BAB II

LANDASAN TEORI

2.1. Belajar

Belajar merupakan kegiatan yang berproses dan termasuk unsur yang sangat fundamental dalam penyelenggaraan setiap jenis dan jenjang pendidikan. Belajar tidak hanya dapat dilakukan di sekolah saja, namun dapat dilakukan dimana-mana, seperti di rumah ataupun di lingkungan masyarakat.

2.2. Bahasa Inggris

Bahasa Inggris adalah sebuah bahasa yang berasal dari Inggris, merupakan bahasa utama di Britania Raya (termasuk Inggris), Amerika Serikat, serta banyak negara lainnya, dan termasuk rumpun bahasa Jermanik Barat. Bahasa ini berawal dari kombinasi antara beberapa bahasa lokal yang dipakai oleh orang-orang Norwegia, Denmark, dan Anglo-Saxon dari abad ke-6 sampai 10. Lalu pada tahun 1066 dengan ditaklukkan Inggris oleh *William the Conqueror*, sang penakluk dari Normandia, Perancis Utara, maka bahasa Inggris dengan sangat intensif mulai dipengaruhi bahasa Latin dan bahasa Perancis. Dari seluruh kosakata bahasa Inggris modern, diperkirakan ±50% berasal dari bahasa Perancis dan Latin.

2.2.1. Tujuan Belajar Bahasa Inggris

Tujuan belajar bahasa Inggris itu sederhana sekali, diantaranya:

1. Agar mampu menulis dalam bahasa Inggris (*writing*).
2. Agar mampu membaca dan memahami tulisan dalam bahasa Inggris (*reading*).

3. Agar mampu berbicara dalam bahasa Inggris (*speaking*).
4. Agar mampu mendengar/memahami ucapan yang diucapkan dalam bahasa Inggris (*Listening*).

2.3. *Game*

Dalam bahasa Indonesia "*Game*" berarti "permainan". Permainan yang dimaksud dalam *game* juga merujuk pada pengertian sebagai "kelincahan intelektual" (*intellectual playability*). Sementara kata "*game*" bisa diartikan sebagai arena keputusan dan aksi pemainnya. Ada target-target yang ingin dicapai pemainnya. Kelincahan intelektual, pada tingkat tertentu, merupakan ukuran sejauh mana *game* itu menarik untuk dimainkan secara maksimal.

2.3.1. Sejarah *Game*

Pada tahun 1952, seorang mahasiswa Universitas Cambridge bernama A.S. Gouglas membuat permainan OXO (tic-tac-toe) dalam versi grafik. Permainan ini ia kembangkan ketika hendak mendemonstrasikan tesisnya tentang interaksi antara manusia dan komputer. Pada tahun 1958 William Higin Botham mendesain sebuah *game* dengan judul *Tennis For Two* yang dimainkan dalam *oscilloscope*, dan kemudian ada pula Steve Russel pada tahun 1961 dengan *game* berjudul *Spacewar* yang dibuat dalam komputer mainframe DEC PDP-1 saat mereka menjalani studi di MIT.

Sejarah video *game* tidak hanya mengenai orang-orang yang berperan di dalamnya, tetapi juga mengenai berbagai perusahaan *game* yang mempunyai kasus ironis. Atari adalah perusahaan Amerika dengan nama Jepang, dan perusahaan.

Jepang bernama SEGA didirikan oleh orang Amerika. Magnavox yang memulai peran perusahaan ini telah berumur 1 abad, dan Nintendo sebagai perusahaan yang mempopulerkan kembali video *game* juga sama tuanya, serta tidak ada yang pernah berpikir bahwa Sony yang merupakan perusahaan penemu banyak barang-barang elektronik mulai dari transistor radio hingga alat perekam video, akan membuat sebuah *console* yang menjadi produk dengan tingkat penjualan yang tinggi hingga saat ini.

2.3.2. Jenis-Jenis *Game*

Jenis-jenis *game* dapat diklasifikasikan menjadi beberapa tipe antara lain adalah :

1. ***Action games***, biasanya meliputi tantangan fisik, teka-teki (*puzzle*), balapan, dan beberapa konflik lainnya. Dapat juga meliputi masalah ekonomi sederhana, seperti mengumpulkan benda-benda.
2. ***Real Time Strategy*** **(RTS)** adalah *game* yang melibatkan masalah strategi, taktik, dan logika. Contoh *game* jenis ini adalah *Age of Empire*, *War Craft*, dan sebagainya.
3. ***Role Playing Games*** **(RPG)**, kebanyakan *game* jenis ini melibatkan masalah taktik, logika, dan eksplorasi atau penjelajahan. Dan juga kadang meliputi teka-teki dan masalah ekonomi karena pada *game* ini biasanya melibatkan pengumpulan barang-barang rampasan dan menjualnya untuk mendapatkan senjata yang lebih baik. Contoh dari *game* ini adalah *Final Fantasy*, *Ragnarok*, *Lord of The Rings*, dan sebagainya.

4. ***Real World Simulation***, meliputi permainan olahraga dan simulasi masalah kendaraan termasuk kendaraan militer. *Games* ini kebanyakan melibatkan masalah fisik dan taktik, tetapi tidak masalah eksplorasi, ekonomi dan konseptual. Contohnya seperti adalah *game Championship Manager*.

5. ***Construction and Management***, seperti *game Roller Coster Tycoon* dan *The Sims*. Pada dasarnya adalah masalah ekonomi dan konseptual. *Game* ini jarang yang melibatkan konflik dan eksplorasi, dan hampir tidak pernah meliputi tantangan fisik.

6. ***Adventure games***, mengutamakan masalah eksplorasi dan pemecahan teka-teki. Namun terkadang meliputi masalah konseptual, dan tantangan fisik namun sangat jarang.

7. ***Puzzle games***, ditujukan untuk memecahkan suatu masalah tertentu. Hampir semua semua tantangan disini menyangkut masalah logika yang biasanya dibatasi oleh waktu.

8. ***Slide scrolling games***, pada jenis *game* ini karakter dapat bergerak ke samping diikuti dengan gerakan *background*. Contoh *game* tipe seperti ini adalah Super Mario, Metal Slug, dan sebagainya.

2.3.3. Komponen *Game*

Di dalam sebuah *game*, terdapat komponen-komponen dalam *game*:

1. Grafik

Grafik merupakan suatu hal yang penting dalam *game*, dimana grafik ini akan membuat *game* menjadi daya tarik user untuk memainkan *game*. pada awalnya

game menggunakan grafik monokrom dengan layar hitam putih sampai sekarang sudah menggunakan teknologi 3D, semakin bagus grafik maka *game* tersebut semakin seperti dalam dunia nyata.

2. Suara

suara menjadi komponen *game* karena dengan adanya suara pada *game* maka *game* tersebut akan menarik sehingga kita dapat merasakan kondisi dalam *game* tersebut. selain itu, dengan adanya suara, kita tidak bosan bermain *game*.

3. AI(kecerdasan Buatan)

Kecerdasan buatan atau AI(artificial Inteligence) menjadi komponen *game* karena dengan hal tersebut *game* menjadi lebih menantang. Dengan menggunakan AI, *game* dapat berpikir untuk mengalahkan pemain sehingga membuat *game* lebih seru.

4. Skenario *Game*

Skenario *game* menjadi komponen *game* karena dengan adanya skenario dalam *game*, si pemain seolah-olah bermain sesuai dengan petunjuk yang ada di *game* tersebut, sehingga dapat dikatakan *game* tersebut memiliki cerita sehingga membuat pemain lebih asyik memainkan *game* tersebut.

5. *Multiplayer*

Multiplayer bisa menjadi komponen *game* karena dengan hal tersebut, pemain dapat bermain dengan teman-temannya dan orang lain sehingga *game* itu dapat dimainkan lebih dari 1 orang. dengan kata lain, *multiplayer* adalah sebuah mode dalam *game* dimana *game* dimainkan oleh pemain lainnya sehingga bermain *game* secara bersama-sama.

2.4 *Game* Edukasi

Education games (permainan edukatif) **menurut Andang Ismail (2006:120)** *Permainan edukatif adalah suatu kegiatan yang sangat menyenangkan dan dapat merupakan cara atau alat pendidikan yang bersifat mendidik. Dengan kata lain, permainan edukatif merupakan sebuah bentuk kegiatan mendidik yang dilakukan dengan menggunakan cara atau alat yang bersifat mendidik pula.*

Sehingga permainan edukatif bermanfaat untuk meningkatkan kemampuan berbahasa, berpikir, serta bergaul dengan lingkungannya. Disamping itu, permainan edukatif juga bermanfaat untuk menguatkan dan menerampilkan anggota badan si anak, mengembangkan kepribadian, mendekatkan hubungan antara pengasuh dengan anak didik, serta menyalurkan kegiatan anak.

Dari uraian diatas dapat disimpulkan bahwa *education games* (permainan edukatif) adalah sebuah permainan yang digunakan dalam proses pembelajaran dan dalam permainan tersebut mengandung unsur mendidik atau nilai-nilai pendidikan.Selain itu, untuk pemilihan permainan, diusahakan agar seluruh aspek yang dimiliki anak dapat berkembang dengan baik,baik dari segi *kognitif*, efektif dan juga psikomotorik.*Game* edukasi sangat menarik untuk dikembangkan.

Ada beberapa kelebihan dari *game* edukasi dibandingkan dengan metode edukasi konvensional. Salah satukelebihan utama *game* edukasi adalah pada visualisasi dari permasalahan nyata.Massachussets Insitute of Technology (MIT) berhasil membuktikan bahwa *game* sangat berguna untuk meningkatkan logika

dan pemahaman pemain terhadap suatu masalah melalui proyek *game* yang dinamai Scratch.

2.4.1. Kriteria *Game* Edukasi

Menurut Hurd dan Jenuings (2009: 5), perancangan *Education game* yang baik haruslah memenuhi kriteria dari *education game* itu sendiri. Berikut ini adalah beberapa kriteria dari sebuah *education game*, yaitu:

a. 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*.

b. 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.

c. 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.

d. 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.

e. Relevan *(Relevance)*

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

f. Objektifitas *(Objectives)*

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

g. 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.

2.4.2. *English Education Game*

English Education Game adalah permainan yang bermanfaat untuk melatih siswa belajar bahasa Inggris.*Game* ini dikhususkan untuk siswa sekolah tingkat dasar,dikarenakan banyak siswa yang tidak begitu mengerti bahasa Inggris,maka dari itu *game* ini dirancang untuk membantu siswa belajar bahasa Inggris dan melatih kecerdasaan siswa.

2.5. Java

Java adalah bahasa pemrograman berorientasi objek murni yang dibuat berdasarkan kemampuan-kemampuan terbaik bahasa pemrograman objek sebelumnya (C++, Ada,Simula). Java diciptakan oleh James Gosling, developer dari Sun Microsystems pada tahun 1991.

Adapun Karakteristik *Java* antara lain :

a. Sederhana (*Simple*)
b. Berorientasi Objek (*Object Oriented*)
c. Terdistribusi (*Distributed*)
d. Interpreted
e. Robust
f. Aman (*Secure*)
g. Architecture Neutral
h. Portable
i. Performance
j. Multithreaded
k. Dinamis

2.5.1. Sejarah Singkat Java

Bahasa Pemrograman Java Terlahir untuk pertama kalinya dari sebuah proyek asuhan *Sun Microsystem* yang dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy beserta dengan sembilan rekan mereka dari *Sun Microsystem.* proyek ini berjalan selama 18 bulan, dari awal tahun 1991 hingga

musim panas 1992. Nama Peoyeknya adalah "*The Green Project*" Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang.

Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program *Java* Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai **"*7″ (Star Seven).**

Setelah era STAR 7 berakhir, sebuah anak perusahaan TV kabel Tertarik akan hal tersebut. maka dari itu bersama beberapa orang dari *the green project* mulai memusatkan perhatiannya pada hal ini. perusahaan ini akhirnya mengalami peningkatan yang signifikan dalam rentangan waktu yang cukup cepat, diaman jumlah karyawan yang terdahulu hanya 13 orang kini telah mencapai 70 orang. pada era ini penggunaan internet mulai dirintis guna demi kelancaran hubungan komunikasi mereka dalam menyampaikan ide-ide ataupun masukan-masukan yang bermanfaat untuk proyek tersebut.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.02 dibuka. Kesuksesan mereka ini di pubilkasikan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 mei 1995.

2.6 *Java Development Kit* (JDK)

Java Development Kit berisi perangkat lunak dan alat-alat yang Anda butuhkan untuk mengkompilasi, debug, dan menjalankan applet dan aplikasi bahwa Anda telah ditulis menggunakan bahasa pemrograman Java. JDK *(Java Development Kit)* sebagai komponen utama kumpulan alat pemrograman, termasuk Javac, jar, pengarsipan, yang paket perpustakaan kelas yang terkait ke dalam *file* JAR tunggal. Alat ini juga membantu mengelola *file* JAR, Javadoc - generator dokumentasi, yang secara otomatis menghasilkan dokumentasi dari komentar kode sumber, JDB - debugger, JPS - alat status proses, yang menampilkan informasi proses untuk proses Java saat ini, Javac yang disassembler *file* kelas , dan komponen lain begitu banyak.

JDK juga dilengkapi dengan *Java Runtime Environment* yang lengkap, biasanya disebut runtime pribadi. Ini terdiri dari *Java Virtual Machine* dan semua perpustakaan kelas hadir dalam lingkungan produksi, serta library tambahan hanya berguna untuk pengembang, dan seperti perpustakaan internasionalisasi dan perpustakaan IDL. Apa yang baru dalam versi ini: Versi 7 Update 2 berisi dukungan untuk Oracle Solaris 11 dan dukungan untuk Firefox 5 dan kemudian.

2.6.1. Komponen *Java Development Kit* (JDK)

Komponen *Java Development Kit* antara lain compiler Javac, interpreter Java disebut juga *Java virtual machine* atau *Java runtime environment*, applet viewer (*appletviewer*), debugger (jdb), *Java Class Library* (jcl), header dan stub generator (Javah), dan yang penting yaitu *Java documentation* (Javadoc).

Penjelasan penggunaan komponen JDK :

1. *Kompilator (*Javac*)*

   Berfungsi untuk kompilasi *file* source code : *.Java menjadi *.*Class* Syntax umum: Javac nama_*file*.*Java*.

2. *Interpreter (*Java*)*

   Bertugas untuk menjalankan bytecode (*.*Class*) Syntax umum :*Java* nama_*file*.*Class* .

3. *Applet Viewer*

   Digunakan untuk menjalanakan applet viewer , namun sekarang sudah digantikan browser. Syntax umum : appletviewer nama_*file*.html.

4. *Java Debugger*

   Bertugas untuk melakukan debugging aplikasi Java. Syntax umum : jdb option.

5. *Java Class File Diassembler* (*Java*p)

   Bertugas membuat daftar method dan attribute public dari suatu kelas Syntax Javap namaKelas.

6. *Java Header and Stub Generator*

   Bertugas menerjemahkan bahasa yang ditulis dalam bahasa Java menjadi bahasa pemrograman C. Syntax umum : *Java*h namaKelas

7. *Java Documentation Generator*

   Menampilkan pustaka kelas, interface, constructor, dan method standard yang telah dibuat vendor.Dari hasil instalasi JDK, dokumentasi ini dapat dilihat pada C:\Java\docs\api\index.html dan Dari hasil instalasi NetBeans, dapat dilihat pada C:\Program *File*s\NetBeans3.6\doc\junit\index.html.

8. *Source Code Java API*

Source code ini dapat diperoleh dari *file* src.zip. Untuk pemrogram pemula,lingkungan pemrograman Java dapat diringk menjadi :

1. *Editing source code* menggunakan editor teks, seperti Notepad atau TextPad
2. *Compiling* menggunakan *keyword* Javac melalui command prompt (dapat juga dari *editor teks* seperti TextPad)
3. *Executing* menggunakan :

a. Command prompt untuk *Java application*(atau dari editor teks TextPad)

b. *Browser* atau *appletviewer* untuk *Java* applet.

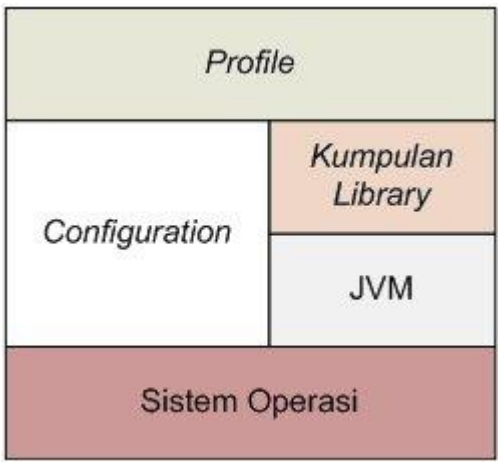
2.7. J2ME

Menurut M.Shalahuddin dan Rosa A.S (2010 : 3 - 4) *Micro Edition atau yang biasa disebut J2ME adalah lingkungan pengembangan yang didesain untuk meletakkan perangkat lunak Java pada barang elektronik beserta perangkat pendukungnya*.

Pada J2ME, jika perangkat lunak berfungsi baik pada sebuah perangkat maka belum tentu juga berfungsi baik pada perangkat yang lainnya.J2ME belum tentu juga berfungsi baik pada perangkat yang lainnya.J2ME membawa Java kedunia informasi,komunikasi,dan perangkat komputasi selain perangkat komputer *desktop* yang biasanya lebih kecil dibandingkan perangkat komputer *desktop*.J2ME biasa digunakan pada telepon selular,pager,personal digital assistants (PDA’s) dan sejenisnya.

J2ME adalah bagian dari J2SE, karena itu tidak semua *library* yang ada pada J2SE dapat digunakan pada J2ME.Tetapi J2ME mempunyai beberapa *library* khusus yang tidak dimiliki J2SE.

Arsitektur J2ME dapat dilihat pada gambar berikut :



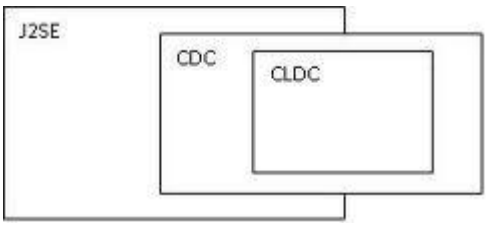
Gambar 2.1 Arsitektur *J2me*

Sumber : M.Shalahuddin dan Rosa A.S Java2.*Pemrograman J2ME* (2010 : 4).

Tekonologi J2ME juga memiliki beberapa keterbatasan,terutama jika diaplikasikan pada ponsel.J2ME sangat tergantung pada perangkat(device) yang digunakan,bisa dari segi merk ponsel,maupun kemampuan ponsel ,dan dukungannya terhadap teknologi J2ME.Misalnya,jika sebuah ponsel tidak memiliki kamera maka jelas J2ME pada ponsel tersebut tidak dapat mengakses kamera.keterbatasan lainnya adalah pada ukuran aplikasi,karena memori pada ponsel sangat terbatas.

Configuration merupakan Java *library* minimum dan kapabilitas yang dipunya oleh para pengembang J2ME,yang maksudnya sebuah mobile device dengan kemampuan Java akan dioptimalkan untuk menjadi sesuai.*Configuration h*anyalah mengatur hal-hal tentang kesamaan sehingga dapat dijadikan ukuran kesesuaian antar – *device*.Misalnya sebuah lampu sepeda dirancang sedemikian rupa sehingga dapat digunakan oleh berjenis-jenis sepeda.Dalam J2ME telah didefinisikan dua buah konfigurasi yaitu CLDC (*Connected Limited Device*

Configuration) untuk perangkat kecil dan CDC (*Connected Device Configuration*) untuk perangkat yang lebih besar.Linkup CLDC dan CDC dapat dilihat pada gambar berikut :



Gambar 2.2 Lingkup *Configuration*

Sumber : M.Shalahuddin dan Rosa A.S Java2.*Pemrograman J2ME* (2010 : 5).

Profile berbeda dengan *Configuration,profile* membahas sesuatu yang spesifik untuk sebuah perangkat.Sebagai contoh misalnya,sebuah sepeda dengan merk tertentu tentu mempunyai cirri spesifik dengan sepeda lain.Dalam J2ME terdapat dua buah profile yaitu MIDP dan *Foundation Profile*.

2.7.1. *Connected Limited Device Configuration*(CLDC)

CLDC atau Co*nnected Limited Device Configuration* adalah perangkat dasar dari J2ME,spesifikasi dasar yang berupa *library* dan API yang diimplementasikan pada J2ME,seperti yang digunakan pada telepon selular,pager, dan PDA.Perangkat tersebut dibatasi dengan keterbatasan memori,sumber daya dan kemampuan memproses.Spesifikasi CLDC pada J2ME adalah spesifikasi minimal dari *package* kelas dan sebagian fungsi *Java Virtual Machine* yang dikurangi agar dapat diimplementasikan dengan keterbatasan sumber daya pada alat-alat tersebut,JVM yang digunakan disebut KVM(*Killobyte Virtual Machine*).

2.7.2. *Connected Device Configuration* (CDC)

CDC atau *Connected Device Configuration* adalah Spesifikasi dari konfigurasi J2ME.CDC merupakan komunitas proses pada Java yang memiliki standarisasi.CDC terdiri dari *virtual machine* dan kumpulan *library* dasar untuk dipergunakan pada *profile* industri.Implementasi CDC pada J2ME adalah *source code* yang menyediakan sambungan dengan macam-macam *platform*.

Berikut perbandingan antara CLDC dan CDC :

Tabel 2.1

Perbandingan antara CLDC dan CDC

CLDC	CDC
Mengimplemantasikan sebagian dari J2SE	Mengimplementasikan seluruh fitur J2SE
JVM yang digunakan adalah KVM	JVM yang digunakan adalah CVM
Digunakan pada perangkat genggam (handphone,PDA,twoway pager)dengan memori terbatas(160-152 KB)	Digunakan pada perangkat genggam(internet TV,Nokia Communicator,car TV)dengan memori minimal 2 MB.
Prosesor : 16/32 bit	Prosesor : 32 bit

Sumber : M.Shalahuddin dan Rosa A.S Java2.*Pemrograman J2ME* (2010 : 7).

2.7.3. MIDP

MIDP atau *Mobile Information Device Profile* adalah spesifikasi untuk sebuah profil J2ME.MIDP memiliki lapisan diatas CLDC,API tambahan untuk daur hidup aplikasi, antarmuka, jaringan dan penyimpangan persisten.

Java MIDP-2.0 adalah bahasa pemrograman yang dibuat oleh Sun untuk perangkat dengan memori dan kapasitas kecil seperti ponsel. Fitur MIDP-2.0 atau dikenal juga sebagai J2ME (*Java 2 Micro Edition*) cukup lengkap baik untuk *networking*nya, tampilan dan kemampuan khusus untuk membangun *game*.

Adapun spesifikasi MIDP 2.0 merupakan pengembangan dari MIDP 1.0. Kebutuhan memori pun bertambah, minimal 256kB memori *nonvolatile* dan 128kB memori *volatile* untuk JVM *heap space*. Untuk spesifikasi lainnya seperti:

1. Keamanan *Networking* dengan HTTPS
2. Push aplikasi dengan *Class Javax.microedition.io.PushRegistry*.
3. *Communication serial Port* telah standarisasi.
4. Pengembangan aplikasi *wireless* dengan *Over-the-Air provisioning* (OTA).
5. Lebih baik untuk aplikasi *game* dengan *package Javax.microedition.lcdui. game.*

2.7.4. *Killobyte Virtual Machine* (KVM)

KVM adalah paket JVM yang di desain untuk perangkat yang kecil.KVM mendukung sebagian dari fitur-fitur JVM,seperti misalnya KVM tidak mendukung operasi floating-point dan finalisasi objek.KVM diimplemtasikan dengan menggunakan C sehingga sangat mudah beradaptasi pada tipe *platform* yang berbeda.

2.8. Komponen-komponen J2ME

J2ME terdiri atas beberapa kompenen, yaitu :

a. *Java Virtual Machine* (JVM)

Komponen ini digunakan untuk menjalankan program-program Javapada emulator.

b. Java API (*Application Programming Interface*)

Komponen ini merupakan kumpulan *library* untuk menjalankan dan mengembangkan program Java pada perangkat yang ada.

c. Tools

Emulator untuk pengembangan aplikasi Java untuk menjalankan Midlet, programmer tidak perlu memiliki dan mencobanya pada ponsel, cukup dengan emulator dari ponsel yang dapat dijalankan pada PC.

Dalam J2ME dibagi menjadi dua bagian yang dikenal dengan istilah configuration dan profile.Dan buah istilah tersebut sangat penting dalampengembangan aplikasi wireless dengan Java.

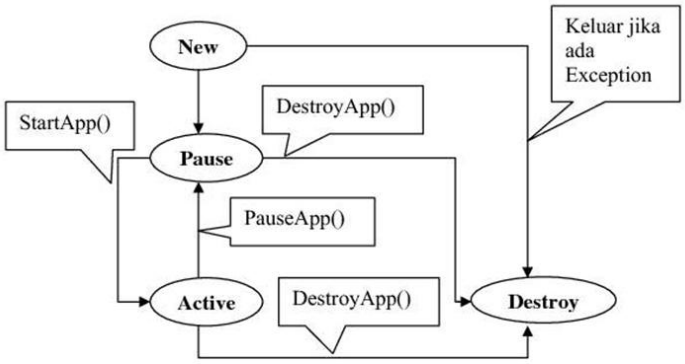
2.9. *Class*

Class yang akan digunakan pada aplikasi ini adalah :

1. *Class* yang berhubungan dengan MIDP adalah :

a. *Javax.microedition.midlet*

Javax.microedition.midlet adalah *Class* yang menyediakan interaksi antar aplikasi dengan lingkungan dimana aplikasi tersebut berjalan.*Class* ini mempunyai 3 kondisi atau *state*.tersebut yang akan menjaga kestabilan tindakan dari suatu aplikasi 3 *state* itu adalah *Pause,active,* dan *destroy.Pause* adalah *state* dimana aplikasi dalam keadaan berhenti sementara untuk menunggu proses selanjutnya.*Active* adalah *state* dimana aplikasi berjalan secara normal.*Destroy* adalah stae dimana aplikasi ditutup/dimatikan.Dari ketiga *state* inilah dapat dibentuk lifecycle dari sebuah aplikasi MIDP,seperti pada gambar dibawah ini.



Gambar 2.3 *Lifecycle* dari sebuah aplikasi MIDP

Sumber : *Sun Microsystem.Mobile Information* Device pro*file*(JSR-37).Palo Alto :*Sun Microsystem*,inc.(2000:121)

b. *Javax.Microedition.lcdui*

Javax.microedition.lcdui adalah Class UI API yang menyediakan fasilitas *interface* bagi *user* untuk aplikasi MIDP.Pusat abstraksi dari *Class* ini adalah pada layar.Layar adalah suatu objek alat encapsulasi,terutama pada *rendering* grafik *input user*.Hanya satu layar yang dapat tampil disaat tersebut dan user dapat menjelajah melalui *item* yang ada pada layar.MIDP UI terbagi menjadi 2 bagian API,yaitu *High Level* API dan *Low Level* API.*High Level* API biasa digunakan pada aplikasi yang bagian *client* berjalan pada MIDP.Untuk aplikasi semacam ini,kecocokan untuk digunakan alat jenis lain sangat tinggi.*High Level* API dapat diasumsikan adalah API yang harus dapat beradaptasi dengan perangkat keras peralatan lain serta UI yang lain.

Implementasi dari *High Level* API adalah layar dengan *subClass screen.Low Level API* didesain untuk aplikasi yang membutuhkan ketepatan penempatan dan pengolahan elemen grafik,serta akses pada *low level input event*.Dengan memakai *Low Level* API,aplikasi dapat melakukan control penuh dalam penggambaran

grafik,merespon pada event sederhana seperti tombol ditekan dan dilepas,akses pada tombol yang konkrit dan *input* lainnya.Implementasi dari *Low Level* API *Class canvas* dan *graphic.*

c. *Class* yang berhubungan dengan CLDC adalah :

- *Javax.microedition.io*

  Javax.microedition.io adalah *Class* untuk koneksi secara umum pada CLDC.*Class* ini yang menangani manajemen jaringan pada sebuah aplikasi.

- *Javax.util*

  Javax.util adalah *Class* yang berisi kumpulan *framework*,turunan dari *Class* ,fasilitas tanggal dan waktu dan macam - macam *Class utility.Class* ini merupakan Class yang juga terdapat pada Java 2 Standard edition(J2SE).

- *Javax.io*

  Javax.io adalah *Class* yang menyediakan sistem *input* dan *output* melalui data stream.*Class* ini menangani bidang data *stream* pada suatu aplikasi,juga menangani pengubahan data pada bentuk *binary stream*.

d. *Class* yang berhubungan dengan Bluetooth adalah :

- *Javax.bluetooth*

2.10. *Record Management System* (RMS)

RMS merupakan implementasi *database* sederhana untuk digunakan dalam MIDlet Java yang berorientasi *record*. Jika aplikasi kita ingin memanfaatkan fasilitas RMS ini, kita harus mengimport paket *Java*x.microedition.rms.*.

Didalamnya terdapat *Class – Class* untuk manajemen RMS salah satunya *Class Record Store* yang menyediakan fungsi - fungsi untuk menambah, mengubah, menghapus data dan manipulasi lainnya.

- **Paket *Javax*.microedition.rms**

Paket ini menyediakan fungsi-fungsi untuk implementasikan *database* RMS pada MIDlet. Paket ini terdiri atas:

a. Interface

1. ***RecordComparator***

   Mendefinisikan komparator atau pembanding ketika melakukan komparasi/perbandingan antara 2 *record*.

2. ***RecordEnumerator***

   Merepresentasikan *enumerator record* yang bersifat *biderictional*(dua arah).

3. ***RecordFilter***

   Mendefinisikan *filter* yang digunakan untuk menganalisis sebuah *record* pada *RecordStore*, apakah memenuhi kriteria tertentu sesuai dengan filter yang diberikan atau tidak.

4. ***RecordListener***

   Mendeteksi *record* yang ditambahkan, diubah, atau dihapus dalam sebuah *RecordStore*.

b. Kelas

1. ***RecordStore***

   Merupakan kelas utama yang harus Anda gunakan untuk membuat objek *RecordStore*.

c. Exception

1. ***InvalidRecordIDException***

   Jika ditemukan *RecordId* yang tidak *valid*, eksepsi ini akan dilemparkan.

2. ***RecordStoreException***

   Eksepsi umum yang digunakan jika terjadi kesalahan pada manajemen Objek *RecordStore*.

3. ***RecordStoreFullException***

   Jika tempat penyimpanan *RecordStore* sudah penuh.

4. ***RecordStoreNotFoundException***

   Jika tempat penyimpanan *RecordStore* tidak ditemukan.

5. ***RecordStoreNotOpenException***

   Jika tempat penyimpanan *RecordStore* tidak dibuka (*opened*).

2.11. J2ME *Wireless Toolkit*

2.11.1. *KToolbar*

KToolbar merupakan bagian utama dari aplikasi untuk pengembangan Java MIDP atau lebih dikenal dengan istilah MIDlet. *KToolbar* merupakan lingkunganpengembangan minimal yang disediakan oleh J2ME Wireless Toolkit untuk pengembangan MIDlet.Untuk melakukan pengembangan aplikasi MIDlet melaui *KToolbar* , hal pertama yang harus dilakukan adalah membuat proyek.

Proyek ini secara fisik akandiletakkan pada **C:\WTK2.5\apps\{nama-proyek}**dengan asumsi J2ME Wireless Toolkit diinstal pada C:\WTK20\. Proyek yang dibuat terdiri atas direktori-direktori berikut:

a. Src , adalah direktori yang berisi berkas-berkas kode Java.
b. Bin, adalah direktori yang berisi berkas-berkas JAD, JAR dan manifest.
c. Res , adalah direktori yang berisi berkas-berkas sumber, misalnya gambar.
d. Lib, adalah direktori yang berisi berkas-berkas kelas Java yangdiperlukan dalam format gambar atau zip.

2.12. NetBeans

NetBeans mengacu pada dua hal, yakni *Platform* untuk pengembangan aplikasi desktop *Java*, dan sebuah Integrated Development Environment (IDE) yang dibangun menggunakan *Platform* NetBeans. ***Platform* NetBeans** memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut 'modul'. Sebuah modul adalah suatu arsip Java (*Java archive*) yang memuat kelas-kelas *Java* untuk berinetraksi dengan NetBeans Open API dan *file* manifestasi yang mengidentifikasinya sebagai modul. Aplikasi yang dibangun dengan modul-modul dapat dikembangkan dengan menambahkan modul-modul baru. Karena modul dapat dikembangkan secara independen, aplikasi berbasis *Platform* NetBeans dapat dengan mudah dikembangkan oleh pihak ketiga secara mudah dan *powerful.*

Pengembangan NetBeans diawali dari Xelfi, sebuah proyek mahasiswa tahun 1997 di bawah bimbingan Fakultas Matematika dan Fisika Universitas Charles, Praha. Sebuah perusahaan kemudian dibentuk untuk proyek tersebut dan

menghasilkan versi komersial NetBeans IDE hingga kemudian dibeli oleh Sun Microsystem pada tahun 1999. Sun kemudian menjadikan NetBeans open source pada bulan Juni tahun 2000. Sejak itu komunitas NetBeans terus berkembang.

2.12.1. *Platform* NetBeans

Platform NetBeans adalah *framework* yang dapat digunakan kembali (*reusable*) untuk menyederhanakan pengembangan aplikasi desktop. Ketika aplikasi berbasis *Platform* NetBeans dijalankan, kelas Main dari *Platform* dieksekusi. Modul-modul yang tersedia ditempatkan di sebuah *registry* di dalam memori, dan tugas *startup* modul dijalankan. Secara umum, kode modul dimuatkan ke dalam memori hanya ketika iadiperlukan.

Aplikasi dapat menginstal modul secara dinamis. Aplikasi dapat memasukkan modul *Update Center* untuk mengijinkan pengguna aplikasi men-download *digitally-signed upgrade* dan fitur-fitur baru secara langsung ke dalam aplikasi yang berjalan. Penginstalan kembali sebuah upgrade atau rilis baru tidak memaksa pengguna untuk men-download keseluruhan aplikasi lagi.

Platform NetBeans menawarkan layanan-layanan yang umum bagi aplikasi desktop, mengijinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi. Fitur-fitur yang disediakan oleh *Platform* NetBeans: Manajemen antarmuka (missalnya : menu&toolbar).

1. Manajemen pengaturan pengguna.
2. Manajemen penyimpanan (menyimpan dan membuka berbagai macam data).
3. Manajemen jendela.

4. Wizard framework (mendukung dialog langkah demi langkah).

2.13. Pengujian *Software*

Menurut Roger S. Pressman (2002: 596) " *Pengujian sistem adalah sederetan pengujian yang berbeda yang tujuan utamanya adalah sepenuhnya menggunakan sistem berbasis komputer.*"

Pengujian perangkat lunak dilakukan untuk mendapatkan suatu perangkat lunak yang layak untuk digunakan. Pengujian merupakan proses menjalankan dan mengevaluasi sebuah perangkat lunak secara manual maupun otomatis dengan tujuan untuk menguji apakah perangkat lunak tersebut sudah sesuai dengan apa yang diharapkan.

Ada Banyak teknik pengujian yang dapat digunakan untuk menguji perangkat lunak, diantaranya: Pengujian *Black Box* dan Pengujian *White Box*.

1. Pengujian *Black Box*
   a. Digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang.
   b. Kebenaran perangkat lunak yang diuji hanya dilihat berdasarkan keluaran yang dihasilkan dari data atau kondisi masukan yang diberikan untuk fungsi yang ada tanpa melihat bagaimana proses untuk mendapatkan keluaran tersebut.
   c. Dari keluaran yang dihasilkan, kemampuan program dalam memenuhi kebutuhan pemakai dapat diukur sekaligus dapat diiketahui kesalahan-kesalahannya.

   Beberapa jenis kesalahan yang dapat diidentifikasi :

   1. Fungsi tidak benar atau hilang.

2. Kesalahan antar muka.
3. Kesalahan pada struktur data (pengaksesan basis data).
4. Kesalahan inisialisasi dan akhir program.
5. Kesalahan performasi.

2. Pengujian *White Box*

a. Digunakan untuk mengetahui cara kerja suatu perangkat lunak secara *internal*.
b. Pengujian dilakukan untuk menjamin operasi-operasi *internal* sesuai dengan spesifikasi yang telah ditetapkan dengan menggunakan struktur kendali dari prosedur yang dirancang.

Pelaksanaan pengujian *white box* :

a. Menjamim seluruh *independent path* dieksekusi paling sedikit satu kali. *Independent path* adalah jalur dalam program yang menunjukkan paling sedikit satu kumpulan proses ataupun kondisi baru.
b. Menjalani *logical decision* pada sisi dan *false*.
c. Mengeksekusi pengulangan (*looping*) dalam batas-batas yang ditentukan
d. Menguji struktur data *internal*.