BAB 2
LANDASAN TEORI

2.1. Game

Game merupakan suatu permainan yang dapat dilakukan bail secara individu maupun berkelompok untuk menyelesaikan misi pada *game* tersebut. *Game* biasanya digunakan sebagai media hiburan, media edukasi dan untuk mengisi waktu luang.

2.1.1. Pengertian *Game*

Game merupakan salah satu media hiburan yang paling populer untuk semua kalangan usia. *Game* dapat dimainkan dengan aturan tertentu sehigga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan *refreshing*. Pengertian *game* menurut beberapa ahli:

1. Menurut David Parlett, *Game* adalah sesuatu yang memiliki "akhir dan cara mencapainya": artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya
2. Menurut Roger Caillois seorang sosiolog Perancis, dalam bukunya yang berjudul Les jeux et les hommes menyatakan game adalah aktivitas yang mencakup karakteristik berikut:
   a. *Fun* (bebas bermain adalah pilihan bukan kewajiban).
   b. *Separate* (terpisah).
   c. *Uncertain, non-productive, governed by rules* (ada aturan).
   d. *Fictitious* (pura-pura).
3. Menurut Clark C.Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan)
4. 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].

Jadi *game* adalah kegiatan yang memiliki tujuan untuk dicapai dan dibatasi dengan peraturan bagi para pemainnya yang bersifat menyenangkan dan untuk membuat *game* maka haruslah memahami teknik dan metode animasi. *Game* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut untuk dilesaikan dengan cepat dan tepat.

2.1.2. Klasifikasi *Game*

Game bisa diklasifikasikan kedalam beberapa hal, diantaranya klasifikasi berdasarkan *platform*, berdasarkan *genre* dan berdasarkan sudut pandang permainan (*game view point*).

2.1.2.1. Berdasarkan Platform

Tiga kategori utama platform pada *game* adalah *personal computer (PC)*,
console, dan *mobile* [3].

1. *Personal Computer (PC)*

   PC game adalah *game* yang dibuat untuk computer baik berbasis Windows, Mac, ataupun Linux. *PC* menyediakan kekuatan grafis dan pemrosesan yang kuat yang memungkinkan pengembang untuk membuat game yang mutakhir. Tetapi kelemahan game *PC* adalah mahal karena pengguna harus menggunakan *hardware* yang *up-to-date* untuk memainkan game PC dengan baik. Selain itu, kelemahan game PC bagi pengembang adalah banyaknya konfigurasi PC membuat sulit bagi pengembang untuk memastikan game tersebut berjalan dengan benar pada semua setup PC.

2. *Console*

   Console adalah hardware yang dibuat oleh pihak ketiga seperti Sony, Microsoft, dan Nintendo. *Console* terhubung ke televisi dan tujuan utamanya adalah untuk bermain *game*. *Game Console* sangat menarik bagi *game*

pengembang karena mereka hanya perlu memikirkan satu konfigurasi *hardware* ketika membuat *software* untuk konsol. Sangat kontras dengan PC yang memiliki opsi konfigurasi yang tak terbatas.

3. Mobile

Mobile platform terdiri atas sesuatu yang *portable* dan bisa digenggam, termasuk ponsel, PDA, iPods, dan handheld game seperti Nintendo DSi atau Sony PSP. *Game mobile* memiliki kontrol yang sederhana jika dibandingkan dengan PC.

2.1.2.2. Berdasarkan *Genre*

Menurut buku "*Fundamentals of Game Design*", *genre game* bisa dibagi menjadi 9, yaitu [4]:

1. *Action*

Action game adalah *game* dimana kebanyakan dari tantangan yang disajikan merupakan dari *tes physical skill* dan koordinasi pemain. Salah satu *sub-genre action game* adalah *shooters game*, baik yang 2D maupun 3D seperti *First Person Shooters (FPS).*

2. *Strategy*

Strategy game menantang pemain untuk mencapai kemenangan dengan perencanaan, khususnya melalui perencanaan serangkaian tindakan yang dilakukan melawan satu lawan atau lebih. Kemenangan diraih dengan perencanaan matang dan pengambilan keputusan yang optimal.

3. *Role Playing Game (RPG)*

RPG adalah *game* dimana pemain mengontrol satu atau lebih karakter yang biasanya di desain oleh pemain itu sendiri, dan memandu mereka melewati berbagai rintangan yang diatur oleh komputer. Perkembangan karakter dalam hal kekuatan dan kemampuannya adalah kunci dari game jenis ini.

4. Sports

Sports game mensimulasikan berbagai aspek dari olahraga atletik nyata maupun imajiner, apakah itu memainkan pertandingan, me-manage tim dan karir, atau keduanya. Salah satu contoh game jenis ini adalah *Pro Evolution Soccer*

2012 (PES 2012), dimana pemain bisa memainkan pertandingan, menjadi manajer tim, maupun menjadi pemain dan mengembangkan karirnya sendiri.

5. *Vechicle Simulation*

Vechicle simulation membuat *feeling* mengendarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. *Performa* dan karakteristik kasar mesin harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner.

6. *Construction and Management Simulation*

CMS game adalah *game* tentang proses. Tujuan pemain bukan untuk mengalahkan musuh, tetapi membangun sesuatu dengan konteks proses yang sedang berjalan. Semakin pemain mengerti dan mengontrol proses, semakin sukses sesuatu yang ia bangun. *Game* seperti ini biasanya menyediakan dua jenis permainan, yaitu mode bebas dimana pemain bebas membangun sesuatu,dan mode misi dimana terdapat skenario hal apa yang harus dibangun oleh pemain.

7. *Adventure/ Platform*

Sebuah game petualangan bukanlah suatu kompetisi atau simulasi. Sebuah game petualangan juga tidak menawarkan proses untuk dikelola atau lawan harus dikalahkan melalui strategi dan taktik. Sebaliknya, sebuah game petualangan adalah sebuah cerita interaktif tentang karakter yang dikendalikan oleh pemain. Penyampaian cerita dan eksplorasi adalah elemen inti dari game ini. Penyelesaian teka-teki dan tantangan konseptual adalah bagian besar dari permainan. Nyaris seluruh *game* dengan *genre* ini adalah permainan *single player* karena lebih menekankan dari segi sudut cerita dan karakternya, membuat desain *multi player* untuk *game* petualangan ini sulit dibuat. *Game* dengan *genre* ini menawarkan petualangan yang panjang, dan sering kali sejumlah *mini-games* ada di dalam sebuah *game adventure*. *Game genre* ini akan menempatkan kamu pada seorang tokoh yang harus menjalani kisahnya sendiri, dan biasanya harus menempuh waktu bermain lama untuk menjalani `cerita` yang ada di dalam permainan ini. Seperti Harry Potter, Super Mario, dll.

8. *Artificial Life and Puzzle Game*

Artificial Life game adalah *game* yang membuat tiruan dari kehidupan sebenarnya. Biasanya ada dua jenis game ini, tiruan kehidupan manusia, contohnya The SIMS, dan tiruan kehidupan binatang, contohnya Tamagochi.

9. *Online Game*

Istilah *online game* disini mengacu kepada *multiplayer game* dimana mesin dari para pemain terhubung dengan jaringan.

2.1.2.3 Berdasarkan Sudut Pandang Pemain

Sebuah game biasanya mempunyai sudut pandang permainan tersendiri disesuaikan berdasarkan genre game yang diambil. Berikut beberapa macam sudut pandang permainan yang biasa digunakan [4]:

1. *Side Scrolling*

   Sudut pandang permainan yang terlihat dari samping dan memungkinkan karakter utama untuk bergerak dari kiri ke kanan serta memungkinka background pada game seolah-olah bergeser mengikuti pergerakan karakter utama.

2. *Top Down*

   Sudut pandang permainan yang memungkinkan karakter utama bermanuver ke empat arah namun cara permainannya sendiri bergeser dari bawah ke atas, dan biasanya game yang menggunakan sudut pandang permainan jenis ini adalah *shooter game*.

3. *Isometric*

   Sudut pandang permainan yang memungkinkan permainan terlihat diantara sisi *Side Scrolling* dan juga *Top Down*, dan biasanya diterapkan pada *game* dengan *genre RTS (Real Time Strategy).*

2.2. Edukasi

Edukasi adalah sesuatu yang bersifat menyeluruh. Edukasi tidak hanya berupa pelajaran ataupun diktat kuliah. Edukasi atau pendidikan memiliki makna yang lebih dalam daripada hanya sekedar mengajari atau mendidik. Mendidik berarti membentuk karakter dan pola pikir seseorang. Dalam hal ini tentunya ke

arah yang baik. Ketika seseorang meniru perilaku sesuatu atau seseorang, ini juga merupakan bagian dari pendidikan. Maka daripada itu, sebenarnya implementasi pendidikan dalam *game* tidak hanya mengacu pada pelajaran semata. Ia harus mencakup semua tatanan pendidikan,meski dengan spesifikasi tertentu.

Penggunaan *game* sebagai sarana pendidikan sebetulnya bukan merupakan hal yang tabu ataupun salah. Karena *game* bersifat "*entertain*" atau menghibur, yang disini adalah bermain. Psikologi manusia adalah lebih suka bermain daripada belajar serius. Dalam *game*, pendidikan diberikan lewat praktek atau pembelajaran dengan praktek (*learning by doing*) [5].

2.2.1. *Game Education*

Penerapan *education game* bermula dari perkembangan video game yang sangat pesat dan menjadikannya sebagai media efektif yang interaktif dan banyak dikembangkan di perindustrian. Melihat kepopuleran *game* tersebut, para pendidik berpikir bahwa mereka mempunyai kesempatan yang baik untuk menggunakan komponen rancangan game dan menerapkannya pada kurikulum dengan penggunaan industri berbasis game. *Game* harus memiliki desain antar muka yang interaktif dan mengandung unsur menyenangkan[6].

Education game adalah *game* yang khusus dirancang untuk mengajarkan user suatu pembelajaran tertentu, pengembangan konsep dan pemahaman dan membimbing mereka dalam melatih kemampuan mereka, serta memotivasi mereka untuk memainkannya[6].

2.2.2. Kriteria *Education Game*

Menurut Hurd dan Jenuings perancangan *education game* yang baik haruslah memenuhi kriteria dari educationgameitu sendiri. Berikut ini adalah beberapa kriteria dari sebuah educationgame, yaitu [6]:

1. Nilai Keseluruhan (*Overall Value*)

   Nilai keseluruhan dari suatu *game* terpusat pada desain dan panjang durasi *game*. Aplikasi ini dibangun dengan desain yang menarik dan interaktif. Untuk penentuan panjang durasi, aplikasi ini menggunakan fitur timer.

2. Dapat Digunakan (*Usability*)

   Mudah digunakan dan diakses adalah poin penting bagi pembuat game. Apliksi ini merancang sistem dengan *interface* yang user *friendly* sehingga user dengan mudah dapat mengakses aplikasi.

3. Keakuratan (*Accuracy*)

   Keakuratan diartikan sebagai bagaimana kesuksesan model / gambaran sebuah *game* dapat dituangkan ke dalam percobaan atau perancangannya. Perancangan aplikasi ini harus sesuai dengan model game pada tahap perencanaan.

4. Kesesuaian (*Appropriateness*)

   Kesesuaian dapat diartikan bagaimana isi dan desain game dapat diadaptasikan terhadap keperluan user dengan baik. Aplikasi ini menyediakan menu dan fitur yang diperlukan user untuk membantu pemahaman user dalam menggunakan aplikasi.

5. Relevan (*Relevance*)

   Relevan artinya dapat mengaplikasikan isi *game* ke target user. Agar dapat relevan terhadap user, sistem harus membimbing mereka dalam pencapaiantujuan pembelajaran. Karena aplikasi ini ditujukan untuk anak-anak maka desain antarmuka harus sesuai dengan nuansa anak - anak, yaitu menampilkan warna-warna yang ceria.

6. Objektifitas (*Objectives*)

   Objektifitas menentukan tujuan user dan kriteria dari kesuksesan atau kegagalan. Dalam aplikas ini objektivitas adalah usaha untuk mempelajari hasil dari permainan.

7. Umpan Balik (*Feedback*)

   Untuk membantu pemahaman user bahwa permainan (*performance*) mereka sesuai dengan objek *game* atau tidak, *feedback* harus disediakan. Aplikasi ini menyajikan animasi dan efek suara yang mengindikasikan kesuksesan atau kegagalan permainan.

2.3. AI (*Artificiall Iintelligence*)

Sebagian kalangan menerjemahkan *Artificiall Iintelligence* sebagai kecerdasan buatan, kecerdasan *artifisial*, intelijensia *artifisial*, atau intelejensia buatan. Pada bukunya,[7] Suyanto sengaja tidak menerjemahkan istilah *Artificiall Iintelligence* ke bahasa Indonesia karena istilah tersebut sudah sangat akrab bagi orang Indonesia. Begitu juga dengan singkatan istilah tersebut, yaitu AI, sudah sangat melekat di berbagai media ilmiah maupun non ilmiah.

Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Stuart Russel dan Peter Norvig mengelompokkan definisi AI, yang diperoleh dari beberapa *textbook* berbeda, ke dalam empat kategori[7], yaitu :

A. *Thinking humanly* : *the cognitive modeling approach* Pendekatan ini dilakukan dengan dua cara sebagai berikut[7] :

1. Melalui introspeksi : mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berpikir. Tetapi, seorang psikolog Barat mengatakan "*how do you know that you understand*?" Bagaimana anda tahu bahwa anda mengerti? Karena pada saat anda menyadari pemikiran anda. Sehingga definisi ini terkesan mengada-ngada dan tidak mungkin dilakukan.
2. Melalui eksperimen-eksperimen psikologi.

B. *Acting humanly : the Turing test approach*

Pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer berintelijensia untuk menguji apakah komputer tersebut mampu mengelabui seorang manusia yang menginterogasinya melalui *teletyp*e (komunikasi berbasis teks jarak jauh). Jika *interrogator* tidak dapat membedakan yang diinterogasi adalah manusia atau komputer, maka komputer berintelijensia tersebut lolos dari *Turning test.* Komputer tersebut perlu memiliki kemampuan : *Natural Language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision, Robotics*. *Turing test* sengaja menghindari interaksi fisik antara interrogator dan komputer karena simulasi fisik manusia tidak memerlukan intelijensia.

C. *Thinking rationally : the laws of thought approach*

Terdapat dua masalah dalam pendekatan ini, yaitu [7]:

1. Tidak mudah untuk membuat pengetahuan informal dan menyatakan pengetahuantersebut ke dalam formal term yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%.
2. Terdapat perbedaan besar antara dapat memecahkan masalah "dalam prinsip" dan memecahkannya "dalam dunia nyata".

D. *Acting rationally : the rational agent approach*

Membuat inferensi yang logis merupakan bagian dari suatu rational agent. Hal ini disebabkan satu-satunya cara untuk melakukan aksi secara rasional adalah dengan menalar secara logis. Dengan menalar secara logis, maka bisa didapatkan kesimpulan bahwa aksi yang diberikan akan mencapai tujuan atau tidak. Jika mencapai tujuan, maka agent dapat melakukan aksi berdasarkan kesimpulantersebut.

Thinking humanly dan *acting humanly* adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio, yakni reflex dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnyaoleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika menggunakan definisi ini, maka banyak produk komputasi cerdas saat ini yang tidak layak disebut sebagai produk AI.

Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally*. Oleh karena itu, definisi AI yang paling tepat untuk saat ini adalah *acting rationally* dengan pendekatan *rational agent*. Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut.

2.4. Algoritma A*

Algorima A*, dapat juga disebut sebagai Algoritma A Star, merupakan salah satu contoh algoritma pencarian yang cukup popular di dunia. Beberapa terminologi dasar yang terdapat pada algoritma ini adalah simpul (*node*), simpul

asal/mulai (*source node*), simpul akhir/tujuan (*destination node*), simpul sekarang (*current node), openlist, closedlist, bestpath*, biaya (*cost*), halangan (*unwalkable/obstacle*). Simpul (*node*) adalah petak-petak kecil sebagai representasi dari area pencarian (*pathfinding*). Bentuknya dapat berupa persegi, lingkaran, maupun segitiga. Simpul asal/mulai (*source node*) adalah sebuah terminologi untuk posisi awal sebuah benda. Simpul akhir/tujuan (*destination node*) adalah tempat tujuan yang ingin dicapai pada pencarian. Simpul sekarang (*current node*) adalah simpul terbaik sebelumnya yang dipilih dan menjadi titik acuan untuk membangkitkan simpul tetangganya (*adjacent*). *Open list* adalah tempat menyimpan data simpul yang diakses dari simpul asal/mulai (*source node)* atau dari tetangga simpul sekarang (*current node*) yang belum pernah berada di *open list* maupun *closed list. Closed list* adalah tempat menyimpan simpul yang pernah menjadi simpul sekarang (*current node).* Biaya biasanya disimbolkan dengan huruf F (total *cost*) adalah nilai yang diperoleh dari penjumlahan nilai G (*actual cost)* dengan nilai H (*heuristic cost*). Halangan/penghalang adalah simpul yang tidak dapat dilalui. Awal dari algoritma ini bekerja adalah dengan menentukan terlebih dahulu simpul asal (*source node*) dan simpul tujuan (*destination node*).

Algoritma A* tanpa fungsi *heuristic* yang baik akan memperlambat pencarian dan dapat menghasilkan rute yang tidak tepat. Untuk menghasilkan rute yang benar –benar tepat, maka fungsi *heuristic*-nya harus *underestimate* terhadap biaya dari suatu *node* ke final *node*. Fungsi *heuristic* yang sempurna akan membuat algoritma A* langsung menuju final *node* tanpa harus mencari ke arah-arah lain. Sehingga jika fungsi *heuristic*-nya terlalu underestimate akan menyebabkan algoritma ini beranggapan bahwa ada rute yang lebih baik dari rute yang ada. Untuk fungsi *heuristic* yang *underestimate,* bila nilainya terlalu rendah akan menyebabkan algoritma ini seperti algoritma *Dijkstra* yang mencari ke segala arah yang mungkin. Hal ini dikarenakan tidak cukupnya informasi mengenai masalah yang dihadapi, sehingga menyebabkan algoritma A* melakukan pencarian lebih banyak dan lebih lama. Jika algoritma ini diharapkan melakukan pencarian dengan lebih cepat tanpa memperoleh rute terpendek, maka

fungsi *heuristic*-nya suatu saat dapat *overestimate.* Beberapa fungsi h*euristic* yang umum digunakan pada algoritma pathfinding A* adalah sebagai berikut [8] :

1. *Manhattan Distance*

Manhattan Distance adalah fungsi *heuristic* standar untuk algoritma A*. Digunakan pada aplikasi yang memiliki 4 arah gerakan (tidak dapat bergerak diagonal).

$$h(n) = d * (abs(Xn - Xgoal) + abs (Yn - Ygoal)) \quad \ldots 2.1$$

Dimana :

a. h adalah nilai *heuristic*.
b. d adalah nilai biaya. Dimana nilai d didapat dari nilai minimum cost perpindahan antara *node*.
c. Xn adalah koordinat X dari *node* pertama pada *grid*.
d. Xgoal adalah koordinat X dari final *node*.
e. Yn adalah koordinat dari *node* pertama pada *grid*.
f. Ygoal adalah koordinat Y dari final *node*.

2. *Straight Line Distance*

Straight Line Distance adalah fungsi *heuristic* yang digunakan pada aplikasi yang dapat bergerak ke segala arah / sudut.

$$h(n) = sqrt((Xn – Xgoal)2 + (Yn – Ygoal)2) \quad \ldots 2.2$$

Dimana :

a. h adalah nilai *heuristic*.
b. Xn adalah koordinat X dari *node* pertama pada *grid*
c. Xgoal adalah koordinat X dari final node
d. Yn adalah koordinat dari *node* pertama pada *grid*
e. Ygoal adalah koordinat Y dari final *node*.

3. *Diagonal Distance*

Diagonal Distance adalah fungsi *heuristic* yang digunakan pada aplikasi yang memiliki delapan arah gerakan (dapat bergerak diagonal).

$$h(n) = d * \max abs (Xn – Xgoal), abs (Yn –Ygoal) \quad \ldots 2.3$$

Dimana :

a. h adalah nilai *heuristic*.
b. d adalah nilai biaya. Dimana nilai d didapat dari nilai minimum cost perpindahan antar *node*.
c. Xn adalah koordinat X dari node pertama pada *grid*.
d. Xgoal adalah koordinat X dari final *node*.
e. Yn adalah koordinat dari *node* pertama pada *grid*.
f. Ygoal adalah koordinat Y dari final *node*.

2.5. *Tools* yang Digunakan

Tools yang digunakan untuk merancang dan membangun *game* edukasi Menggembala dan Pemanfaatan Kerbau ini, baik itu untuk pemrograman maupun analisis.

2.5.1. *Object Oriented Programming* (OOP)

Object Oriented Programming (OOP) menerapkan sifat yang lebih modular agar setiap program dapat lebih mudah dikembangkan. Dalam OOP dibutuhkan memori lebih besar dibandingkan dengan program *procedural* (tradisional)[9].

Dua objek yang identik akan memerlukan dua area memori berbeda walaupun dari sisi data dan proses keduanya memiliki jumlah dan jenis yang sama. Hal ini disebabkan karena data dan proses pada kedua objek tersebut dipisahkan oleh komputer.

Secara garis besar yang menjadi ciri dari OOP adalah adanya proses abstraksi (*abstraction*), pengkapsulan (*encapsulation*), penurunan sifat (*inheritance*), dan polimorfisme (*polymorphism*) pada objek-objek yang dibentuk. *Object Oriented Programming* (OOP) dibagi menjadi beberapa ciri utama[9], yaitu :

A. Kelas

Kelas (*class*) merupakan contoh abstrak dari sebuah objek yang telah terbentuk dari proses penyederhanaan, dengan kata lain kelas (*class*) merupakan cikal bakal dari objek (*object*), kemudian contoh nyata atau perwujudan dari sebuah objek dinamakan i*nstance*. Sehingga apabila kita mempunyai sebuah kelas

manusia, maka beberapa *instance*s (wujud nyata) dari kelas manusia adalah Prima, Aulia, Dewi, dan masih banyak yang lainnya. Perbedaan antara kelas (*class*) dengan objek (*object*) dalam OOP dibagi menjadi dua[10], yaitu :

1. *Class* merupakan rancangan (*design*) dan object merupakan perwujudan dari suatu class.
2. *Class* bersifat abstrak sementara *object* bersifat *konkrit* (atau nyata).

Di dalam kelas, terdapat *access modifier, class modifier,* dan *class body.* dibawah ini merupakan penjelasan tentang isi dari sebuah kelas.

1. *Access Modifier*

Access modifier merupakan *keyword* yang digunakan untuk menentukan spesifikasi tingkat akses suatu variable atau method (anggota kelas). Pengaksesan yang dimaksud bisa berupa pengaksesan dalam kelas yang sama, turunan, maupun di luar kelas di mana *variable* dan *method* dideklarasikan. Dalam Java, a*ccess modifier* terdiri atas :

a. *Default*

Access modifier ini hanya menspesifikasikan kelas-kelas (*classes*) di paket yang sama yang bisa mengakses variable dan method suatu kelas. Anggota kelas dengan *default access* bisa melihat kelas lain di paket yang sama. Apabila tidak ada *access modifier* pada suatu pendeklarasian kelas, berarti yang dimaksud adalah *default access*.

b. *Public*

Apabila suatu anggota kelas (*class member*) dideklarasikan sebagai *public,* maka anggota tersebut akan bisa digunakan oleh siapa saja, yaitu baik oleh kelas yang sama, kelas lain, maupun lingkungan luar kelas.

c. *Protected*

Access modifier protected menspesifikasikan atau menentukan anggota kelas (*class member*) yang hanya bisa diakses oleh *method-method* yang ada di dalam kelas yang sama dan subkelas-subkelas turunan dari kelas tersebut.

d. *Private*

Access modifier private merupakan tingkatan akses yang sangat terbatas. Pada keadaan ini tidak ada kelas lain yang dapat mengakses anggota kelas *private*, sekaligus termasuk subkelas-subkelasnya.

2. *Class Modifier*

a. *Public*

Apabila suatu kelas (*class*) dideklarasikan sebagai public, maka kelas tersebut akan bisa digunakan oleh kelas lain maupun (termasuk dari lingkungan luar) tanpa memperhatikan apakah kelas lain tersebut berasal dari paket (*package*) yang sama atau tidak.

b. *Abstract*

Apabila suatu kelas (*class*) dideklarasikan sebagai *abstract,* maka kelas tersebut tidak akan bisa diwujudkan menjadi suatu objek. Kelas ini disusun dari satu atau lebih metode abstrak, artinya *method-method* dalam kelas ini dideklarasikan akan tetapi tanpa badan (tidak dimplementasikan).

c. *Final*

Apabila suatu kelas (*class*) dideklarasikan sebagai *final*, maka kelas tersebut tidak akan bisa diturunkan lagi menjadi kelas turunan yang lain. Dengan kata lain sudah selesai atau berakhir (*final*). Tujuannya adalah untuk mencegah perluasan yang tidak diinginkan.

d. *Extends*

Class modifier extends digunakan untuk mendeklarasikan kelas turunan dari kelas yang lainnya.

e. *Implements*

Class modifier implements digunakan untuk mendeklarasikan sebuah kelas yang akanmengimplementasikan satu atau lebih *interface Java*. Apabila *interface Java* yang akan dimplementasikan lebih dari satu, maka pendeklarasiannya dipisahkan dengan tanda koma “,”.

B. Objek

Dalam kenyataannya, sebuah objek dalam OOP adalah sebuah persilangan yang berbagi-pakai (*share*) sejumlah ciri dari objek umum dengan fitur *(feature*) dari sebuah bentuk komputer[10].

Sebuah objek secara praktis pemrograman berorientasi objek bisa didefinisikan sebagai berikut :

1. Setiap objek dimiliki oleh kelas objek, sehingga sebuah objek tidak bisa hadir tanpa sebuah kelas yang mendefinisikannya. Dengan kata lain objek adalah wujud (*instance*) dari sebuah kelas.
2. Sebuah objek (dan kelas yang memuatnya) adalah sebuah pengkapsulan (e*ncapsulation*) yang memasukkan data dan operasi untuk pemrosesannya.
3. Atribut-atribut (*attributes*) objek membantu untuk menyimpan dan menjaga status objek. Atribut-atribut ini menentukan apa yang diingat mengenai objek. Methode objek adalah satu-satunya cara untuk mengakses data dan memodifikasi statusnya. Cara pengaksesan dan pemodifikasian data dilakukan dengan mengirimkan sebuah pesan ke objek tersebut.

C. Abstraksi

Abstraksi dapat didefinisikan sebagai suatu proses melakukan desain class dan menentukan data dan method yang akan dimiliki oleh sebuah class[16]. Sebuah *method* abstrak mendefinisikan sebuah antarmuka dalam kelas dasar dan meninggalkan implementasi pada kelas turunan. Kelas abstrak adalah sebuah kelas yang berisi satu atau beberapa method abstrak.

D. Pengkapsulan

Pengkapsulan (*encapsulation)* merupakan proses pembungkusan atau penyederhanaan dari beberapa data atau method menjadi sebuah objek (*object*) atau kelas (*class*)[10].

E. Pewarisan Sifat

Penurunan atau pewarisan sifat (*inheritance*) ini merupakan cirri utama dari OOP dimana sifat-sifat yang terdapat pada kelas induk (*base class*) akan dimiliki oleh kelas turunannya (*derived class*)[10].

2.5.2. UML (*Unified Modeling Language*)

Menurut Martin Fowler UML adalah bahasa pemodelan, bukan suatu metode. UML tidak memiliki notasi atas proses yang merupakan bagian penting dari metode, UML terdiri atas sembilan diagram. Kesembilan diagram tersebut, yaitu: [11] .

A. *Use-Case Model Diagram*

Use-Case Diagram menggambarkan interaksi antara sistem dengan luar sistem dan sistem dengan *user*. Dengan kata lain, *Use-Case Diagram* secara grafis menggambarkan siapa yang akan menggunakan sistem dan dalam cara apa user ingin berinteraksi dengan sistem.

B. *Class Diagram*

Melukiskan struktur sistem dalam bentuk objek. Disni digambarkan objek *class* yang membangun sistem beserta hubungan antar *class.*

C. *Squence Diagram*

Secara grafis menggambarkan bagaimana objek saling berinteraksi melalui pesan dalam melakukan suatu operasi atau melakukan *use case*.

D. *Activity Diagram*

Digunakan untuk menggambarkan urutan aktivitas secara berurutan dari proses bisnis atau *Use Case*.

2.5.3. *Adobe Flash Professional CS5*

Adobe Flash (dahulu bernama *Macromedia Flash*) adalah salah satu perangkat lunak komputer yang merupakan produk unggulan *Adobe Systems*. *Adobe Flash* digunakan untuk membuat gambar vektor maupun animasi gambar tersebut. Berkas yang dihasilkan dari perangkat lunak ini mempunyai *file extension .swf* dan dapat diputar di penjelajah *web* yang telah dipasangi *Adobe Flash Player*. *Flash* menggunakan bahasa pemrograman bernama *ActionScript* yang muncul pertama kalinya pada *Flash* 5. Sebelum tahun 2005, *Flash* dirilis oleh *Macromedia Flash* 1.0 diluncurkan pada tahun 1996 setelah *Macromedia* membeli program animasi vektor bernama *Future Splash*. Versi terakhir yang diluncurkan di pasaran dengan menggunakan nama „*Macromedia*“ adalah

Macromedia Flash 8. Pada tanggal 3 Desember 2005 *Adobe Systems* mengakuisisi *Macromedia* dan seluruh produknya, sehingga nama *Macromedia Flash* berubah menjadi *Adobe Flash*. *Adobe Flash* merupakan sebuah program yang didesain khusus oleh *Adobe* dan program aplikasi standar *authoring tool profesional* yang digunakan untuk membuat animasi dan bitmap yang sangat menarik untuk keperluan pembangunan situs web yang interaktif dan dinamis. *Flash* didesain dengan kemampuan untuk membuat animasi 2 dimensi yang andal dan ringan sehingga *Flash* banyak digunakan untuk membangun dan memberikan efek animasi pada *website*, CD interaktif dan yang lainnya. Selain itu, aplikasi ini juga dapat digunakan untuk membuat animasi logo, *movie*, *game*, pembuatan navigasi pada situs web, tombol animasi, banner, menu interaktif, interaktif form isian, *e-card*, screen saver, dan pembuatan aplikasi-aplikasi web lainnya. Dalam *Flash* terdapat teknik-teknik membuat animasi, fasilitas *actionscript*, filter, custom easing dan dapat memasukkan video lengkap dengan fasilitas *playback* FLV. Keunggulan yang dimiliki oleh *Flash* ini adalah ia mampu diberikan sedikit kode pemrograman baik yang berjalan sendiri untuk mengatur animasi yang ada di dalamnya atau digunakan untuk berkomunikasi dengan program lain seperti HTML, PHP, dan Database dengan pendekatan XML, dapat dikolaborasikan dengan web, karena mempunyai keunggulan antara lain kecil dalam ukuran *file output*-nya. *Movie-movie Flash* memiliki ukuran *file* yang kecil dan dapat ditampilkan dengan ukuran layar yang dapat disesuaikan dengan keinginan. Aplikasi *Flash* merupakan sebuah standar aplikasi industri perancangan animasi *web* dengan peningkatan peraturan dan perluasan kemampuan integrasi yang lebih baik. Banyak fitur baru dalam *Flash* yang dapat meningkatkan kreativitas dalam pembuatan isi media yang kaya dengan memanfaatkan kemampuan aplikasi tersebut secara maksimal. Fitur-fitur baru ini membantu kita lebih memusatkan perhatian pada desain yang dibuat secara cepat, bukannya memusatkan pada cara kerja dan penggunaan aplikasi tersebut. *Flash* juga dapat digunakan untuk mengembangkan secara cepat aplikasi-aplikasi *web* yang kaya dengan pembuatan script tingkat lanjut. Di dalam aplikasinya juga tersedia sebuah alat untuk men-*debug script*. Dengan

menggunakan *Code hint* untuk mempermudah dan mempercepat pembuatan dan pengembangan isi *ActionScript* secara otomatis [12].

2.5.4. ActionScript

Salah satu kelebihan *Adobe Flash* CS5 dibanding perangkat lunak animasiyang lain yaitu adanya *ActionScript*. *ActionScript* adalah bahasa pemrograman *Adobe Flash* CS5 yang digunakan untuk membuat animasi atau interaksi.[13]

ActionScript mengizinkan untuk membuat intruksi berorientasi action (lakukan perintah) dan instruksi berorientasi *logic* (analisis masalah sebelum melakukan perintah). Sama dengan bahasa pemrograman yang lain, *ActionScript* berisi banyak elemen yang berbeda serta strukturnya sendiri. Kita harus merangkainya dengan benar agar *ActionScript* dapat menjalankan dokumen sesuai dengan keinginan. Jika tidak merangkai semuanya dengan benar, maka hasil yang didapat kan akan berbeda atau *file flash* tidak akan bekerja sama sekali. *ActionScript* juga dapat diterapkan untuk *action* pada *frame*, tombol, *movie clip*, dan lain-lain. *Action frame* adalah *action* yang diterapkan pada frame untuk mengontrol navigasi *movie, frame,* atau objek lain-lain.[14]

Salah satu fungsi *ActionScript* adalah memberikan sebuah konektivitas terhadap sebuah objek, yaitu dengan menuliskan perintah-perintah didalamnya.

Tiga hal yang harus diperhatikan dalam *ActionScript* yaitu:

1. *Event*

Event merupakan peristiwa atau kejadian untuk mendapatkan aksi sebuah objek. Event pada *Adobe Flash CS 5* ada empat yaitu:

a. *Mouse event* adalah *Event* yang berkaitan dngan penggunaan *mouse*.
b. *Keyboard Event* adalah kejadian pada saat menekan tombol *keyboard.*
c. *Frame Event* adalah *Event* yang diletakkan pada *keyframe.*
d. *Movie Clip Event* adalah *Event* yang disertakan pada *movie clip*.

2. Target

Target adalah objek yang dikenai aksi atau perintah. Sebelum dikenai aksi atau perintah, sebuah objek harus dikonversi menjadi sebuah simbol dan memiliki

nama instan. Penulisan nama target pada skrip harus menggunakan tanda petik ganda (“ “).

3. *Action*

Pemberian *action* merupakan lagkah terakhir dalam pembuatan interaksi antar objek. *Action* dibagi menjadi dua antara lain:

a. *Action Frame*: adalah action yang diberikan pada *keyframe.* Sebuah *keyframe* akan ditandai dengan huruf a bila pada *keyframe* tersebut terdapat sebuah *action*.
b. *Action Objek*: adalah *action* yang diberikan pada sebuah objek, baik berupa tombol maupun *movie clip*.