BAB 2
LANDASAN TEORI

2.1 Matematika

Matematika merupakan ilmu universal yang mendasari perkembangan teknologi modern, mempunyai peran penting dalam berbagai disiplin dan memajukan daya pikir manusia. Perkembangan pesat di bidang teknologi informasi dan komunikasi dewasa ini dilandasi oleh perkembangan matematika di bidang teori bilangan, aljabar, analisis, teori peluang dan matematika diskrit. Untuk menguasai dan mencipta teknologi di masa depan diperlukan penguasaan matematika yang kuat sejak dini[1].

2.1.1 Anak Usia SD Dalam Pembelajaran Matematika di SD

Anak usia SD adalah anak yang berada pada usia sekitar 7 sampai 12 tahun. Menurut Piaget anak usia sekitar ini masih berpikir pada tahap operasi konkrit artinya siswa siswa SD belum berpikir formal. Ciri-ciri anak-anak pada tahap ini dapat memahami operasi logis dengan bantuan benda-benda konkrit, belum dapat berpikir deduktif, berpikir secara transitif. Contoh : 2 + 2 = 4, 4 + 2 = 6, 6 + 2 = 8, 10 + 2 = 12. Proses ini sudah dapat dipahami oleh siswa.

Sebagaimana kita ketahui, matematika adalah ilmu deduktif, formal, hierarki dan menggunakan bahasa simbol yang memiliki arti yang padat. Karena adanya perbedaan karakteristik antara matematika dan anak usia SD, maka matematika akan sulit dipahami oleh anak SD jika diajarkan tanpa memperhatikan tahap berpkir anak SD. Seorang guru hendaknya mempunyai kemampuan untuk menghubungkan antara dunia anak yang belum dapat berpikir secara deduktif agar dapat mengerti matematika yang bersifat deduktif[3].

2.1.2 Bermain Sambil Beajar Matematika

Bermain pada umumnya tidak dapat dipisahkan dari anak-anak, karena disamping memenuhi kebutuhan akan bermain dapat juga menambah atau memperkaya pengalaman anak. Dengan keinginan anak bermain, orang tua atau pendidik dapat memanfaatkannya untuk menanamkan pengertian akan pelajaran misalnya belajar matematika[3].

2.1.3 Operasi Hitung Sederhana (Penjumlahan, Pengurangan, Perkalian, dan Pembagian)

Operasi penjumlahan pada dasarnya merupakan suatu aturan yang mengaitkan setiap pasang bilangan dengan bilangan yang lain. Operasi penjumlahan ini mempunyai beberapa sifat yaitu: sifat pertukaran (komutatif), sifat identitas, dan sifat pengelompokan (asosiatif)[3].

Operasi pengurangan merupakan kebalikan dari operasi penjumlahan, tetapi operasi pengurangan tidak memiliki sifat yang dimiliki operasi penjumlahan. Operasi pengurangan tidak memenuhi sifat pertukaran, sifat identitas, dan sifat pengelompokan.

Operasi perkalian dapat didefinisikan sebagai penjumlahan berulang. Misalkan pada perkalian 4 x 3 dapat didefinisikan sebagai 3 + 3 + 3 + 3 = 12 sedangkan 3 x 4 dapat didefinisikan sebagai 4 + 4 + 4 = 12. Secara konseptual, 4 x 3 tidak sama dengan 3 x 4, tetapi jika dilihat hasilnya saja maka 4 x 3 = 3 x 4.

Operasi pembagian dapat didefinisikan sebagai pengurangan berulang. Secara matematis ditulis sebagai a : b = a – b – b – b .... = 0. Misal, 24 : 3 = 24 – 3 – 3 – 3 – 3 – 3 – 3 – 3 – 3 = 0. Berarti 24 : 3 = 8. Hasil ini ditunjukkan oleh banyaknya angka 3 yang muncul sebagai bilangan pengurangnya. Operasi pembagian adalah kebalikan dari operasi perkalian.

2.2 *Game*

Game berasal dari kata bahasa inggris yang berarti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Dalam game, ada target-target yang ingin dicapai pemainnya. Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, play dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan[4].

2.2.1 Bagian dari Game

Menurut Teresa Dillon elemen-elemen dasar sebuah *game* adalah [4]:

1. Game Rule

Game rule merupakan aturan perintah, cara menjalankan, fungsi objek dan karakter di dunia permainan Dunia *Game* Dunia *game* bisa berupa pulau, dunia khayal, dan tempat-tempat lain yang sejenis yang dipakai sebagai setting tempat dalam permainan *game*.

2. Plot

Plot biasanya berisi informasi tentang hal-hal yang akan dilakukan oleh player dalam *game* dan secara detail ,perintah tentang hal yang harus dicapai dalam *game*.

3. Tema

Di dalam biasanya ada pesan moral yang akan disampaikan.

4. Character

Pemain sebagai karakter utama maupun karakter yang lain yang memiliki ciri dan sifat tertentu *Object* Merupakan sebuah hal yang penting dan biasanya digunakan pemain untuk memecahkan masalah, adakalanya pemain harus punya keahlian dan pengetahuan untuk bisa memaninkannya.

5. Text, Graphic dan Sound

Game biasanya merupakan kombinasi dari media teks, grafik maupun suara, walaupun tidak harus semuanya ada dalam permainan *game*.

6. Animasi

Animasi ini selalu melekat pada dunia *game* , khususnya untuk gerakan karakter-karakter yang ada dalam *game*, properti dari objek.

7. User Interface

Merupakan fitur-fitur yang mengkomunikasikan *user* dengan *game*.

2.2.2 *Game* Edukasi

Menurut Randel *game* sangat berpotensi untuk menumbuhkan kembali motivasi belajar anak yang mengalami penurunan. Berdasarkan penelitian yang dilakukan oleh Randel pada tahun 1991 tercatat bahwa pemakaian *game* sangat bermanfaat pada materi-materi yang berhubungan dengan matematika, fisika dan kemampuan berbahasa (seperti studi sosial, biologi dan logika)

Game yang memiliki content pendidikan lebih dikenal dengan istilah game edukasi. *Game* berjenis edukasi ini bertujuan untuk memancing minat belajar anak terhadap materi pelajaran sambil bermain, sehingga dengan perasaan senang diharapkan anak bisa lebih mudah memahami materi pelajaran yang disajikan. Jenis ini sebenarnya lebih mengacu kepada isi dan tujuan game, bukan jenis yang sesungguhnya.

Menurut Edward *game* merupakan sebuah alat yang efektif untuk mengajar karena mengandung prinsip-prinsip pembelajaran dan teknik instruksional yang efektif digunakan dalam penguatan pada *level-level* yang sulit

Maja Pivec membuktikan *game* edukasi berhasil diterapkan untuk pendidikan formal khususnya di militer, ilmu kedokteran, fisika, training dan lain sebagainya. Model *game* edukasi yang merupakan perpaduan antara instructional content dan karakteristik game terbukti mampu meningkatkan motivasi pemain game serta mendapatkan keluaran ilmu pengetahuan ketikan process pembelajaran game itu berlangsung[4].

2.2.2.1 Prinsip Game Edukasi

Menurut Foremen beberapa prinsip yang harus diterapkan dalam aplikasi sebuah *game* edukasi adalah[4]:

1. *Individualization*

Materi pembelajaran (pengetahuan) dibuat sesuai dengan kebutuhan individual dari pembelajar, sedangkan *game* mengadopsi *level individual* dari pemain.

2. *Feedback Active*

Adanya *feedback* yang sesuai dengan cepat untuk memperbaiki pembelajaran dan mengurangi ketidaktahuan pembelajar terhadap materi yang disampaikan, sedangkan *game* menyediakan *feedback* dengan cepat dan konstektual.

3. *Active Learning*

Adanya kecenderungan untuk menyertakan pelajar secara aktif dalam menciptakan penemuan dan pengetahuan baru yang membangun, sedangkan *game*

menyediakan suatu lingkungan yang membantu terjadinya penemuan baru tersebut.

4. *Motivation*

Pelajar termotivasi dengan *reward* yang diberikan dalam aktivitas permainan, sedangkan *game* melibatkan pengguna berjam-jam untuk mencapai tujuan.

5. *Scaffolding*

Pelajar secara berangsur-angsur ditantang dengan tingkat kesulitan yang makin tinggi dan dapat melangkah lebih maju untuk mencapai kemenangan dari permainan, sedangkan *game* dibangun secara multi *level*, pemain tidak bisa bergerak ke *level* yang lebih tinggi sampai dia mampu menyelesaikan permainan di *level* yang ada.

6. *Transfer*

Pelajar mengembangkan kemampuan untuk mentransfer pengetahuan dari satu orang ke orang yang lain, sedangkan *game*, sedangkan *game* mengijinkan pemain untuk menstransfer informasi dari suatu konteks ke konteks yang lain

7. *Assessment*

Setiap individu mempunyai kesempatan untuk menilai pelajaran mereka sendiri atau membandingkannya dengan orang lain.

2.2.2.2 Dampak Game Edukasi

Ada dampak positif dan negatif sebuah *game*, sekalipun *game* edukasi[4].

1. Dampak Positif *Game* Edukasi

Menurut Edward *game* memiliki banyak dampak positif antara lain:

a. *Game* banyak digunakan orang untuk mengajarkan suatu pengetahuan dan membangun ketrampilan baik dibidang edukasi, bisnis maupun militer.
b. *Game* effektif digunakan untuk membangun kemampuan matematika dan membaca pada anak, dibuktikan dengan penelitian yang dilakukan oleh Murphy dan kawan-kawan pada tahun 2002.
c. *Game* terbukti effektif untuk membantu anak-anak penderita ashma dan diabetes mengelola kebiasaan hidup sehat, penelitian dilakukan oleh

Lieberman pada tahun 1997 dan McPhershon dan kawan-kawan pada tahun 2006.

d. Banyak bisnis menggunakan game edukasi untuk membangun ketrampilan karyawan mereka seperti Cisco mengajarkan karyawan mereka pengenalan *tools* dasar dan *security network* melalui sebuah *game*.
e. Pada tahun 2007 militer AS menggunakan game dalam *training* personil militer seperti simulasi penerbangan pesawat dan pemakaian sistem persenjataan.

2. Dampak Negatif *Game* Edukasi

Tidak semua *game* memberi dampak yang diinginkan, penelitian yang dilakukan Anderson pada tahun 2004 dan Batholow 2006 memberikan bukti bahwa game juga memiliki dampak negatif seperti:

a. Meningkatkan sifat aggresiv cognitiv, aggresiv behavior.
b. Memberikan efek permusahan dan efek psikologis.
c. Menurunkan kebiasaaan prososial.
d. Dari segi kesehatan game dapat memberikan dampak meningkatatnya detak jantung, tekanan darah aktivitas otak.

Dampak negarif dari game ini secara umum berlaku pada pemain yang masih anak-anak dan remaja yang memiliki tidak emosional yang tinggi. Ketika penelitian dilakukan terhadap orang tua maka dampak ini sangat kecil pengaruhnya.

2.2.2.3 *Game Mobile Learning*

Kecanggihan teknologi komunikasi memberi nuansa berbeda dalam proses belajar mengajar sesorang. Teknologi komunikasi mengubah lokasi belajar dari kelas ke tempat dimana saja peserta didik dapat belajar. Dengan demikian, teknologi komunikasi mendorong terjadinya evolusi pada lokasi belajar. Menurut prawiradilaga belajar tidak lagi hanya berlangsung di sekolah dan di kelas, belajar dapat terjadi dimana saja selama ada bahan ajar dan peserta didik merasa nyaman dengan situasi itu.

Disisi lain perkembangan teknologi telah menciptakan terobosan-terobosan dalam pembelajaran. Peserta didik bersinggungan dengan perangkat-perangkat

teknologi komunikasi bergerak seperti *handphone* yang dengan berbagai fasilitasnya telah menjadi gelombang kecenderungan baru yang memungkinkan peserta didik (pembelajar) belajar secara *mobile* atau lebih dikenal sebagai *mobile learning*.

Pemanfaatan *handphone* sebagai sarana mobilitas dalam sebuah mobile learning dan penyajian materi dalam format *game* menjadi sebuah wacana untuk menyajikan pengetahuan dalam sebuah hiburan. Belajar dengan senang, belajar dimanapun kapanpun, tanpa sekat tanpa batas[4].

2.3 Algortima Dijkstra

Algoritma Dijkstra merupakan algoritma tercepat untuk menghitung jarak minmal dari *vertex* ke *vertex* lainnya di graph yang tidak mengandung nilai negatif pada setiap *edges*. Algoritma ini ditemukan oleh ilmuwan komputer Belanda Edsger Dijstra pada tahun 1959[8].

2.3.1 Deskripsi Algoritma Dijkstra

Algoritma merupakan perluasan dari algoritma *breadth-first search*, dimana urutan node (*vertex*) tidak ditentukan oleh jumlah dari setiap sisi akarnya, tetapi sebagai jarak dari akar (jumlah bobot dari semua sisi sepanjang jalan dari akar ke *node* yang diberikan). Sebagai konsekuensi algoritma Dijkstra proses hanya dilakukan pada *node*, dimana jalan terpendek sudah ditemukan.

Algoritma Dijkstra akan menyimpan semua node dalam antrian prioritas dengan mencari jarak dari simpul dari akar - pada iterasi pertama dari algoritma, akar dinisialisasi 0, jarak semua *node* lain sama dengan tak terhingga. Dalam setiap langkah algoritma Dijkstra mengambil dari antrian *node* dengan prioritas tertinggi (paling tidak jarak dari akar) yang akan diperoses dan mengevaluasi ulang jarak semua keturunan yang diproses dari *node*. Ini berarti bahwa algoritma memeriksa semua keturunan bahwa kondisi berikut ini berlaku :

$$distance_{processed} + edgeWeight_{processed,descendant} < distance_{descendant}$$

Jika sudah habis, algoritma perubahan sesuai jarak keturunan dan menetapkan *node* olahan sebagai induknya. Ketika semua keturunan diperiksa, algoritma mengambil lagi node dengan prioritas tertinggi dan mengulangi proses. Algoritma Dijkstra berakhir, ketika antrian kosong (semua node diproses).

Hal ini penting untuk mengetahui bahwa algoritma Dijkstra mensyaratkan bahwa bobot semua sisi non-negatif. Jika prosedur ini tidak dapat menentukan apakah jalur terpendek untuk node itu sudah ditemukan atau tidak.

2.3.2 Complexity

Kompleksitas asimtotik algoritma Dijkstra tergantung pada pelaksanaan antrian prioritas. Jika diimplementasikan menggunakan pencarian sekuensial daripada kompleksitas adalah $O(|N|^2)$, dimana $|N|$ banyaknya *node* dari graph. Ketika *benary heap* digunakan, the complexity is $O(|E|.\log_2|N|)$ $|E|$ banyaknya *edge* dari dijkstra.

2.3.3 Code

```
/**
 * Dijkstra's algorithm
 * @param d matrix of legths, position [0 1] = 2 means that from node 0 leads an edge to node 1 of length 2
 * @param from root node
 * @return tree an ancestors (denotes path from the node to the root node)
 */
procedure int[] doDijkstra(d, from) {
    //insert all nodes to the priority queue, node from has a distance 0, all others infinity
    Q = InsertAllNodesToTheQueue(d, from)
    CLOSED = {} //closed nodes - empty set
    predecessors = new array[d.nodeCount] //array of ancestors

    while !Q.isEmpty() do
        node = Q.extractMin()
        CLOSED.add(node) //close the node

        //contract distances
        for a in Adj(node) do //for all descendants
            if !CLOSED.contains(a) //if the descendatn was not closed yet
                //and his distance has decreased
                if Q[node].distance + d[node][a] < Q[a].distance
                    //zmen prioritu (vzdalenost) uzlu
                    Q[a].distance = Q[node].distance + d[node][a]
                    //change its ancestor
                    predecessors[a] = node

    return predecessors
```

Gambar 2.1 *Code* Algoritma Dijkstra

2.4 Algoritma *Fisher-Yates*

Fisher-Yates Shuffle (dinamai berdasarkan penemunya, Ronald Fisher dan Frank Yates) digunakan untuk mengubah urutan masukan yang diberikan secara acak. Permutasi yang dihasilkan oleh algoritma ini muncul dengan probabilitas yang sama[9].

Metode Fisher-Yates secara umum adalah :

1. Ketika masih ada elemen tersisa untuk diacak
2. Ambil elemen secara acak dari elemen yang tersisa
3. Kemudian tukar dengan elemen saat ini.

2.4.1 Code

Berikut Algoritma Fisher Yates dalam javascript :

```
function shuffle(array) {
  var m = array.length, t, i;

  // While there remain elements to shuffle…
  while (m) {

    // Pick a remaining element…
    i = Math.floor(Math.random() * m--);

    // And swap it with the current element.
    t = array[m];
    array[m] = array[i];
    array[i] = t;
  }

  return array;
}
```

Gambar 2.2 *Code* Algoritma Fisher-Yates

2.5 Pemograman Berbasis Objek

Pemograman berorientasi objek (*Object-Oriented Programing*) adalah pembangunan perangkat lunak yang mengoragisasikan preangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan. Pemograman berorientasi objek merupakan suatu cara bagaimana system perangkat lunak dibangun melalui pendakatan objek[11].

2.5.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu tehnik atau cara pendekatan dalam melihat permasalahan dan sistem. Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Berikut adalah beberapa konsep dasar yang harus dipahami tentang pemograman berorientasi objek :

1. Kelas (*class*)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut.

2. Objek (*object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi

(status) dan mempunyai operasi yang dapat diterapkan atau berpengaruh pada status objeknya. Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebbuah objek.

3. Metode (*method*)

Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode atau Operasi dapat berasal dari event, aktifitas atau aksi, keadaan, fungsi, dan kelakuan dunia nyata.

4. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek.

5. Abstraksi (*abstraction*)

Prinsip untuk mempresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapulasi (*encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi objek sehingga objek lain tidak mengetahui cara kerjannya.

7. Pewarisan (*inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*interface*)

Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang di deklarasikan tanpa isi.

9. Reusability

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.

11. Komunikasi Antar Objek

Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dari suatu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. Package

Package adalah sebuah container atau kemasan yang dapat digunakan untuk mengelompokan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.5.2 Analisis Berorientasi Objek

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahapan unntuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek.

2.5.3 Desain Berorientasi Objek

Disain berorientasi objek atau *Object Oriented Design* (OOD) adalah tahapan perantara untuk memetakan spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasi objek.

Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat pemodelan beorientasi objek, diantaranya adalah UML (*Unified Modeling Language*).

2.5.4 Bahasa Pemograman yang Mendukung PBO

1. Bahasa Pemograman Smalltalk

Smaltalk merupakan salah satu bahasa pemograman yang dikembangkan untuk mendukung pemograman berorientasi objek mulai tahun 1978.

2. Bahasa Pemograman Eiffel

Effiel merupakan bahasa pemograman yang dikembangkan untuk mendukung pemograman berorientasi objek mulai than 1985 oleh Bertrand Mater dan *compiler* Eiffel selesai dibuat pada tahun 1987.

3. Bahasa Pemograman C++

Bahasa pemograman C++ merupakan pengembangan lebih lanjut dari bahasa pemograman C untuk mendukung pemograman berorientasi objek.

4. Bahasa Pemograman PHP

PHP dibuat pertama kali oleh seorang perekayasa perangkat lunak (*softwaew engineering*) yang bernama Rasmus Lerdoff. Dengan beberapa versi yang telah diperbaharui. PHP 5 sudah mendukung pemograman berorientasi objek.

5. Bahasa Pemograman Java

Java dikembangkan oleh perusahaan Sun Microsystem. Java menurut Sun Microsystem adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Java merupakan bahasa pemograman objek murni karena semua kode program dibungkus dalam kelas.

2.6 Pemodelan Sistem

Pemodelan sistem yang berorientasi objek dapat dimodelkan dengan menggunakan UML (*Unified Modeling Language*). UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. Berikut merupakan bagian-bagian dari UML[12].

1. Use Case

Use Case merupakan pemodelan dari kebutuhan sistem terhadap interaksi pengguna kepada sistem. *Use Case* dapat menjadi gambaran kejadian dimana pengguna melakukan interaksi terhadap sistem.

2. Activity Diagram

Activity Diagram merupakan sebuah notasi diagram yang menjelaskan tentang sebuah sekenario yang terjadi pada sebuah *use case* dimana untuk memudahkan pembacaaan sekenario.

3. Class Diagram

Class Diagram merupakan pemodelan dari sebuah *class-class* yang akan dibangun pada sebuah sistem. *Class diagram* akan menggambarkan semua relasi yang terjadi terhadap *class-class*.

4. Sequence Diagram

Sequence diagram merupakan pemodelan untuk menggambarkan interaksi antar pengguna dan sistem, dimana pada suatu *use case* akan dijelaskan secara sekuen bagaimana porses interaksi anatara pengguna dan sistem dijelaskan. *Sequence diagram* akan menjelaskan keterkaitan *class-class* yang akan dibangun.

2.7 *Tools* yang Digunakan

Berikut adalah *tools* yand digunakan sebagai pendukung pembangunan aplikasi *game*.

2.7.1 Android *Development Tools Bundle*

Android *Development Tools Bundle* merupakan kumpulan dari beberapa softwawre untuk membangun aplikasi Android yang disediakan oleh Android[14]. Di dalam Android *Development Tools Bundle* terdapat software, yaitu :

2.7.1.1 Eclipse

Eclipse adalah sebuah IDE (Integrated Development Environment) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (platform-independent)[18]. Berikut ini adalah sifat dari Eclipse:

1. Multi-platform
2. Mulit-language: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. Multi-role: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Secara standar Eclipse selalu dilengkapi dengan JDT (Java Development Tools), plug-in yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (Plug-in Development Environment) untuk

mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java. Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (*C/C++ Development Tools*).

2.7.1.2 ADT (*Android Development Tools*) Plugins for Eclipse

Android Development Tools (ADT) adalah *plugin* yang didesain untuk IDE Eclipse yang memberikan kita kemudahan dalam mengembangkan aplikasi angroid dengan menggunakan IDE Eclipse. Dengan menggunakan ADT untuk Eclipse akan memudahkan kita dalam membuat aplikasi projek android, membuat GUI aplikasi, dan menambah komponen-komponen yang lainnya, begitu juga kita dapat melakukan *running* aplikasi menggunakan Android SDK melalui Eclipse[14].

2.7.1.3 Android SDK (*Sun Development Kit*)

Android SDK (*Sun Development Kit*) adalah *tools* API (*Application Programing Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan pemograman Java[14].

2.7.2 JDK (*Java Development Kit*) Oracle

Java Development Kit merupakan perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode Java menjadi bytecode yang dapat dimengerti dan dapat dijalankan oleh Java Runtime Environtment.

Java Development Kit wajib terinstall pada komputer yang akan melakukan proses pembuatan aplikasi berbasis Java. Namun Java Development Kit tidak wajib terinstall di komputer yang akan menjalankan aplikasi yang dibangun menggunakan Java[19].

2.7.3 Reason

Reason merupakan sebuah program yang dirancang untuk membuat musik melalui pemograman musik yang bisa disebut dengan MIDI (*Musical Instrument Digital Interface*)[20].

2.7.4 CorelDraw

CorelDraw merupakan sebuah software untuk membuat sebuah grafis dan gambar-gambar[22].

2.7.5 Cubase

Cubase merupakan sebuah software untuk membantu perekaman musik baik secara digital melalui MIDI ataupun secara manual (memainkan alat oleh manusia)[21].