BAB II

TINJAUAN PUSTAKA

II.1. Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yagn diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi sutau pake disebut sebagai suatu paket atau *suite* aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice.org*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki atarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan 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. (*Dahlan Abdullah ; 2013 : 152*)

Jenis-jenis Software Aplikasi :

1. *Software* aplikasi hiburan, contohnya yaitu winamp untuk mendengarkan musik, games dan sebagainya untuk hiburan.
2. *Software* aplikasi pendidikan yaitu software digunakan untuk mempelajari atau mereferensikan tentang pendidikan atau pengetahuan.

3. *Software* aplikasi bisnis yaitu software yang digunakan untuk aplikasi bisnis
4. *Software* aplikasi khusus
5. *Software* aplikasi untuk produtivitas kerja.

II.2. Permainan

Permainan (*games*) adalah setiap kontes antara pemain yang berinteraksi satu sama lain dengan mengikuti aturan-aturan tertentu untuk mencapai tujuan tertentu pula. Jadi permainan adalah cara bermain dengan mengikuti aturan-aturan tertentu yang dapat dilakukan secara individu maupun berkelompok guna mencapai tujuan tertentu. Alat permainan adalah semua alat bermain yang dapat digunakan oleh peserta didik untuk memenuhi naluri bermainnya dan memiliki barbagai macam sifat, seperti bongkar pasang, mengelompokkan, memadukan, mencari padanannya, merangkai, membentuk, atau menyusun sesuai dengan bentuk aslinya.(*Yumarlin MZ ; 2013 ; 76-77*)

II.2.1. Jenis-jenis Permainan

Ada beberapa macam permainan yang memiliki aturan-aturan tertentu dan tujuan tertentu pula. Adapun macam-macam permainan tersebut adalah sebagai berikut :

1. Permainan Individual

   Permainan ini peserta didik memainkan untuk menguji kemampuan sendiri karena sebagian besar permainan itu dilakukannya sendiri. Peserta didik bermain tanpa menghiraukan apa yang dilakukan oleh peserta didik lain disekitarnya. Contoh permainan individual adalah lompat tali, menyusun puzzle, menyusun balok-balok, dsb. *Yumarlin MZ* (2013 : 77)

2. Permainan Beregu

   Permainan beregu ini mempunyai aturanaturan yang diberikan sebelum permainan dimulai. Aturan permainan harus dimengerti oleh setiap pemain dan bersedia mengikuti aturan permainan. *Yumarlin MZ* (2013 :78)

3. Permainan Kooperatif

   Permainan ini ditandai dengan adanya kerjasama atau pembagian tugas dan pembagian peran antara peserta didik yang terlibat dalam permainan tersebut untuk mencapai tujuan dari kegiatan bermain. Permainan kerjasama dapat dilihat saat peserta didik mengerjakan suatu proyek atau tugas secara bersama-sama dalam kelompok kecil atau kelompok besar sekaligus. Bermain dengan bekerjasama ini bias dimulai oleh peserta didik sendiri atau dengan arahan dari guru. Permainan ini dapat mengembangkan keterampilan sosial dan konstruktif bagi peserta didik. Dalam permainan ini peserta didik dapat berperan serta dalam usaha untuk belajar memecahkan masalah secara bersama-sama. *Yumarlin MZ* (2013 : 78)

4. Permainan Sosial

   Permainan sosial adalah kegiatan bermain peserta didik dengan teman-temannya sendiri. Pada permainan ini peserta didik berpartisipasi dalam kegiatan bermain dengan peserta didik lainnya sesuai perannya masing-masing yang sudah disepakati sebalumnya. Contohnya seperti permainan polisi dengan pencuri, atau lompat tali beregu. *Yumarlin MZ* (2013 : 78)

5. Permainan dengan aturan tertentu

   Permainan ini ditandai dengan adanya kegiatan bermain yang menggunakan aturan-aturan tertentu. Dalam permainan ini peserta didik diharapkan dapat bersikap sportif. Contoh dari permainan ini adalah sepak bola, permainan ular tangga, monopoli, gobak sodor, dsb. *Yumarlin MZ* (2013 : 78)

II.3. *Android*

Android merupakan sistem operasi yang berisi *middleware* serta aplikasi-aplikasi dasar. Basis sistem operasi *Android* yaitu kernel *linux* 2.6 yang telah diperbaharui untuk *mobile device*. Pengembangkan aplikasi android menggunakan bahasa pemrograman *Java*. Yang mana konsep-konsep pemrograman *Java* berhubungan dengan Pemrograman Berbasis Objek (OOP)). Selain itu pula dalam pengembangan aplikasi *Android* membutuhkan *software development kit* (SDK) yang disediakan *Android*, SDK ini memberi jalan bagi programmer untuk mengakses *application programming interface* (API) pada *Android*. (*Arzan Muharom ; 2013 ; 2*)

II.4. Penglihatan Warna

Ada tiga jenis gangguan penglihatan terhadap warna, yaitu :

1. *Monochromacy*

   Monochromacy adalah keadaan dimana seseorang hanya memiliki sebuah sel *pigmencones* atau tidak berfungsinya semua sel *cones*. *Monochromacy* ada dua jenis, yaitu *rod monochromacy* dan *cone monochromacy.*

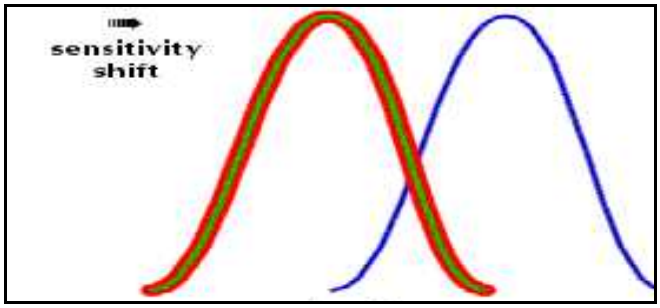
a. *Rod monochromacy* (*typical*) adalah jenis buta warna yang sangat jarang terjadi, yaitu ketidakmampuan dalam membedakan warna sebagai akibat dari tidak berfungsinya semua *cones* retina . Penderita *rod monochromacy* tidak dapat membedakan warna sehingga yang terlihat hanya hitam, putih dan abu-abu.

b. *Cone monochromacy* (*atypical*) adalah tipe *monochromacy* yang sangat jarang terjadi yang disebabkan oleh tidak berfungsinyadua sel *cones*. Penderita *cone monochromacy* masih dapat melihat warna tertentu, karena masih memiliki satu sel *cones* yang berfungsi. (*Rahmadi Kurnia ; 2010 : I-27*)

2. *Dichromacy*

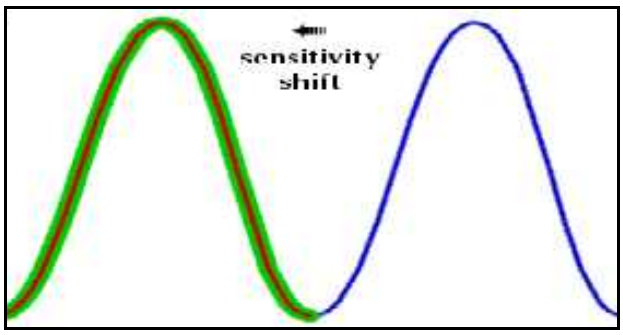
Dichromacy adalah jenis buta warna dimana salah satu dari tiga sel *cone* tidak ada atau tidak berfungsi. Akibat dari disfungsi salah satu sel pigmen pada *cone*, seseorang yang menderita dikromatis akan mengalami gangguan penglihatan terhadap warna-warna tertentu. *Dichromacy* dibagi menjadi tiga bagian berdasarkan sel pigmen yang rusak. (*Rahmadi Kurnia ; 2010 : I-27*)

a. *Protanopia* adalah salah satu tipe *dichromacy* yang disebabkan oleh tidak adanya *photoreseptor* retina merah . Pada penderita *protanopia*, penglihatan terhadap warna merah tidak ada. *Dichromacy* tipe ini terjadi pada 1% dari seluruh pria. *Protanopia* juga dikenal dengan buta warna merah-hijau seperti terlihat pada gambar II.1. berikut :



Gambar II.1. Perubahan sensitivitas panjang gelombang warna merah
Sumber : Rahmadi Kurnia ; 2010 : I-27

b. *Deutanopia* adalah gangguan penglihatan terhadap warna yang disebabkan tidak adanya *photoreseptor* retina hijau . Hal ini menimbulkan kesulitan dalam membedakan *hue* pada warna merah dan hijau (*red-green hue discrimination*). Seperti terlihat pada gambar II.2. berikut :



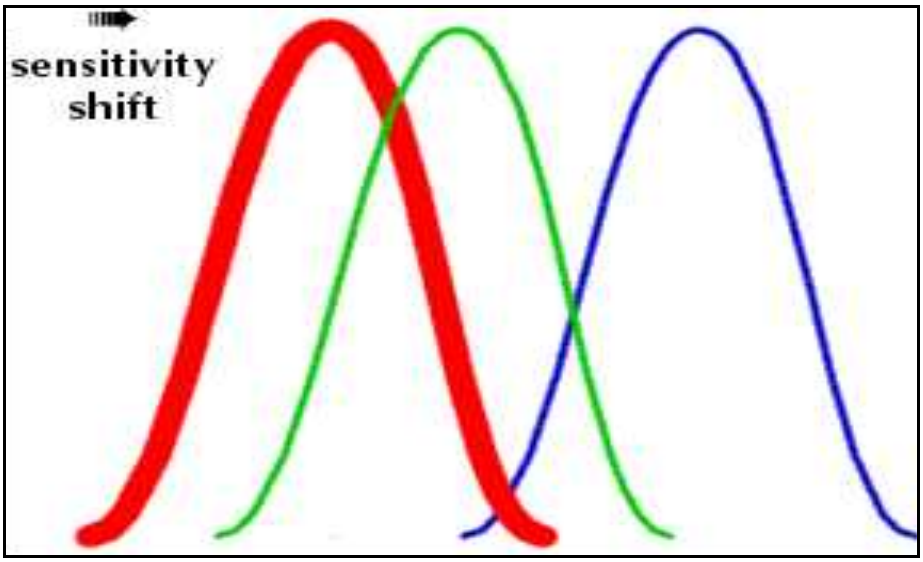
Gambar II.2. Perubahan sensitivitas panjang gelombang warna hijau
Sumber : Rahmadi Kurnia ; 2010 : I-27

c. *Tritanopia* adalah keadaan dimana seseorang tidak memiliki *short-wavelength cone*. Seseorang yang menderita *tritanopia* akan kesulitan dalam membedakan warna biru dan kuning dari spektrum cahaya tampak. *Tritanopia* disebut juga buta warna biru-kuning dan merupakan tipe *dichromacy* yang sangat jarang dijumpai. (*Rahmadi Kurnia ; 2010 : I-27*)

3. *Anomalous trichromacy*

Anomalous trichromacy adalah gangguan penglihatan warna yang dapat disebabkan oleh faktor keturunan atau kerusakan pada mata setelah dewasa. Penderita *anomalous trichromacy* memiliki tiga sel *cones* yang lengkap, namun terjadi kerusakan mekanisme sensitivitas terhadap salah satu dari tiga sel reseptor warna tersebut. (*Rahmadi Kurnia ; 2010 : I-27*)

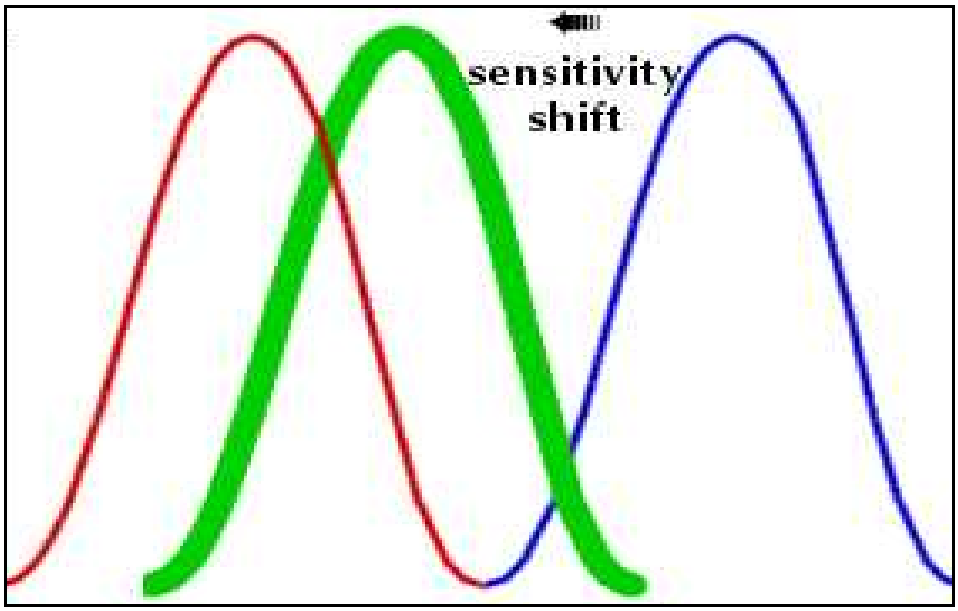
a. *Protanomaly* adalah tipe *anomalous trichromacy* dimana terjadi kelainan terhadap *long-wavelength* (*red*) *pigment*, sehingga menyebabkan rendahnya sensitifitas terhadap cahaya merah. Artinya penderita *protanomaly* tidak akan mampu membedakan warna dan melihat campuran warna yang dapat dilihat oleh mata normal, berikut terlihat pada gambar II.3



Gambar II.3. Pergeseran panjang gelombang warna merah
Sumber : Rahmadi Kurnia ; 2010 : I-27

Penderita juga akan mengalami penglihatan yang buram terhadap warna spektrum merah. Hal ini mengakibatkan mereka dapat salah membedakan warna merah dan hitam.

b. *Deuteranomaly* disebabkan oleh kelainan pada bentuk pigmen *middle-wavelength* (*green*).*deuteranomaly* tidak mampu melihat perbedaan kecil pada nilai *hue* dalam area spektrum untuk warna merah, orange, kuning, dan hijau. Penderita salah dalam menafsirkan *hue* dalam region warna tersebut karena *hue*-nya lebih mendekatiwarna merah.. Perbedaan antara keduanya yaitu penderita *deuteranomaly* tidak memiliki masalah dalam hilangnya penglihatan terhadap kecerahan (*brigthness*). Seperti terlihat pada gambar 4. (*Rahmadi Kurnia ; 2010 : I-27*)



Gambar II.4. Pergeseran panjang gelombang warna Hijau
Sumber : Rahmadi Kurnia ; 2010 : I-27

c. *Tritanomaly* adalah tipe *anomolous trichromacy* yang sangat jarang terjadi, baik pada pria maupun wanita. Pada *tritanomaly*, kelainan terdapat pada *shortwavelength pigment* (*blue*). Pigmen biru ini bergeser ke area hijau dari spektrum warna. Tidak seperti *protanomaly* dan *deuteranomaly*, *tritanomaly* diwariskan oleh kromosom 7. Inilah alasan mengapa penderita *tritanomaly* sangat jarang ditemui. (*Rahmadi Kurnia ; 2010 : I-27*)

II.5. UML *(Unified Modelling Language)*

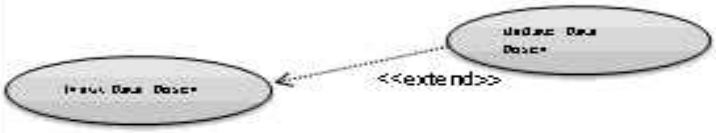
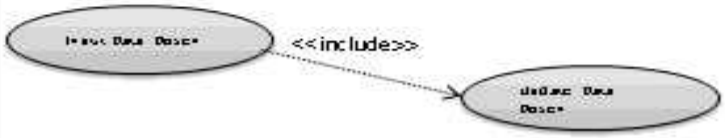
Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented*

Design), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson,* yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. *(Yuni Sugiarti ; 2013 : 33)*

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterangkan dibawah ini.

1. *Use Case Diagram*

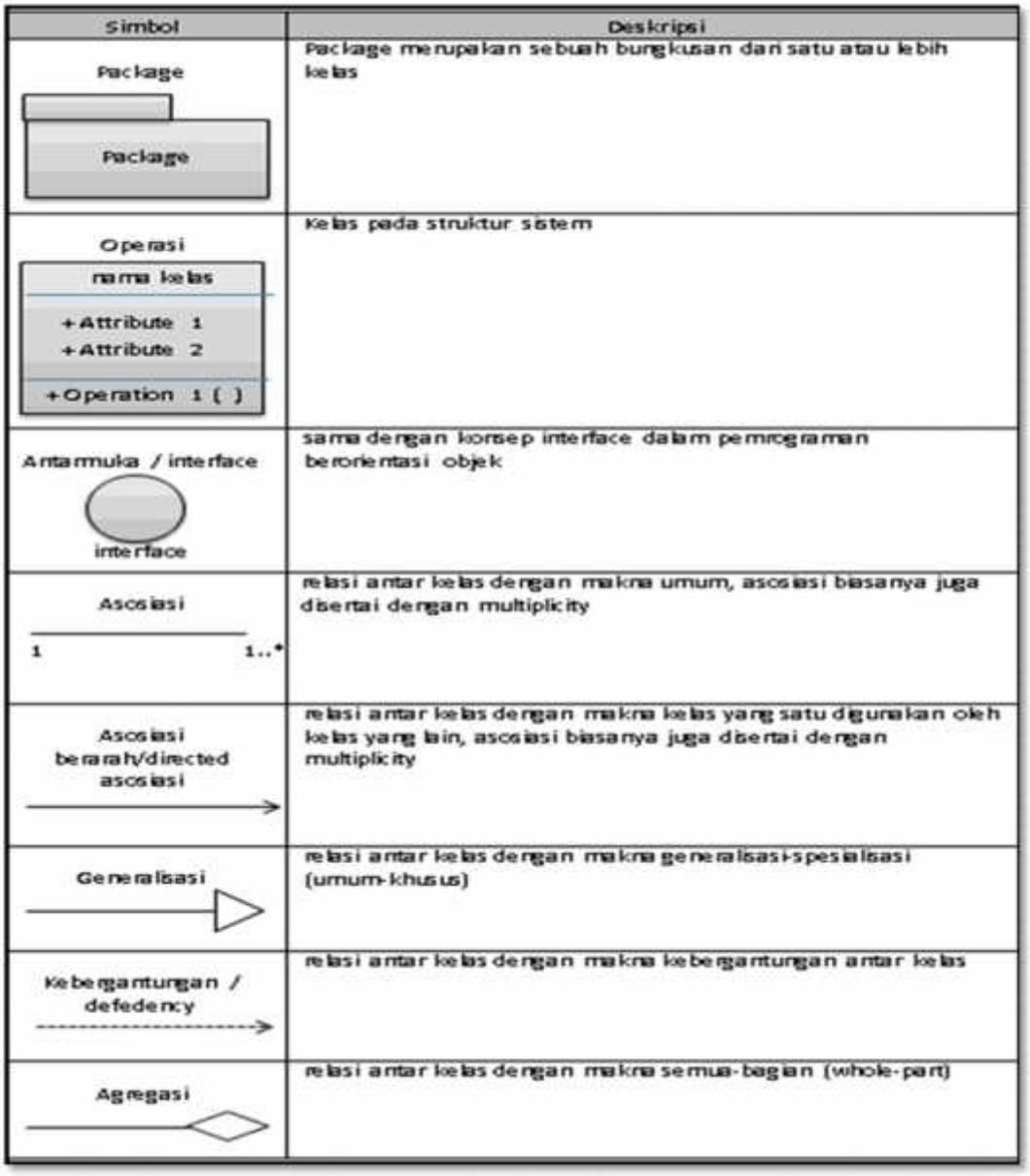
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. 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 *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; 2013 : 41)*

Simbol	Deskripsi
Use Case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case
Aktor  nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
Extend  <<extend>>	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjukan pada use case yang dituju contoh : 
Include <<include>> 	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

Gambar II.5. *Use Case* Diagram
Sumber : (Yuni Sugiarti ; 2013 ; 42)

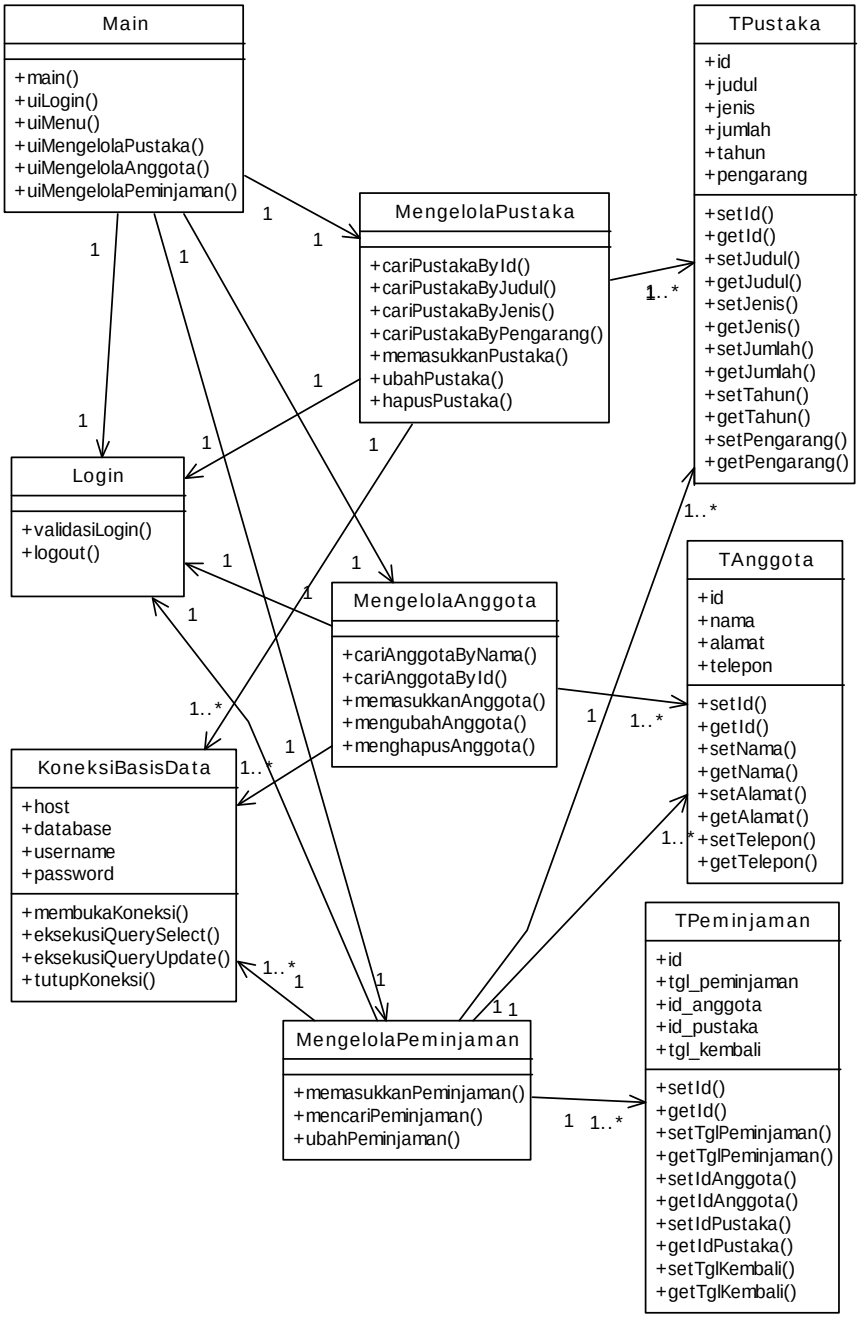
2. *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :



Simbol	Deskripsi
Package Package	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi nama kelas +Attribute 1 +Attribute 2 +Operation 1 ( )	Kelas pada struktur sistem
Antarmuka / interface interface	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 1 1..*	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / defedency	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi	relasi antar kelas dengan makna semua-bagian (whole-part)

Gambar II.6. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)



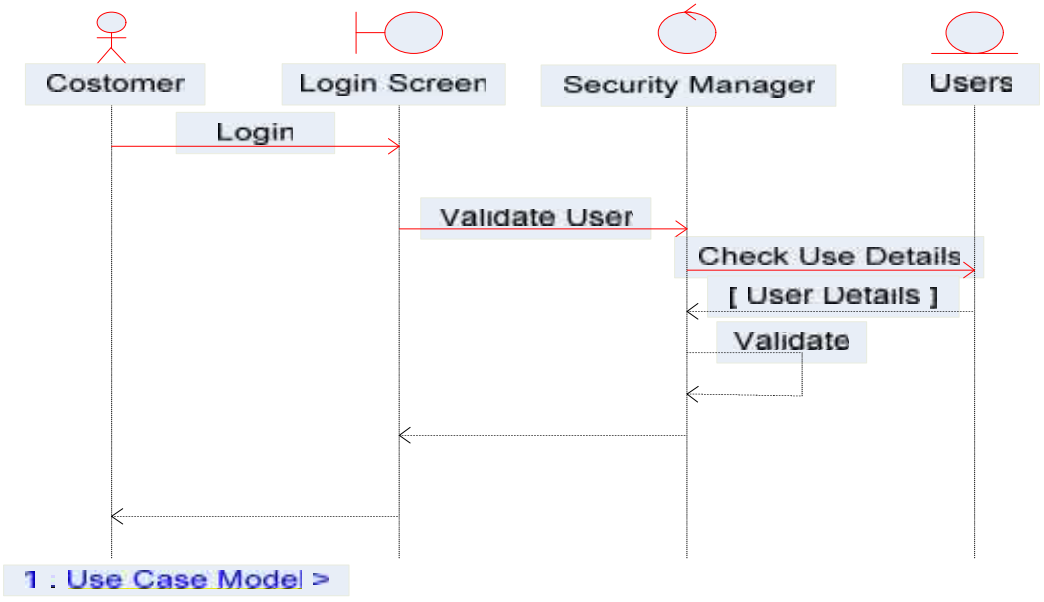
Gambar II.7. Contoh *Class* Diagram

Sumber : (Yuni Sugiarti ; 2013 : 63)

3. *Sequence Diagram*

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

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

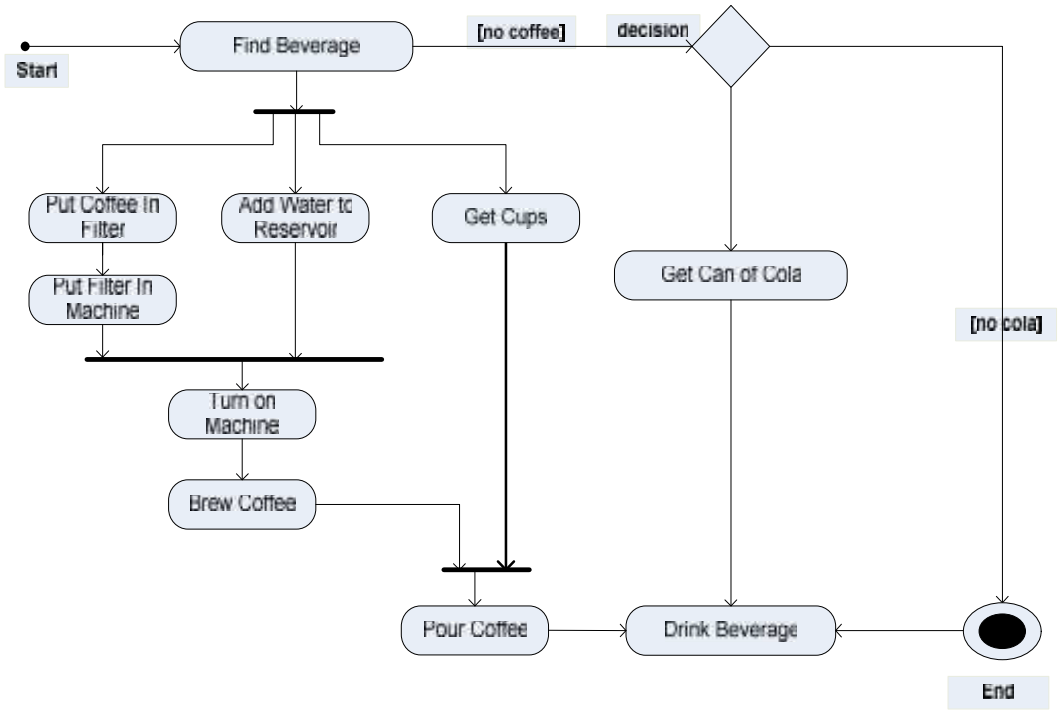


Gambar II.8. Contoh *Sequence* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. *(Yuni Sugiarti ; 2013 : 76*



Gambar II.9. *Activity* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)

II.6. *Android* SDK

Android SDK (*Software Development Kit*) *Android* SDK adalah *tools* API (*Aplication Programming Interface*) yang diperlukan untuk mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java*. Beberapa fitur-fitur *Android* yang paling penting adalah mesin *Virtual Dalvik* yang dioptimalkan untuk perangkat *mobile*, *integrated browser* berdasarkan e*ngine open source WebKit*, Grafis yang dioptimalkan dan didukung oleh libraries grafis 2D, grafis 3D berdasarkan spesifikasi *opengl* ES 1.0 (Opsional akselerasi perangkat keras), kemudian *SQLite* untuk penyimpanan data (*database*). Fitur-fitur android lainnya termasuk media yang mendukung *audio*, *video*, dan gambar, juga ada fitur *bluetooth*, EDGE, 3G dan *WiFi*, dengan fitur kamera, GPS, dan

kompas. Selanjutnya fitur yang juga turut disediakan adalah lingkungan *Development* yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk debugging, profil dan kinerja memori, dan *plugin* untuk IDE *Eclipse*.(*Alicia Sinsuw ; 2013 ; 2*)

II.7. *Java*

Java merupakan bahasa pemrograman berorientasi objek daan bebas *platform*, dikembangkan oleh *SUN Micro System* dengan jumlah keunggulan yang memungkinkan *java* dijadikan sebagai bahasa pengembang *entreprise. Java* merupakan bahasa yang *powerfull* yang bisa digunakan dalam hampir semua bentuk pengembangan *software*. Anda dapat menggunakan java untuk membuat game, aplikasi *desktop*, aplikasi *web*, aplikasi *enterprise*, aplikasi jaringan, dan lain-lain. Yang menarik adalah bahwa java bias digunakan untuk membuat laporan yang dapat berjalan di atas HP, PDA, dan peralatan lain yang dilengkapi dengan *Java Virtual Machine*(JVM). (*Atik Rusmayanti ; 2014 ; 2*)