BAB II
TINJAUAN PUSTAKA

2.1. Teori Umum

Teori ini membahas tentang pengertian-pengertian yang berkaitan dengan judul laporan akhir.

2.1.1. Pengertian Aplikasi

Asropudin (2013:6), menjelaskan bahwa "*application* atau aplikasi merupakan *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya Microsoft Word, Microsoft Excel".

Sutarman (2012:285), menjelaskan bahwa "program aplikasi adalah program-program yang dibuat oleh suatu perusahaan komputer untuk para pemakai yang beroperasi dalam bidang-bidang umum, seperti toko, penerbitan, komunikasi, penerbangan, perdagangan, dan sebagainya".

Berdasarkan pengertian diatas penulis menyimpulkan bahawa aplikasi atau program aplikasi adalah *software* atau program-program yang dibuat untuk membantu para pemakai atau *user* dalam mengerjakan tugas-tugas tertentu.

2.1.2. Pengertian Simulasi

Seaparamita (2012), menjelaskan bahwa "simulasi adalah suatu cara untuk menduplikasi/menggambarkan ciri, tampilan, dan karakteristik dari suatu sistem nyata. Ide awal dari simulasi adalah untuk meniru situasi dunia nyata secara matematis, kemudian mempelajari sifat dan karakter operasionalnya, dan akhirnya membuat kesimpulan dan membuat keputusan berdasar hasil dari simulasi."

Kamus Besar Bahasa Indonesia (2014) menjelaskan bahwa "simulasi adalah metode pelatihan yang meragakan sesuatu dalam bentuk tiruan yang mirip dengan keadaan yang sesungguhnya atau penggambaran suatu sistem atau proses dengan peragaan berupa model statistik atau pemeranan."

Berdasarkan pengertian diatas penulis menyimpulkan bahawa secara keseluruhan, simulasi merupakan pemeragaan atau peniruan atas suatu kejadian dengan sesuai dengan kejadian aslinya.

2.1.3. Pengertian Ujian Nasional

Shvoong(2014), menjelaskan bahwa ”Ujian nasional merupakan penilaian pada akhir proses pembelajaran di sekolah.Penilaian merupakan serangakaian kegiatan untuk memperoleh, menganalisis,dan menafsirkan data tentang proses dan hasil belajar siswa yang dilakukansecara sistematis dan berkesinambungan sehingga menjadi informasi yangbermakna dalam mengambil keputusan.

Didalam Peraturan Menteri Pendidikan dan Kebudayaan Indonesia Nomor 97 tahun 2013 tentang kriteria kelulusan peserta didik dari satuan pendidikan dan penyelenggaraan ujian sekolah/madrasah/pendidikan kesetaraan dan ujian nasional,dijelaskan juga bahwa Ujian Nasional yang selanjutnya disebut UN adalah kegiatan pengukuran dan penilaian pencapaian standar kompetensi lulusan secara nasional pada mata pelajaran tertentu.

Jadi, dapat disimpulkan bahwa Ujian Nasional adalah Penilaian pada akhir proses pembelajaran dilakukan ujian untuk mendapatkan kelayakan dan kesetaraan kualitas pendidikan pada tingkat nasional.

2.1.4. Pengertian Aplikasi Desktop

Dew Omenn (2013) menjelaskan bahwa ”*desktop application* atau aplikasi desktop adalah suatu aplikasi yang dapat berjalan sendiri atau independen tanpa menggunakan *browser* atau koneksi internet disuatu komputer otonom.”

Rafyrpl101(2013), menjelaskan bahwa “aplikasi berbasis *desktop* merupakan aplikasi yang dijalankan pada masing-masing komputer atau klien. Aplikasi berbasis *desktop* harus diinstall terlebih dahulu ke dalam komputer agar dapat digunakan.

Berdasarkan pengertian diatas penulis dapat menyimpulkan bahwa aplikasi berbasis desktop adalah aplikasi yang berjalan pada komputer yang dapat digunakan secara langsung ketika kode program selesai dikompilasi.

2.1.5 Pengertian Sekolah

Kamus Besar Bahasa Indonesia Edisi Baru (2009:1244), menjelaskan bahwa ”sekolah adalah bangunan atau lembaga untuk belajar dan mengajar serta tempat menerima dan member pelajaran”.

2.1.6 Pengertian Judul Secara Keseluruhan

Dari uraian diatas dapat disimpulkan bahwa Aplikasi Simulasi Ujian Nasional pada SMA Negeri 1 Rambang Dangku Berbasis Desktop merupakan suatu aplikasi yang berjalan langsung dikomputer tanpa media perantara(*browser*, *web server*), berfungsi untuk membantu siswa dalam mengerjakan latihan ujian nasional dan membantu guru dalam mengkoreksi hasil latihan siswa tersebut.

2.2. Teori Khusus

2.2.1. *UML (Unified Modeling Language)*

Rosa dan Shalahuddin (2013:133), menjelaskan tentang pengertian UML sebagai berikut :

> *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.

Gambar 2.1. Tampilan Logo *UML*

Widodo dan Herlawati (2011:6-7), menjelaskan tentang kegunaan UML sebagai berikut :

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 2.1. 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)

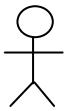
2.2.2. Jenis-Jenis Diagram *UML*

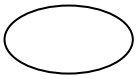
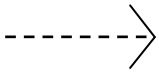
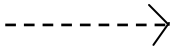
2.2.2.1. *Use case* Diagram

Rosa dan Shalahuddin (2013:155), menjelaskan tentang *use case* diagram sebagai berikut :

> *Use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem . Adapun simbol-simbol yang digunakan dalam *use case* adalah sebagai berikut:

Tabel 2.2. 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>> 	*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>> 	*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: Rosa dan Shalahuddin (2013:156-158)

Komponen pembentuk diagram *use case* adalah:

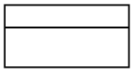
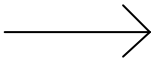
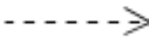
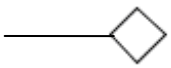
1. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use case*, aktivitas / sarana yang disiapkan oleh bisnis / sistem.
3. Hubungan (*link*), aktor mana saja yang terlibat dalam *use case* ini.

2.2.2.2. *Class* Diagram

Rosa dan Shalahuddin (2013:141), menjelaskan tentang *class* diagram sebagai berikut :

> *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 2.3. 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)*.

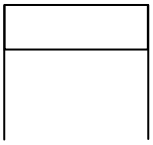
Sumber: Rosa dan Shalahuddin (2013:146-147)

2.2.2.3. Activity Diagram

Rosa dan Shalahuddin (2013:161), menjelaskan tentang *activity* diagram sebagai berikut :

> A*ctivity* 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 2.4. 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.

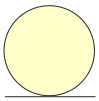
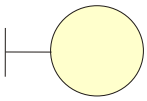
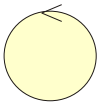
Sumber: Rosa dan Shalahuddin (2013:162-163)

2.2.2.4. Sequence Diagram

Rosa dan Shalahuddin (2013:165), menjelaskan tentang *sequence* diagram sebagai berikut :

> Diagram *sequence* 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 2.5. 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 menggambaran 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

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

2.2.3 Kamus Data (*Data Dictionary*)

Rosa dan Shalahuddin (2013:73), menjelaskan bahwa "kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan *(input)* dan keluaran *(output)* dapat dipahami secara umum (memiliki standar cara penulisan)".

Adapun simbol-simbol yang digunakan dalam kamus data adalah:

Tabel 2.6. Simbol-simbol Kamus Data

Simbol	Keterangan
=	disusun atau terdiri dari
+	Dan
[|]	baik...atau...
{ }^{n}	n kali diulang/bernilai banyak
( )	data opsional
..	batas komentar

Sumber: Rosa dan Shalahuddin (2013:73)

2.3. Teori Program

2.3.1. Basis Data *(Database)*

Rosa dan Shalahuddin (2013:43), menjelaskan bahwa "basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat".

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

Kadir (2008:3), menjelaskan bahwa "*database* berarti koleksi data yang saling terkait. Secara praktis, basis data dapat dianggap sebagai suatu penyusunan data yang terstruktur yang disimpan dalam media pengingat (*hard disk*) yang tujuannya adalah agar data tersebut dapat diakses dengan mudah dan cepat".

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.

2.3.2. *JAVA*

2.3.2.1 Pengertian *Java*

Supriyatno (2010:10), menjelaskan tentang pengertian *java* sebagai berikut :

> Java merupakan bahasa pemrograman berorientasi objek yang dapat berjalan pada platform yang berbeda baik *Windows*, *Linux*, serta system operasi lainnya. Jadi kita dapat membuat sebuah aplikasi dengan java pada system operasi linux dan selanjutnya menjalankan atau menginstal aplikasi tersebut pada system operasi *windows* dan juga sebaliknya tanpa mengalami masalah. Dengan menggunakan java, kita dapat mengembangkan banyak aplikasi yang dapat digunakan pada lingkungan yang berbeda, seperti pada :*Desktop*, *Mobile*, *Internet*, dan lain-lain.

2.3.2.2 Kelebihan Java

Rickyanto (2003:3), menjelaskan tentang kelebihan *java* sebagai berikut :

> 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.
>
> 1. Java memiliki *array* yang tidak memerlukan pointer sehingga memudahkan para programmer.
> 2. 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 (= =).

3. Java menghilangkan multiple inheritance pada C++ dan menggunakan *interface* yang memiliki kemampuan yang sama tetapi lebih sederhana.

2.3.2.3 Paket Instalasi *Java*

Supriyatno (2010:10), menjelaskan tentang paket instalasi pada *java* sebagai berikut :

> Untuk menginstalasi dan menggunakan *Java,Sun Microsystem* selaku pengembang java menyediakan paket instalasi sesuai dengan kebutuhan kita dalam membangun aplikasi. Berikut ini uraian singkat mengenai paket aplikasi java yang tersedia.
>
> 1. *J2ME*(*Java* 2 *Micro Edition*)
> Paket instalasi ini dapat digunakan untuk mengembangkan software yang berjalan pada perangkat yang memiliki memori dan sumber daya yang kecil, seperti pada *Handphone*, *PDA*, dan *Smartcard*.
>
> 2. *J2SE*(*Java* 2 *Standard Edition*)
> Paket instalasi ini dapat digunakan untuk mengembangkan aplikasi yang berjalan pada lingkungan *workstation*, seperti aplikasi *desktop*.
>
> 3. *J2EE*(*Java* 2 *Enterprise Edition*)
> Paket instalasi ini dapat digunakan untuk mengembangkan aplikasi pada lingkungan internet maupun aplikasi skala *enterprise*."

2.3.2.4 Deklarasi Kelas dalam Java

Ifnu Bima (2011:20), menjelaskan tentang *class* dalam *java* sebagai berikut :

> *Class* di dalam *java* dideklarasikan menggunakan *keyword class* diikuti dengan nama *class*.Setelah nama *class* ada kurung kurawal buka ({) menandai awal dari *class* dan kurung kurawal tutup (}) yang menandai akhir dari *class.*

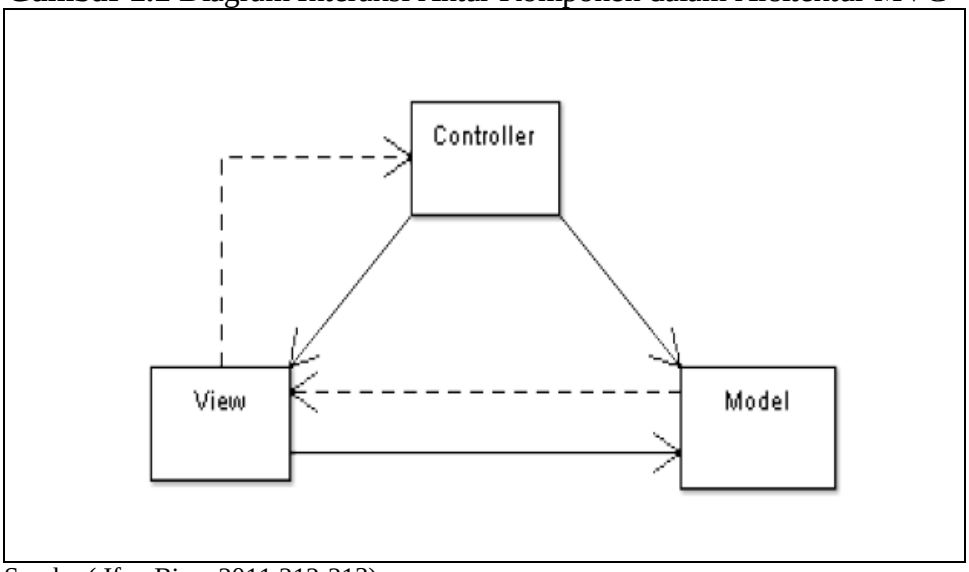
2.3.3 Pengertian MVC (*Model view controller*)

Ifnu Bima (2011:212-213) menjelaskan tentang *MVC* sebagai berikut :

> MVC adalah arsitektur aplikasi yang memisahkan kode-kode aplikasi dalam tiga lapisan, *Model, View dan Control.* MVC termasuk dalam arsitektural design pattern yang menghendaki organisasi kode yang terstruktur dan tidak bercampur aduk. Ketika aplikasi sudah sangat besar

dan menangani struktur data yang kompleks, harus ada pemisahan yang jelas antara domain *model*, komponen *view* dan *kontroler* yang mengatur penampilan *model* dalam *view.*

Gambar 2.2 Diagram Interaksi Antar Komponen dalam Arsitektur MVC

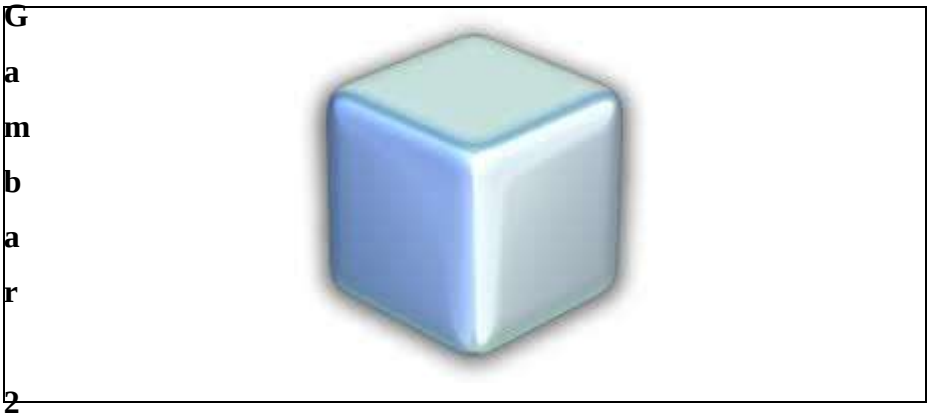


Sumber(Ifnu Bima 2011:212-213)

2.3.4 Netbeans

Nishom (2012), menjelaskan tentang netbeans sebagai berikut :

> 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 2.3 Diagram interaksi antar komponen dalam arsitektur MVC

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.

2.3.4.1 Fitur-Fitur Dari Platform Netbeans

Nishom (2012),menjelaskan tentang fitur-fitur di dalam netbeans sebagai berikut :

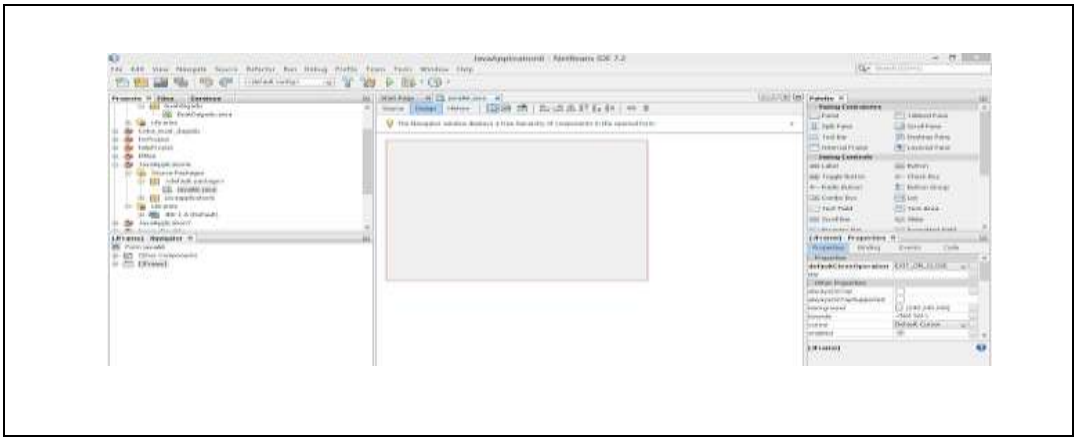
> Netbeans IDE merupakan sebuah IDE *open source* yang ditulis sepenuhnya dengan bahasa pemrograman java menggunakan platform netbeans. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (J2SE, web, EJB, dan aplikasi *mobile*). Fitur lainnya adalah sistem proyek berbasis Ant, kontrol versi,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

2.3.4.2 Komponen GUI Netbeans

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. Komponen GUI pada NetBeans antara lain:

a. GUI *builder*

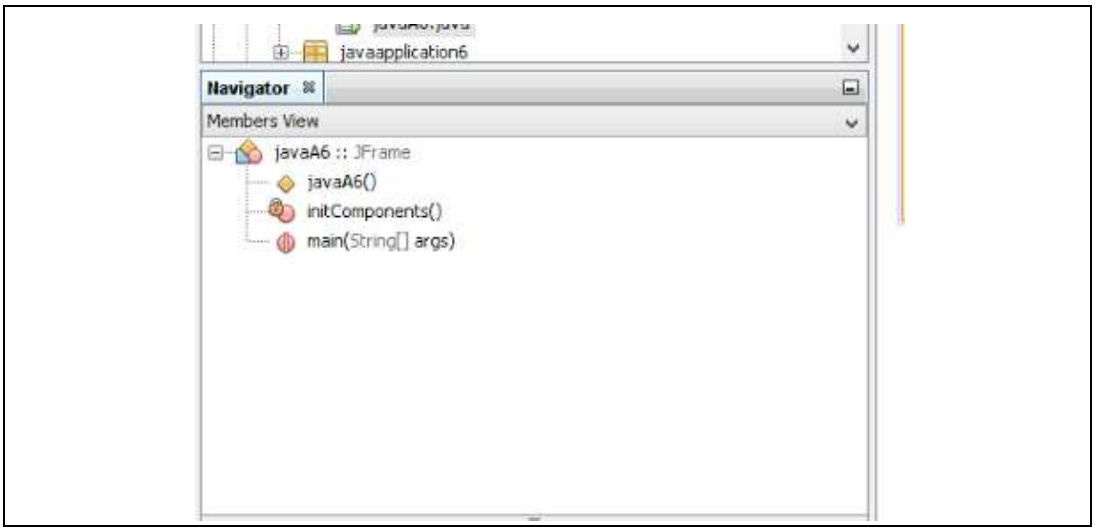
GUI *builder* merupakan jendela utama yang didalamnya terdapat komponen untuk merancang GUI.



Gambar 2.4 GUI *Builder*

b. *Navigator Windows*

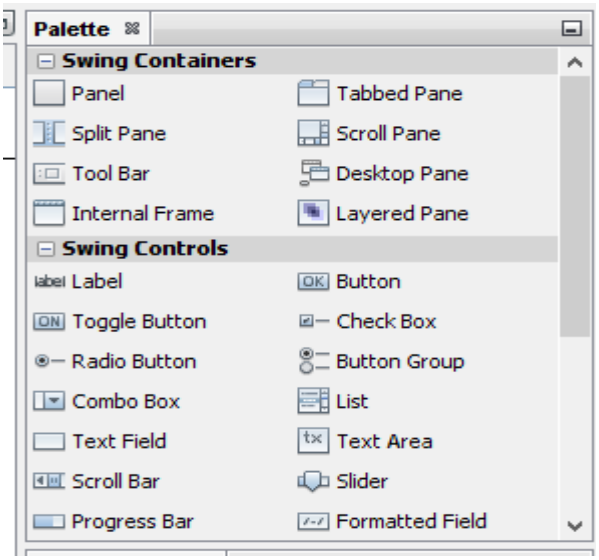
Navigator windows merupakan jendela yang menampilkan pohon pewarisan dari semua komponen form yang di buka spserti button, label, menu, timer, dan sebagainya.



Gambar 2.5 *Navigator Windows*

c. *Palete Windows*

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



Gambar 2.6 *Palete Windows*

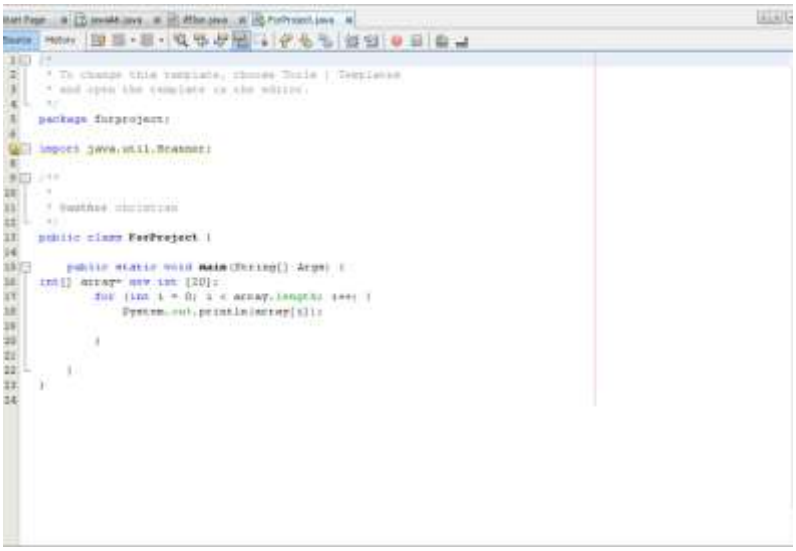
d. Properties Windows

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

Gambar 2.7 *Properties Windows*

e. *Source Area*

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



Gambar 2.8 *Source Area*

2.3.5 MySQL

Kadir (2008:2), menjelaskan tentang *Mysql* sebagai berikut :

> *MySQL* (baca: mai-se-kyu-el) merupakan *software* yang tergolong *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 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 gratis.
> Kelebihan dari *MySQL* adalah dapat digunakan untuk aplikasi *multi-user* (banyak pengguna) dan menggunakan bahasa *query* (permintaan) standar *SQL (Structured Query Language)*.
> Kekurangan dari *MySQL* adalah untuk koneksi ke bahasa pemrograman *visual* seperti *VB*, Delphi, dan Foxpro, *MySQL* kurang *support*, karena koneksi ini menyebabkan *field* yang dibaca harus sesuai dengan koneksi dari program visual tersebut, dan ini yang menyebabkan *MySQL* jarang dipakai dalam program visual serta data yang ditangani belum begitu besar."

2.3.5.1 Fungsi-fungsi *MySQL*

Kadir (2008:360-379), menjelaskan tentang fungsi-fungsi mysql sebagai berikut:

Tabel 2.7. Fungsi-fungsi *MySQL*

No	Nama Fungsi	Kegunaan	Bentuk Pemanggilan
1	*MySQL_connect()*	Membuat hubungan ke *database MySQL* yang terdapat pada suatu *host*.	mysql_connect (*host*, nama_pemakai, *password*)
2	*MySQL_close()*	Menutup hubungan ke *database MySQL*.	mysql_close (pengenal_hubungan)
3	*MySQL_select_db()*	Memilih *database*	mysql_select_db (database, pengenal_hubungan)
4	*MySQL_query()*	Mengeksekusi permintaan terhadap sebuah tabel atau sejumlah tabel.	mysql_query (permintaan, pengenal_hubungan)
5	*MySQL_db_query()*	Menjalankan suatu permintaan terhadap suatu *database*.	mysql_db_query (*database*, permintaan, pengenal_hubungan)
6	*MySQL_num_rows()*	Memperoleh jumlah baris dari suatu hasil permintaan (*query*) yang menggunakan SELECT.	mysql_num_rows (pengenal_hasil)
7	*MySQL_affected_rows()*	Memperoleh jumlah baris yang dikenai operasi INSERT, DELETE, dan UPDATE.	mysql_affected_rows ([pengenal_hubungan])
8	*MySQL_num_fields()*	Memperoleh jumlah kolom pada suatu hasil	mysql_num_fields

		permintaan.	(pengenal_hasil)
9	*MySQL_fecth_row()*	Menghasilkan suatu *array* yang berisi seluruh kolom dari sebuah baris pada suatu himpunan hasil.	mysql_fetch_row (pengenal_hasil)
10	*MySQL_fecth_array()*	Mempunyai kegunaan serupa dengan mysql_fetch_row(). Hanya saja, setiap kolom akan disimpan dua kali pada *array* hasil.	mysql_fetch_array (pengenal_hasil)
11	*MySQL_fecth_field( )*	Memperoleh informasi suatu kolom.	mysql_fetch_field (pengenal_hasil [, nomor_kolom])
12	*MySQL_data_seek()*	Memindah *pointer* pada suatu himpunan hasil supaya menunjuk ke baris tertentu.	mysql_data_seek (pengenal_hasil, nomor_baris)
13	*MySQL_field_seek()*	Memindah *pointer* pada suatu himpunan hasil supaya menunjuk ke kolom tertentu.	mysql_data_seek (pengenal_hasil, nomor_kolom)
14	*MySQL_create_db()*	Menciptakan *database MySQL*.	mysql_create_db (database [, pengenal_hubungan])
15	*MySQL_drop_db()*	Menghapus *database MySQL*.	mysql_drop_db (database [, pengenal_hubungan])
16	*MySQL_list_dbs()*	Menghasilkan daftar *database MySQL*.	mysql_list_dbs ([pengenal_hubungan])
17	*MySQL_list_tables( )*	Memperoleh daftar nama tabel dalam suatu *database*.	mysql_list_tables (database [, pengenal_hubungan])
18	*MySQL_list_fields()*	Memperoleh daftar nama kolom dalam suatu *database*.	mysql_list_fields (database [, pengenal_hubungan])

Sumber: Kadir (2008:360-379)