BAB II

TINJAUAN PUSTAKA

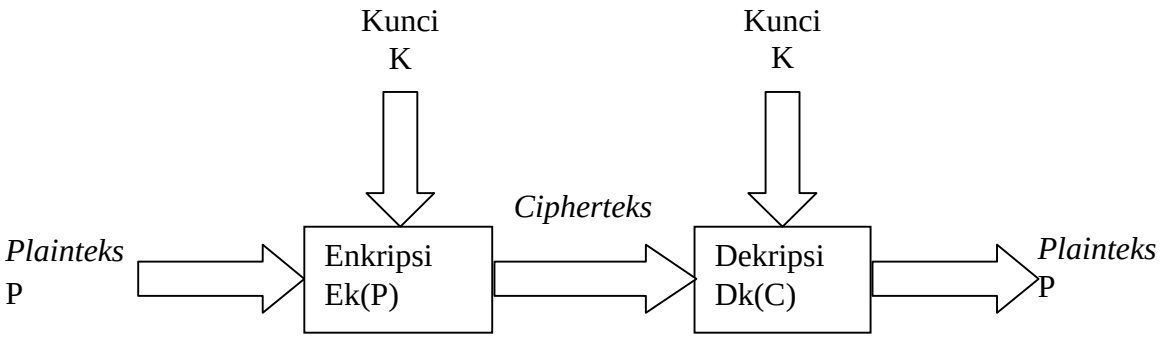
II.1 Kriptografi

II.1.1 Pengertian Kriptografi

Kriptografi berasal dari bahasa yunani, menurut bahasa dibagi menjadi dua kripto dan *graphia*, kripto berarti *secret* (rahasia) dan *graphia* berarti writing (tulisan). Sehingga kriptografi dapat diartikan sebagai "tulisan yang dirahasiaan". Menurut terminologinya kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ketempat yang lain. (Dony Ariyus; 2006 : 9).

Selain itu, kriptografi juga dapat diartikan sebagai ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasian data, keabsahan data, integritas data, serta autentifikasi data.

Dalam kriptografi, pesan yang belum disandikan disebut plainteks, sedangkan pesan yang telah disandikan disebut cipherteks. Terdapat dua proses pada kriptografi yang berguna untuk menyandikan dan mengekstraksi pesan yang telah disandikan. Proses tersebut antara lain enkripsi dan dekripsi. Enkripsi adalah proses menyandikan plainteks menjadi cipherteks. Sedangkan dekripsi merupakan proses mengembalikan cipherteks menjadi plainteks semula, agar dapat diketahui informasi yang terkandung didalamnya. (Arifin Luthfi P; 2010: 1).



Gambar II.1. Skema Enkripsi dan Dekripsi Menggunakan Kunci.
Sumber : Arifin Luthfi P, 2010, 1.

Misalkan:

C = Chiperteks

P = Plainteks

Fungsi enkripsi *E* memetakan *P* ke *C*,

$E(P) = C$

Fungsi dekripsi *D* memetakan *C* ke *P*,

$D(C) = P$

Fungsi enkripsi dan dekripsi harus memenuhi sifat:

$D(E(P)) = P$

II.1.2 Algoritma Kriptografi

Algoritma ditinjau dari asal usul kata, kata algoritma mempunyai sejarah yang menarik, kata ini muncul didalam kamus *Webster* sampai akhir tahun 1957 hanya menemukan kata *algorism* yang mempunyai arti proses perhitungan dengan bahasa Arab. Algoritma berasal dari nama penulis buku Arab yang terkenal yaitu Abu Ja'far Muhammad ibnu Musa al-khuwarizmi (al-Khuwarizmi dibaca oleh orang barat menjadi *algorism*). Kata *algorism* lambat laun berubah menjadi

algorithm.

Defenisi terminologinya algoritma adalah urutan langkah-langkah logis untuk penyelesaian masalah yang disusun secara sistematis. Algoritma kriptografi merupakan langkah- langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut (Dony ariyus; 2006: 13). Algoritma kriptografi terdiri dari tiga fungsi dasar yaitu :

1. Algoritma Enkripsi : Enkripsi merupakan hal yang sangat penting dalam kriptografi yang merupakan pengamanan data yang dikirimkan terjaga kerahasiannya. Pesan asli disebut *plainteks* yang dirubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan dengan cipher atau kode. Sama halnya dengan kita tidak mengerti akan sebuah kata, maka kita akan melihatnya didalam kamus atau daftar istilah-istilah. Beda halnya dengan enkripsi, untuk merubah *plainteks* ke bentuk *cipherteks* kita menggunakan algoritma yang dapat menkodekan data yang kita inginkan.
2. Algoritma Dekripsi : dekripsi merupakan kebalikan dari enkripsi, pesan yang telah dienkripsi dikembalikan kebentuk asalnya (*Plainteks*) disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan yang digunakan untuk enkripsi.
3. Kunci : kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi, kunci terbagi jadi dua bagian kunci pribadi (*private key*) dan kunci umum (*public key*).

II.1.3 Tujuan Kriptografi

Dari paparan awal dapat dirangkumkan bahwa kriptografi bertujuan untuk memberi layanan keamanan. Yang dinamakan aspek-aspek keamanan sebagai berikut :

1. Kerahasiaan (*confidentiality*)

   Adalah layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak. Di dalam kriptografi layanan ini direalisasikan dengan menyandikan *plainteks* menjadi *cipherteks*. Misalnya pesan “harap datang pukul 8” disandikan menjadi “trxC#45motyptre!%”. istilah lain yang senada dengan confidentiality adalah *secrecy* dan *privacy*.

2. Integritas data (*data integrity*)

   Adalah layanan yang menjamin bahwa pesan masih asli atau belum pernah dimanipulasi selama pengiriman. Dengan kata lain, aspek keamanan ini dapat diungkapkan sebagai pertanyaan: “ apakah pesan yang diterima masih asli atau tidak mengalami perubahan (modifikasi)?”.

3. Otentikasi (*authentication*)

   Adalah layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user autehentication*). Dua pihak yang saling berkomunikasi harus dapat mengotentikasi satu sama lain sehingga ia dapat memastikan sember pesan.

4. Nirpenyangkalan (*non-repudiation*)

   Adalah layanan untuk menjaga entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan.

II.2 *Vigenere Cipher*

Vigenere Cipher merupakan salah satu metode kriptografi klasik yang cukup terkenal. *Vigenere cipher* merupakan bentuk *polyalphabetic substitution* yang mengenkripsi teks alfabet menggunakan sekumpulan *IDE* dari *caesar cipher*. *Vigenere cipher* sudah dibuat berulang kali. Pada dasarnya, metode ini dikemukakan pada tahun 1553 oleh *Giovan Battista Bellaso* dalam bukunya yang bernama *La cifra del. Sig. Giovan Battista Bellaso*. (Denver ; 2012: 1)

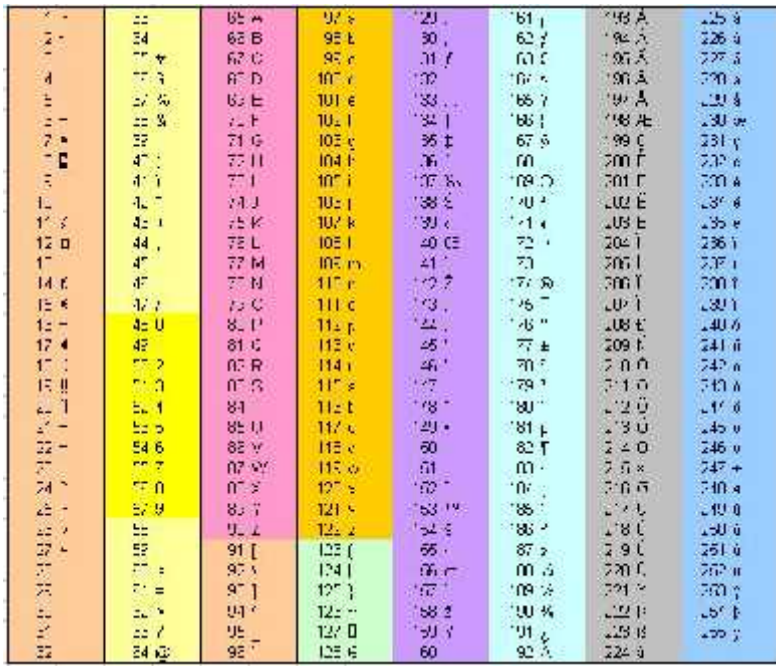
Vigeneree Cipher ini terkenal karena mudah untuk digunakan dan diimplementasikan. Metode ini juga terkenal karena tidak cukup rentan terhadap analisis frekuensi kemunculan huruf. Pada masa kejayaannya sandi ini dijuluki *le chiffre indechifferable*. Metode ini berhasil dipecahkan oleh metode *kasiski*, yang ditemukan oleh *Friedrich Kasiski*. Pada tahun 1863 *Friedrich Kasiski* merupakan orang yang sukses menemukan cara menyerang *vigenere cipher*. Penyerangan pertama menggunakan pengetahun dari *plainteks* atau pengenalan kata sebagai kunci. (Denver ; 2012: 1)

Untuk melakukan enkripsi, *vigenere cipher* ini menggunakan bujur sangkar *vigenere* untuk memetakan karakter *cipherteks*. Pada bujur sangkar *vigenere* ini, kolom paling kiri menunjukan karakter kunci dan baris paling atas menunjukan karakter *plainteks*, karakter-karakter pada baris lainnya menunjukan karakter *cipherteks*. Karakter *cipherteks* tersebut diperoleh dengan menggunakan prinsip *caesar cipher*, yang di mana pergeseran huruf ditentukan dengan nilai desimal dari huruf-huruf yang bersangkutan, di mana untuk karakter berjumlah 26 karakter: a=0, b=1, c=2,d=3,.., z=25. (Rangga Wisnu Adi Permana; 2007: 2).

Seperti ditunjukkan pada tabel II.1 dibawah ini :

Tabel II.1. ASCII

Plainteks



Sumber : Graha Ilmu, 2006, 35.

Panjang kunci dari *vigenere cipher* lebih kecil atau sama dengan panjang dari plainteks, jika panjang kunci lebih kecil daripada panjang plainteks maka kunci akan diulang hingga panjangnya sama dengan panjang plainteks. Tinjau plainteks “THAMA” dengan kunci “SAYAA”, maka proses enkripsi dengan vigènere cipher adalah sebagai berikut: (Bhimantyo Pamungkas; 2007: 2).

Plainteks : THAMA

Kunci : SAYAA

Cara menentukan *cipherteks* pada sistem ini, pada tabel II.2 bisa dilihat pada posisi horizontal merupakan *plainteks* dan pada posisi vertikal adalah kunci, jika *plainteks* huruf K, maka lihat posisi letak huruf K pada *plainteks* tabel dan

posisi huruf T pada posisi kunci, jika sudah menemukan tarik garis lurus kebawah dari *plainteks* dan garis lurus kesamping dari posisi kunci dan akan menemukan huruf S, maka huruf Ç tersebut yang akan manjadi cipherteks dan begitulah seterusnya. Hasil enkripsi seluruhnya adalah :

Plainteks : THAMA

Kunci : SAYAA

Cipherteks : çÉÚÎÂ

Proses : Plainteks + Key = Cipherteks

(t=11b6) + (s=115) = 231(ç)

(h=104) + (a=97) = 201(É)

(a=97) + (y=121) = 218(Ú)

(m=109) + (a=97) = 206(Î)

(a=97)+(a=97)= 194(Â)

Cipherteks : çÉÚÎÂ

Berikut adalah rumus sandi vigenere ciipher dengan kombinasi ascii

P + K = C

Jika hasil penjumlahan P + K lebih besar dari nilai 255 maka

E = Enval – Nilai Max (255) – 1

Contoh = 271 -255 – 1

= 15

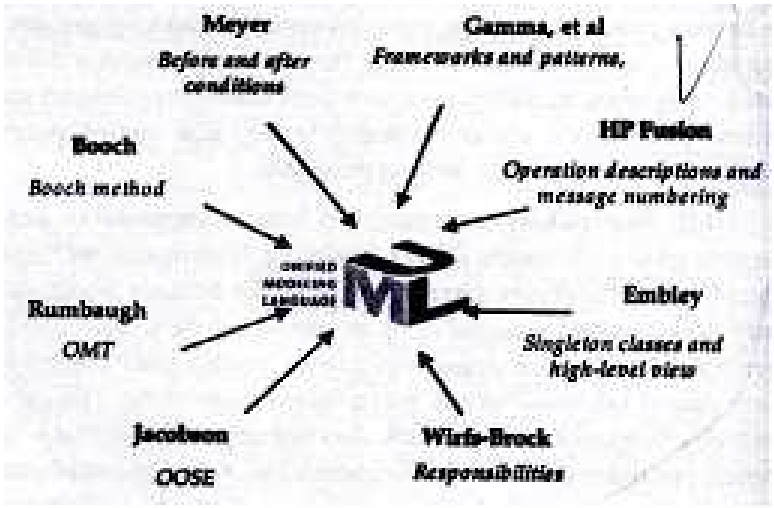
II.3 Konsep UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia perkembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi perkembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (sharing) dan mengkomunikasikan rancangan dengan baik (Munawar ; 2005 : 17).

UML merupakan kesatuan bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *Object Oriented Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan interatif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi.Keunggulan metode Booch adalah pada detail dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstuktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, disain sistem, desain objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan dan *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, desain dan implementasi dan model pengujian (test Model). Keunggulan

metode ini adalah mudah dipelajari karena memiliki notaasi sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya. Unsur-unsur yang membentuk UML ditunjukkan dalam Gambar II.2

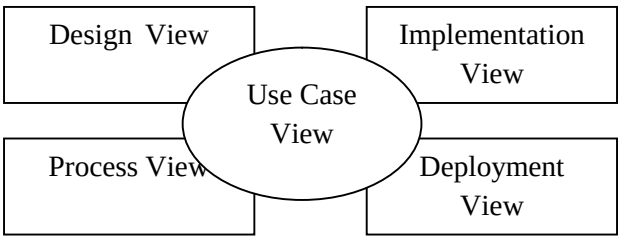


Gambar II.2 Unsur-unsur yang membentuk UML
Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 18

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam OOAD (*Object Oriented Analysis* dan *Design*). UML tidak hanya domain dalam penotasian dilingkungan OO tetapi juga populer di luar lingkungan OO. Ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan bahasa *pemrograman*. Sebagai sebuah sketsa UML bisa berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil.

Dengan cetak biru ini maka akan bisa diketahui informasi detil tentang coding program (*Forward engineering*) atau bahkan membaca program dan mengimplementasikannya kembali ke dalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokkumentasi asli hilang atau bahkan belum perna dibuat sama sekali. Sebagai bahasa pemrograman, UML dapat diterjemahkan diagram yang ada di UML menjadi kode program siap untuk dijalankan.

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahhwa struktur sebuah sistem dideskripsikan dalam *view* dimana salah satu diantaranya *use case view*. *use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain. Model 4+1 *view* ditunjukkan pada gambar II.3



Gambar II.3 Model 4+1 *View*
Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 20

Kelima *view* tersebut tidak berhubungan dengan diagram yang dideskripsikan di UML. Setiap *view* berhubungan dengan perspektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut.

Use case view mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tarik bagi *end user*, analis dan tester. Pandangan ini mendefenisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari peran dan sering dikatakan yang mendrive proses perkembangan perangkat lunak.

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi definisi komponen program, class-class utama bersama-sama dengan spesifikasi data, perilaku dan interaksinya. Informasi yang terkandung di *view* menjadi pergantian para progremer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan diimplementasikan.

Implementasi *view* menjelaskan komponen-komponen visi yang akan dibangun. Hal ini berbeda dengan komponen logic yang dideskripsikan pada *design view*. Termasuk disini diantaranya *file exe*, *library* dan *database*. Informasi yang ada di *view* dan integrasi siistem.

Proses *view* berhubungan dengan hal-hal yang berkaitan dengan *concurrency do* dalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti jaringan komputer dimana sistem akan dijalankan. Kedua view ini menunjukan kebutuhan non fungsional dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja (Munawar;2005:17-21).

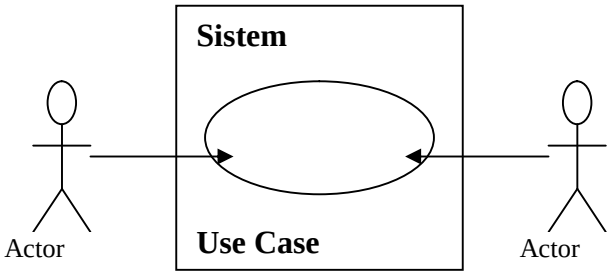
II.3.1 Use Case Diagram

Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain object dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*.

Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system / sub system boundary. Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Ilustrasi *actor, use case* dan *system* ditunjukkan pada gambar II.4



Gambar II.4 *Use Case* Diagram
Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 64

Untuk mengidentifikasi *actor*, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks targer sistem. *Actor* adalah *abstraction* dari orang dan sistem yang lain mengaktifkan fungsi dari target sistem. Orang atau sistem bila muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan use case, tetapi tidak memiliki kontrol atas use *case*.

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. Oleh kerena itu sangat penting untuk memilih abstraksi yang cocok. Use case dibuat berdasarkan keperluan actor. Use case harus merupakan ’apa’ yang dikerjakan software aplikasi, bukan ’bagaimana’ software aplikasi mengerjakannya. Setiap use case harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan actor. Namun use case boleh terdiri dari beberapa kata dan tidak boleh ada dua use case yang memiliki nama yang sama (Munawar ; 2005 : 63-66).

II.3.2 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 :

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Susunan kelas suatu sistem yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main, kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem, kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case*, kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
4. Kelas yang diambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

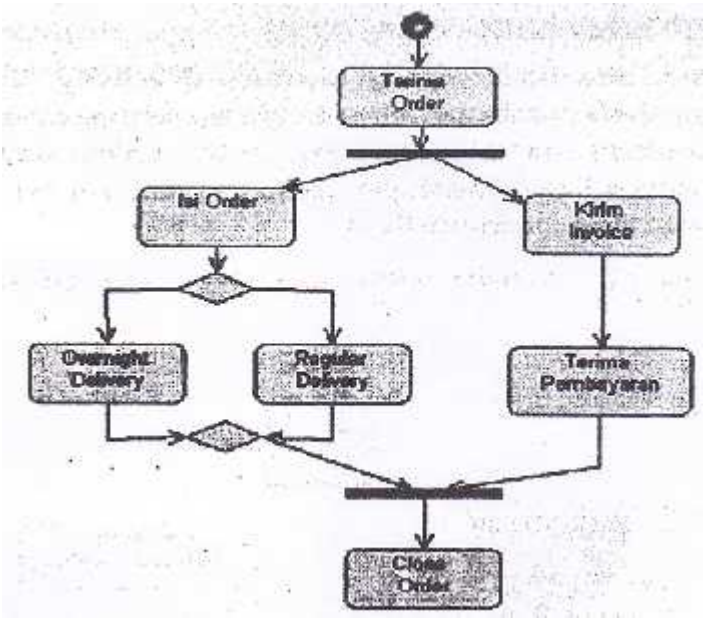
Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohension* dan *coupling*. *Cohension* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan

instruksi antara metode yang satu dengan yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah (Rosa A.S dan M. Shalahuddin ; 2011 : 122-123).

II.3.3 Activity Diagram

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan flowchart tidak bisa(Munawar ; 2005 : 87). Contoh *activity diagram* sederhana ditunjukkan pada gambar II.5



Gambar II.5 Contoh *Activity Diagram* Sederhana
Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 111

II.3.4 Squence Diagram

Squence diagram digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sebuah contoh objek dan pesan yang diletakkan diantara objek-objek ini didalam *use case*.

Komponen utama *Squence diagram* terdiri dari atas objek yang dituliskan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical* (Munawar ; 2005 : 109).

1. Objek / *participant*

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram. Setiap *participant* dihubungkan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation*. *Activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*. *Activation* mewakili sebuah eksekusi operasi dari participant. Panjang kotak ini berbanding lurus dengan durasi *activation.* Bentuk *participant* dapat dilihat pada gambar II.6

Gambar II.6 Bentuk *Participant*
Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 88

2. *Messege*

Sebuah *messege* bergerak dari suatu *participant* ke *participant* yang lain dan dari *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri.

Sebuah *message* bisa jadi *simple*, *synchronous* atau *asynchoronous*. *Message* yang *simple* adalah sebuah perpindahan (transfer), contoh dari satu *participant* ke *participant* yang lainnya. Jika suatu *participant* mengirimkan sebuah *message* tersebut akan ditunggu sebelum di proses dengan urusannya. Namun jika *message asynchoronous* yang dikirimkan, maka jawabannya atas *message* tersebut tidak perlu ditunggu. Simbol *message* pada *squnence diagram* dapat dilihat pada gambar II.7

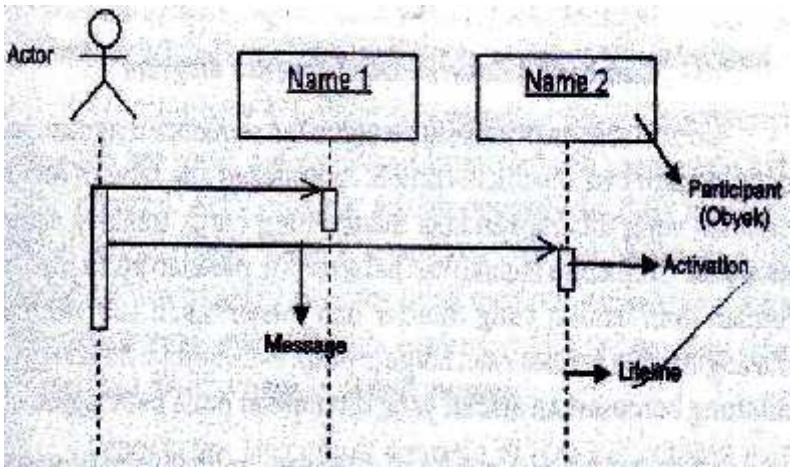
Gambar II.7 Bentuk *Messege*
Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 88

3. *Time*

Time adalah diagram yang mewakili waktu pada arah vertikal. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat kebawah.

Terdapat dua dimensi pada *squence diagram* yaitu dimensi dari kiri ke kanan menunjukkan tata letak participant dan dimensi dari atas ke bawah

menunjukkan lintasan waktu. Simbol-simbol yang ada pada *squence diagram* ditunjukkan pada gambar II.8



Gambar II.8 Bentuk *Time*
Sumber : Pemodelan Visual dengan UML, Munawar, 2005 : 89

II.4 Penjelasan Peta

Peta merupakan penggambaran secara grafis atau bentuk skala (perbandingan) dari konsep mengenai bumi. Hal ini berarti bahwa peta merupakan alat untuk menyampaikan informasi mengenai ilmu bumi. Peta merupakan media yang *universal* untuk komunikasi sehingga dapat mudah dipahami dan dimengerti oleh setiap orang dengan mengabaikan budaya dan bahasa. Sebuah peta merupakan kumpulan gagasan, penggambaran tunggal, konsep-konsep mengenai ilmu bumi yang secara terus menerus mengalami perubahan (Merriam, 1996). Seperti apa peta dahulu diketahui, pengetahuan dasar mengenai peta sama seperti halnya filosofi. Yang mana sering terdapat perbedaan dengan pemetaan modern.

II.4.1 Klasifikasi Peta

Peta dapat diklasifikasi menjadi jenis, yakni :

1. Peta Umum adalah peta yang manampilkan bentuk fisik permukaan bumi suatu wilayah. Contoh : Peta kota Medan Polonia.
2. Peta khusus adalah peta yang menampakkan suatu keadaan atau kondisi khusus suatu daerah tertentu atau keseluruhan daerah bumi. Contohnya adalah peta persebaran hasil tambang, peta curah hujan, peta pertanian perkebunan, peta iklim, dan lain sebagainya.

II.4.2 Macam-Macam dan Jenis Warna Peta

1. Warna Laut

a. hijau : 0 - 200 meter dpl / ketinggian

b. kuning : 200 - 500 meter dpl / ketinggian

c. coklat muda : 500 - 1500 meter dpl / ketinggian

d. coklat : 1500 - 4000 meter dpl / ketinggian

e. coklat berbintik hitam : 4000 - 6000 meter dpl / ketinggian

f. coklat kehitam-hitaman : 6000 meter dpl lebih / ketinggian

2. Warna Darat

a. biru pucat : 0 - 200 meter / kedalaman

b. biru muda : 200 - 1000 meter / kedalaman

c. biru : 1000 - 4000 meter / kedalaman

d. biru tua : 4000 - 6000 meter / kedalaman

e. biru tua berbintik merah : 6000 meter lebih / kedalaman.

II.4.3 Jenis Skala Pada Peta

Skala peta adalah perbandingan jarak di peta dengan jarak sesungguhnya dengan satuan atau tehnik tertentu.

Berikut ini adalah beberapa macam skala, yaitu:

1. Skala angka / skala pecahan

Skala angka adalah skala yang menunjukkan perbandingan antara jarak di peta dan jarak sebenarnya dengan angka. Contoh skala angka 1 : 50.000. Skala angka 1:50.000, artinya jarak satu. Satuan yang tergambar pada peta sama dengan 50.000 satuan di permukaan bumi. Berarti 1 cm di peta mewakili 50.000 cm jarak di lapangan. Jika ada 2 buah kata, yaitu kata A dan B pada sebuah peta yang berskala 1 : 50.000 adalah 20 cm maka jarak sesungguhnya antara kota A dan B adalah :

= 20 cm x 50.000
= 1.000.000 cm
= 10 km

2. SkalaVerbal

Skala Verbal adalah skala yang dinyatakan dengan kalimat atau secara verbal. Skala ini sering terdapat pada peta-peta yang tidak menggunakan satuan pengukuran matrik, seperti peta-peta di Inggris. Contoh skala verbal adalah 1 inchi to 1 mile, artinya 1 inchi pada peta menyatakan jarak 1 mil dilapangan. Apabila 1 mil = 63.360 inchi, maka skala tersebut bila dinyatakan dalam skala angka menjadi 1 : 63.360

3. Skala Grafik

Skala Grafik ditunjukkan oleh garis lurus yang dibagi dalam beberapa ruas, dan setiap ruas menunjukkan satuan panjang yang sama.

II.4.4 Sistem Koordinat Peta

Kalau kita memperhatikan sebuah peta, kita akan melihat garis-garis membujur (menurun) dan melintang (mendatar) yang akan membantu kita untuk menentukan posisi suatu tempat di muka bumi. Garis-garis koordinat tersebut memiliki ukuran (dalam bentuk angka) yang dibuat berdasarkan kesepakatan. Perpotongan antara garis bujur dan garis lintang tersebut dinamakan koordinat peta.

Dengan adanya sistem koordinat, masyarakat menjadi saling memehami posisi masing- masing di permukaan bumi. Dengan sistem koordinat pula, pemetaan suatu wilayah menjadi lebih mudah.

Saat ini terdapat dua sistem koordinat yang biasa digunakan di Indonesia, yaitu sistem koordinat Bujur Lintang dan sistem koordinat UTM (*Universal Transverse)*.

1. Sistem Koordinat Bujur Lintang

   Sistem koordinat bujur-lintang (atau dalam bahasa Inggris disebut *Latitud longitude*), terdiri dari dua komponen yang menentukan, yaitu :

a. Garis dari atas ke bawah (vertikal) yang menghubungkan kutub utara dengan kutubselatan bumi, disebut juga garis lintang (*Latitude*).
b. Garis mendatar (*horizontal*) yang sejajar dengan garis khatulistiwa, disebut

juga garis bujur (*Longitude*).

2. Sistem Koordinat UTM (Universal Transverse Mercator)

   Koordinat Universal Transverse Mercator atau biasa disebut dengan UTM, memang tidak terlalu dikenal di Indonesia karena lebih sering menggunakan koordinat bujur-lintang. Dalam pemetaan partisipatif, agar masyarakat memahaminya disarankan menggunakan dua koordinat yaitu Bujur Lintang dan UTM. Dalam sistem koordinat UTM garis bujurnya hanya menggunakan arah timur yang dalam bahasa Inggris ditulis "East" dan dalam peta disingkat (E), atau dalam bahasa Indonesia ditulis "Timur" dan disingkat (T). Cara menulis koordinatUTM adalah 48 M 0817750 mT UTM 9070450 mU, artinya adalah sebagai berikut :

a. Letak koordinat UTM itu berada berada di zona 48M UTM.
b. Memiliki koordinat bujur 0817750 mT (terletak 817 km dari sebelah Timur awal zona 48).
c. Memiliki koordinat Lintang 9070450 (terletak 950 km ke arah selatan garis khatulistiwa)

II.4.5 Posisi Pada Peta

Yang dimaksud posisi pada peta pada aplikasi ini adalah data atribut atau keterangan yang diinputkan oleh pengguna aplikasi yang akan dienkripsi agar data tersebut aman dari pihak-pihak yang kurang bertanggung jawab, dan koordinat peta.

II.5 Pemrograman Visual Basic

II.5.1 *Visual basic 2010*

Visual basic merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi windows. *Visual basic 2010* merupakan aplikasi pemrograman yang menggunakan teknologi .NET Framework. Teknologi .NET Framework merupakan komponen windows yang terintegrasi serta mendukung pembuatan, penggunaan aplikasi, dan halaman web. Teknologi .NET Framework mempunyai 2 komponen utama, yaitu CLR (Common Language Runtime) dan class library. CLR digunakan untuk menjalankan aplikasi yang berbasis .NET, sedangkan Library adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi. (Wahana Komputer; 2010: 2).

II.5.2. Sistem *Visual Basic 2010*

Sebelum menginstal *visual basic 2010*, komputer harus memenuhi beberapa persyaratan agar *visual basic 2010* dapat dijalankan dengan baik. Adapun, persyaratan (system requirments) yang harus dipenuhi dapat dilihat pada tabel II.2. berikut :

Tabel II.2. Sistem *Requirements Visual Basic 2010*

Sistem	Syarat minimal	Syarat yang direkomendasikan
Arsitektur	x86 dan x64	
Sistem Operasi	Microsoft Windows 7 Service pack 2 Microsoft Windows Server 2003 Windows Vista	
Prosesor	CPU 1.6 GHz (Giga Hertz)	Windows XP dan Windows Server 2003: CPU 2,2 GHz atau yang lebih tinggi. Windows Vista: CPU 2,4 GHz.
RAM	Windows XP dan Windows Server 2003 384 MB (Mega Byte) Windows Vista: 768 MB.	RAM 10 24 MB/1GB atau yang lebih besar.
Harddisk	Tanpa MSDN R Ruang kosong harddisk pada drive penginstalan 2 GB. Sisa ruang harddisk kosong 1 GB.	Kecepatan harddisk 7200 RPM atau yang lebih tinggi.
	Tanpa MSDN R Ruang kosong harddisk pada drive penginstalan 3,8 GB (MSDN diinstal full). 2,8 GB untuk menginstal MSDN default. Kecepatan harddisk 5400 RPM.	
Display Layar	1024x768 display	1280x1024 display

(Sumber : Andi, 2010, 2.)

II.5.3. Mengenal Area Kerja *Visual basic 2010*

Setelah berhasil menginstal *visual studio 2008* yang didalamnya terdapat *visual basic 2010*, maka selanjutnya adalah mencoba menjalankan dan mengenal lingkungan kerja *visual basic 2010* Lingkungan kerja *visual basic* atau disebut *Integrated Development Environment* (*IDE*) adalah suatu lingkungan kerja tempat programmer melakukan pemrograman yang didukung oleh *compiler*, *editor* baik *editor grafis* maupun kode, dan lain sebagainya untuk memudahkan pemrograman. (Wahana Komputer; 2010: 3).

II.5.3.1. Membuka *IDE Visual basic 2010*

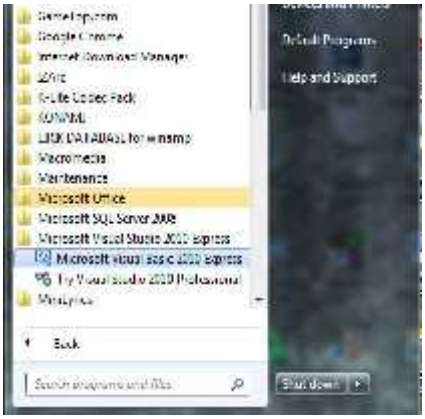
Langkah membuka lingkungan kerja *visual basic 2010* sebagai berikut :

1. Ikuti salah satu cara berikut untuk membuka *IDE* visual basic :

    - Klik ganda shortcut *visual basic 2010* pada desktop.

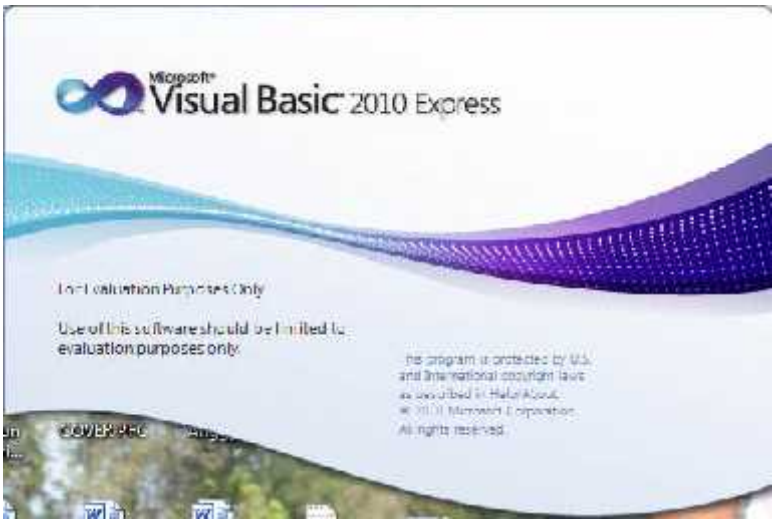
Gambar II.9. Ikon *shortcut* pada desktop
Sumber : Wahana Komputer, 2010, 4.

    - Melalui menu *start* > *Microsoft Visual Studio 2010* > *Microsoft Visual Studio 2010.*



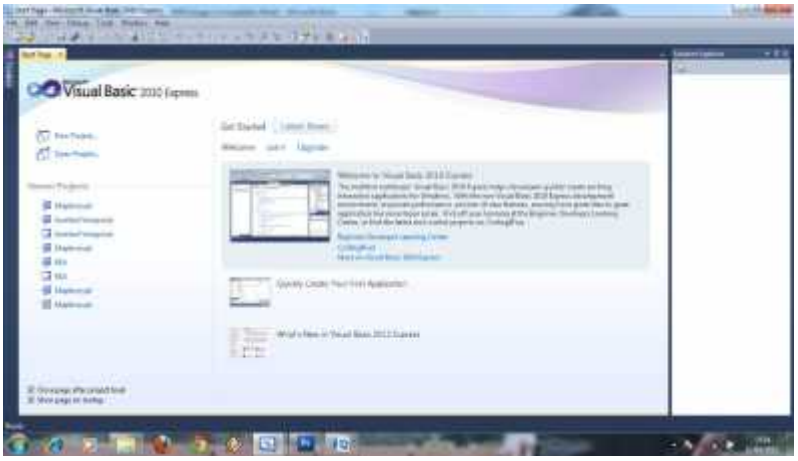
Gambar II.10. Membuka *visual studio 2010* melalui menu
Sumber : Wahana Komputer, 2010, 4.

2. Pada saat pertama kali menjalankan *visual basic 2010*, akan muncul kotak dinformasi proses setting *IDE visual basic 2010*. Setelah proses setting selesai, kotak *splash screen* sesuai dengan gambar II.11. akan muncul :



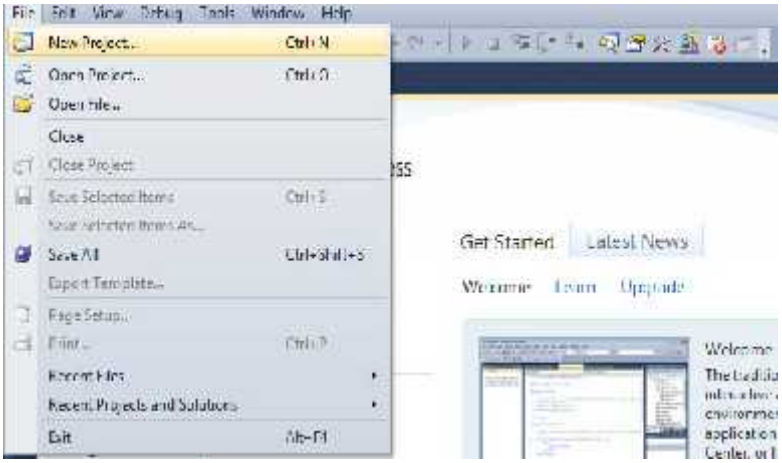
Gambar II.11. *Splash screen* visual studio 2008.
Sumber : Wahana Komputer, 2010, 4

3. Setelah itu, akan terbuka lingkungan kerja atau *IDE* dari *visual basic 2010* yang ditunjukkan pada gambar II.12. berikut :



Gambar II.12. Area kerja *visual basic 2010* saat pertama kali dijalankan
Sumber : Junindar, 2010, 2.

4. Untuk membuat sebuah project baru menggunakan Visual Studio .NET 2010, klik menu **File | New | Project.**



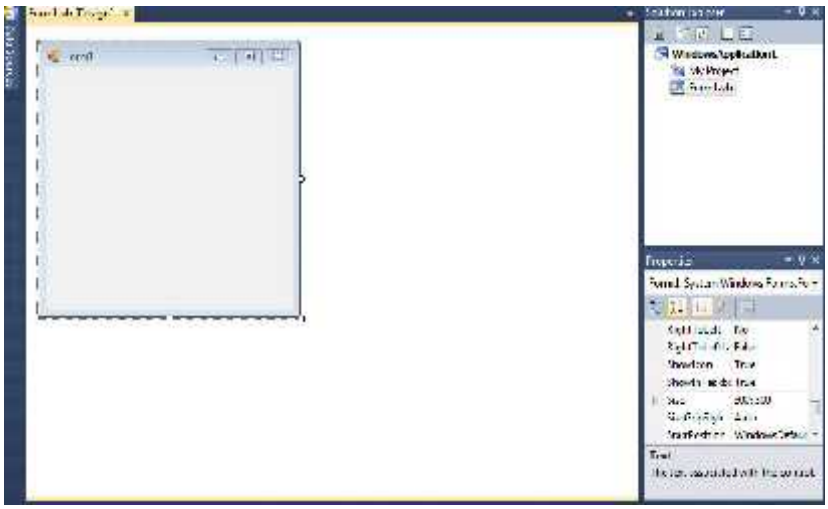
Gambar II.13. Menu New Project
Sumber : Junindar, 2010, 3

5. Setelah itu akan muncul kotak dialog **New Project**. Pada kotak dialog **New Project** terdapat beberapa pilihan tool untuk pengembangan aplikasi, seperti Visual Basic, Visual C# dan Visual C++. Pilih **Visual Basic** kemudian pilih

Windows Form Application. Beri nama project yang akan dibuat pada bagian **Name** dan direktori tempat menyimpan project pada bagian **Location**.

Gambar II.14. Kotak Dialog New Project
Sumber : Junindar, 2010, 2

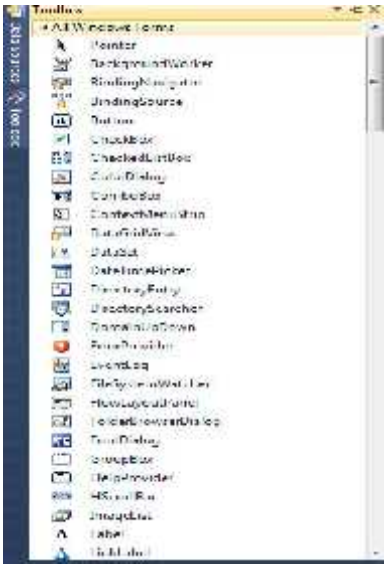
6. Selanjutnya muncul Visual Basic IDE tempat untuk membangun aplikasi Visual Basic .NET 2010.



Gambar II.15. IDE Visual Studio .NET 2010
Sumber : Junindar, 2010, 4

Pada project Visual Basic untuk Windows Application secara default telah terdapat sebuah form. Form tersebut bernama **Form1**. Pada form inilah tempat untuk meletakkan kontrol-kontrol atau komponen-komponen untuk membuat sebuah aplikasi windows. Form dan kontrol-kontrol dari program aplikasi inilah yang biasanya disebut dengan GUI (Graphical User Interface) atau antar muka dari program. Jadi user akan berinteraksi dengan sebuah program aplikasi melalui GUI. Pada IDE Visual Basic .NET 2008 **terdapat Menu, Toolbar, Toolbox, Server Explorer, Solution Explorer** dan **Properties Window.**

7. Toolbox adalah tempat dimana kontrol-kontrol dan komponen-komponen diletakkan. Kontrol dan komponen yang terdapat pada toolbox dipakai dalam pembuatan program aplikasi. Untuk menampilkan windows toolbox, klik pada tombol toolbox yang terdapat pada toolbar.



Gambar II.16. Toolbox
Sumber : Junindar, 2010, 4