BAB II
LANDASAN TEORI

2.1 Teori Umum

2.1.1 Komputer

Sutarman (2012:3), "Komputer Merupakan suatu rangkaian peralatan elektronik yang bekerja secara bersama-sama. Komputer dapat melakukan rangkaian pekerjaan secara otomatis melalui instruksi (program) yang diberikan dan alat pengolahan data menjadi informasi melalui proses tertentu".

2.1.2 Sistem

Yakub (2012:3), sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, terkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk tujuan tertentu.

Hartono (2013:10), sistem yakni suatu benda atau entitas yaitu himpunan dari berbagai bagian atau komponen, dan sekaligus juga suatu proses atau metode atau cara untuk mencapai tujuan yaitu saling berhubungan secara terorganisasi berdasar fungsi-fungsinya.

Ladjamudin (2013:1), sistem adalah suatu urutan kegiatan yang saling berhubungan, berkumpulan bersama-sama untuk mencapai suatu tujuan tertentu.

Sutarman (2012:86), sistem adalah kumpulan dari bagian-bagian (subsistem) yang terkait menjadi satu bentuk mekanisme kerja yang memberikan fungsi dan manfaat tertentu.

Kristanto (2008:1), "*System* adalah jaringan kerja dari prosedur – prosedur yang saling berhubungan, berkumpul bersama – sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu".

2.1.3 Karakteristik Sistem

Al Fatta (2007:5), berikut adalah karakteristik sistem yang dapat membedakan suatu sistem dengan sistem lainnya:

a. Batasan (*boundary*) : Pengembangan dari suatu elemen atau unsur mana yang termasuk dalam sistem dan mana yang diluar sistem.

b. Lingkungan (*environment*) : Segala sesuatu diluar sistem, lingkungan yang menyediakan asumsi, kendala, dan input terhadap suatu sistem.
c. Masukan (*input*) : Sumber daya (data, bahan baku, peralatan, energi) dari lingkungan yang dikonsumsi dan dimanipulasi oleh suatu sistem.
d. Keluaran (*output*) : Sumber daya atau produk (informasi, laporan, dokumen, tampilan layar komputer, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.
e. Komponen (*component*) : Kegiatan – kegiatan atau proses dalam suatu sistem yang mentransformasikan input menjadi bentuk setengah jadi (output). Komponen ini bisa merupakan subsistem dari sebuah sistem.
f. Penghubung (*interface*) : Tempat dimana komponen atau sistem dan lingkungannya bertemu dan berinteraksi.
g. Penyimpanan (*storage*) : Area yang dikuasai dan digunakan untuk penyimpanan sementara dan tetap dari informasi, energi, bahan baku, dan sebagainya. Penyimpanan merupakan suatu media penyangga diantara komponen tersebut bekerja dengan berbagai tingkatan yang ada dan memungkinkan komponen yang berbeda dari data yang sama.

2.1.4 Klasifikasi Sistem

Kristanto (2008:5), dari berbagai sudut pandang sistem dapat diklasifikasikan menjadi beberapa bagian yaitu:

a. Sistem Abstak dan Sistem Fisik

   Sistem abstrak merupakan sistem yang tidak bisa dilihat secara mata biasa dan biasanya sistem ini berupa pemikiran atau ide-ide. Contoh dari sistem abstrak ini adalah filsafat. Sistem fisik merupakan sistem yang bisa dilihat secara mata biasa dan biasanya sering digunakan oleh manusia. Contoh dari sistem fisik ini adalah sistem akuntansi, sistem komputer dan sebagainya.

b. Sistem Alamiah dan Sistem Buatan

   Sistem alamiah merupakan sistem yang terjadi karena pengaruh alam. Misalnya sistem rotasi bumi, sistem gravitasi dan sebagainya. Sistem buatan

merupakan sistem yang dirancang dan dibuat oleh manusia. Misalnya, sistem pengolahan gaji.

c. Sistem Tertutup dan Sistem Terbuka

Sistem tertutup merupakan sistem yang tidak berhubungan dengan bagian luar sistem dan biasanya tidak terpengaruh oleh kondisi di luar sistem. Sedangkan sistem terbuka merupakan sistem yang berhubungan dengan bagian luar sistem.

2.1.5 Tahap Pengembangan Sistem

Kristanto (2008:41), "Siklus pengembangan sistem adalah kumpulan-kumpulan kegiatan dari analisis pendesain dan *user* dari sistem informasi yang dilaksanakan untuk dikembangkan dan diimplementasikan". Siklus pengembangan sistem terdiri dari aktivitas-aktivitas:

1. Penyelidikan Awal dan Studi Kelayakan

Sebelum tahapan ini dilakukan yang perlu diketahui dan dipertimbangkan adalah alasan timbulnya gagasan untuk membuat sistem informasi yang baru.

a. Penyelidikan Awal

Pada tahap penyelidikan awal, analisis belajar dari pemakai mengenai apa yang diharapkan dari sebuah sistem informasi yang baru. Hal-hal yang perlu diperhatikan dalam tahapan ini adalah:

i. Mencoba memahami dan menjelaskan apa yang diharapkan oleh pemakai (sistem informasi bagaimana yang mereka perlukan).
ii. Menentukan ruang lingkup dari studi sistem informasi.
iii. Menentukan kelayakan dari masing-masing alternatif dengan memperkirakan keuntungan atau kerugian yang didapat.

b. Studi Kelayakan

Studi kalayakan merupakan tahap yang paling penting, karena di dalamnya menyangkut berbagai aspek sistem baru yang diusulkan. Laporan mengenai studi kelayakan harus disampaikan kepada manajemen, yang pada gilirannya akan

memberikan beberapa perubahan, menyarankan untuk diadakan penelitian lebih mendalam atau memutuskan untuk segera dilaksanakan.

2. Penentuan Kebutuhan-kebutuhan Sistem

Kebutuhan-kebutuhan sistem yang akan dikembangkan meliputi masukan, keluaran, operasi, dan *resources*, untuk memenuhi kebutuhan organisasi masa kini dan masa mendatang. Pada tahap penentuan kebutuhan sistem ini dilakukan evaluasi untuk memastikan keadaan sistem yang baru.

Sasaran pertama pada tahap ini adalah mendefenisikan apa yang seharusnya dapat dilakukan oleh sistem baru. Kemudian menentukan kriteria yang dapat digunakan untuk mengevaluasi keandalan sistem yang baru:

a. Pengembangan Kriteria Evaluasi

   Dengan mengembangkan kriteria evaluasi sebelum melakukan pengembangan sistem baru, berarti telah diterapkan suatu metode pengukuran yang valid yang dapat digunakan untuk mengevaluasi keandalan sistem baru. Evaluasi sistem baru merupakan hal yang sangat penting karena sistem tidak akan bisa dipasang (diinstal) tanpa sekumpulan kriteria evaluasi yang valid. Artinya standar keadaan sebaiknya sejalan dengan sasaran-sasaran sistem.

b. Teknik Memperoleh Informasi

   Penentuan teknik pengumpulan data terkait erat dengan jenis instrumen yang digunakan. Tujuan penelitian serta cakupan sample yang akan dijadikan sumber data sangat memperngaruhi pemilihan kita akan jenis instrumen yang paling tepat, serta dengan teknik seperti apa instrumen tersebut akan digunakan.

c. Strategi Penentuan Kebutuhan Sistem

   Konsep aliran data seperti yang telah diberikan pendekatannya oleh Gene & Sarson hanya menggunakan empat buah simbol sehingga sederhana pemakaiannya.

d. Strategi Analisa Keputusan

Analisa keputusan digunakan untuk mempermudah komunikasi antara pemakai dan analis. Ada dua jenis analisa keputusan yang dapat digunakan yaitu tabel keputusan dan pohon keputusan.

3. Desain Sistem

Untuk melakukan perbaikan terhadap sistem informasi, terlebih dahulu harus dipahami dengan jelas kondisi sistem yang ada sekarang dan yang dihadapi, setelah itu sasaran dan kebutuhan sistem di masa yang akan datang. Kemudian baru dapat dimasukkan ide-ide secara bersama-sama ke dalam suatu desain yang akan memenuhi tujuan-tujuan yang telah ditetapkan. Untuk itu dapat digunakan analisa terstruktur dengan diagram-diagram aliran data. Pada proses desain sistem, terdapat proses pemindahan dari apa yang harus dilakukan sistem dan bagaimana sistem nanti akan melakukannya.

a. Desain Pengembangan Model Sistem

Penggunaan teknik-teknik terstruktur melibatkan pengembangan model-model baik untik sistem yang ada maupun sistem yang baru. Terdapat empat buah model dalam hal ini, yaitu:

1. Model fisik dari sistem pada saat itu.
2. Model logik dari sistem pada saat itu.
3. Model fisik dari sistem yang baru.
4. Model logik dari sistem yang baru.

b. Desain *output*

Ada beberapa cara untuk menampilkan hasil keluaran atau desain, yang paling umum adalah *output* berbentuk laporan di media kertas. Selain itu, yang paling banyak digunakan adalah *output* dalam bentuk tabel dan yang berbentuk grafik atau bagan.

c. Desain Kode (Pengkodean)

Kebutuhan untuk melakukan desain kode dilakukan pada saat:

1. Sebuah sistem baru akan mengimplementasikan pada suatu organisasi.
2. Kode yang telah ada tidak mungkin lagi dikembangkan karena strukturnya tidak memungkinkan.

3. Dua atau lebih organisasi dengan sistem kode yang berbeda bergabung menjadi satu, sehingga diperlukan kode tunggal untuk gabungan kedua organisasi tersebut. Suatu kode harus didesain sedemikian rupa sehingga proses identifikasi dan *retrieval* (pengambilan data) dapat berjalan secara efisien.

d. Desain *Input*

Terdapat dua jenis *input* yang ada pada sistem berbasis komputer yaitu:

1. *Batch input* merupakan metode pengumpulan data transaksi tradisional untuk pengolahan data dengan komputer.
2. *On-line input* merupakan pengumpulan data secara langsung dihubungkan dengan komputer.

e. Desain Database *(File)*

Hal yang paling penting pada saat melakukan desain *file* adalah pengetahuan akan struktur dari file yang akan didesain, misalnya *file* pegawai. Sebuah file menyimpan record-record yang jenisnya sama. Satu atau lebih informasi tersebut yang disedut juga sebagai field dapat digunakan sebagai kunci bagi sebuah *record*.

4. Implementasi dan Evaluasi

Untuk melihat apakah desain yang dinyatakan dengan spesifikasi sistem tersebut sesuai dengan kebutuhan pemakai maka manajemen proyek harus dapat memastikannya dan bagaimana proyek dapat dipastikan untuk dapat diselesaikan dengan biaya yang telah diperkirakan serta tepat pada waktunya. Dengan demikian pada tahap implementasi dan evaluasi merupakan tahapan yang paling menyita banyak waktu dan membutuhkan perhatian yang khusus.

2.2 Teori Judul

2.2.1 Aplikasi

Sutabri (2012 : 147), “Program aplikasi atau biasa disebut dengan aplikasi merupakan alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya”.

2.2.2 Pengolahan data

Laksito (2011), "Pengolahan Data adalah manipulasi data agar menjadi bentuk yang lebih berguna. Pengolahan data ini tidak hanya berupa perhitungan numeris tetapi juga operasi-operasi seperti klasifikasi data dan perpindahan data dari satu tempat ke tempat lain. Secara umum, kita asumsikan bahwa operasi-operasi tersebut dilaksanakan oleh beberapa tipe mesin atau komputer, meskipun beberapa diantaranya dapat juga dilakukan secara manual.

2.2.3 Simpan Pinjam Koperasi

Sucitrawan(2012), "Koperasi Simpan Pinjam (KOSIPA) adalah sebuah koperasi yang modalnya diperoleh dari simpanan pokok dan simpanan wajib para anggota koperasi. Kemudian modal yang telah terkumpul tersebut dipinjamkan kepada para anggota koperasi dan terkadang juga dipinjamkan kepada orang lain yang bukan anggota koperasi yang memerlukan pinjaman uang, baik untuk keperluan komsumtif maupun untuk modal kerja. Kepada setiap peminjam, KOSIPA menarik uang administrasi setiap bulan sejumlah sekian persen dari uang pinjaman.

2.2.4 Koperasi

Sucitrawan (2012), "Koperasi merupakan salah satu bentukbadan hukum yang sudah lama terkenal di Indonesia. Pelopor pengembangan perkoperasian di Indonesia adalah Bung Hatta, beliau juga dikenal sebagai bapak koperasi Indonesia. Koperasi merupakan suatu kumpulan dari orang-orang yang mempunyai tujuan atau kepentingan bersama. Jadi koperasi merupakan bentuk dari sekelompok orang yang memiliki tujuan bersama. Kelompok orang inilah yang akan menjadi anggota koperasi berdasarkan asas kekeluargaan dan gotong royong.

2.2.5 Data

Kristanto (2008:8), "Data adalah sesuatu yang nyata atau setengah nyata yang dapat mengurangi derajat ketidakpastian tentang suatu keaadan atau kejadian. Sebagai contoh, informasi yang menyatakan bahwa nilai rupiah akan naik, akan mengurangi ketidakpastian mengenai jadi tidaknya sebuah investasi akan dilakukan".

2.3 Teori Program

2.3.1 JAVA

Rickyanto (2003:2), " Java adalah Suatu teknologi di dunia *software* komputer selain merupakan suatu bahasa pemrograman, java juga merupakan suatu *platform*.

Secara jelasnya java merupakan teknologi dimana teknologi tersebut mencakup java sebagai bahasa pemrograman yang memiliki sintaks dan aturan pemrograman tersendiri, juga mencakup java sebagai platform dimana teknologi ini memiliki virtual machine dan library yang diperlukan untuk menulis dan menjalankan program yang ditulis dengan bahasa pemrograman java.

2.3.2 Kelebihan Java

Rickyanto (2003:3), " java merupakan suatu teknoligi yang unik dan revolusioner dan merupakan teknologi pertama di dunia software yang memiliki semboyan "*write once,run anywhere*". Semboyan tersebut telah terbukti karena banyak program java dapat dijalankan di berbagai platform system operasi, seperti Linux, Windows maupun Unix.

Java telah mengatasi masalah portabilitas yang sering menjadi kendala dan hambatan dalam pembuatan suatu aplikasi software. Mengapa? Karena *software developer* harus mengeluarkan banyak tenaga, pikiran dan waktu untuk menghasilkan aplikasi yang dapat berjalan di operating system atau *platform* lain.

Java juga didesain untuk menghasilkan program dengan seminimal mungkin bug karena kemampuan sebagai berikut:

1. Java di desain untuk menghilang alokasi memori dan dealokasi memori secara manual. Java memiliki *garbage collection* otomatis yang mencegah adanya *memory leak*. *Memory leak* adalah masalah yang sering dihadapi programmer C dan C++ dimana memori yang digunakan untuk objek atau variable yang sudah tidak digunakan tidak didealokasikan sehingga menyebabkan kehabisan memori karena proses alokasi maupun dealokasi yang tidak diatur dengan baik.
2. Java memiliki *array* yang tidak memerlukan pointer sehingga memudahkan para programmer.
3. Java menghilangkan banyak kebingungan apabila terjadi proses assignment (pemberian nial) pada statemen kondisional seperti berikut:

   If [varnya=5]

   Kode diatas menyebabkan program java tidak dapat dikompilasi karena java membedakan tanda = yang digunakan untuk pemberian nilai dan untuk pengecekan kondisi *true* atau *false* yang harus menggunakan tanda = ganda (= =).
4. Java menghilangkan multiple inheritance pada C++ dan menggunakan *interface* yang memiliki kemampuan yang sama tetapi lebih sederhana.

2.3.3 Tipe Data pada *JAVA*

Rickyanto (2003:37),"Dalam pemrograman sudah pasti kita berurusan dengan data. Oleh sebab itu anda perlu mengerti bagaimana menangani data. Sebagaimana bahasa pemrograman pada umumnya, kita mengenal adanya variable yang digunakan untuk menyimpan nilai atau data. Pada bab ini akan dibahas apa saja tipe data yang dapat digunakan dan bagaimana mendeklarasikan variable.

Java dikenal sebagai bahasa pemrograman dengan sifat *strongly typed* di mana artinya adalah anda harus mendeklarasikan tipe data dari semua variable,

dan apabila anda lupa atau salah mengkuti aturan pendeklarasian variable maka anda akan mendapatkan error pada saat proses kompilasi

Java memiliki tipe data yang dapat dikategorikan menjadi dua, yaitu tipe data primitive dan referensi:

2.3.3.1 Tipe Data Primitif

Ada delapan macam tipe data primitive dalam pemrograman Java, Yaitu:

1. Tipe Data *bolean*

   Tipe data Boolean adalah tipe data paling sederhana, yakni untuk menyatakan suatu nilai kebenaran *TRUE* atau *FALSE* .

2. Tipe Data *Integer*

   Tipe data *integer* adalah tipe data numerik yang kita gunakan apabila tidak berurusan dengan pecahan atau bilangan desimal. Anda dapat melihat tipe data numerik yang termasuk integer pada tabel dibawah ini:

Tabel 2.1
Tipe data Integer

Tipe Data	Deskripsi
Byte	Memiliki nilai integer dari -128 sampai +127 dan menempati 1 byte (8 bits) di memori
Short	Memiliki nilai integer dari -32768 sampai 32767 dan menempati 2 byte (16 bits) di memori
Int	Memiliki nilai dari -2147483648 sampai 2147483647 dan menempati 4 byte (32 bits) di memori
Long	Memiliki nilai dari -9223372036854775808 sampai 9223372036854775807 dan menempati 8 byte (64 bits) di memori

Sumber: Rickyanto (2003:38)

3. Tipe Data *Floating point*

   Tipe data *Floating-Point* merupakan tipe data primitif dalam java dan digunakan untuk menangani bilangan *floating point* yang kita gunakan untuk

menangani bilangan decimal atau perhitungan yang lebih detail dibanding *integer*.

Ada dua macam *floating point*, yaitu:

a. *Float* : meimiliki nilai -3.4×10^{8} sampa $+3.4 \times 10^{8}$ dan menempati 4 *byte* di memori.

b. *Double*: Memiliki nilai -1.7×10^{308} sampai $+1.7 \times 10^{308}$

   Semua bilangan pecahanan atau decimal dalam java tanpa diakhiri huruf "f" akan dianggap *double*.

   Sedangkan bilangan yang ingin anda kategorikan sebagai *float* harus diakhiri dengan huruf "F". misalnya :

   4.22F

   2.314f

   Sedang kan untuk bilangan *double*, anda dapat memilih untuk menambah huruf D, karena secara default bilangan dengan koma atau pecahan atau decimal akan dianggap *double*.

4. Tipe Data Char

   Char adalah karakter tunggal yang didefinisikan dengan diawali dan diakhiri dengan tanda ‘ (petik tunggal). Char berbeda dengan *string*, karena *string* bukan merupakan tipe data primitif tetapi sudah merupakan objek.

   Tipe char mengikuti aturan *Unicode* sehingga anda dapat menggunakan kode untuk \u kemudian diikuti bilangan dari 0 sampai 65535, tetapi biasanya yang digunakan adalah bilangan heksadesimal dari 0000 sampai FFFF.

   Jadi sah-sah saja bila anda mendefinisikan karakter seperti berikut:

   ‘\u023’

   Untuk mengetahui lebih lanjut mengenal Unicode, anda dapat mengunjungi situs http://www.unicode.org.

Selain karakter biasa juga terdapat karakter khusus yang didefinisikan dengan cara mengawalinya menggunakan tanda \ seperti table berikut.

Tabel 2.2
Kharakter Khusus

Kode	Nama	Nilai Unicode
\b	*Backspace*	\u0008
\t	*Tab*	\u0009
\n	*Linefeed*	\u000a
\r	*Carriage return*	\u000d
*	*Double quote*	\u0022
\’	*Single quote*	\u0027
\\	*Backslash*	\u005c

Sumber: Rickyanto(2003:40)

2.3.3.2 Tipe Data Referensi

Kelebuhan pemrograman berorientas objek adalah bahwa kita dapat mendefinisikan tipe data baru yang merupakan objek dari class tertentu. Tipe data ini digunakan untuk mereferensikan objek atau class tertentu, Seperti string.

2.3.4 Pengertiaan *MySQL*

Kadir (2008:2), “MySQL merupakan *software* yang tergolong sebagai DBMS (*Database Management System*) yang bersifat *Open Source*. *Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat MySQL), selain itu tentu saja bentuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi, dan bisa diperoleh dengan cara men-*download* (mengunduh) di internet secara geratis ”.

Sebagai *software* DBMS MySQL memiliki sejumlah fitur seperti yang dijelaskan di bawah ini.

1. *Muliplatform*

   MySQL tersedia pada beberapa platform (Windows, Linux, Unix, dan lain-lain).

2. Andal, Cepat dan Mudah Digunakan

   MySQL tergolong sebagai *database server* (*server* yang melayani permintaan terhadap *database*) yang andal dapat menangani database fungsi untuk mengakses *database*, dan sekaligus mudah untuk digunakan. Berbagai *tool* pendukung juga bersedia (walaupun dibuat oleh pihak lain). Perlu diketahui, MySQL dapat menangani sebuah *table* yang berukuran dalam *terabyte* (1 *terabyte* = 1024 *gigabyte*). Namun ukuran yang sesungguhnya sangat bergantung pada batasan sistem operasi, sebagai contoh pada sistem solaris 9/10 batasan ukuran file sebesar 16 *terabyte*.

3. Jaminan Keamanan Akses

   MySQL mendukung pengamanan *database* dengan berbagai kriteria pengaksesan. Sebagai gambaran, dimungkinkan untuk mengatur *user* tertentu agar bisa mengakses data yang bersifat rahasia (misalnya gaji pegawai), sedangkan *user* lain tidak boleh. MySQL juga mendukung konektivitas ke berbagai *software*. Sebagai contoh, dengan menggunakan ODBC *(Open Database Connectivity), database* yang ditangani MySQL dapat diakses melalui program yang dibuat dengan *Visual Basic*. MySQL juga mendukung program klien yang berbasis java utnuk berkomunikasi dengan *database* MySQL melalui JDBC (*Java Database Connectivity*). MySQL juga bisa diakses melalui aplikasi berbasis *Web*, misalnya dengan menggunakan PHP.

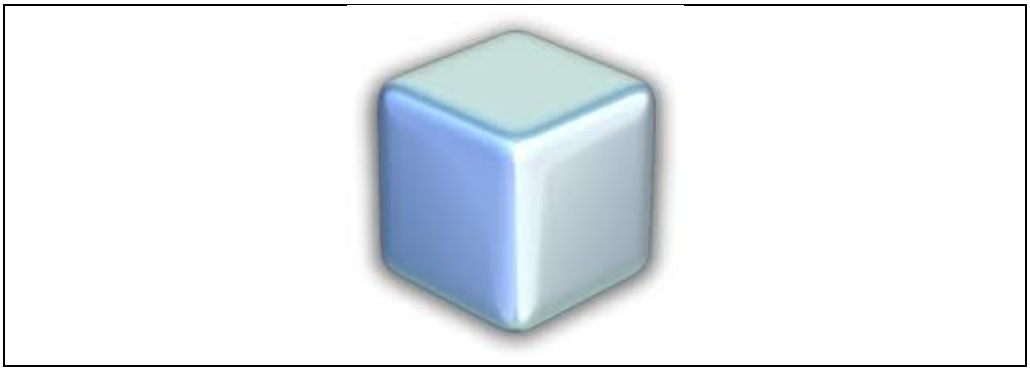
4. Dukungan SQL

   Seperti tersirat dalam namanya, MySQL mendukung perintah SQL *(Structured Query Language).* Sebagaimana diketahui, SQL merupakan standar dalam pengaksesan *database* relasional. Pengetahuan akan SQL akan memudahkan siapa pun untuk menggunakan MySQL.

2.3.5 JAVA dengan Java Netbeans

Nishom (2012),"Netbeans merupakan sebuah aplikasi *Integrated Development Environment* (IDE) yang berbasiskan Java dari *Sun Microsystems* yang berjalan di atas *swing*. *Swing* merupakan sebuah teknologi Java untuk pengembangan aplikasi *desktop* yang dapat berjalan pada berbagai macam

platform seperti windows, linux, Mac OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan *Graphic User Interface* (GUI), suatu kode editor atau text, suatu *compiler* dan suatu *debugger*.



Gambar 2.1
Lambang Java Netbean

Netbeans juga digunakan oleh sang programmer untuk menulis, meng-compile, mencari kesalahan dan menyebarkan program netbeans yang ditulis dalam bahasa pemrograman java namun selain itu dapat juga mendukung bahasa pemrograman lainnya dan program ini pun bebas untuk digunakan dan untuk membuat *professional desktop, enterprise, web, and mobile applications* dengan Java *language*, C/C++, dan bahkan *dynamic languages* seperti PHP, *JavaScript, Groovy,* dan *Ruby*. NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). *Sun Microsystems* mendirikan proyek kode terbuka *NetBeans* pada bulan Juni 2000 dan terus menjadi sponsor utama. Dan saat ini pun *Netbeans* memiliki 2 produk yaitu *Platform Netbeans* dan *Netbeans* IDE. Platform Netbeans merupakan framework yang dapat digunakan kembali (*reusable*) untuk menyederhanakan pengembangan aplikasi *desktop* dan Platform NetBeans juga menawarkan layanan-layanan yang umum bagi aplikasi *desktop*, mengijinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi.

2.3.6 Fitur-Fitur Dari *Platform* Netbeans

a. Manajemen antarmuka (misal: menu & *toolbar*)
b. Manajemen pengaturan pengguna
c. Manajemen penyimpanan (menyimpan dan membuka berbagai macam data)
d. Manajemen jendela
e. *Wizard framework* (mendukung dialog langkah demi langkah)

Netbeans IDE merupakan sebuah IDE *open source* yang ditulis sepenuhnya dengan bahasa pemrograman java menggunakan platform netbeans. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (J2SE, web, EJB, dan aplikasi *mobile*). Fitur lainnya adalah sistem proyek berbasis Ant, kontrol versi, dan r*efactoring*.

Versi terbaru saat ini adalah NetBeans IDE 5.5.1 yang dirilis Mei 2007 mengembangkan fitur-fitur Java EE yang sudah ada (termasuk Java *Persistence support*, EJB-3 dan JAX-WS). Sementara paket tambahannya, *NetBeans Enterprise Pack* mendukung pengembangan aplikasi perusahaan Java EE 5, meliputi alat desain visual SOA, skema XML, *web service* dan pemodelan UML. NetBeans C/C++ Pack mendukung proyek C/C++.

Modularitas: Semua fungsi IDE disediakan oleh modul-modul. Tiap modul menyediakan fungsi yang didefinisikan dengan baik, seperti dukungan untuk bahasa pemrograman Java, editing, atau dukungan bagi CVS. NetBeans memuat semua modul yang diperlukan dalam pengembangan Java dalam sekali download, memungkinkan pengguna untuk mulai bekerja sesegera mungkin. Modul-modul juga mengijinkan NetBeans untuk bisa dikembangkan. Fitur-fitur baru, seperti dukungan untuk bahasa pemrograman lain, dapat ditambahkan dengan menginstal modul tambahan. Sebagai contoh, Sun Studio, Sun Java Studio *Enterprise*, dan *Sun Java Studio Creator* dari *Sun Microsystem* semuanya berbasis *NetBeans* IDE.

Fitur fitur yang terdapat dalam netbeans antara lain:

1. *Smart Code Completion*: untuk mengusulkan nama variabel dari suatu tipe, melengkapi *keyword* dan mengusulkan tipe parameter dari sebuah method.
2. *Bookmarking*: fitur yang digunakan untuk menandai baris yang suatu saat hendak kita modifikasi.
3. *Go to commands*: fitur yang digunakan untuk jump ke deklarasi variabel, source code atau file yang ada pada project yang sama.
4. *Code generator*: jika kita menggunakan fitur ini kita dapat meng-*generate constructor*, *setter* and *getter* method dan yang lainnya.
5. *Error stripe*: fitur yang akan menandai baris yang *error* dengan memberi *highlight* merah.

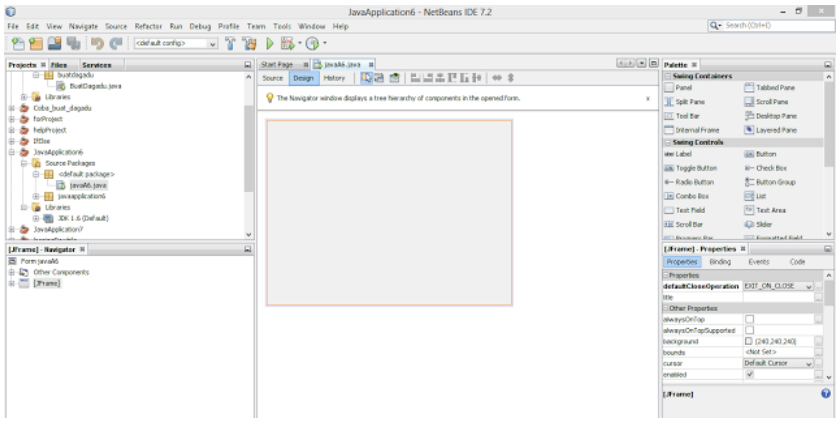
2.3.7 Komponen GUI Netbeans

Saragih (2013), ”Pada aplikasi java, komponen GUI disimpan pada kontainer yang di sebut dengan *form*. Bahasa pemograman java mengumpulkan komponen antar muka dari *form* GUI yang telah terbentuk. Pada NetBeans komponen GUI ini telah tersedia. Hal ini mempermudah anda dalam merancang dan membangun *form* Java.

Komponen GUI pada NetBeans antara lain:

a. GUI *builder*

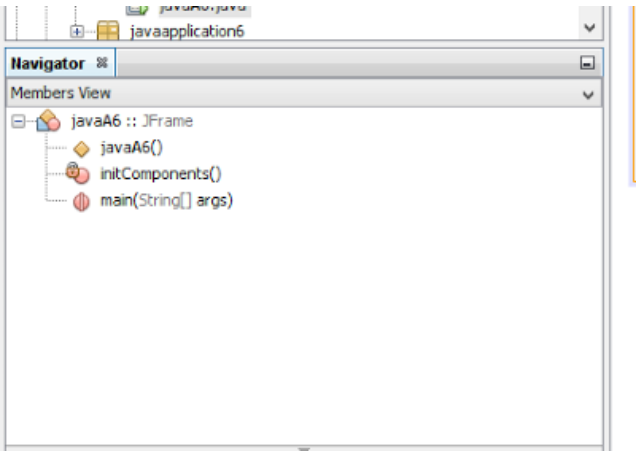
GUI *builder* merupakan jendela utama yang didalamnya terdapat komponen untuk merancang GUI.



Gambar 2.2
GUI *Builder*

b. *Navigator Windows*

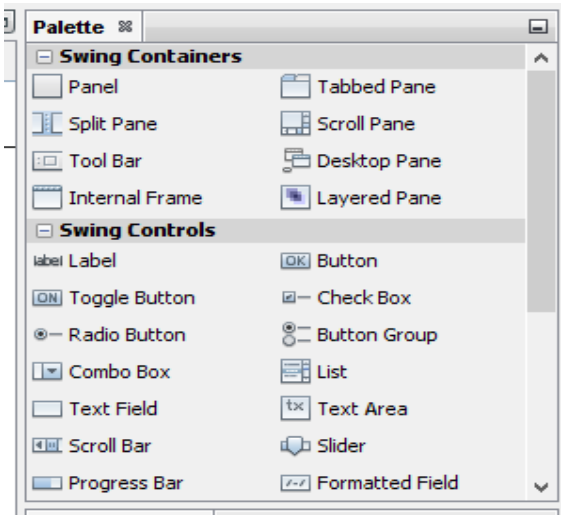
Navigator windows merupakan jendela yang menampilkan pohon pewarisan dari semua komponen *form* yang di buka seperti *button*, label, menu, *timer*, dan sebagainya.



Gambar 2.3
Navigator Windows

c. *Palete Windows*

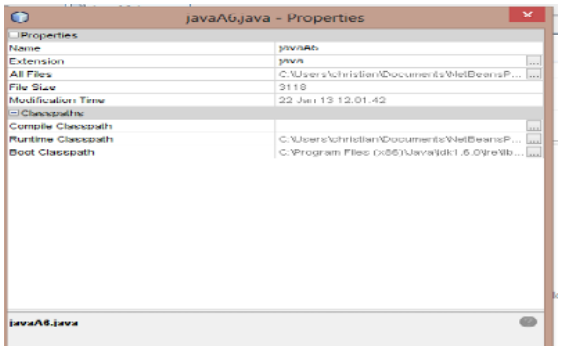
Palete windows adalah jendela yang menampilkan daftar semua komponen *swing* yang dapat dimasukan ke dalam *form* seperti *label, button*, menu dan lainnya



Gambar 2.4
Palete Windows

d. *Properties Windows*

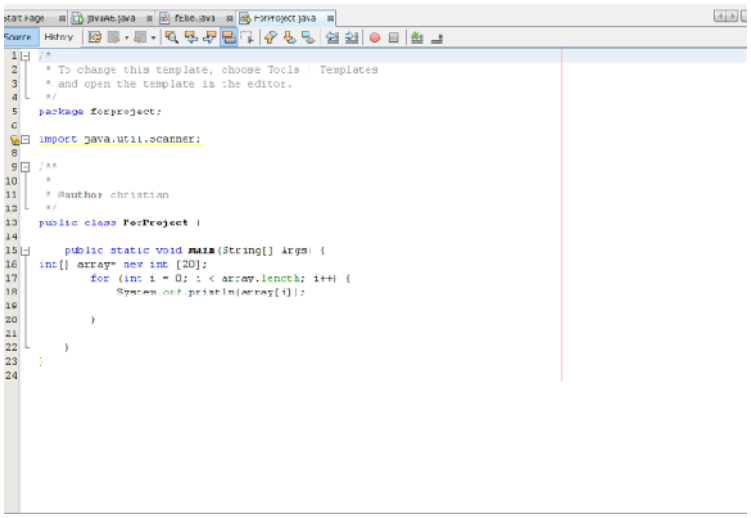
Properties windows merupakan jendela yang dapat di ubah memilih komponen yang akan di pergunakan



Gambar 2.5
Properties Windows

e. *Source Area*

Source area merupakan jendela yang di gunakanuntuk menambahkan kode program pada pemograman JAVA



Gambar 2.6
Source Area

2.3.8 JasperReport

Bima (2011:262), *JasperReports* adalah *library reporting* paling populer di Java, kepopuleran *JasperReports* sangat ditunjang dengan adanya *tools* visual *designer* yaitu *iReport.* Dengan menggunakan *iReports* membuat *report* menjadi sangat cepat dan akurat.

Feature-feature JasperReports juga sudah sangat lengkap, berikut ini adalah daftar featurenya:

a. *Pixel-perfect page-oriented* atau *continuous output* untuk web atau print
b. *Dashboards, tables, crosstabs, charts dan gauges*
c. *Web-based and pixel-perfect reports*
d. *Report output* dalam format PDF, XML, HTML, CSV, XLS, RTF, TXT

e. *Sub-reports* dapat dengan mudah digunakan untuk menampilkan *layout* yang kompleks

f. *Support barcode*

g. Dapat melakukan rotasi posisi text secara visual

h. *Styles library*

i. *Drill-down / hypertext link,* termasuk *support* untuk PDF *bookmarks*

j. Tidak ada batasan ukuran *report*

k. *Conditional printing*

l. *Multi datasource* dalam satu report

m. *Internationalized* dan *Localizable.*

2.3.9 *iReport*

Softmaniak (2013)," *Ireport* adalah *visual designer* untukmembuat laporan yang komplek, menggunakan *jasperReport Library*.

2.4 Teori Khusus

2.4.1 Database

Kristanto (2008:79), "*Database* (basis data) adalah kumpulan data, yang dapat digambarkan sebagai aktifitas dari satu atau lebih organisasi yang berelasi".

Sutarman (2012:15), "*database* merupakan sekumpulan *file* yang saling berhubungan dan terorganisasi atau kumpulan *record-record* yang menyimpan data dan hubungan diantaranya".

Jadi, *database* atau basis data merupakan kumpulan data-data yang saling berhubungan dan terorganisasi yang disimpan dalam media pengingat (*hard disk)* agar dapat diakses dengan mudah dan cepat.

2.4.2 Kamus Data (Data Dictionary)

Kristanto (2008:72), "Kamus data (*data dictonary)* adalah kumpulan elemen-elemen atau simbol-simbol yang digunakan untuk membantu dalam penggambaran atau pengidentifikasi setiap *field* atau *file* di dalam sistem".

Simbol-simbol yang ada dalam kamus data adalah sebagai berikut:

Tabel 2.3
Simbol-simbol Kamus Data

No.	Simbol	Arti
1	=	Terdiri atas
2	+	Dan
3	()	Opsional
4	[]	Memilih salah satu alternatif
5	**	Komentar
6	@	Identifikasi atribut kunci
7	|	Pemisah alternatif simbol []

2.4.3 *Unified Modeling Language* (UML)

Sukamto dan Shalahuddin (2013:133), "*UML* (*Unified Modeling Language)* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek".

UML menyediakan serangkaian gambar dan diagram yang sangat baik. Beberapa diagram memfokuskan diri pada ketangguhan teori *object-oriented* dan sebagian lagi memfokuskan pada detail rancangan dan konstruksi. Semua dimaksudkan sebagai sarana komunikasi antar *team programmer* maupun dengan pengguna.

Widodo dan Herlawati (2011:6-7), *UML* diaplikasikan untuk maksud tertentu, biasanya antara lain untuk:

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Tabel 2.4.
Tipe Diagram *UML*

No.	Diagram	Tujuan
1	*Class*	Memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi
2	*Package*	Memperlihatkan kumpulan kelas-kelas, merupakan dari diagram komponen
3	*Use case*	Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas)
4	*Sequence*	Diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu
5	*Communication*	Sebagai pengganti diagram kolaborasi *UML* 1.4 yang menekankan organisasi struktural dari obyek-obyek yang menerima serta mengirim pesan
6	*Statechart*	Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status *(state),* transisi, kejadian serta aktivitas
7	*Activity*	Tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem
8	*Component*	Memperlihatkan organisasi serta kebergantungan sistem / perangkat lunak pada komponen-komponen yang telah ada sebelumnya
9	*Deployment*	Memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*)

Sumber: Widodo dan Herlawati (2011:10-12)

2.4.4 Jenis-Jenis Diagram *UML*

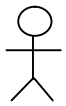
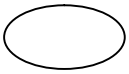
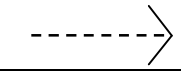
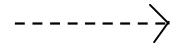
2.4.4.1 *Use case* Diagram

Sukamto dan Shalahuddin (2013:155), "*use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat".

Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Adapun simbol-simbol yang digunakan dalam *use case* adalah sebagai berikut:

Tabel 2.5.
Simbol-simbol *Use case* Diagram

No.	Simbol	Nama	Deskripsi
1		*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
2		*Use case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
3		*Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4		*Extend*	Relasi *use case* tambahan ke sebuah *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5		*Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungai yang satu adalah fungsi yang lebih umum dari lainnya.
6		*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.

Sumber: Sukamto dan Shalahuddin (2013:156-158)

Komponen pembentuk diagram *use case* adalah:

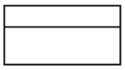
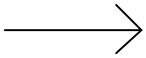
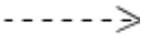
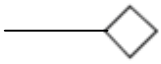
1. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use case*, aktivitas / sarana yang disiapkan oleh bisnis / sistem.
3. Hubungan (*link*), aktor mana saja yang terlibat dalam *use case* ini.

2.4.4.2 *Class* Diagram

Sukamto dan Shalahuddin (2013:141), "*class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem".

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Adapun simbol-simbol yang digunakan dalam *class* diagram adalah sebagai berikut:

Tabel 2.6.
Simbol-simbol *Class* Diagram

No	Gambar	Nama	Deskripsi
1		*Class*	Kelas pada stuktur sistem.
2		*Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3		*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4		*Directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
5		*Generalization*	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus).
6		*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
7		*Aggregation*	Relasi antar kelas dengan makna semua bagian *(whole-part)*.

Sumber: Sukamto dan Shalahuddin (2013:146-147)

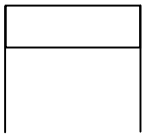
2.4.4.3 *Activity* Diagram

Sukamto dan Shalahuddin (2013:161) "*activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak".

Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Adapun simbol-simbol yang digunakan dalam *activity* diagram adalah sebagai berikut:

Tabel 2.7.
Simbol-simbol *Activity* Diagram

No	Simbol	Nama	Deskripsi
1		Status awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2		Aktivitas	Aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja.
3		Decision	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		*Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5		Status akhir	Status akhir yang dilakukan sebuah sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Sumber: Sukamto dan Shalahuddin (2013:162-163)

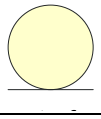
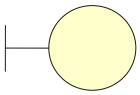
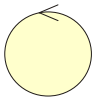
2.4.4.4 *Sequence* Diagram

Sukamto dan Shalahuddin (2013:165) , "diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek".

Sequence diagram menunjukkan urutan *event* kejadian dalam suatu waktu. Komponen *sequence* diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu

yang ditunjukkan dengan progress vertikal. Simbol-simbol yang digunakan dalam *sequence* diagram adalah:

Tabel 2.8.
Simbol-simbol *Sequence* Diagram

No	Simbol	Nama	Keterangan
1		*An Actor*	Menggambarkan orang yang sedang berinteraksi dengan sistem
2		*Entity Class*	Menggambarkan hubungan kegiatan yang akan dilakukan
3		*Boundary Class*	Menggambarkan sebuah penggambaran dari *form*
4		*Control Class*	Menggambarkan penghubung antara boundary dengan tabel
5		*A focus of control*	Menggambarkan tempat mulai dan berakhirnya sebuah *message* (pesan)
6		*A line of life*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi