BAB II

TINJAUAN PUSTAKA

II.1. Jaringan Komputer

Jaringan komputer adalah himpunan interkoneksi sejumlah komputer *autonomous*. Dua buah komputer dikatakan interkoneksi apabila keduanya bisa berbagi *resources* yang dimiliki, seperti saling bertukar data/informasi, berbagi *printer*, media penyimpanan (*hard disk, floppy disk, CD ROM, flash disk*). Data berupa teks, *audio* maupun *video* mengalir melalui media jaringan, baik kabel maupun gelombang elektronik sehingga memungkinkan pengguna jaringan komputer bertukar *file*/data atau *printer* yang sama, menggunakan *hardware*/*software* yang terhubung dalam jaringan. Jadi jaringan komputer dapat dikatakan sebagai kumpulan beberapa buah komputer yang terhubung satu sama lain dan dapat saling berbagi *resources*. (*Sulistiyo, IJSN,* 2012, 2)

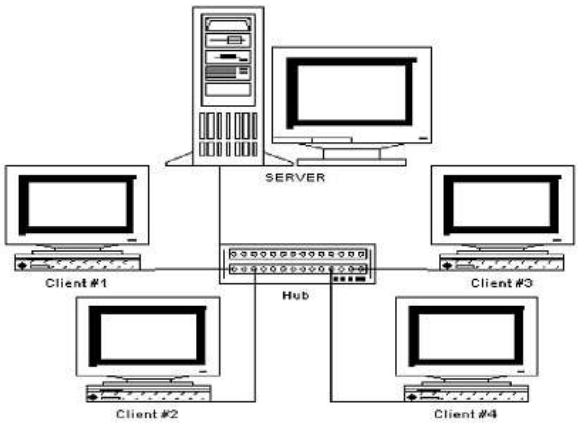
Jaringan komputer adalah sebuah kumpulan komputer, *prinnter*, dan peralatan lainnya yang saling terhubung. Informasi dan data bergerak melalui kabel-kabel sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data. *(Edy Victor Haryanto ; 2012 : 12)*

Jaringan komputer adalah komunikasi data antar komputer, yaitu minimal 2 komputer. Jaringan komputer dapat dilakukan melalui media kabel ataupun nirkabel (wireless). Pada sistem pemesanan makanan ini, jaringan komputer dilakukan melalui media kabel antara 2 komputer. Interface yang digunakan adalah DB-25 atau port paralel. Data yang dikirimkan antar

komputer adalah berupa kode biner 1 dan 0, yang dirancang pada masing-masing program yaitu pada program *server* dan program client. Ada tiga tipe jaringan yang umum yang digunakan antara lain : Jaringan *Workgroup*, Jaringan LAN dan Jaringan WAN.

II.1.1. Jaringan LAN

LAN (*Local Area Network*) adalah suatu kumpulan komputer, dimana terdapat beberapa unit komputer (client) dan 1 unit komputer untuk bank data (*server*). Antara masing-masing *client* maupun antara *client* dan *server* dapat saling bertukar *file* maupun saling menggunakan printer yang terhubung pada unitunit komputer yang terhubung pada jaringan LAN. (*Ruri Hartika Zain , Jurnal Momentum, 2012, Vol.13 No.2)*



Gambar II.1. Jarngan LAN
Sumber : Ruri Hartika Zain (*Jurnal Momentum, 2012, Vol.13 No.2)*

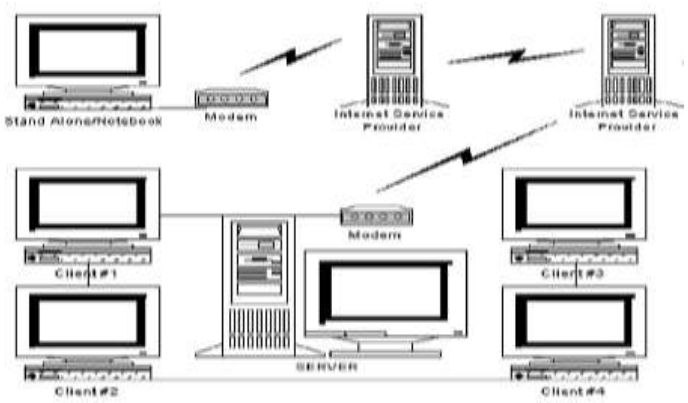
II.1.2. Jaringan MAN

Metropolitan Area Network atau MAN, merupakan Jenis Jaringan Komputer yang lebih luas dan lebih canggih dari Jenis Jaringan Komputer

LAN. Disebut *Metropolitan Area Network* karena Jenis Jaringan Komputer MAN ini biasa digunakan untuk menghubungkan jaringan komputer dari suatu kota ke kota lainnya. Untuk dapat membuat suatu jaringan MAN, biasanya diperlukan adanya operator telekomunikasi untuk menghubungkan antar jaringan komputer. (*Ruri Hartika Zain , Jurnal Momentum, 2012, Vol.13 No.2)*

II.1.3. Jaringan WAN

WAN (*Wide Area Network*) adalah kumpulan dari LAN dan/atau *Workgroup* yang dihubungkan dengan menggunakan alat komunikasi modem dan jaringan Internet, dari/ke kantor pusat dan kantor cabang, maupun antar kantor cabang. Dengan sistem jaringan ini, pertukaran data antar kantor dapat dilakukan dengan cepat serta dengan biaya yang relatif murah. Sistem jaringan ini dapat menggunakan jaringan Internet yang sudah ada, untuk menghubungkan antara kantor pusat dan kantor cabang atau dengan PC *Stand Alone*/*Notebook* yang berada di lain kota ataupun negara.



Gambar II.2. Jarngan WAN
Sumber : Ruri Hartika Zain (*Jurnal Momentum, 2012, Vol.13 No.2)*

II.1.4. Konsep *Client Server*

Secara umum *server* adalah komputer yang berisi program baik sistem operasi maupun program aplikasi yang menyediakan layanan kepada komputer atau program lain yang sama ataupun berbeda. Jenis *server* sangat banyak, salah satunya adalah Terminal *Server*, dimana terminal *server* bertindak sebagai sebuah *multiplexer* yang memungkinkan sejumlah komputer kecil, atau terminal-terminal yang lain mengakses ke sebuah titik LAN yang sama. Terminal *server* dapat digunakan untuk menyediakan akses ke komputer pusat untuk sejumlah terminal dengan menggunakan biaya yang rendah. (*Sulistiyo, IJSN,* 2012, 2)

Keunggulan tipe jaringan *Client Server* adalah sebagai berikut :

1. Terdapat administrator jaringan yang mengelola sistem keamanan dan administrasi jaringan, sehingga sistem keamanan dan administrasi jaringan akan lebih terkontrol.
2. Komputer *server* difungsikan sebagai pusat data, komputer klien dapat mengakses data yang ada dari komputer klien manapun. Apabila terdapat komputer klien yang rusak, pengguna masih dapat mengakses data dari komputer klien yang lain.
3. Pengaksesan data lebih tinggi karena penyediaan dan pengelolaaan fasilitas jaringan dilakukan oleh komputer *server*. Dan komputer *server* tidak terbebani dengan tugas lain sebagai *workstation*.

4. Pada tipe jaringan *Client Server*, sistem *backup* data lebih baik, karena backup data dapat dilakukan terbpusat di komputer *server*. Apabila data pada komputer klien/*workstation* mengalami masalah atau kerusakan masih tersedia *backup* pada komputer *server.*

Kelemahan tipe jaringan *Client Server* adalah sebagai berikut :

1. Biaya mahal, karena membutuhkan komputer yang memiliki kemampuan yang difungsikan sebagai komputer *server*.
2. Kelancaran jaringan tergantung pada komputer *server*. Bila komputer *server* mengalami gangguan maka maka jaringan akan terganggu.

II.1.5. Jaringan *Wi-Fi*

Jaringan *Wireless* atau jaringan nirkabel yang sering disebut WLAN/*Wireless* LAN merupakan jaringan LAN biasa yang tidak menggunakan kabel untuk transfer datanya. Media untuk transfer data adalah gelombang radio. Jaringan ini cakupannya bervariasi, bisa sekedar mencakup satu area kecil, ataupun area yang sangat luas.

Pada umumnya jaringan *wireless* ini dikombinasikan juga dengan jaringan kabel. Terutama untuk *backbone* biasanya tetap menggunakan kabel. Selain itu untuk menghubungkan satu jaringan *wireless* ke jaringan *wireless* lain, juga digunakan jaringan kabel. Ini karena jaringan kabel lebih stabil dalam transfer datanya. *(Edy Winarno ST, M.Eng, Ali Zaki ; 2013 : 105)*

II.1.6. Jaringan *Ad-Hoc*

Di jaringan komputer *wireless*, mode *ad-hoc* adalah metode yang mengungkinkan piranti *wireless* yang berada di jangkauan sinyal untuk mencari dan berkomunikasi secara *peer to peer* tanpa melibatkan *access point*. Untuk bisa menerapkan *ad-hoc*, tiap adaptor *wireless* harus diset ke mode *ad-hoc* dan bukannya di mode infrastruktur. Selain itu semua adaptor *wireless* (*Wifi*) harus diset menggunakan satu nama SSID dan *channel* yang sama. *(Edy Winarno ST, M.Eng, Ali Zaki ; 2013 : 114)*

II.2. Komunikasi Data

Kata komunikasi merupakan suatu kata yang dapat diartikan sebagai cara untuk menyampaikan atau menyebarluaskan data dan informasi, sedangkan kata informasi berarti berita, pikiran, pendapat dalam berbagai bentuk. Manusia dapat melakukan komunikasi dengan berbagai cara, berbicara secara langsung, berbisik, mengirim surat dan lain sebagainya.

Dari berbagai cara komunikasi manusia ini masih terdapat banyak kekurangan dan kelemahan yaitu :

1. Jarak yang jauh (bahkan sampai menyeberangi lautan),
2. Waktu yang lama untuk menyampaikan pesan.
3. Biaya yang relatif mahal.

Kekurangan tersebut bisa diatasi seiring dengan perkembangan teknologi informasi. Dengan teknologi komunikasi sekarang ini, hanya dengan menekan beberapa tombol maka jarak dan waktu untuk melakukan komunikasi tidak lagi menjadi kendala yang berarti.

Teknologi komunikasi terus dikembangkan dengan tujuan memudahkan manusia dalam melakukan komunikasi. Para ahli terdorong untuk mengembangkan teknik komunikasi jarak jauh yang lebih efisien dengan metode telekomunikasi yang memanfaatkan teknologi elektronika, yang dikenal dengan istilah teknik komunikasi data.

Komunikasi data merupakan cara mengirimkan data menggunakan sistem transmisi elektronik dari satu komputer ke komputer lain atau dari satu komputer ke terminal tertentu. Sedangkan data itu sendiri merupakan sinyal elektromagnetik yang dibangkitkan oleh sumber data yang ditangkap dan dikirimkan ke terminal penerima. (*Ariyus ; 2008 ; 3*)

II.2.1. Tujuan Komunikasi Data

Tujuan komunikasi data adalah mengirim data dalam jumlah besar dan waktu yang singkat dengan cara yang efisien dan ekonomis dari suatu tempat ke tempat lain tanpa ada kesalahan. (*Edy Victor Hariyanto;2012:2*)

Komunikasi data memiliki keuntungan yang dapat diterapkan pada beberapa perangkat komputer, yaitu :

1. Memungkinkan penggunaan komputer atau terminal secara terpusat (sentralisasi) ataupun tersebar (desentralisasi) sehingga mendukung menajemen dalam hal kontrol.
2. Mempermudah kemungkinan pengolahan dan pengaturan data yang ada dalam berbagai macam sistem komputer.
3. Mengurangi waktu untuk pengolahan data.

4. Mendapatkan data langsung dari sumbernya (mempertinggi kehandalan).
5. Mempercepat penyebaran informasi.

II.3. Pembahasan Umum Java

Java menurut definisi dari *Sun* adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. *Java2* adalah generasi kedua dari *Java platform* (generasi awalnya adalah *Java Development Kit*). *Java* berdiri atas sebuah mesin *interpreter* yang diberi nama *Java Virtual Machine* (JVM). JVM inilah yang akan membaca *bytecode* dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu, bahasa *Java* disebut sebagai bahasa pemrograman *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM. (*Shalahuddin 2008 : 01*)

II.4. *Android*

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka.

Android merupakan generasi baru *platform mobile*, *platform* yang memberikan pengembang untuk melakukan pengembangan sesuai dengan yang diharapkannya. Sistem operasi yang mendasari *Android* dilisensikan dibawah GNU, *General Public Lisensi* Versi 2 (GPLv2), yang sering dikenal dengan istilah

"*Copyleft"* lisensi dimana setiap perbaikan pihak ketiga harus terus jatuh dibawah *terms. Android* didistribusikan dibawah lisensi *Apache Software* (ASL/*Apache2*), yang memungkinkan untuk distribusi kedua dan seterusnya. Komersialisasi pengembang (produsen *handset* khususnya dapat memilih untuk meningkatkan *platform* tanpa harus memberikan perbaikan mereka ke masyarakat *open source*. Sebailknya pengembang dapat keuntungan dari perangkat tambahan seperti perbaikan dan mendistribusikan ulang pekerjaan mereka di bawah lisensi apapun yang mereka inginkan. Pengembang aplikasi *Android* diperbolehkan untuk mendistribusikan aplikai mereka dibawah skema lisensi apapun yang mereka inginkan. (*Ichwan, Jurnal Informatika, 2013, No. 1, Vol. 4*)

II.5. *Eclipse*

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform-independent*). *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 populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*. (*Mario, Transient, 2013, Vol. 2, No. 1*)

II.6. 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 dirilis *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.

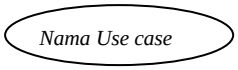
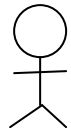
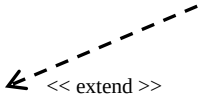
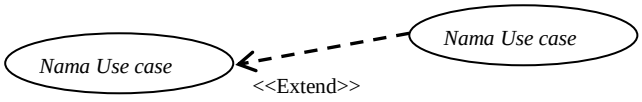
Dalam pembuatan skripsi ini penulis menggunakan diagram *Use case* yang terdapat di dalam UML. Adapun maksud dari *Use case* Diagram diterangkan dibawah ini.

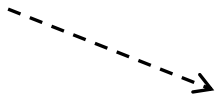
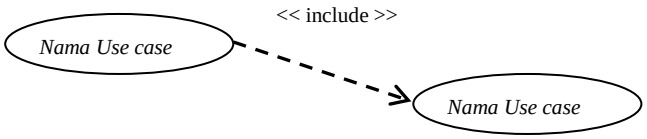
1. *Use case Diagram*

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 ; S.T, M.Kom ; 2013 : 41)*

Tabel II.1. Simbol *Use case*

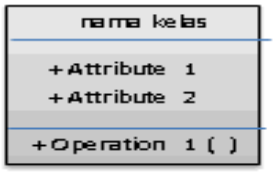
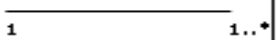
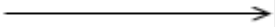
Simbol	Deskripsi
Use case 	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor, biasanya menggunakan kata kerja di awal frase nama *use case*
Aktor  *Nama Aktor*	Orang atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sistem itu sendiri. Jadi walaupun simbol aktor adalah gambar orang, tapi aktor belum tentu merupakan orang. Biasanya menggunakan kata benda di awal frase nama aktor
Asosiasi / Association	Komunikasi antar aktor dan *use case* yang berpartisipasi *use case* atau *use case* berinteraksi dengan aktor.
Extend 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walupun tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambhan 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; 

Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 42)

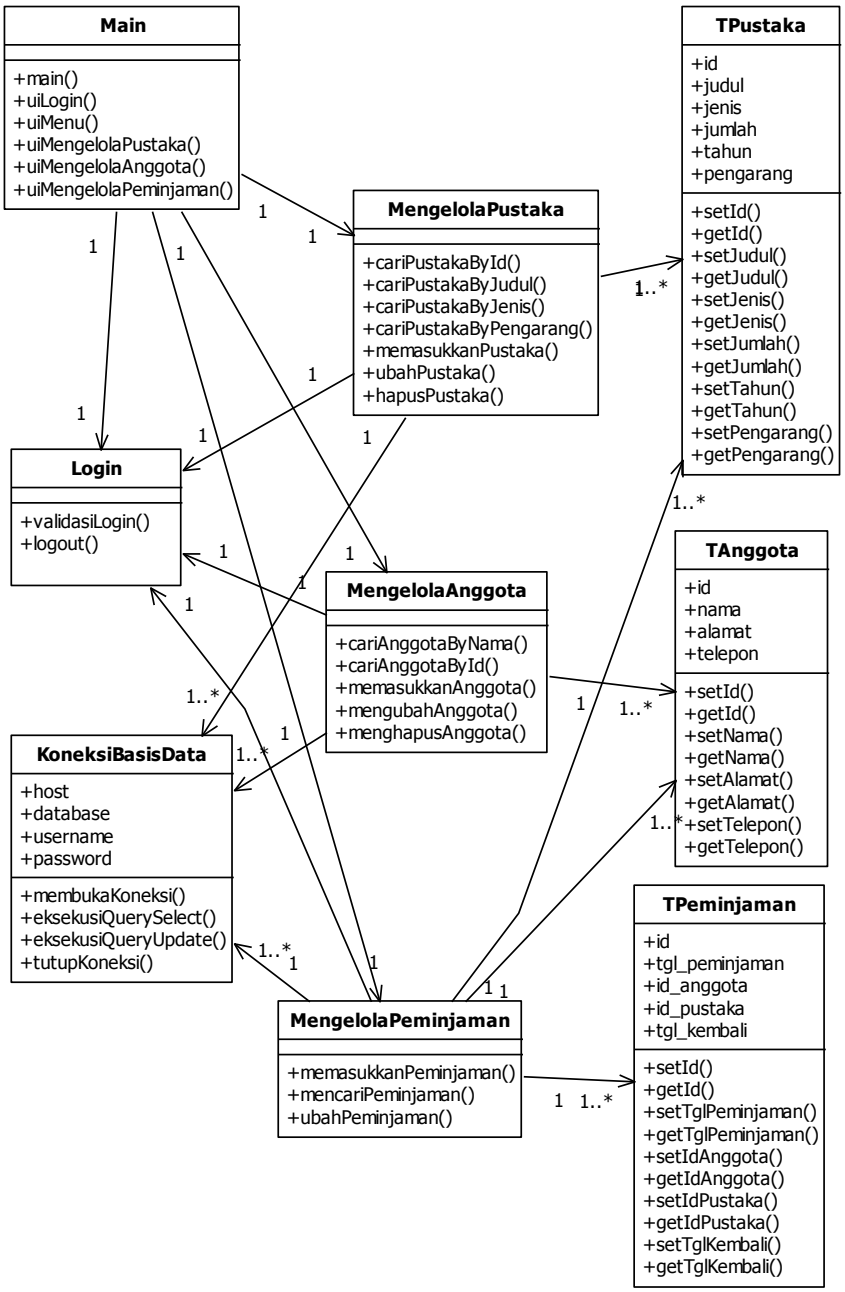
2. *Class Diagram*

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 merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface  interface	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 	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.3. *Class* Diagram

Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 59)



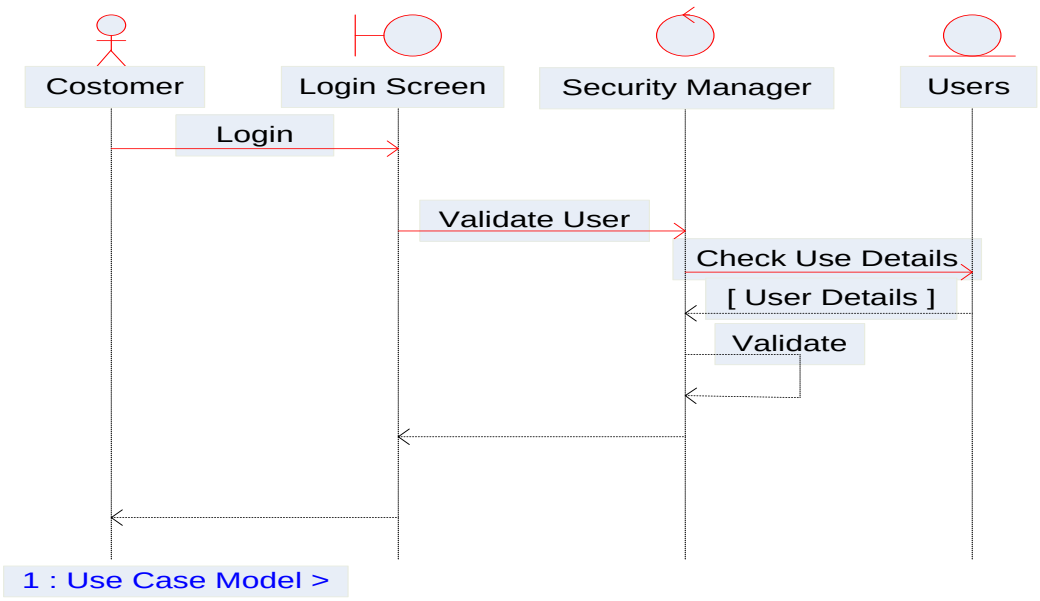
Gambar II.4. Contoh Class Diagram

Sumber : Yuni Sugiarti, S.T, M.Kom (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.5. Contoh *Sequence* Diagram
Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 63)

4. *Activity Diagram*

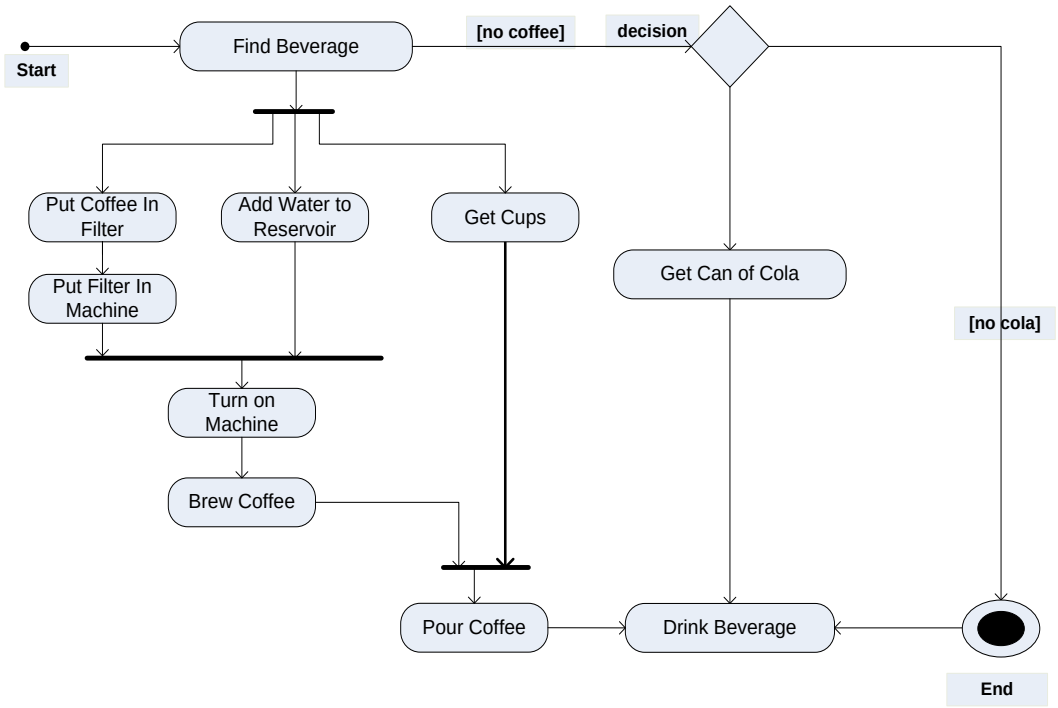
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.



Gambar II.6. *Activity* Diagram

Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 76)