BAB 2

LANDASAN TEORI

2.1 *Game*

Game adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan *refreshing*. Teori *game* pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Teori itu dikemukakan oleh John von Neumann and Oskar Morgenstern yang berisi :

"*Game* terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi."[16].

2.1.1 Pengertian *Game*

Pengertian *game* menurut beberapa ahli [6]:

1. Menurut Agustinus Nilwan dalam bukunya Pemrograman Animasi dan *Game* Profesional terbitan Elex Media Komputindo, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah

memahami pembuatan *game*. Atau jika ingin membuat *game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.

2. Menurut Clark C. Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
3. Menurut Chris Crawford, seorang komputer *game* designer mengemukakan bahwa *game*, pada intinya adalah sebuah interaktif, aktivitas yang berpusat pada sebuah pencapaian, ada pelaku aktif (player), ada pelaku pasif (NPC).

2.1.2 Klasifikasi *Game*

2.1.2.1 Berdasarkan Jenis *Platform* Yang Digunakan

1. *Arcade games,* yaitu yang sering disebut *ding-dong* di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki box atau mesin yang memang khusus di design untuk jenis *video game*s tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa masuk dan menikmati , seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya)
2. *PC Games* , yaitu *video game* yang dimainkan menggunakan Personal Komputer.

3. *Console games*, yaitu *video game*s yang dimainkan menggunakan *console* tertentu, seperti *Playstation* 2, *Playstation* 3, *XBOX* 360, dan *Nintendo Wii*.
4. *Handheld games*, yaitu yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana, contoh *Nintendo DS* dan *Sony PSP*.
5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobilephone* atau PDA.

2.1.2.2 Berdasarkan *Mode* Permainanya

1. *Casual Game Mode*

   Game dengan *mode casual* adalah *game video* yang ditargetkan pada khalayak *game*r *casual*, maksudnya adalah pengguna yang menggambarkan tipe bermain *video game* dengan waktu atau minat bermain *game* yang terbatas dibandingkan dengan *game hardcore* [7]. Oleh karena itu, *game casual* biasanya dibuat dengan aturan-aturan yang sederhana dan mudah dalam memainkannya, tujuannya adalah untuk menyajikan pengalaman *pick-up-and-play* bahwa orang-orang dari hampir semua kelompok usia atau tingkat keterampilan dapat menikmati *video game*.
2. *Hardcore Game Mode*

   Game dengan *mode hardcore* biasanya dimainkan oleh pemain yang memiliki waktu luang banyak untuk bermain *game*. *Game hardcore*

dibuat dengan aturan-aturan yang cukup banyak yang mengharuskan pemain memiliki keahlian khusus untuk memainkannya.

2.1.2.3 Berdasarkan *Genre* Permainanya

1. Aksi *Shooting*, (tembak-tembakan , atau hajar-hajaran bisa juga tusuktusukan, tergantung cerita dan tokoh di dalamnya), *video game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari *game* jenis ini adalah tembak, tembak dan tembak.
2. *Fighting* (pertarungan) Ada yang mengelompokan *video game fighting* di bagian Aksi, namun penulis berpendapat berbeda, jenis ini memang memerlukan kecepatan reflek dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya), pengenalan karakter dan timing sangatlah penting, iya, *combo*-pun menjadi esensial untuk mengalahkan lawan secepat mungkin. Berbeda seperti *game action* pada umumnya yang hanya melawan *Artificial Intellegence* atau istilah umumnya melawan komputer saja, pemain jenis *fighting game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya. *Seri Street Fighter, Tekken*, *Mortal Combat*, *Soul Calibur* dan *King of Fighter* adalah contohnya.
3. Petualangan. Bedanya dengan jenis *video game* aksi-petualangan, reflek dan kelihaian pemain dalam bergerak, berlari hingga memecut

atau menembak tidak diperlukan di sini. *Video Game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat.

4. Strategi. Kebalikan dari *video game* jenis action yang berjalan cepat dan perlu refleks secepat kilat, *video game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. *Video game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusal pelatihan tempur, tergantung dari tema ceritanya.
5. *Puzzle*. *Video game* jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula *game* jenis ini adalah juga unsur *game* dalam *video game* petualangan maupun *game* edukasi. Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman.

6. Olahraga. Singkat padat jelas, bermain sport di *PC* atau *consol* anda. Biasanya *game*nya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti NBA JAM. Contohnya pun jelas, Seri Winning Eleven, seri NBA, seri FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater, dll

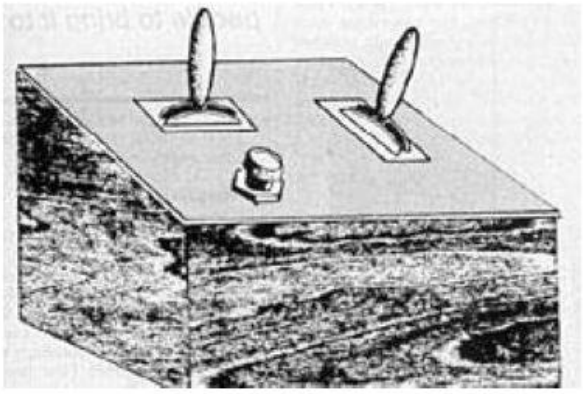
2.1.3 *Game Control*

2.1.3.1 *Game Control* Pertama

Alat *control* pertama yang dimainkan pada komputer yang tidak memiliki perangkat keras khusus yang dirancang untuk *control game* [8]. Mereka berjalan pada apapun yang tersedia di institusi dimana mereka dikembangkan (biasanya di komputer atau oloskop), dan menggunakan *switch* digital atau tombol-tombol yang melekat pada variabel resistor untuk input.

1. *Spacewar*!

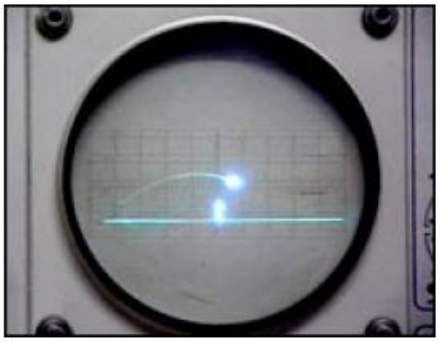
   Aksi pertama yang diketahui dalam *game* yang berbasis komputer adalah *Spacewar*!. Ini adalah contoh alat *control* pertama yang mempengaruhi pengembangan alat *control* yang baru. *Control* yang terlibat adalah 5 *switch* yang dapat mengendalikan rotasi kiri dan kanan, mendorong, menembakan misil dan *hyperspace*. Alat *control* ini terhubung ke mesin utama dengan kawat.



Gambar 2.1 Sebuah Sketsa Kotak Kontrol Untuk *Spacewar*!.

2. *Pong and The Paddle*

Secara teknis, *game* pertama adalah *Tennis for Two* pada tahun 1958 yang dikembangkan oleh Willy Niginbotham. *Control*-nya melalui sebuah tombol pada kotak *control* yang melekat di komputer dengan kawat, yang kemudian dikenal sebagai *Paddle*. Cara penggunaannya yaitu dengan memutarkan kenop yang disesuaikan dengan sudut bola pada saat dikembalikan. Ini merupakan desain *controller* yang sangat standar, hal ini disebabkan karena sumberdaya untuk pengembangan *game controller* masih terbatas. Walaupun demikian, pengembang *paddle* ini kemudian merancang kembali untuk penggunaan yang lebih baik. *Paddle* ini digunakan sebagai pengendali utama untuk *consol video game* pertama yaitu Magnavox Odyssey pada tahun 1972.


,

Gambar 2.2 *Tennis For Two Being Played*

3. *Atari and first consol multigame*

Antara awal 1970-am dan awal 1980-an puluhan *game consol* dikembangkan dengan *game-game* dan *control game* yang baru. Pada pengembangannya, pada tahun 1977 *game consol Atari* menciptakan *Atari* 2600 dengan *game control joystick* dan driving *controller*s (pengendali mengemudi). Pada tahun 1982, mereka menciptakan *Atari* 5200 dan menambahkan *game control* menggunakan *mouse, trackball, keyboard*, dan pada tahun 1986 *Atari* 7800 menambahkan *gamepads*. Selama pengembangannya, *controller* terbaik yang diciptakan oleh *Atari* adalah *joystick* yang disertakan sengan semua versi perangkat kerasnya. *Joystick Atari* memiliki dasar persegi dengan tongkat silinder tunggal mengarah ke atas. Salah satu faktor penting dalam desain *controller* adalah kemampuannya untuk digunakan dalam berbagai jenis *game*. Sebuah pistol memberikan *control* yang sempurna untuk *game* dalam kisaran target, tetapi tidak berguna untuk dipakai dalam *game* yang lainnya.



Gambar 2.3 *Atari* 2600 *Joystick*

2.1.3.2 *PC Control*

Sejajar dengan perkembangan *consol game* rumahan adalah *game PC*. *Game PC* tidak benar-benar memisahkan diri dari *consol game* hingga awal 1990-an. Pada masa itu terdapat 3 jenis *game* yang terkuat, yaitu *First-Person Shooter (FPS), Real Time Strategy (RTS)* dan Simulasi Penerbangan. *Control* pada *game* simulasi penerbangan menggunakan *joystick analogue* yang di desain mirip dengan *control* pesawat pada umumnya, sedangkan FPS dan RTS telah mengebangkan gaya *control* mereka sendiri dengan konsep unik sesuai dengan konsep *game*-nya.

2.1.3.3 3D and Analogue Control

Sony Playstation dan *Sega Saturn* merupakan *consol* 3D pertama yang dirilis. Mereka berdua menambahkan tombol digital di bahu perangkat control *game*-nya, dan menambahkan jumlah tombol dan *D-pad*.

1. *DPads* dan *Game 3D*

   Masalah utama bahwa *D-pad* tidak dapat bekerja dengan baik dalam lingkungan 3D, terutama pada perspektif orang ketiga. Dalam *game* 3D, skema *control* baru harus dikembangkan untuk mengatasi ekstra dimensi. Dalam *game*nya digunakan *control* atas dan bawah pada *D-pad* untuk bergerak maju dan mundur, dan kiri dan kanan untuk memutar karakter di tempat. Hal ini terlihat kasar dan lambat, rasanya lebih seperti mengendarai tank dari pada mengendalika sesorang. Jelas bahwa untuk memiliki *control* 3D yang baik, pemain harus mampu untuk bergerak bebas dan dalam arah yang lebih baik.

2. *Analogue Stick Control*

   Consol game pertama yang menyediakan *control analogue* pertama adalah *Nintendo* 64 (N64). *Controller* N64 memiliki desain eksentrik dan dapat melakukan 3 *event* yang berbeda (kiri dan tengah, kiri dan kanan, dan tengah dan kanan). *Control* ini dikembangkan dengan tujuan untuk menyediakan *control* 3D yang baik.



Gambar 2.6 N64 *Controller*

2.1.3.4 *Controller* Baru

Terlepas dari *controller* pada *game consol* dan puluhan skema *controller* baru yang unik, sebagian besar dirancang untuk digunakan dengan *game* tertentu.

1. *Rhythm Game*s

   Game music *game* merupakan *game* dimana pemain harus mensimulasikan beberapa aspek musik atau menari. Karena sifat unik dari *game-game* ini, mereka sering memiliki *controller* baru yang dirancang untuk *game*nya. Dalam *game*nya, pemain harus berdiri dilantai dan menekan arah panah atas, bawah, kiri dan kanan sebelum melewati batas atas layar.



Gambar 2.7 *Dance Dance Revolution Pad*

2. *Touch Screens*

   Nintendo DS adalah *consol game* genggam yang menggunakan layar sentuh sebagai salah satu *control*-nya. Interaksi dengan *touchscreen* dilakukan melalui jari, *stylus* atau *thumbpad* khusus, tergantung pada tingkat akurasi yang diperlukan.



Gambar 2.8 *Nintendo DS Lite* and *Stylus*

3. *Motion Kontrol*

   Perkembangan terbaru dalam *video game consol* komersial adalah penambahan *control* gerak. *Nintendo Wii* dan *Sony Playstation* 3 telah menambahkan fungsi ini. *Controller Wii* merupakan pengembangan radikal dari semua *controller* sebelumnya. *Controller Wii* berbentuk menyerupai remote *control* televisi. *Remote* Wii memiliki *accelerometer* didalamnya yang dapat memberikan kemampuan untuk merasakan gerakan yang diciptakan oleh pemain.



Gambar 2.9 *Nintendo Wii Remote and Nunchuk*

2.2 *Mobile Game*

Menurut Lam [3], *mobile game* merupakan *game* yang tidak hanya dapat dimainkan melalui telepon seluler, namun dapat dikembangkan dalam berbagai

macam *mobile handset* seperti PDA, *symbian* OS dan *Microsoft Smartphone*. *Mobile game* dimainkan dengan menggunakan teknologi yang ada di dalam alat itu sendiri.

2.2.1 Sejarah Singkat *Mobile Game*

Pada awalnya, *game* pada telepon seluler hanya bersifat sebagai fitur tambahan, namun pada kenyataannya banyak konsumen telepon seluler yang menyambut baik fitu tambahan ini. Kini, telepon seluler tidak hanya berfungsi sebagai alat komunikasi tapi juga berfungsi untuk memberikan hiburan bagi para penggunanya.

Sejarah *mobile game* dimulai pada tahun 1997-an dimana *Nokia* merilis tipe *game* baru yang dapat dimainkan dengan menggunakan telepon seluler. *Game* pertama yang sangat popular adalah "*Snake*". Pada tahun 1999-an, muncul lagi trend baru *mobile game* dengan *game* berbasis WAP dan *game* berbasis SMS. Di era ini, diririslah *game I-Mode*, layanan internet *wireless*, yang begitu popular khususnya di Jepang dan Negara lain. Karena sudah adanya dukungan jaringan internet, kini telepon selulerpun sudah mampu digunakan untuk bermain *game* secara *multiplayer*. Di tahun 2001, dengan adanya perkembangan spesifikasi dari perangkat telepon seluler, *game* telah dimainkan pada layar berwarna dengan dukungan grafik yang lebih baik dan sudah mampu untuk di-*download*. Pada masa itu, pengguna telepon seluler mulai berlomba-lomba membeli dan menyimpan berbagai hiburan ke dalam telepon seluler. Telepon seluler-pun berkembang ke arah menggunakan *microcomputer*, sehingga

dukungan grafik dan suara yang baik dapat digunakan pada *mobile game*. Pada tahun 2003, nokia memperkenalkan *platform N-Gage* dan *game* online dengan *platform* tersebut. Hingga saat ini, teleppon seluler telah banyak yang mendukung grafik 3D, sehingga resolusi tampilan yang disuguhkan menjadi lebih baik dan memori yang memadai. Berbagai *platform* pemrograman *game* seperti *J2ME, Symbian, Windows Mobile*, kini sudah didukung dengan pustaka yang semakin lengkap.

2.2.2 Kelebihan *Mobile Game*

Menurut nugraha (2010) dengan artikel berjudul : *Mobile Video Game*s di *iPhone* vs *Android*" :

> *"Memang video games hanya salah satu fitur yang bias kita manfaatkan di smartphone dan kita sendiri kalaupun mau nge-game serius lebih baik menggunakan consol game, namun eksistensi video games di smartphone tidak bias diremehkan. Apalagi dengan banyak munculnya games yang juga berbasis social networking. Semua game yang kita lakukan bias kita share atupun kita interaksikan dengan social networking. Saya rasa ini merupakan nilai tambah dari mobile games saat ini (selain kepraktisan dan juga nilai portabilitasnya yang tidak dimiliki oleh consol game)."*

2.3 *Android*

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* yang terbuka bagi para pengembang untuk mencipyakan aplikasi mereka. Awalnya, *Google Inc*. Membeli *Android Inc*, yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*, kemudian untuk mengembangkan *Android*, dibentuklah *Open Headset Alliance*, konsursium

dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk *Google*, *HTC*, *Intel*, *Motorola*, *Qualcomm*, T-*Mobile*, dan *Nvidia*.[9]

2.3.1 Sejarah *Android*

Pada saat perilisan perdana *Android*, 5 Noveber 2007, *Android* bersama *Open Headset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Dilain pihak, *Google* merilis kode-kode *Android* dibawah lisensi *Apace*, sebuah lisensi perangkat lunak dan open *platform* perangkat seluler. Didunia ini terdapat dua jenis distributor sistem operasi *Android*. Pertama yang mendapat dukungan penuh dari *google* atau *Google mail services* (GMS) dan kedua adalah benar-benar bebas distribusinya tanpa dukungan langsung *Google* atau dikenal sebagai *Open handset Distribution* (OHD) [9].

Sektar september 2007 *Google* mengenalkan *Nexus One*, salah satu jenis *Smartphone* yang menggunakan *Android* sebagai sistem operasinya. Telepon seluler ini diproduksi oleh HTC *Comporation* dan tersedia di pasaran pada 5 januari 2008. Pada 9 desember 2008, diumumkan anggota baru yang bergabung dalam program keja *Andoid ARM Holdings*, *Atheros Commucation*, diproduksi oleh *Asustek Computer Inc*, *Garmin Ltd*, *Softbank*, *Soni Ericsson*, *Toshiba Corp*, dan *Vodaphone Group Plc*. Seiring pembentukan open *Handset Alliance*, perangkat mobile yang merupakan modivikasi *kernel Linux* 2.6. sejak *Android* di rilis telah dilakukan berbagai pembaruan berupa bug dan penambahan fitur.

2.3.2 *The Dalvik Virtual Machine* (*DVM*)

Salah satu elemen kunci dari *Android* adalah *DVM*. *Android* bejalan didalam *DVM* bukan di *Java Vurtual Machine* (*JVM*), sebenarnya banyak persamaan dengan *JVM* seperti *JME* (*Java Mobile* Edition), tetapi *android* mengunakan *Virtual Machine* sendiri yang dikostumisasi dan dirancang untuk memastikan bahwa beberapa fitur-fitur berjalan lebih efisien di perangkat *mobile*.

Semua *hardware* yang berbasis *android* dijalankan dengan menggunakan *Virtual Machine* untuk eksekusi aplikasi, pengembang tidak perlu kewatir tentang implementasi perangkat keras tertentu. *DVM* mengeksekusi *Executable File*, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The Executable File* diciptakan dengan mengubah kelas bahasa *java* dan dikompilasi menggunakan *tools* yang disediakan dalam *SDK Android*.

2.3.3 *Android SDK* (*Software Development Kit*)

Android SDK adalah *tools API* (*Application Programing Interface*) yang diperlukan untuk mulai mengembangkan aplikasi *platform Android* menggunakan bahasa pemrograman *Java*. *Android* merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di *release* oleh *Google*. Saat ini disediakan *Android SDK* sebagai alat bantu dan *API* untuk mulai mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *java*. Sebagai *platform* aplikasi netral, *Android* memberi anda kesempatan untuk membuat aplikasi yang kita butuhkan yan bukan

merupakan aplikasi bawaan *Handphone/smartphone*. Beberapa fitur-fitur *Android* yang paling penting adalah:

1. *Framework* aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mensin *Virtual Dalvik* dioptimalkan untuk perangkat *mobile*.
3. *Integrated browser* berdasarkan *engine open source WebKit*.
4. Grafis yang dioptimalkan dan didukung oleh *libraries grafis 2D,* grafis *3D* berdasarkan spesifikasi *opengl* ES 1,0 (*Operasional akselerasi hardware*).
5. *SQLite* untuk penyimpanan data (database).
6. Media *Support* yang mendukung *audio, video,* dan gambar.
7. *Bluetooth, EDGE, 3G,* dan *Wifi*
8. Kamera, *GPS*, kompas, dan *accelerometer*.

2.3.4 *Eclipse*

Eclipse merupakan komunitas *open source* yang bertujuan menghasilkan *platform* pemrograman terbuka. *Eclipse* terdiri dari *framework* yang dapat dikembangkan lebih lanjut, peralatan bantu untuk membuat dan memanage *software* sejak awal hingga diluncurkan. *Platform Eclipse* didukung oleh ekosistem besar yang terdiri dari vendor tekonologi, start-up inovatif, universitas, riset institusi serta individu. Banyak orang mengenal *Eclipse* sebagai *IDE* (*integrated development environment*) untuk bahasa *Java*, tapi *Eclipse* lebih dari sekedar *IDE* untuk *Java*.

Komunitas *Eclipse* memiliki lebih dari 60 proyek *open source*. Proyek-proyek ini secara konsep terbagi menjadi 7 kategori [10]:

1. *Enterprise Development*
2. *Embedded and Device Development*
3. *Rich Client Platform*
4. *Rich Internet Applications*
5. *Application Framework*s
6. *Application Lifecycle Management* (*ALM*)
7. *Service Oriented Architecture* (*SOA*)

2.3.5 Memulai *Java* dengan *Eclipse*

Program komputer adalah rangkaian instruksi yang diberikan agar komputer dapat bekerja. Suatu pekerjaan yang mungkin sederhana bagi manusia tidak dapat dimengerti oleh komputer. Sintaks bahasa pemrograman ditentukan berdasarkan apa yang bisa dilakukan oleh komputer, misalnya *loop* (perulangan), cabang (branch), atau fungsi. Hanya program dengan sintaks yang benar yang dapat dikompilasi atau diinterpretasi yang pada akhirnya bisa dijalankan di komputer. *Compiler* akan memberikan pesan kesalahan apabila ada kesalahan dalam sintaks sehingga kita memperbaikinya. Secara umum, aplikasi memiliki 4 komponen, yaitu [10]:

1. Definisi paket (*package*)

   Package dalam *Java* merupakan kumpulan dari berbagai kode yang terangkum dalam satu paket. Untuk memudahkan penulisan dan

pembagian logika suatu program, satu paket terbagi menjadi beberapa berkas (*file*) di mana setiap *file* memiliki fungsi atau tugas yang sangat khusus, misalnya satu *file* berfungsi untuk mendeklarasikan konstanta dan kelas, sementara *file* yang lain berisi implementasi kelas dan prosedurnya. Pada contoh aplikasi SelamatDatang di atas, paket ini hanya berisi satu buah *file* yang isinya terdiri dari satu kelas dan satu metode. Definisi paket tidak selalu diperlukan, tetapi hal ini merupakan kebiasaan baik untuk melatih kita berfikir secara logis dan sistematis.

2. Komentar

   Komentar tidak akan diproses oleh kompiler tetapi berguna bagi programmer lain. Bahasa *Java* memiliki 3 jenis komentar :

   1. /* *text* */ — Compiler akan mengabaikan kata kata antara /* dan */
   2. /** documentation */ — Ini merupakan komentar yang dipergunakan khusus untuk dokumentasi. Kompiler akan mengabaikan komentar dari /* hingga */. Alat bantu *javadoc* akan memproses komentar dokumentasi untuk membuat dokumentasi secara otomatis dari sumber program.
   3. // text — Kompiler akan mengabaikan segala sesuatu dari // hingga akhir baris

3. Definisi kelas (*class*)

   Kelas merupakan bagian integral dari bahasa *Java* karena *Java* merupakan bahasa berorientasi objek. Setiap aplikasi harus terdiri dari satu kelas. Di sini kita definisikan kelas SelamatDatang sebagai kelas utama.

public class SelamatDatang {

...

}

4. Metode main

Dalam bahasa pemrograman *Java*, setiap aplikasi harus memiliki satu buah metode main yang bentuknya seperti berikut :

public static void main(String[] args) {

...

}

2.4 Latar Belakang Karakter

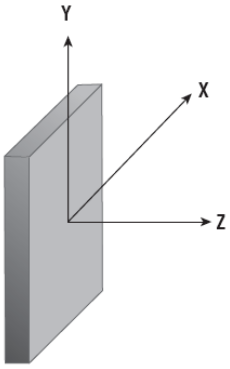
Sebagai anak bangsa kadang kita tidak mengetahui kebudayaan sendiri, sementara orang luar negeri malah tertarik dengan kebudayaan Indonesia yang unik, menarik dan khas. Bahkan sebagian budaya kita diklaim oleh Negara lain mulai dari reog ponorogo, batik, rendang, hingga lagu rasa sayange. Banyak sekali kebudayaan yang sangat unik dan menarik di Indonesia, salah satunya cerita rakyat. Ratusan cerita rakyat yang didongengkan secara turun temurun semakin dilupakan karena tergeser oleh film dan dongeng dari Barat yang diakses melalui televise. “padahal cerita-cerita rakyat yang merupakan salah satu bentuk kearifan lokal penuh dengan nasihat-nasihat dan mendidik yang bermanfaat untuk generasi muda kita,” kata peneliti Balai Bahasa Medan (BBM) Agus Bambang Hermanto di Medan, Sumatra Utara, Rabu (10/6). Lutung Kasarung adalah salah satu cerita

rakyat dari Jawa Barat yang kini sudah hampir terlupakan. Padahal, cerita rakyat lutung kasarung mengandung filosofi lokal dengan nilai-nilai moral di dalamnya. Ada beberapa nilai moral yang dapat di petik, yakni dampak negatif atas perilaku memandang rendah orang lain, keutamaan sifat pemaaf dan tidak pendendam, dan pekerja keras untuk bangkit dari keterpurukan.

2.5 *Accelerometer* dan *Procedural Content Generation* (PCG)

2.5.1 *Accelerometer*

Accelerometer, seperti namanya, digunakan untuk mengukur percepatan. Percepatan didefinisikan sebagai laju perubahan kecepatan, sehingga dapat mengukur seberapa cepat akselerasi pada perangkat berubah dalam arah tertentu. Dengan menggunakan *accelerometer*, anda dapat mendeteksi gerakan, laju perubahan dari kecepatan gerakan yang terjadi pada perangkat. Sangat penting untuk dicatat bahwa *accelerometer* tidak mengukur kecepatan, sehingga anda tidak bisa mengukur kecepatan secara langsung berdasarkan pembacaan *accelerometer* tunggal. Sebaliknya, anda perlu untuk mengukur perubahan dalam percepatan dari waktu ke waktu [11]. Percepatan dapat diukur oleh tiga sumbu arah : *forward-backward* (*longitudinal*), *left-right* (literal), dan *up-down* (*vertical*). Sensor manager melaporkan perubahan sensor di semua tiga sumbu arah. Seperti yang di ilustrasikan pada gambar di bawah ini :



Gambar 2.10 Tiga Sumbu Arah *Accelerometer*.[11]

1. *Vertical* (*up-down*), dimana arah positif berarti ke atas gerakan perangkat yang sedang di tinggikan dan sebaliknya adalah nilai positif
2. *Longitudinal* (*forward-backward*), dimana gerakan kedepan yang menandakan percepatan positif dan gerakan kebelakang yang menandakan percepatan negatif.
3. *Sideways lateral* (*left-right*), dimana gerakan kearah kanan dari perangkat yang mewakili nilai positif dan gerakan perangkat kea rah kiri yang mewakili nilai negatif.

Sensor *manager* mempertimbangkan perangkat "diam" ketika menghadap ke atas pada permukaan yang datar dalam orientasi *potret*. Seperti dijelaskan sebelumnya, kita dapat memantau perubahan percepatan dengan menggunakan sensor *listener*. Untuk memanggil kelas *SensorListener* dari Sensor *Manager* yaitu dengan menggunakan *code SENSOR_ACCELEROMETER* yang nantinya dapat meminta pembaruan nilai-nilai *accelerometer* dan tingkat update sensor seperti potongan kode berikut :

SensorManager=

(SensorManager)getSistemService(Context.SENSOR_SERVICE);

```
sm.registerListener(mySensorListener,
SensorManager.SENSOR_ACCELEROMETER,
SensorManager.SENSOR_DELAY_UI);
```

Sensor *Listener* yang kita buat harus menerapkan *onSensorChanged method* yang akan dipicu ketika perubahan terjadi dalam percepatan disepanjang salah satu dari tiga sumbu yang telah dijelaskan sebelumnya. *onSensorChanged method* menerima *array float* yang berisi nilai percepatan di ketiga sumbu dalam format baku. Sensor *Manager* mencakup konstanta indeks yang dapat kita gunakan untuk mengambil nilai percepatan yang dibutuhkan, seperti terlihat pada potongan kode berikut :

```
SensorListener mySensorListener = new SensorListener() {
public void onSensorChanged(int sensor, float[] values) {
if (sensor == SensorManager.SENSOR_ACCELEROMETER) {
float xAxis = values[SensorManager.DATA_X];
float yAxis = values[SensorManager.DATA_Y];
float zAxis = values[SensorManager.DATA_Z];
float raw_xAxis = values[SensorManager.RAW_DATA_X];
float raw_yAxis = values[SensorManager.RAW_DATA_Y];
float raw_zAxis = values[SensorManager.RAW_DATA_Z];
// TODO apply the acceleration changes to your application.
}
}
public void onAccuracyChanged(int sensor, int accuracy) { }
```

};

2.5.2 *Procedural Content Generation* (*PCG*)

Procedural content generation (*PCG*) adalah metode yang digunakan untuk penempatan objek-objek yang diperlukan dalam pembentukan dunia *game* (*platform,* batu-batuan, pohon dan sebagainya). Konten *game* merupakan factor penting dalam menjaga pemain agar tidak cepat merasa bosan dan tetap terlibat dalam dunia *game*. Namun permintaan para pelaku *game* untuk pembuatan konten baru semakin meningkat, sementara produksi konten manual oleh desainer *game* semakin mahal [2]. Berbeda dengan produksi konten secara manual, *procedural content generation* dapat mengkombinasikan semua konten sederhana yang diperlukan dalam sebuah dunia *game* dengan menempatkan konten-kontennya secara acak. Jadi apabila ingin menciptakan kebun yang indah, kita hanya membutuhkan satu konten pohon dan satu konten lampu taman.

Saat ini, produksi berkualitas tinggi *game* komersial memerlukan pekerjaan beberapa ratus orang, termasuk seniman, desainer, *programmer*, dan insinyur *audio,* dan banyak dari mereka menghasilkan conten *game* yang baru. Akibatnya, produksi konten telah tumbuh ke titik dimana ia telah menjadi hambatan kedua anggaran setelah pemasaran [14]. Teknik *procedural* merupakan alternatif untuk membuat dunia *game* dalam waktu terbatas tanpa menempatkan beban yang besar pada *game* desainer konten. Gagasan utama dari *Procedural Conten Generation* ini adalah bahwa konten *game* tidak ditempatkan secara manual oleh desiner manusia, tetapi oleh komputer dengan menjalankan prosedur

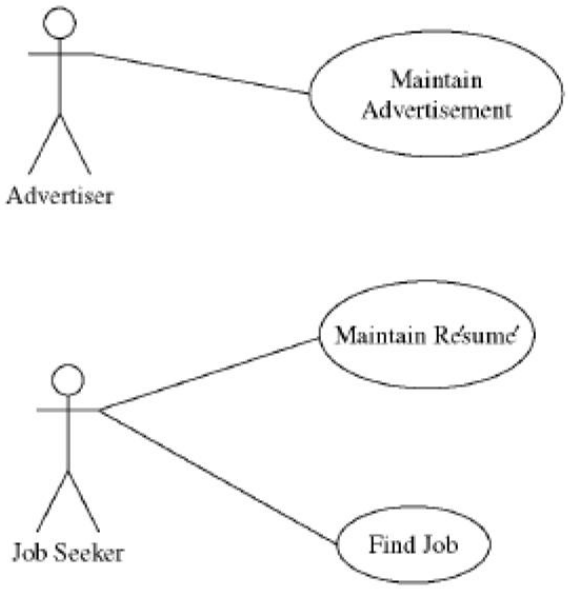
yang jelas tentunya. Untuk menghindari kehilangan *control* atas proses desain yang di inginkan, desainer masih dapat mempengaruhi hasil akhir dengan menyesuaikan parameter yang diperlukan dari prosedur.

2.6 UML (Unified *Mode*lling *Language*)

*Unified Mode*ling *Language* (UML) adalah keluarga notasi grafis yang didukung oleh meta-*mode*l tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemorgraman berorientasi objek (OO) [5].

2.6.1 *Use Case Diagram*

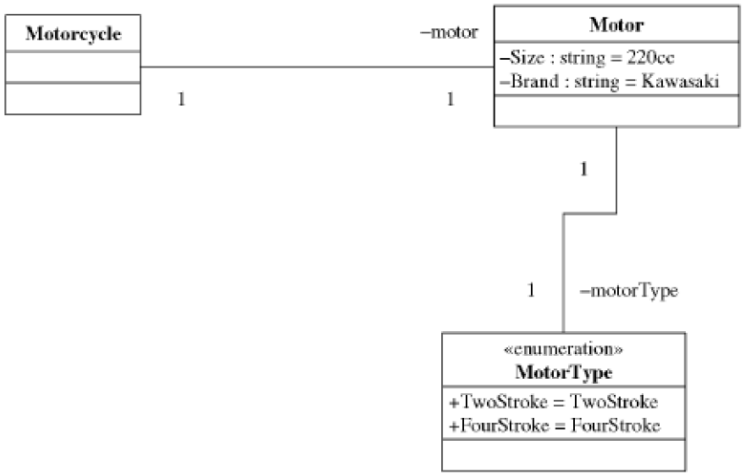
Use case diagram menggambarkan tujuan penting fungsionalitas *sistem* yang akan ditampilkan pada user. Sebua use case merepresentasikan sebuah interaksi antara actor dengan sistem. *Use case* diagram dapat membantu dalam penyusunan *requirement* sebuah *sistem*, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua fitur yang ada pada sistem [12].



Gambar 2.11 Contoh *Use Case* Diagram [12]

2.6.2 Class Diagram

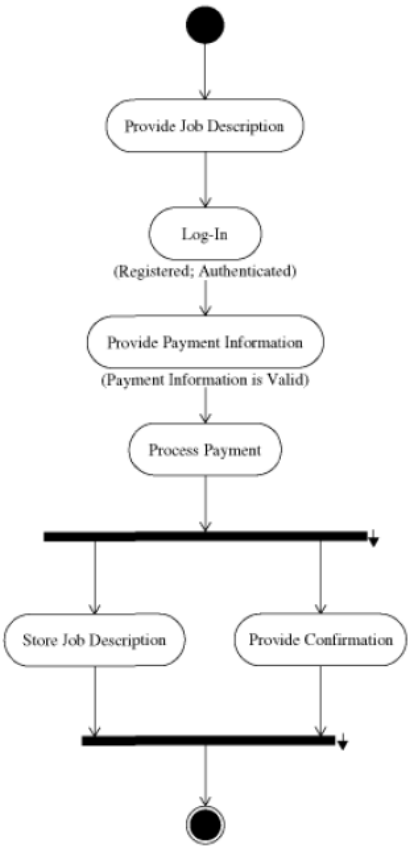
Class diagram digunakan untuk memperlihatkan *class* pada sistem dan hubungan antara *class*. *Class* diagram tidak menjelaskan fungsionalitas atau bagaimana *instances class* berinteraksi. *Class* menggambarkan keadaan (*atribut*/*property*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (*method*/fungsi) [12].



Gambar 2.12 Contoh *Class Diagram* [12]

2.6.3 Activity Diagram

Activity diagram digunakan untuk menggambarkan aliran aktivitas baik dalam proses bisnis maupun *use case*. Diagram ini juga dapat digunakan untuk memodelkan aksi yang akan dilakukan saat sebuah operasi dieksekusi dan memodelkan hasil dari aksi tersebut [12].

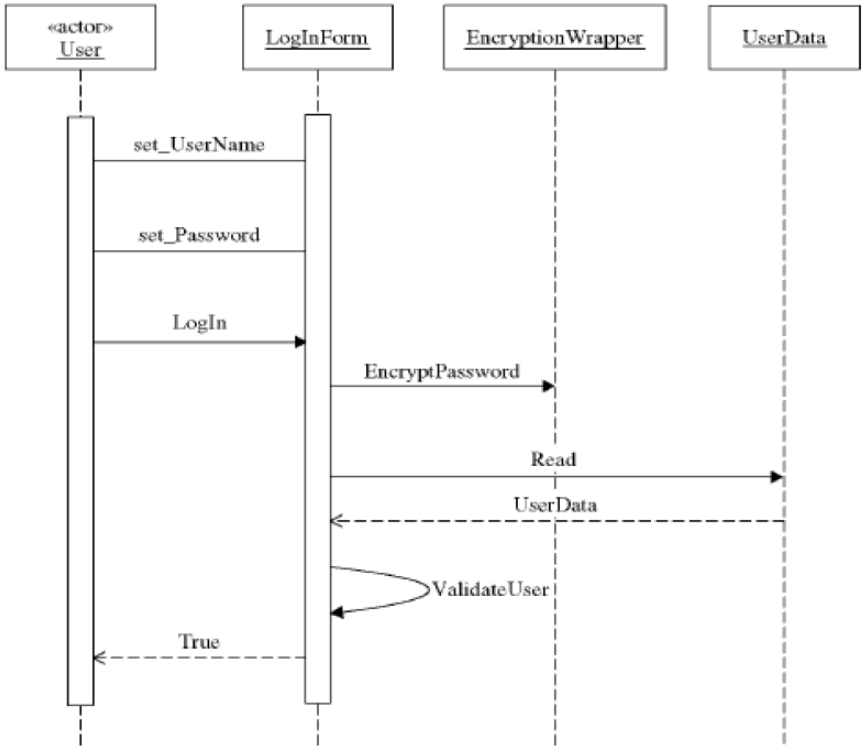


Gambar 2.13 Contoh *Activity Diagram* [12]

2.6.4 Sequence Diagram

Sequence diagram biasanya digunakan untuk menggambarkan skenario atau rangkaian lengkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan *output* tertentu. Diawali dari apa yang mend-*trigger* aktivitas

tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk *actor*, memiliki *lifeline vertical*. *Message* digambarkan sebagai garis berpanah dan satu objek ke objek lainnya. Pada fase desain berikutnya *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message* [12].



Gambar 2.14 Contoh *Sequence Diagram* [12]

2.7 Skala Pengukuran

Skala pengukuran merupakan kesepakatan yang digunakan sebagai acuan untuk menentukan panjang pendeknya interval yang ada dalam alat ukur, sehingga alat ukur tersebut bila digunakan dalam pengukuran akan menghasilkan data kuantitatif [3]. Dengan skala pengukuran ini, maka nilai variabel yang diukur 10

dengan instrumen tertentu dapat dinyatakan dalam bentuk angka, sehingga akan lebih akurat, efisien, dan komunikatif.

1. Skala *Likert*

Skala *Likert* digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Dengan skala *Likert*, maka variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item–item instrumen yang dapat berupa pernyataan atau pertanyaan. Jawaban setiap item instrumen yang menggunakan skala likert mempunyai gradasi dari sangat positif sampai sangat negatif, yang dapat berupa kata–kata antara lain: sangat setuju dengan skor 5, setuju dengan skor 4, ragu–ragu dengan skor 3, tidak setuju dengan skor 2, sangat tidak setuju dengan skor1.

Instrument peneletian yang menggunakan skla likert dapat dibuat dalam bentuk *checklist* ataupun pilihan ganda.

Contoh :

Berilah jawaban pertanyaan berikut sesuai dengan pernyataan anda, dengan cara memberi tanda (√) pada kolom yang tersedia.

Tabel 2.1 Tabel pertanyaan skala likert

No.	Pertanyaan	Jawaban				
		SS	ST	RG	TS	STS
1.	Prosedur kerja yang baru itu akan segera		√			

2.	diterapkan di perusahaan anda.					

SS = Sangat Setuju diberi skor 5
ST = Setuju diberi skor 4
RG = Ragu-ragu diberi skor 3
TS = Tidak Setuju diberi skor 2
STS = Sangat Tidak Setuju diberi skor 1

Kemudian dengan teknik pengumpulan data angket, maka instrument tersebut misalnya diberikan kepada 100 orang karyawan yang diambil secara random. Dari 100 orang pegawai setelah dilakukan analis misalnya:

25 Orang menjawab SS
40 Orang menjawab ST
5 Orang menjawab RG
20 Orang menjawab TS
10 Orang menjawab STS

Berdasarkan data tersebut 65 orang (40+25) atau 65% karyawan menjawab setuju dan sangat setuju. Jadi kesimpulannya mayoritas karyawan setuju adanya metode kerja baru.

Data interval tersebut juga dapat dianalisis dengan menghitung rata-rata jawaban berdasarkan skoring setiap jawaban dari responden. Berdasarkan skor yang telah ditetapkan dapat dihitung sebagai berikut :

Jumlah skor untuk 25 orang yang menjawab SS = 25 x 5= 125
Jumlah skor untuk 40 orang yang menjawab ST = 40 x 4= 160
Jumlah skor untuk 5 orang yang menjawab RG = 5 x 3 = 15

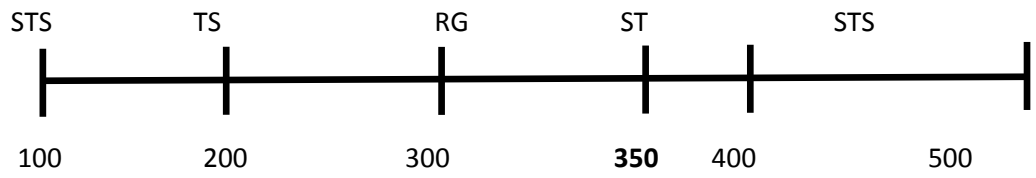
Jumlah skor untuk 20 orang yang menjawab TS = 20 x 2= 40

Jumlah skor untuk 10 orang yang menjawab STS = 10 x 1= 10

Jumlah total = 350

Jumlah skor ideal (kriterium) untuk seluruh item = 5 x 100 = 500 (seandainya semua menjawab SS). Jumlah skor yang diperoleh dari penelitian 350. Jadi berdasarkan data tu maka tingkat persetujuan terhadap metode kerja baru itu = (350 : 500) x 100% = 70% dari yang diharapkan (100%)

Secara kontinum dapat digambarkan seperti berikut :



Gambar 2.15 Gambar interpretasi skor persetujuan metode kerja baru

2. Skala *Guttman*

Skala pengukuran dengan tipe ini, akan didapat jawaban yang tegas, yaitu ”ya–tidak”, ”benar–salah”, ”pernah–tidak pernah”,”positif–negatif” dan lain–lain. Data yang diperoleh dapat berupa data interval atau rasio 11 dikhotomi (dua alternatif). Jadi kalau pada skala *likert* terdapat 3, 4, 5, 6, 7 interval, dari kata “sangat setuju” sampai “sangat tidak setuju”, maka pada skala Guttman hanya ada dua interval yaitu ”setuju” atau ”tidak setuju”.

Penelitian menggunakan skala Guttman dilakukan bila ingin mendapatkan jawaban yang tegas terhadap suatu permasalahan yang ditanyakan.

Contoh :

1. Bagaimana pendapat anda, bila orang itu menjabat pimpinan di perusahaan ini ?
   a. Setuju
   b. Tidak setuju
2. Pernahkah pimpinan melakukan pemeriksaan di ruang kerja anda ?
   a. Tidak pernah
   b. Pernah

3. Rating Scale

Dari ketiga skala pengukuran seperti yang telah dikemukan, data yang diperoleh semuanya adalah data kualitatif yang kemudian dikuantitatifkan. Tetapi dengan *rating scale* data mentah yang diperoleh berupa angka kemudian ditafsirkan dalam pengertian kualitatif. Yang penting bagi penyusun instrumen dengan *rating scale* adalah harus dapat mengartikan setiap angka yang diberikan pada alternatif jawaban pada setiap item instrumen.