclipse 3.0
2. Eclipse 3.1
3. Callisto
4. Europa
5. Ganymede
6. Galileo
7. Helios
8. Indigo
9. Juno
10. Kepler

2.7 Software Develpment Kit (SDK)

Software Development Kit (SDK) adalah suatu kit atau library dari bahasa pemrograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya SDK terdiri dari kumpulan tools yang dibutuhkan. Misalnya bahasa

pemrograman java, mempunyai SDK yang berisi suatu *library* yang dapat digunakan untuk membuat suatu aplikasi berbasis java [2].

2.8 Java Development Kit (JDK)

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang berbasis Java, Sedangkan JRE adalah sebuah implementasi dari Java Virtual Machine yang benar-benar digunakan untuk menjalankan program java.Biasanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber kompiler java, bundling, debuggers, development libraries dan lain sebagainya. Perbedaan JDK dengan SDK *(Software Development Kit)* yaitu JDK adalah sebuah SDK tetapi sebuah SDK tidak harus menjadi sebuah JDK [2].

2.9 Android Development Tools (ADT)

Android Development Tools (*ADT*) adalah *plugin* untuk Eclipse *Intergrated Development Environment* (*IDE*) yang dirancang untuk memberikan lingkungan yang terpadu di mana untuk membangun aplikasi Android.

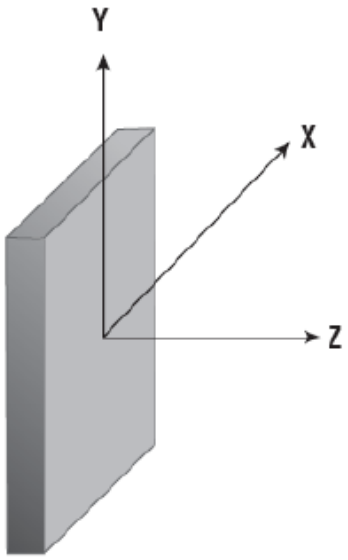
ADT memperluas kemampuan Eclipse untuk membiarkan para *developer* lebih cepat dalam membuat proyek baru Android, membuat aplikasi UI, menambahkan komponen berdasarkan Android *Framework* API, *debug* aplikasi dalam pengunaan Android SDK, dan membuat file APK untuk mendistribusikan aplikasi.

Mengembangkan aplikasi di Eclipse dengan ADT sangat dianjurkan dan merupakan cara tercepat untuk memulai membuat aplikasi Android, karena banyak kemudahan-kemudahan sebagai tools yang terintegrasi seperti, custom XML editor, dan *debug panel ouput*. Selain itu ADT memberikan dorongan luar biasa dalam mengembangkan aplikasi Android [2].

2.10 Sensor Accelerometer

Accelerometer, seperti namanya, digunakan untuk mengukur percepatan. Percepatan didefinisikan sebagai laju perubahan kecepatan, sehingga dapat mengukur

seberapa cepat akselerasi pada perangkat berubah dalam arah tertentu. Dengan menggunakan *accelerometer*, anda dapat mendeteksi gerakan, laju perubahan dari kecepatan gerakan yang terjadi pada perangkat. Sangat penting untuk dicatat bahwa *accelerometer* tidak mengukur kecepatan, sehingga anda tidak bisa mengukur kecepatan secara langsung berdasarkan pembacaan *accelerometer* tunggal. Sebaliknya, anda perlu untuk mengukur perubahan dalam percepatan dari waktu ke waktu. Percepatan dapat diukur oleh tiga sumbu arah : *forward-backward* (*longitudinal*), *left-right* (literal), dan *up-down* (*vertical*). Sensor manager melaporkan perubahan sensor di semua tiga sumbu arah. Seperti yang di ilustrasikan pada gambar 2-7 [5]:



Gambar 2.8 Accelerometer

1. *Vertical* (*up-down*), dimana arah positif (yAxis) berarti ke atas gerakan perangkat yang sedang di tinggikan dan sebaliknya adalah nilai negative (raw_yAxis).
2. *Longitudinal* (*forward-backward*), dimana gerakan kedepan yang menandakan percepatan positif (zAxis) dan gerakan kebelakang yang menandakan percepatan negative (raw_zAxis).
3. *Sideways lateral* (*left-right*), dimana gerakan kearah kanan dari perangkat yang mewakili nilai positif (xAxis) dan gerakan perangkat kea rah kiri yang mewakili nilai negative (raw_xAxis).

Sensor *manager* mempertimbangkan perangkat diam ketika menghadap ke atas pada permukaan yang datar dalam orientasi *potret*. Seperti dijelaskan sebelumnya, kita dapat memantau perubahan percepatan dengan menggunakan sensor *listener*.

Sensor *Listener* yang di buat harus menerapkan *onSensorChanged method* yang akan dipicu ketika perubahan terjadi dalam percepatan disepanjang salah satu dari tiga sumbu yang telah dijelaskan sebelumnya. *onSensorChanged method* menerima *array float* yang berisi nilai percepatan di ketiga sumbu dalam format baku. Sensor *Manager* mencakup konstanta indeks yang dapat kita gunakan untuk mengambil nilai percepatan yang dibutuhkan, seperti terlihat pada contoh potongan kode berikut [5]:

```
SensorListener mySensorListener = new SensorListener() {
public void onSensorChanged(int sensor, float[] values) {
if (sensor == SensorManager.SENSOR_ACCELEROMETER) {
float xAxis = values[SensorManager.DATA_X];
float yAxis = values[SensorManager.DATA_Y];
float zAxis = values[SensorManager.DATA_Z];
float raw_xAxis = values[SensorManager.RAW_DATA_X];
float raw_yAxis = values[SensorManager.RAW_DATA_Y];
float raw_zAxis = values[SensorManager.RAW_DATA_Z];
// TODO apply the acceleration changes to your application.
}
}
public void onAccuracyChanged(int sensor, int accuracy) { }
}
```

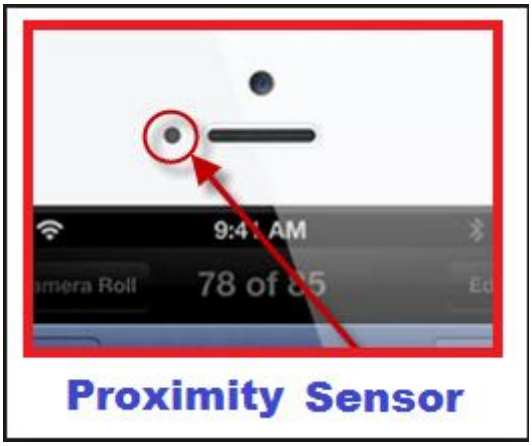
2.11 Sensor Proximity

Sensor *proximity* adalah sebuah sensor yang bisa mendeteksi keberadaan benda tanpa kontak fisik. Sensor *proximity* memancarkan medan elektromagnetik atau sinar radiasi elektromagnetik (misalnya inframerah) dan mendeteksi

perubahan bidang dengan mengembalikan sinyal. Ada empat jenis teknologi sensor *proximity*, diantaranya [3]:

1. *Electrical* (*Inductive* dan *Capacitive*)
2. *Optical* (IR dan Laser)
3. *Magnetic*
4. *Sonar*

Dari beberapa jenis sensor *proximity*, jenis sensor *proximity* dengan tipe *optical* paling banyak digunakan pada android, karena jenis sensor *proximity* dengan tipe *optical* lebih murah dan lebih sederhana dibandingkan dengan jenis sensor yang lainnya. Sensor *proximity* pada ponsel android biasanya terdapat di depan ponsel, tepatnya diatas layar ponsel seperti pada gambar 2-8.



Gambar 2.9 Posisi Sensor *Proximity* Pada Perangkat Android

Sensor proximity memungkinkan untuk menentukan seberapa jauh suatu objek dari perangkat. Biasanya digunakan untuk mematikan layar saat pengguna membuat panggilan atau menerima panggilan telepon. Contoh kode berikut menunjukkan bagaimana mendapatkan sebuah *instance* dari *sensor proximity* standar.

```
private SensorManager mSensorManager;
private Sensor mSensor;
…
mSensorManager = (SensorManager) getSystemService
```

```
(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDafaultSensor (Sensor.TYPE.PROXIMITY);
```

Kebanyakan *sensor proximity* mengembalikan jarak *absolut* dalam cm, tetapi beberapa *sensor proximity* juga hanya menentukan nilai dekat dan jauh. Kode berikut menunjukkan bagaimana menggunakan *sensor proximity*.

```
// Dapatkan sebuah instance dari layanan sensor, dan menggunakannya untuk mendapatkan instance dari sensor tertentu.
mSensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
mProximity = mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
}
public final void onAccuracyChanged(Sensor sensor, int accuracy) {}
public final void onSensorChanged(SensorEvent event) {
float distance = event.values[0];
// Lakukan sesuatu dengan data sensor ini.
}
protected void onResume() {
// Daftarkan listener untuk sensor
super.onResume();
mSensorManager.registerListener(this, mProximity,
SensorManager.SENSOR_DELAY_NORMAL);
}
}
```