Bab 4

Low Level User Interface

4.1 Tujuan

Setelah mempelajari bab ini, Pelajar diharapkan mampu untuk :

- Memahami event handling level rendah dalam MIDP
- Menggambar dan menampilkan teks, gambar, garis, kotak, dan sudut
- Menentukan warna, huruf, dan coretan untuk operasi menggambar
- Memahami dan menggunakan class Canvas dan Graphic
- Mengetahui bagaimana menggunakan GAME API
- Menggambar grafik berskala

4.2 Pengenalan

Pada bab sebelumnya, kita telah membahas tentang bagaimana cara membuat user interface level tinggi seperti list, form, dan field input. Mereka bersifat user interface level tinggi dan programmer tidak perlu khawatir tentang menggambar pixel layar atau mengatur posisi teks pada layar. Semua program telah menetapkan jenis komponen dan label elemen. Sistem tersebut akan menangani gambar pada layar, scrolling dan layout.

Satu kelemahan ketika hanya menggunakan komponen user interface level tinggi adalah program tidak memiliki kendali penuh sebuah layar. Ada saat dimana kita ingin menggambar sebuah garis, gambar beranimasi dan mempunyai kendali untuk mengatur pixel pada layar.

Pada bab ini, kita akan berhadapan langsung dengan layar. Kita akan mempelajari class Canvas, dimana akan menjadi pendukung dari proses menggambar kita. Kita juga akan menyelidiki ke dalam class Graphic, dimana memiliki metode untuk menggambar garis, kotak, sudut, dan teks. Kita juga akan membahas huruf, warna dan gambar.

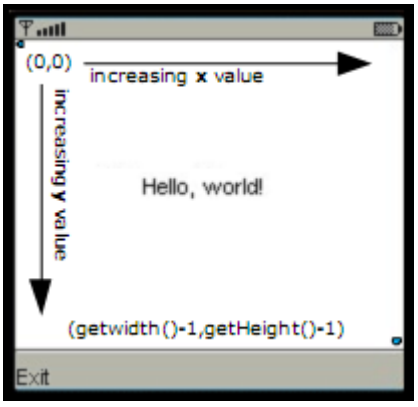
4.3 Canvas

Canvas adalah subclass dari Displayable. Itu adalah sebuah class abstrak yang harus di-extend sebelum sebuah aplikasi dapat menggunakan fungsi-fungsi yang ada. Canvas dapat digabungkan dengan subclass Displayable level tinggi yaitu Screen. Program dapat pindah ke dan dari Canvas dan Screen.

Canvas menggambarkan metode-metode event handling kosong. Aplikasi harus mengesampingkan mereka untuk handle event. Class Canvas menggambarkan sebuah metode abstrak yang disebut paint(). Aplikasi menggunakan class Canvas harus menyediakan sebuah implementasi untuk metode paint().

4.3.1 Sistem Koordinat

Sistem koordinat dari Canvas adalah berbasis nol. Koordinat x dan y dimulai dengan nol. Pojok kiri atas dari Canvas berkoordinat (0,0). Koordinat x bertambah dari kiri ke kanan. Sedangkan koordinat y bertambah dari atas ke bawah. Metode getWidth() dan getHeight() mengembalikan nilai lebar dan tinggi berturut-turut. Pojok kanan bawah pada layar memiliki koordinat (getWidth()-1,getWidth()-1). Setiap perubahan yang terjadi pada ukuran yang diberikan untuk area menggambar pada Canvas dilaporkan kepada aplikasi oleh metode sizeChanged(). Ukuran yang tersedia pada Canvas mungkin saja berubah jika ada pergantian antara mode layar full dan normal atau penambahan dan pengurangan sebuah komponen seperti Command.



Gambar 1: Sistem Koordinat

4.3.2 ”Hello,World!”

Gambar 2: Hello World MIDlet menggunakan canvas

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloCanvasMIDlet extends MIDlet {
  private Display display;
  HelloCanvas canvas;
  Command exitCommand = new Command("Exit", Command.EXIT, 0);
  public void startApp() {
    if (display == null){
      canvas = new HelloCanvas(this, "Hello, world!");
      display = Display.getDisplay(this);
```

```
    }
    display.setCurrent(canvas);
  }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {}
  protected void Quit(){
    destroyApp(true);
    notifyDestroyed();
  }
}

class HelloCanvas extends Canvas implements CommandListener {
  private Command exitCommand = new Command("Exit", Command.EXIT, 0);
  private HelloCanvasMIDlet midlet;
  private String text;
  public HelloCanvas(HelloCanvasMIDlet midlet, String text) {
    this.midlet = midlet;
    this.text = text;
    addCommand(exitCommand);
    setCommandListener(this);
  }
  protected void paint(Graphics g) {
    // membersihkan layar dengan mengisi semua layar dengan warna putih
    g.setColor(255, 255, 255 );
    g.fillRect(0, 0, getWidth(), getHeight());
    // mengatur warna tulisan dengan warna hitam
    g.setColor(0, 0, 0);
    // dan menulis sebuah text
    g.drawString(text,
    getWidth()/2, getHeight()/2,
    Graphics.TOP | Graphics.HCENTER);
  }
  public void commandAction(Command c, Displayable d) {
    if (c == exitCommand)
      midlet.Quit();
  }
}
```

Dengan midlet “Hello,World!”, kita membuat sebuah class yang ber-extend Canvas

```
class HelloCanvas extends Canvas implements CommandListener {
```

Kemudian kita menambahkan perintah (“Exit”) dan mengatur command listener nya.

```
addCommand(exitCommand);
setCommandListener(this);
```

Kita menciptakan command listener dengan mengimplementasikan class CommandListener. Ini berarti membuat class yang memiliki metode commandAction.

```
class HelloCanvas extends Canvas implements CommandListener {
...
public void commandAction(Command c, Displayable d) {
...
```

Inti dari program ini adalah metode paint(). Set pertama dari pemanggilan metode adalah membersihkan layar.

```
g.setColor(255, 255, 255 );
g.fillRect(0, 0, getWidth(), getHeight());
```

Dan kemudian grafik memanggil metode drawString() untuk menampilkan “Hello,World!” pada layar :

```
// mengatur warna tulisan dengan warna hitam
g.setColor(0, 0, 0);
// dan menulis sebuah text
g.drawString(text, getWidth()/2, getHeight()/2,
Graphics.TOP | Graphics.HCENTER);
```

4.3.3 Perintah

Seperti halnya list, textBox, dan form, Canvas juga mempunyai Command yang disediakan dan dapat merespon untuk event Command. Langkah-langkah untuk menambahkan Command ke dalam Canvas adalah :

1. Buatlah objek Command.

```
private Command exitCommand = new Command("Exit", Command.EXIT, 0);
```

2. Gunakan useCommand() untuk menambahkan perintah ke dalam canvas(atau Form, list, text box)

```
addCommand(exitCommand);
```

3. Gunakan setCommandListener() untuk mendaftarkan class mana yang akan mendapat event command untuk perintah dalam Displayable.

```
setCommandListener(this);
```

4. Buatlah sebuah commandListener dengan mengimplementasikan class commandListener dan menyediakan sebuah metode commandAction().

```
class HelloCanvas extends Canvas implements CommandListener {
...
public void commandAction(Command c, Displayable d) {
```

```
  if (c == exitCommand){
    // do something
  }
}
```

4.3.4 Event key

Subclass dari Canvas dapat merespon sebuah event tombol dengan metode-metode sebagai berikut :

keyPressed(int keyCode)	Dipanggil ketika kunci ditekan
keyRepeated(int keyCode)	Dipanggil ketika kunci diulang
keyReleased(int keyCode)	Dipanggil ketika kunci dilepas

Canvas mendefinisikan kode tombol ini : KEY_NUM0, KEY_NUM1, KEY_NUM2, KEY_NUM3, KEY_NUM4, KEY_NUM5, KEY_NUM6, KEY_NUM7, KEY_NUM8, KEY_NUM9, KEY_STAR, and KEY_POUND. Untuk mendapatkan “String” nama sebuah kunci, gunakan metode getKeyName(int keyCode).

4.3.5 Aksi Permainan

Masing-masing kode tombol bisa dipetakan menjadi sebuah aksi game. Sebuah key code bisa dipetakan kepada aksi sebuah game. Class Canvas mendefinisikan aksi game ini : UP, DOWN, LEFT, RIGHT, FIRE, GAME_A, GAME_B, GAME_C, GAME_D. Sebuah program dapat menerjemahkan sebuah key code ke dalam aksi game menggunakan metode getGameAction(keyCode). Metode getKeyCode(int gameAction) mengembalikan key code yang berkaitan dengan suatu aksi game. Sebuah aksi game dapat memiliki lebih dari satu key code yang berkaitan dengannya. Jika terdapat lebih dari satu key code yang berkaitan dengan aksi game, hanya satu key code yang akan dikembalikan.

Sebuah aplikasi perlu menggunakan getGameAction(int keyCode) daripada langsung menggunakan kode tombol yang telah didefinisikan. Secara normal, jika suatu program ingin merespon kunci “UP”, sebaiknya menggunakan kunci KEY_NUM2 atau key code yang spesifik untuk tombol UP. Program menggunakan metode ini tidaklah portable sejak sebuah perangkat memiliki layout kunci yang berbeda dan key code yang berbeda pula. KEY_NUM2 mungkin menjadi kunci “UP” untuk sebuah perangkat, tetapi mungkin juga menjadi kunci “LEFT” untuk perangkat lainnya. GetGameAction() akan selalu mengembalikan “UP”, tidak terikat pada kunci mana yang ditekan selama dia adalah kunci “UP” di dalam konteks dari layout kunci sebuah perangkat.

4.3.6 Event Pointer

Disamping dari event key, program MIDP juga dapat mengatasi event pointer. Hal ini bersifat benar jika sebuah perangkat memiliki sebuah pointer dan hal tersebut diimplementasikan di dalam sistem JAVA pada sebuah perangkat. Metode hasPointerEvents() mengembalikan nilai true jika sebuah perangkat mendukung pointer yang bersifat ditekan dan dilepaskan. Metode hasPointerMotionEvents() mengembalikan nilai true jika sebuah perangkat mendukung event gerakan dari pointer.

```
public boolean hasPointerEvents()
public boolean hasPointerMotionEvents()
```

Sebuah event dapat di-generate oleh aktivitas pointer sebagai berikut : pointerPressed, pointerReleased dan pointerDragged. Sebuah aplikasi mengesampingkan metode-metode yang untuk diperhatikan ketika event ini terjadi. Koordinat (x,y) dari event (ketika pointer ditekan, dilepas atau digeser) ditetapkan didalam metode-metode callback ini.

```
protected void pointerPressed(int x, int y)
protected void pointerReleased(int x, int y)
protected void pointerDragged(int x, int y)
```

4.4 Grafik

Class Graphic adalah class utama untuk menulis teks, menggambar, garis, kotak dan sudut. Dia memiliki metode untuk menentukan warna, huruf, dan coretan.

4.4.1 Warna

Class Display memiliki metode untuk menentukan apakah sebuah perangkat memiliki fasilitas yang mendukung layar berwarna atau layar monochrome pada sebuah perangkat.

`public boolean isColor()`	Mengembalikan nilai true jika mendukung layar berwarna dan sebaliknya.
`public int numColors()`	Mengembalikan nomor warna(atau level abu-abu jika sebuah perangkat tidak mendukung warna) yang didukung sebuah perangkat

Untuk mengatur warna yang digunakan untuk metode grafik berikutnya, gunakan metode setColor(). SetColor() memiliki dua bentuk:

```
public void setColor(int red, int green, int blue)
public void setColor(int RGB)
```

Pada bentuk pertama, Anda menentukan komponen warna merah, hijau, dan biru. Pada bentuk kedua komponen warna ditentukan dalam bentuk 0x00RRGGBB. Pemanggilan setColor(int) pada contoh berikut akan melakukan hal yang sama:

```
int red, green, blue;
...
setColor(red, green, blue);
setColor( (red<<16) | (green<<8) | blue );
```

Metode lainnya untuk memanipulasi warna adalah :

`public int getColor()`	Mengembalikan warna terbaru dalam bentuk integer.
`public int getRedComponent()`	Mengembalikan komponen merah sebagai warna terbaru

`public int` **`getGreenComponent`**`()`	Mengembalikan komponen hijau sebagai warna terbaru
`public int` **`getBlueComponent`**`()`	Mengembalikan komponen biru sebagai warna terbaru
`public int` **`getGrayScale`**`()`	Mengembalikan nilai abu-abu sebagai warna terbaru
`public int` **`getColor`**`()`	Mengembalikan warna terbaru dalam bentuk integer.
`public void` **`setGrayScale`**`(int value)`	Memilih nilai abu-abu untuk mengganti operasi menggambar

4.4.2 Huruf

Sebuah huruf memiliki tiga atribut yaitu bentuk, type, dan ukuran. Huruf tidak diciptakan oleh aplikasi. Sebagai gantinya, sebuah aplikasi meminta sistem untuk memilih model atribut huruf dan sistem mengembalikan huruf yang sesuai dengan model atribut yang diminta. Sistem tidak menjamin akan mengembalikan semua atribut huruf yang dipilih. Jika sistem tidak memiliki huruf yang sesuai dengan permintaan, dia akan mengembalikan sebuah huruf hampir mirip dengan atribut yang diminta.

Huruf adalah sebuah class yang terpisah. Seperti yang dinyatakan di atas, aplikasi tidak menciptakan objek huruf. Sebagai gantinya, metode-metode statis getFont() dan getDefaultFont() digunakan untuk meminta sebuah huruf dari sistem.

`public static Font` **`getFont`**`(int face, int style, int size)`	Mengembalikan sebuah huruf dari sistem yang sesuai dengan atribut
`public static Font` **`getDefaultFont`**`()`	Mengembalikan huruf default yang digunakan oleh sistem
`public static Font` **`getFont`**`(int fontSpecifier)`	Mengembalikan huruf yang digunakan untuk komponen UI level tinggi. FontSpecifier bisa jadi : FONT_INPUT_TEXT atau FONT_STATIC_TEXT

Face adalah salah satu dari FACE_SYSTEM, FACE_MONOSPACE, atau FACE_PROPORTIONAL.

Style bisa jadi STYLE_PLAIN atau kombinasi STYLE_BOLD, STYLE_ITALIC dan STYLE_UNDERLINED. Kombinasi style ditentukan oleh penggunaan bitwise operator OR (|). Sebuah style huruf tebal(bold) dan garis miring(italic) dideklarasikan sebagai : STYLE_BOLD | STYLE_ITALIC

Ukuran huruf bisa jadi : SIZE_SMALL, SIZE_MEDIUM, SIZE_LARGE

Metode ini mengembalikan atribut huruf tertentu:

```
public int getStyle()
public int getFace()
public int getSize()
public boolean isPlain()
public boolean isBold()
public boolean isItalic()
public boolean isUnderlined()
```

4.4.3 Style Coretan

Metode setStrokeStyle(int style) menetapkan style coretan bahwa akan digunakan untuk menggambar garis, sudut, dan kotak. Style coretan tidak mempengaruhi teks, gambar, dan operasi mewarnai.

`public void setStrokeStyle(int style)`	Mengatur style coretan untuk menggambar garis, sudut, kotak, dll
`public int getStrokeStyle()`	Mengembalikan style coretan terbaru

Nilai valid untuk style adalah SOLID dan DOTTED.

4.4.4 Clipping

Suatu bidang clipping adalah suatu kotak di dalam objek Graphics yang ada. Setiap operasi grafik hanya akan mempengaruhi pixel-pixel didalam area clip. Pixel yang berada diluar clipping tidak akan dipengaruhi oleh setiap operasi grafik.

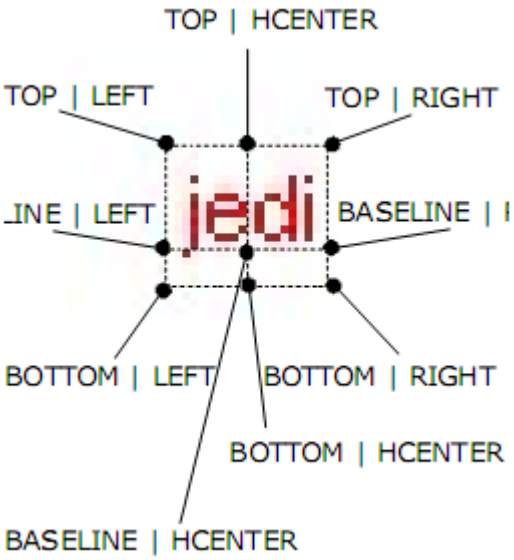
`public void **setClip**(int x, int y, int width, int height)`	Mengatur area clip yang tersedia menjadi kotak, ditentukan oleh koordinat
`public int **getClipX**()`	Mengembalikan offset X dari area clip yang tersedia, sehubungan dengan awal mula dari konteks grafik ini
`public int **getClipY**()`	Mengembalikan offset Y dari area clip yang tersedia
`public int **getClipWidth**()`	Mengembalikan lebar dari area clip yang tersedia
`public int **getClipHeight**()`	Mengembalikan tinggi dari area clip yang tersedia

4.4.5 Anchor Points

Teks digambar sesuai dengan sebuah anchor points. Metode drawString() mengharap sebuah koordinat (x,y) sesuai dengan anchor points.

```
public void drawString(String str, int x, int y, int anchor)
```

Anchor harus suatu kombinasi horisontal yang konstan (LEFT,HCENTER, atau RIGHT) dan vertikal yang konstan (TOP, BASELINE, atau BOTTOM). Horisontal dan vertikal yang konstan harus dikombinasikan mengunakan operator bitwise OR (|). Ini berarti menggambar teks berhubungan dengan baseline dan horisontal tengah akan membutuhkan sebuah nilai anchor BASELINE | HCENTER.



Gambar 3: titik anchor teks

4.4.6 Menggambar Teks

Metode untuk menggambar teks dan karakter adalah :

`public void` **`drawString`**`(String str, int x, int y, int anchor)`	Menggambar teks dalam str menggunakan warna dan huruf yang tersedia. (x,y) adalah koordinat titik anchor
`public void` **`drawSubstring`**`(String str, int offset, int len, int x, int y, int anchor)`	Sama seperti drawString, kecuali ini hanya akan menggambar substring dari offset (berbasis nol) dengan panjang length.
`public void` **`drawChar`**`(char character, int x, int y, int anchor)`	Menggambar karakter dengan warna dan huruf yang tersedia
`public void` **`drawChars`**`(char[] data, int offset, int length, int x, int y, int anchor)`	Menggambar karakter dalam data array karakter, dimulai dari indeks offset dengan panjang length

Berikut adalah beberapa metode dari Font yang berguna dalam menggambar teks:

`public int` **`getHeight`**`()`	Mengembalikan tinggi teks dalam huruf ini. Tinggi dikembalikan termasuk spasi ekstra. Hal ini memastikan bahwa dua teks digambar dengan jarak ini dari titik anchor ke titik anchor lainnya akan berisi cukup ruang antara dua baris teks.
`public int` **`stringWidth`**`(String str)`	Mengembalikan lebar total dalam pixel dari ruang yang ditempati oleh string ini jika digambar menggunakan huruf ini
`public int` **`charWidth`**`(char ch)`	Mengembalikan lebar total dalam pixel dari ruang yang ditempati oleh karakter ini jika digambar menggunaka huruf ini
`public int`	Mengembalikan jarak dalam pixel antara TOP dan BASELINE pada teks,

getBaselinePosition()	berdasarkan pada huruf ini

```
g.setColor(255, 0, 0); // merah
g.drawString("JEDI",
getWidth()/2, getHeight()/2,
Graphics.TOP | Graphics.HCENTER);
g.setColor(0, 0, 255); // biru
Font font = g.getFont();
g.drawString("Java Education & Development Initiative",
getWidth()/2, getHeight()/2+font.getHeight(),
Graphics.TOP | Graphics.HCENTER);
```

Gambar 4: Hasil operasi drawString()

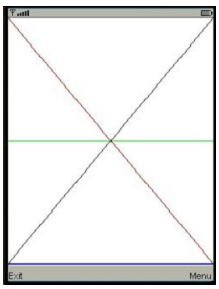
4.4.7 Menggambar garis

Satu-satunya metode grafik untuk menggambar garis didefinisikan sebagai :

```
public void drawLine(int x1, int y1, int x2, int y2)
```

Metode ini menggambar sebuah garis menggunakan warna yang tersedia dan coretan antara koordinat (x1,y1) dan (x2,y2).

```
g.setColor(255, 0, 0); // red
// garis dari pojok kiri atas ke pojok kanan bawah layar
g.drawLine(0, 0, getWidth()-1, getHeight()-1);
g.setColor(0, 255, 0); // green
// garis horisontal pada tengah layar
g.drawLine(0, getHeight()/2, getWidth()-1, getHeight()/2);
g.setColor(0, 0, 255); // blue
// garis horisontal pada bawah layar
g.drawLine(0, getHeight()-1, getWidth()-1, getHeight()-1);
g.setColor(0, 0, 0); // black
// garis dari pojok kiri bawah ke pojok kanan atas layar
g.drawLine(0, getHeight()-1, getWidth()-1, 0);
```



Gambar 5: hasil pemanggilan metode drawLine()

4.4.8 Menggambar kotak

Metode grafik untuk menggambar kotak adalah :

```
public void drawRect(int x, int y, int width, int height)
public void drawRoundRect(int x, int y, int width, int height, int arcWidth,
int arcHeight)
public void fillRect(int x, int y, int width, int height)
public void fillRoundRect(int x, int y, int width, int height, int arcWidth,
int arcHeight)
```

Metode drawRect() menggambar sebuah kotak dengan pojok kiri atas pada koordinat (x,y) dan luas area (width+1 x height+1). Parameter yang sama ada bersama drawRoundRect(). Parameter tambahan arcWidth dan arcHeight adalah diameter horisontal dan vertikal dari busur dari keempat sudut.

Jika Anda akan mengenali, definisi drawRect dan drawRoundRect menetapkan lebar dari kotak yang digambar pada layar adalah dengan width+1 dan tingginya dengan height+1. Hal ini sangat tidak intuitif, tetapi seperti itulah spesifikasi MIDP menggambarkan metode ini. Untuk meng-agravate tidak konsistensi dari “off-by-one” ini, metode fillRect dan fillRoundRect hanya mengisi sebuah area kotak (width x height). Anda akan melihat ketidakcocokan ini jika anda memasukkan parameter yang sama untuk drawRect dan fillRect (dan drawRoundRect vs fillRoundRect). Sisi kanan dan bawah dari kotak digambar oleh kepalsuan drawRect di luar area yang diisi oleh fillRect.

```
// menggunakan tinta hitam untuk drawRect
g.setColor(0, 0, 0);
g.drawRect(8, 8, 64, 32);
// menggunakan tinta kuning untuk fillrect
// untuk menampilkan perbedaan antara drawRect dan fillrect
g.setColor(255, 255, 0);
g.fillRect(8, 8, 64, 32);
```

Gambar 6: hasil dari penggunaan parameter yang sama untuk drawRect dan fillRect

```
// mewarnai warna pena dengan warna hitam
g.setColor(0, 0, 0);
// menggambar kotak pada (4,8) dengan lebar 88 dan tinggi 44
// kotak pada kiri atas
g.drawRect(4,8,88,44);
// elips pada kanan atas
g.drawRoundRect(108,8,88,44,18,18);
// kotak pada kiri bawah
g.fillRect(4,58,88,44);
// elips pada kanan bawah
g.fillRoundRect(108,58,88,44,18,18);
```

Gambar 7:drawRect(), fillRect(), drawRoundRect(), dan fillRoundRect()

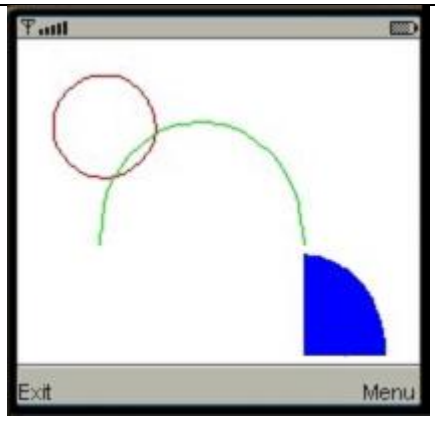
4.4.9 Menggambar Sudut

Metode untuk menggambar bundar atau eclips adalah :

`public void` **`drawArc`**`(int x,int y, int width, int height, int startAngle, int arcAngle)`	Menggambar arc dengan pusat pada (x,y) dan dimensi (width+1 x height+1). Arc digambar mulai dari startAngle dan extend untuk derajat arcAngle. 0 derajat terletak pada jarum jam 3.
`public void` **`fillArc`**`(int x, int y, int width, int height, int startAngle, int arcAngle`	Mewarnai bidang bundar dan eclips yang telah dibuat dengan warna yang tersedia.

```
g.setColor(255, 0, 0);
g.drawArc(18, 18, 50, 50, 0, 360); // menggambar sebuah lingkaran
g.setColor(0, 255, 0);
```

```
g.drawArc(40, 40, 100, 120, 0, 180);
g.setColor(0, 0, 255);
g.fillArc(100, 200, 80, 100, 0, 90);
```



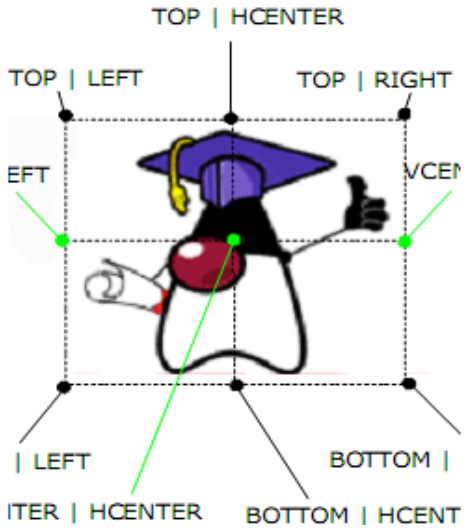
Gambar 8: Hasil pemanggilan metode drawArc dan fillArc

4.4.10 Melukis gambar

Gambar digambar dengan metode drawImage()

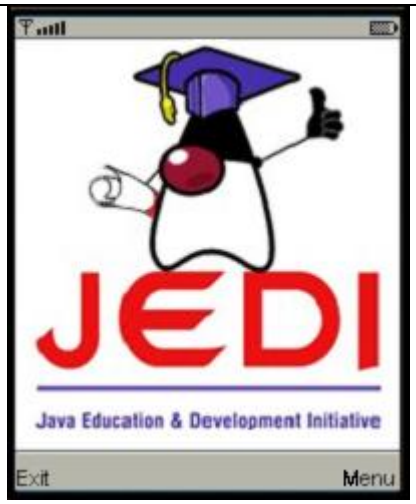
```
public void drawImage(Image img, int x, int y, int anchor)
```

Selama dengan drawString, x dan y adalah koordinat titik anchor. Perbedaannya adalah bahwa vertikal anchor tetap adalah VCENTER yang berdasar BASELINE. Anchor harus berupa kombinasi dari horizontal constant(LEFT, HCENTER atau RIGHT) dan vertical constant (TOP,VCENTER atau BOTTOM). Horizontal dan Vertical Constants dikombinasikan dengan menggunakan operator bitwise OR(|).



Gambar : Image Anchor Points

```
try {
  Image image = Image.createImage("/jedi.png");
  g.drawImage(image,
  getWidth()/2, getHeight()/2,
  Graphics.VCENTER | Graphics.HCENTER);
} catch (Exception e){}
```



Gambar : Output dari drawImage()

4.5 Game API

4.5.1Game API

Aplikasi games memiliki peranan utama pada aplikasi mobile. Sebagian besar aplikasi dibuat pada pangsa pasar mobile adalah games. Action, strategy, board and card games dan sebagainya, seluruhnya terdapat pada aplikasi mobile.Sebagian besar produsen game telah membuat API tersendiri untuk berbagai fungsi bermain game yang hanya akan bekerja pada handset yang dibuat oleh perusahaan tersebut. Hal ini berarti bahwa sebuah game yang dibangun menggunakan API dari salah satu produsen tidak akan berjalan pada device hasil produksi dari produsen lain. Untuk menjembatani perbedaan ini, MIDP versi 2 telah memiliki fungsionalitas dasar yang secara spesifik ditujukan aplikasi game. Class utama Game API dari MIDP adalah class GameCanvas. Class GameCanvas merupakan perluasan dari class Canvas yang kita gunakan dalam pembuatan low-level user interface. Dua kelemahan utama dari class Canvas dalam pemrograman game adalah tidak memadainya kemampuan untuk mengatur proses repaint dan ketidakmampuan untuk mengatur bagaimana pointer events serta quick keys diteruskan pada canvas. Komponen user interface dari MIDP umumnya berupa event driven. Events berupa antrian berurutan dan diteruskan terhadap aplikasi satu persatu, beserta tunda waktu antar waktu dimana event dibuat (key press). GameCanvas memungkinkan aplikasi mengumpulkan events yang terbuat dan melakukan proses repaint pada canvas dengan cepat. Struktur program menjadi lebih bersih karena terdapat rangkaian perulangan utama dimana proses painting dan pengumpulan events dilakukan. GameCanvas menggunakan teknik double buffering. Seluruh proses

pembuatan interface dilakukan di off-screen buffer, kemudian di transfer dari area buffer tersebut menuju area yang terlihat pada canvas. Aplikasi anda harus menggunakan instance method dari class Graphics berupa method getGraphics(). Setiap pemanggilan terhadap method ini mengembalikan sebuah instance baru dari off-screen buffer yang anda gunakan dalam proses pembuatan user interface.

Untuk memperbaharui screen tersebut, anda harus memanggil flushGraphics() untuk melakukan proses repaint secara cepat dengan isi dari off-screen buffer. Perhatikan bahwa anda hanya perlu memanggil method getGraphics() sekali saja, karena sebuah buffer teralokasi setiap kali anda memanggil method ini.

MyCanvas.java

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
public class MyCanvas extends GameCanvas implements Runnable {
  private boolean running;
  private long delay;
  private int currentX, currentY;
  private int screenWidth;
  private int screenHeight;
  public MyCanvas() {
    super(true);
    screenWidth = getWidth();
    screenHeight = getHeight();
    currentX = screenWidth / 2;
    currentY = screenHeight / 2;
    delay = 20;
  }
  public void start() {
    running = true;
    Thread thread = new Thread(this);
    thread.start();
  }
  public void stop() {
    running = false;
  }
  // The Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (running == true) {
    getInput();
```

```
    drawScreen(g);
    try { Thread.sleep(delay); } catch (InterruptedException ie) {}
    }
  }
  private void getInput() {
    int keyStates = getKeyStates();
    if ((keyStates & LEFT_PRESSED) != 0)
      currentX = Math.max(0, currentX - 1);
    if ((keyStates & RIGHT_PRESSED) != 0)
      currentX = Math.min(screenWidth, currentX + 1);
    if ((keyStates & UP_PRESSED) != 0)
      currentY = Math.max(0, currentY - 1);
    if ((keyStates & DOWN_PRESSED) != 0)
      currentY = Math.min(screenHeight, currentY + 1);
  }
  private void drawScreen(Graphics g) {
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x0000ff);
    g.drawString("JEDI", currentX, currentY,
    Graphics.TOP|Graphics.LEFT);
    flushGraphics();
  }
}
```

GameMidlet.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class GameMidlet extends MIDlet {
  private Display display;
  public void startApp() {
    display = Display.getDisplay(this);
    MyCanvas gameCanvas = new MyCanvas();
    gameCanvas.start();
    display.setCurrent(gameCanvas);
  }
  public Display getDisplay() { return display; }
  public void pauseApp() {}
  public void destroyApp(boolean unconditional) {
```

```
    exit();
  }
  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
```

4.5.2 Layers

Layers adalah elemen visual dari sebuah screen. Layer adalah abstract class yang merepresentasikan objects pada screen. Sprite dan TiledLayer adalah subclasses dari class Layer. Tiled Layer adalah rangkaian dari beberapa persegi empat yang berukuran sama dan gambar-gambar yang memadai untuk ditempatkan pada persegi empat tersebut. Layer ini dibangun dengan menempatkan gambar-gambar dan elemen-elemen visual dalam area ini. Sebuah gambar dapat digunakan oleh lebih dari satu persegi empat sehingga dapat menghemat ruang dan memory. Tiled Layers umumnya digunakan sebagai background pada game.

4.5.3 Sprites

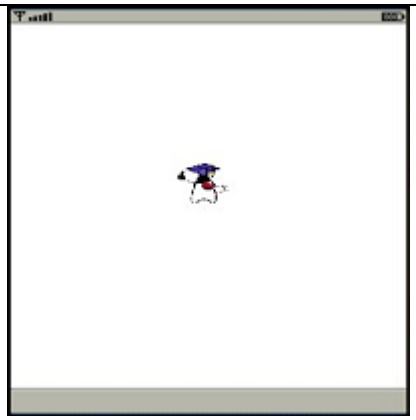
Sprites adalah objects grafis yang anda lihat pada game. Object ini dapat berupa character, kunci, tombol, pintu ataupun peluru. Sebuah sprite bersifat statis ataupun animasi. Animasi sprite terbuat dari beberapa elemen sprite dengan perbedaan-perbedaan kecil dan tersusun sedemikian hingga membentuk kesan bergerak. Rangkaian sprite ini disebut sebagai frame. Contoh kode berikut ini (dari Project Game2) mendemonstrasikan cara penggunaan sprites. Program ini menggunakan sprite sederhana dengan dua frame. Frame ditampilkan menurut penekanan tombol oleh user (UP, DOWN, LEFT atau RIGHT).

GameCanvas dengan Sprite

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
public class MyCanvas extends GameCanvas implements Runnable {
  private boolean running;
  private long delay;
  private int currentX, currentY;
  private int screenWidth;
  private int screenHeight;
  private Sprite sprite;
  public MyCanvas() throws Exception {
    super(true);
    screenWidth = getWidth();
```

```
    screenHeight = getHeight();
    currentX = screenWidth / 2;
    currentY = screenHeight / 2;
    delay = 20;
    Image image = Image.createImage("/jedi.png");
    sprite = new Sprite(image, 32, 32);
  }
  public void start() {
    running = true;
    Thread thread = new Thread(this);
    thread.start();
  }
  public void stop() { running = false; }
  // The Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (running == true) {
    getInput();
    drawScreen(g);
    try { Thread.sleep(delay); } catch (InterruptedException ie) {}
    }
  }
  private void getInput() {
    int keyStates = getKeyStates();
    sprite.setFrame(0);
    if ((keyStates & LEFT_PRESSED) != 0) {
      currentX = Math.max(0, currentX - 1);
      sprite.setFrame(0);
    }
    if ((keyStates & RIGHT_PRESSED) != 0) {
      currentX = Math.min(screenWidth, currentX + 1);
      sprite.setFrame(1);
    }
    if ((keyStates & UP_PRESSED) != 0) {
      currentY = Math.max(0, currentY - 1);
      sprite.setFrame(1);
    }
    if ((keyStates & DOWN_PRESSED) != 0) {
```

```
      currentY = Math.min(screenHeight, currentY + 1);
      sprite.setFrame(0);
    }
  }
  private void drawScreen(Graphics g) {
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x0000ff);
    sprite.setPosition(currentX, currentY);
    sprite.paint(g);
    flushGraphics();
  }
}
```



4.5.4 LayerManager

Class LayerManager akan memberikan kemudahan dalam pengaturan keseluruhan Sprites dan TiledLayers. LayerManager mengatur seluruh Sprites dan TiledLayers yang anda buat. Dan anda tidak perlu untuk membuat seluruh object tersebut satu persatu. LayerManager yang akan melakukannya untuk anda. LayerManager juga mengatur pengurutan objek dari dasar hingga paling atas.

4.6 Scalable 2D Graphics

JSR 226 menyediakan method untuk proses rendering dan transforming atas grafis vector-based 2D. Format gambar raster-based seperti GIF melakukan proses encode terhadap pewarnaan pada tiap-tiap pixel pada area persegi empat yang menunjukkan bentuk gambar. Gambar dengan tipe vector-based hanya memiliki instruksi penggambaran yang menentukan bagaimana pixel-pixel dari gambar tersebut harus diwarnai. Vector tersebut merepresentasikan sebuah gambar yang berukuran jauh lebih kecil, sebuah nilai lebih dalam penggunaan resource pada mobile devices.

Bab 5

Persistence

MIDP menyediakan sebuah API dimana program dapat menyimpan data-data aplikasi secara lokal didalam device tersebut. MIDP Record Management System adalah sebuah fasilitas yang dimiliki oleh MIDlets untuk menyimpan data-data aplikasi pada saat MIDlet invocations. Data akan disimpan dalam non-volatile memory didalam device. Hal ini berarti, data-data program yang telah disimpan tidak akan hilang walaupun program di restart maupun device dimatikan.

5.1 Tujuan

Pada akhir pembelajaran, siswa diharapkan dapat:

- Memahami mengenai konsep dari Record Store
- Membuat dan membuka sebuah Record Store
- Menambah, memanggil kembali, mengupdate, dan mendelete record
- Memanggil record satu persatu (enumerate) record dengan menggunakan RecordEnumerate
- Membuat sebuah Record Comparator
- Membuat sebuah Record Filter

5.2 Record Store

Sebuah Record Store adalah sebuah koleksi daripada record-record. Record Id didalam Record Store selalu unique. Record Id akan secara otomatis dialokasikan pada saat pembentukan sebuah record dan bertindak sebagai index atau primary key. Pemberian record Id dilaksanakan secara sekuensial dan nilai yang diberikan kepada record Id pertama pada setiap Record Store adalah 1 (satu). Pada saat sebuah record dihapus, record id-nya tidak akan bisa digunakan kembali. Jika kita membuat empat buah record dan menghapus record ke-empat, maka record Id selanjutnya yang akan diberikan oleh system adalah 5 (lihat gambar)

Record ID	Byte array
1	Data dari record #1
2	Data dari record #2
...	...
n	Data dari record #n

MIDlets dapat menciptakan lebih dari satu Record Store. Nama dari sebuah record store didalam MIDlet suite haruslah unique. Nama dari record store juga case sensitive dan memiliki panjang maksimal 32 karakter. Pada saat MIDlet suite dihapus dari sebuah device, maka semua record store yang terkoneksi dengan MIDlet didalam suite tersebut juga akan terhapus.

Membuat dan membuka sebuah Record Store

Method-method dibawah ini digunakan untuk membuat dan membuka sebuah record store:

`static` **`RecordStore openRecordStore`**`(String recordStoreName, boolean createIfNecessary)`
`static` **`RecordStore openRecordStore`**`(String recordStoreName, boolean createIfNecessary, int authmode, boolean writable)`
`static` **`RecordStore openRecordStore`**`(String recordStoreName, String vendorName, String suiteName)`

Jika createIfNecessary di-set menjadi true dan Record Store belum ada, maka Record Store akan dibangun. Jika createIfNecessary di-set menjadi false dan Record Store tersebut belum dibuat, maka sebuah RecordStoreNotFoundException akan dijalankan.

Authmode paramater dapat di-set menjadi RecordStore.AUTHMODE_PRIVATE atau RecordStore.AUTHMODE_ANY. Penggunaan AUTHMODE_PRIVATE akan menyebabkan Record Store hanya mampu diakses oleh MIDlet suitesi pemilik MIDlet. Sedangkan setting authmode ke AUTHMODE_ANY akan menyebabkan Record Store untuk diakses oleh MIDlet manapun. Access mode ini dispesifikasikan oleh sebuah writable boolean parameter. Untuk memperbolehkan MIDlet yang lain (diluar MIDlet suite) untuk menggunakan record store tersebut, parameter ini harus diubah menjadi true.

Penggunaan bentuk pertama dari method openRecordStore()akan menyebabkan Record Store untuk dapat diakses oleh MIDlet-MIDlet didalam suite yang sama (authmode di-set ke AUTHMODE_PRIVATE). Untuk membuka sebuah Record Store dari MIDlet suite yang berbeda, bentuk ketiga dari method openRecordStore harus digunakan. Anda harus menspesifikasikan nama vendor (vendorName) dan nama dari Midlet suite (suiteName).

Jika sebuah Record Store terlanjur dibuka, method ini akan mengembalikan reference kepada record store tersebut. System akan tetap menghitung berapa kali Record Store telah dibuka dan setiap Record Store harus ditutup dengan jumlah yang sama pada saat ia dibuka.

Menambahkan sebuah record

`int` **`addRecord`**`(byte[] data, int offset, int numBytes)`

Method addRecord akan membuat record yang baru didalam Record Store dan akan mengembalikan record ID.

Mengambil kembali Record

`byte[]` **`getRecord`**`(int recordId)`
`int` **`getRecord`**`(int recordId, byte[] buffer, int offset)`
`int` **`getRecordSize`**`(int recordId)`

Bentuk pertama dari method getRecord akan mengembalikan copy dari data stored yang ada didalam record tertentu berdasarkan RecordID. Bentuk kedua akan meng-copy data pada paramater byte array

yang telah disediakan. Pada saat menggunakan bentuk kedua, byte array tersebut haruslah dialokasikan terlebih dahulu. Jika ukuran dari record lebih besar daripada ukuran dari parameter, maka akan terjadi ArrayIndexOutOfBoundsException. Anda akan menggunakan method getRecordSize secara berurutan untuk mengetahui ukuran dari record sebelum Anda mulai untuk membacanya.

Meng-update sebuah Record

Anda tidak dapat memodifikasi hanya sebagian dari data record. Jika Anda ingin untuk memodifikasi sebuah record Anda harus:

1) Membaca tiap record dengan menggunakan getRecord
2) Meng-update record didalam memory
3) Memanggil setRecord untuk mengupdate data record

```
void setRecord(int recordId, byte[] newData, int offset, int numBytes)
```

Menghapus Record

```
void deleteRecord(int recordId)
```

Pada saat sebuah record dihapus, record Id akan digunakan kembali di pemanggilan berikutnya pada addRecord. Hal ini berarti, ada sebuah celah didalam Record Id. Oleh karena itu, tidak disarankan untuk menggunakan counter increment untuk membuat list dari keseluruhan record didalam record store. Anda harus menggunakan Record Enumerator untuk mengetahui tiap record didalam sebuah list store.

Menutup sebuah Record Store

```
void closeRecordStore()
```

Record store yang akan ditutup dengan cara pemanggilan method closeRecordStore() tidak akan benar-benar ditutup sampai closeRecordStore() dipanggil sejumlah pemanggilan dari openRecordStore() sebelumnya. Pemanggilan closeRecordStore() lebih dari jumlah pemanggilan openRecordStore() akan berakibat exception RecordStoreNotOpen.

Potongan kode dari contoh RmsExample1 adalah MIDlet sederhana yang mendemonstrasikan bagaimana untuk membuat sebuah record store, menambah record, dan memanggil kembali semua record didalam record store:

```
// Buka dan buatlah record store dengan nama "RmsExample1"
recStore= RecordStore.openRecordStore("RmsExample1", true);
// Masukkan content kedalam record store
for(int recId=1; recId<=recStore.getNumRecords(); recId++){
  recLength = recStore.getRecord(recId, data, 0);
  String item = new String(data, 0, recLength);
  ...
```

```
}
...
// Ini adalah String yang akan kita masukkan kedalam record
String newItem = "Record #" + recStore.getNextRecordID();
// Konversikan String ke byte array
byte[] bytes = newItem.getBytes();
// Tulislah record kedalam record store
recStore.addRecord(bytes, 0, bytes.length);
```

Tips Pemrograman:

1) Record ID dimulai dari 1, bukan 0. Oleh karena itu, apabila menggunakan loop, ingatlah untuk menggunakan 1 sebagai index pertama dan bukan 0.
2) Lebih baik digunakan Record Enumerator daripada menggunakan index incrementing (seperti contoh). Record yang telah dihapus, tetapi masih tetap ingin dibaca pada contoh disini akan menyebabkan InvalidRecordIDException.

Mendapatkan list dari Record Store didalam MIDlet Suite

```
static String[] listRecordStores()
```

Method ini akan mengembalikan array dari nama record store tersebut yang dimiliki oleh MIDlet suite. Jika MIDlet suite tidak memiliki sebuah Record Store, maka method ini akan memiliki nilai pengembalian null.

```
String[] storeNames = RecordStore.listRecordStores();
System.out.println("Record Stores for this MIDlet suite:");
for (int i=0; storeNames != null && i<storeNames.length; i++)
  System.out.println(storeNames[i]);
```

Contoh: RmsListStores

```
Record Stores for this MIDlet suite:
Prefs
RmsExample1
RmsExample2
```

Contoh output dari RmsListStores

Urutan penamaan yang akan dikembalikan tidak akan didefinisikan dan akan diimplementasikan secara independent. Oleh karena itu, apabila Anda ingin untuk menampilkan nama tersebut secara alphabetic, maka Anda harus melakukan sorting array terlebih dahulu.

Menyimpan Data Primitif Java

Sejauh ini, data yang telah dibuat dan dibaca dari Record Store adalah berupa String. CLDC memiliki standard classes dalam manipulasi data primitif. Class tersebut berasal dari standard library platform Java 2, yaitu Standard Edition (J2SE).

Anda dapat menulis data Java primitif dengan mengkombinasikan class ByteArrayOutputStream dan DataOutputStream. Pembacaan tipe data primitive(int, long, short, string, Boolean, dan sebagainya) dapat pula dilakukan dengan menggunakan ByteArrayInputStream dan DataInputStream.

```
ByteArrayOutputStream out = new ByteArrayOutputStream();
DataOutputStream dOut = new DataOutputStream(out);
// Menyimpan sebuah integer
dOut.writeInt(recStore.getNextRecordID() * recStore.getNextRecordID());
// Menyimpan sebuah string
dOut.writeUTF("Record #" + recStore.getNextRecordID());
byte[] bytes = out.toByteArray();
// Menuliskan Record pada Store
recStore.addRecord(bytes, 0, bytes.length);
...
// Menuju Record selanjutnya
byte[] recBytes = enumerator.nextRecord();
ByteArrayInputStream in = new ByteArrayInputStream(recBytes);
DataInputStream dIn = new DataInputStream(in);
int count = dIn.readInt();
String item = dIn.readUTF();
```

Method lain untuk Record Stores

```
long getLastModified()
int getVersion()
```

Sistem merekam bilamana sebuah Record Store mengalami modifikasi terakhir. Method getLastModified memberikan informasi bahwa sebuah Record Store mengalami perubahan terakhir, dalam bentuk long dan sesuai format yang digunakan oleh System.currentTimeMillis().

Seluruh Record Store memiliki version information. Setiap kali sebuah record mengalami modifikasi, maka version number juga akan terupdate. Penggunaan method addRecord, setRecord dan deleteRecord menyebabkan penambahan version number dari record store tersebut.

`static void deleteRecordStore(String recordStoreName)`	Menghapus record store.
`String getName()`	Mengetahui nama dari RecordStore.

`int` **`getNextRecordID`**`()`	Mengetahui recordId dari record selanjutnya untuk disimpan pada record store.
`int` **`getNumRecords`**`()`	Mendapatkan jumlah record yang terdapat pada Record Store.
`int` **`getSize`**`()`	Mengetahui space (dalam bytes) yang dipakai oleh record store.
`int` **`getSizeAvailable`**`()`	Mengetahui sisa space yang tersedia (dalam bytes).
`void` **`setMode`**`(int authmode, boolean writable)`	Mengubah access mode dari RecordStore.

5.3 Record Enumeration

Memeriksa sebuah record store menggunakan inceremented index adalah tidak efisien. Record stores yang telah dihapus akan terlewati jika Record Id dari record tersebut tidak digunakan kembali. Penggunaan record enumeration dapat menyelesaikan permasalahan tersebut dengan melakukan pemeriksaaan pada record yang telah dihapus. Anda juga dapat mengurutkan enumerasi dengan menggunakan method pembanding. Dengan penggunaan method pembanding, anda dapat melewati record yang tidak diharapkan pada output.

```
RecordEnumeration enumerateRecords(RecordFilter filter, RecordComparator comparator, boolean keepUpdated)
```

Method enumerateRecords dari sebuah record store akan menghasilkan enumerasi untuk memeriksa seluruh record pada sebuah record store. Ini adalah cara yang direkomendasikan untuk melewatkan seluruh record dalam record store. Filter dan Comparator akan dibahas dalam pembahasan selanjutnya. Cara paling sederhana dalam menggunakan method ini adalah memberikan nilai null untuk filter dan comparator. Hal ini akan menghasilkan enumerasi dari seluruh record pada sebuah store dalam urutan acak.

```
// Membuka atau membuat sebuah record store dengan nama "RmsExample2"
recStore = RecordStore.openRecordStore("RmsExample2", true);
// Mengambil isi dari record store
RecordEnumeration enumerator
= recStore.enumerateRecords(null, null, false);
while (enumerator.hasNextElement()){
// Mendapatkan record selanjutnya dan konversi byte array menjadi string
String item = new String(enumerator.nextRecord());
// Area kode manipulasi record
...
}
```

5.4 Record Comparator

Pengurutan sebuah enumerasi dapat didefinisikan menggunakan sebuah Record Comparator. Record Comparator digunakan pada method enumerateRecords. Jika anda ingin mengurutkan output dari enumerasi, anda harus membuat comparator dan mengimplementasikannya sebagai parameter kedua pada enumerateRecords.

```
int compare(byte[] rec1, byte[] rec2)
```

Untuk membuat sebuah record comparator, anda harus mengimplementasikan interface RecordComparator. Interface tersebut mendefinisikan method tunggal, compare(byte[] rec1, byte[] rec2). Method ini harus menghasilkan return value, RecordComparator.FOLLOWS atau RecordComparator.PRECEDES. RecordComparator.EQUIVALENT harus dihasilkan jika rec1 adalah ekuivalen terhadap rec2 dalam pengurutan.

```
// Membuat enumerasi, diurutkan menurut alfabet
RecordEnumeration enumerator
= recStore.enumerateRecords(null, new AlphaOrder(), false);
...
// Pengurutan menurut alfabet
class AlphaOrder implements RecordComparator {
  public int compare(byte[] rec1, byte[] rec2){
    String record1 = new String(rec1).toUpperCase();
    String record2 = new String(rec2).toUpperCase();
    if (record1.compareTo(record2) < 0)
      return(PRECEDES);
    else
      if (record1.compareTo(record2) > 0)
        return(FOLLOWS);
      else
        return(EQUIVALENT);
  }
}
```

5.5 Record Filter

Contoh-contoh selama ini membaca seluruh record dari sebuah store. Kita dapat menggunakan sebuah filter untuk mendapatkan hanya record yang kita inginkan. Program Anda harus mengimpelementasikan method match() untuk menyeleksi record. Method tersebut akan menghasilkan nilai true jika record sesuai dengan criteria.

```
boolean matches(byte[] candidate)
```

```
public boolean matches(byte[] candidate){
  boolean isaMatch = false;
```

```
  try {
    ByteArrayInputStream bin = new ByteArrayInputStream(candidate);
    DataInputStream dIn = new DataInputStream(bin);
    int count = dIn.readInt();
    String item = dIn.readUTF();
    // mendapatkan record dengan akhiran "0"
    if (item.endsWith("0"))
      isaMatch = true;
    else
      isaMatch = false;
  } catch (Exception e){items.append(e.toString(), null); }
  return(isaMatch);
}
```

5.6 Record Listener

Sebuah Record Store dapat menggunakan lebih dari satu record listener. Record listener adalah object yang dipanggil pada saat sebuah record ditambahkan, diubah atau dihapus dari record store. Record listeners harus mengimplementasikan interface RecordListener.

Record Listener diregristrasikan pada record store menggunakan method addRecordListener(). Pada saat sebuah record store ditutup, seluruh record listener yang terkait juga akan dihapus. Penggunaan method deleteRecordStore() tidak akan menghasilkan pemanggilan recordDeleted() pada record listener manapun yang terkait.

`void` **`recordAdded`**`(`**`RecordStore`** `recordStore, int recordId)`	Dipanggil saat sebuah record ditambahkan pada record store.
`void` **`recordChanged`**`(`**`RecordStore`** `recordStore, int recordId)`	Dipanggil setelah sebuah record pada record store diubah.
`void` **`recordDeleted`**`(`**`RecordStore`** `recordStore, int recordId)`	Dipanggil setelah sebuah record dihapus dari record store.

Bab 6

Jaringan

Pada bagian ini, kita akan belajar bagaimana menerapkan sebuah MIDlet yang mempunyai kemampuan untuk koneksi kedalam jaringan.

Pada bagian akhir dari sesi ini, siswa diharapkan dapat:

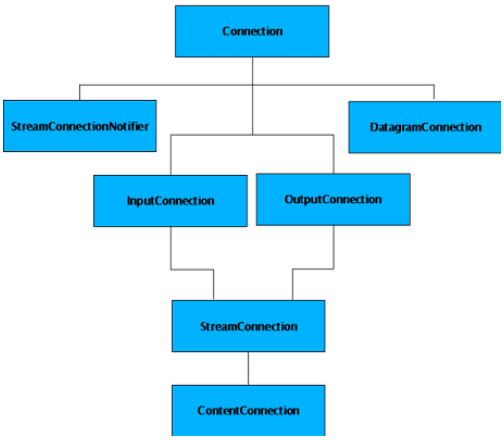
- Mendeskripsikan Generic Connection Framework, dan bagaimana ia dapat digunakan untuk mendukung method koneksi yang berbeda-beda.
- Menspesifikasikan parameter-parameter koneksi dengan menggunakan format pengalamatan GCF URL
- Membuat koneksi HTTP/HTTPS
- Menciptakan MIDlet dengan menggunakan TCP sockets dan UDP datagram

6.1 Generic Connection Framework

Generic Connection Framework mendukung koneksi packet (socket) dan stream (datagram). Sesuai dengan namanya, framework ini menyediakan API dasar bagi koneksi di CLDC. Framework ini menyediakan pondasi umum dari berbagai koneksi seperti HTTP, socket, dan datagram. Walaupun Bluetooth dan serial I/O termasuk kedalam API ini, GCF menyediakan satu set API yang lebih generic dan mendasar yang menjadi abstraksi dari berbagai tipe koneksi. Harus dicatat, bahwa tidak semua tipe koneksi dibutuhkan bagi implementasi sebuah MIDP device.

6.1.1 Hirarki dari GCF Interface

Perluasan dari hirarki GCF interface memungkinkan terjadinya generalization. Sebuah tipe koneksi yang baru mungkin dapat ditambahkan kepada framework ini dengan cara memperluas hirarki.



Gambar 8.1: Hirarki dari GCF Interface

6.1.2 GCF Connection URL

Parameter-parameter koneksi telah dispesifikasikan dengan menggunakan sebuah format pengalamatan:

```
scheme://username:password@host:port/path;parameters
```

1) Scheme adalah sebuah protokol atau method koneksi. Misalnya: http,ftp, https.
2) Username bersifat optional,akan tetapi bila kita ingin mendefinisikannya, harus didahului dengan tanda@
3) Password juga bersifat optional dan hanya dapat dispesifikasikan jika username telah didefinisikan sebelumnya. Jika password didefinisikan, maka ia harus dipisahkan dari username dengan menggunakan tanda titik dua (:)
4) Host-parameter ini wajib dicantumkan. Bisa berupa nama host atau fully qualified domain name (FQDN) atau alamat IP dari host yang dituju.
5) Port-parameter ini juga bersifat optional. Jika tidak dispesifikasikan, maka default port akan digunakan
6) Path
7) parameters-bersifat optional, tetapi harus didahuli dengan titik koma (;) apabila ia dicantumkan

Jika kita menggunakan kurung siku untuk memberi tanda pada parameter-parameter yang bersifat optional pada format pengalamatan diatas, kita dapat mengubah format diatas menjadi seperti berikut:

```
scheme://[username[:password]@]host[:port]/path[;parameters]
```

Format pengalamatan tersebut haruslah sesuai dengan Uniform Resource Indicator (URI) seperti yang didefinisikan pada RFC 2396.Pada MIDP 2.0, hanya skema “http” dan “https” dibutuhkan untuk diimplementasikan pada device.

6.2 Koneksi HTTP

6.2.1 Protokol HTTP

HTTP merupakan kepanjangan dari HyperText Transfer Protocol. Ia merupakan protocol yang digunakan untuk memindahkan web pages dari web server (misal: www.sun.com) kepada web browser. Client(web browser) akan me-request sebuah web page dengan cara mespesifikasikan path dengan command Get atau POST.

Pada method GET, parameter telah dispesifikasikan dan dilekatkan pada URL. Sebagai contoh, untuk memberikan sebuah variable dengan nama “id” dan memiliki nilai 100 kepada index.jsp, url tersebut akan dispesifikasikan sebagai: “http://hostname/index.jsp?id=100”. Parameter tambahan dipisahkan dengan dengan tanda &, "http://hostname/index.jsp?id=100&page=2. Jika method “POST” digunakan, parameter bukanlah menjadi bagian dari URL tetapi dikirim dengan pada baris terpisah pada command POST.

Client / Web Browser	HTTP Server
`GET /index.jsp?id=100 HTTP/1.1`	
	`HTTP/1.1 200 OK`

	Server: Apache-Coyote/1.1 Content-Type: text/html;charset=ISO-8859-1 Date: Wed, 18 Jun 2005 14:09:31 GMT Connection: close <html> <head> <title>Test Page</title> </head> <body> Test Page </body> </html>

Gambar 8.2: Contoh dari transaksi HTTP GET

Client / Web Browser	HTTP Server
GET /non-existent.html HTTP/1.0	
	HTTP/1.1 404 /non-existent.html Server: Apache-Coyote/1.1 Content-Type: text/html;charset=utf-8 Content-Length: 983 Date: Mon, 11 Jul 2005 13:21:01 GMT Connection: close <html><head><title>Apache Tomcat/5.5.7 - Error report</title><style>... <body> HTTP Status 404 ... The requested resource (non-existent.html) is not available. ... </body></html>

Gambar 8.3: Contoh dari transaksi HTTP GET dengan response error

6.2.2 Menciptakan sebuah koneksi HTTP

Anda dapat membuka sebuah koneksi HTTP dengan menggunakan Connector.open() dan meng-casting nya dengan salah satu dari ketiga interface berikut ini: StreamConnection, ContentConnection, dah HTTPConnection. Bagaimanapun, dengan StreamConnection dan ContentConnection, Anda tidak dapat menspesifikasikan dan menurunkan parameter-parameter spesifik dari HTTP dan juga result-nya.

Bila Anda menggunakan StreamConnection, panjang dari sebuah reply, tidak dapat ditentukan sebelumnya. Sedangkan pada ContentConnection atau HTTPConnection, selalu ada cara untuk menentukan panjang dari sebuah reply. Akan tetapi penentuan panjang ini, tidak selalu tersedia. Oleh karena itu, program Anda harus bisa mendapatkan reply tersebut tanpa harus mengetahui panjang content terlebih dahulu.

```
import javax.microedition.io.*;
HttpConnection connection = null;
InputStream iStream = null;
byte[] data = null;
try {
  connection = (HttpConnection) Connector.open("http://www.sun.com/");
  int code = connection.getResponseCode();
  switch (code){
    case HttpConnection.HTTP_OK:
      iStream = connection.openInputStream();
      int length = (int) connection.getLength();
      if (length > 0){
        data = new byte[length];
        int totalBytes = 0;
        int bytesRead = 0;
        while ((totalBytes < length) && (bytesRead > 0)) {
           bytesRead = iStream.read(
           data, totalBytes, length - totalBytes);
           if (bytesRead > 0)
             totalBytes += bytesRead;
        }
      } else {
        //panjang tidak diketahui, baca tiap karakter
        ...
      }
      break;
    default:
      break;
  }
...
```

6.2.3 Handling HTTP Redirects

Terkadang server akan melakukan redirect dari sebuah browser/client ke web page yang lain dengan cara me-reply HTTP_MOVED_PERM (301), HTTP_MOVED_TEMP (302), HTTP_SEE_OTHER (303) atau HTTP_TEMP_REDIRECT (307) daripada menggunakan reply HTTP_OK yang biasa dilakukan. Program Anda harus dapat mengidentifikasi-nya dengan menggunakan getResponseCose(), mendapatkan URI yang baru dari header dengan menggunakan getHeaderField("Location"), dan mendapatkan kembali dokumen dari lokasi yang baru.

```
int code = connection.getResponseCode();
switch(code){
  case HttpConnection.HTTP_MOVED_PERM:
  case HttpConnection.HTTP_MOVED_TEMP:
  case HttpConnection.HTTP_SEE_OTHER:
  case HttpConnection.HTTP_TEMP_REDIRECT:
    String newUrl = conn.getHeaderField("Location");
  ...
```

6.3 Koneksi HTTPS

HTTPS adalah sebuah HTTP diatas sebuah koneksi secure transport. Membuka sebuah koneksi HTTPS, hampir sama untuk membuka koneksi HTTP. Perbedaan utamanya adalah URL akan memberikan kepada Connector.open() dan meng-casting hasilnya kepada HttpsConnection class variable.

Sebuah tipe exception tambahan juga harus dijalankan melalui Connector.open() misalnya IllegalArgumentException, ConnectionNotFoundException, java.io.IOException dan SecurityException. Sebuah CertificateException juga dapat dijalankan untuk melaporkan kesalahan pada certificate.

```
import javax.microedition.io.*;
HttpsConnection connection = null;
InputStream iStream = null;
byte[] data = null;
try {
  connection = (HttpsConnection) Connector.open("https://www.sun.com/");
  int code = connection.getResponseCode();
  ...
} catch (CertificateException ce){
  switch (ce.getReason()){
    case CertificateException.EXPIRED:
    ...
  }
}
```

static byte **BAD_EXTENSIONS**
Mengindikasikan bahwa sertifikat memiliki extenstion yang tidak teridentifikasi.
static byte **BROKEN_CHAIN**
Mengindikasikan bahwa sertifikat terletak didalam sebuah rantai yang tidak terautentikasi pada mata rantai berikutnya.
static byte **CERTIFICATE_CHAIN_TOO_LONG**
Mengindikasikan bahwa sertifikat server dari rantai tersebut melebihi panjang yang disepakati pada policy dari pembuat sertifikat.
static byte **EXPIRED**
Mengindikasikan bahwa sertifikat tersebut telah berakhir jangka waktunya.
static byte **INAPPROPRIATE_KEY_USAGE**
Mengindikasikan bahwa public key dari sertifikat tersebut telah digunakan tidak sesuai dengan ketetapan yang dibuat oleh pembuat sertifikat.
static byte **MISSING_SIGNATURE**
Mengindikasikan bahwa object dari sertifikat tidak memiliki sebuah tanda tangan digital.
static byte **NOT_YET_VALID**
Mengindikasikan bahwa sertifikat tersebut tidak berlaku.
static byte **ROOT_CA_EXPIRED**
Mengindikasikan bahwa root dari public key CA telah habis jangka waktunya.
static byte **SITENAME_MISMATCH**
Indicates a certificate does not contain the correct site name.
static byte **UNAUTHORIZED_INTERMEDIATE_CA**
Mengindikasikan bahwa ada sebuah sertifikat intermediate certificate didalam rantai yang tidak punya otoritas sebagai intermediate CA.
static byte **UNRECOGNIZED_ISSUER**
Mengindikasikan bahwa sertifikat tersebut telah dikeluarkan oleh entity yang tidak teridentifikasi
static byte **UNSUPPORTED_PUBLIC_KEY_TYPE**
Mengindikasikan bahwa tipe public key didalam sertifikat tidak didukung oleh device.
static byte **UNSUPPORTED_SIGALG**
Mengindikasikan bahwa sertifikat telah ditandatangani dengan menggunakan algorithma yang tidak disupport.
static byte **VERIFICATION_FAILED**
Mengindikasikan bahwa sertifikat tersebut gagal di-verifikasi.

Gambar 8.4: Berbagai alasan pada CertificateException

(kutipan dari spesifikasi MIDP 2.0 – JSR 118)

6.4 TCP Sockets

Banyak implementasi dari HTTP dijalankan diatas layer TCP. Jika Anda mengirim data menggunakan layer TCP, data tersebut akan dipotong menjadi bagian yang lebih kecil yang disebut dengan packet. Layer TCK akan memastikan bahwa semua packet akan dikirim oleh sender dan diterima oleh recipient, dengan susunan yang sama seperti pada saat ia dikirimkan. Jika sebuah packet tidak diterima oleh recipient, ia akan mengirimkannya kembali. Hal ini berarti, sekali Anda mengirim sebuah pesan, Anda dapat memastikan bahwa pesan tersebut akan berhasil dikirim kepada recipient dengan format yang sama seperti pada saat Anda mengirimkannya, tanpa ada data yang hilang atau disisipi (dihalangi oleh sebuah siklus tertentu seperti recipient disconnect dari jaringan) .

Layer TCP menangani reassembly dan retransmission pada packet secara transparan. Sebagai contoh, pada protokol HTTP kita tidak perlu khawatir terhadir prosess disassembly dan assembly packet karena hal ini akan dihandle pada layer TCP.

Kadang-kadang, ukuran dari pesan tersebut terkadang terlalu kecil dan sangat tidak efisien untuk dikirimkan sebagai packet tunggal (overhead dari packet sangat tinggi jika dibandingkan dengan payload). Bayangkan banyak packet dikirimkan melalui jaringan dengan satu byte payload dan multi byte overhead(misal 16 bytes). Hal ini akan menyebabkan jaringan sangat tidak effisien, banyak packets membanjiri jaringan dengan hanya satu byte payload. Pada kasus ini, implementasi dari TCP dimungkinkan untuk menunggu sebuah pesan dikirim dengan sukses. Pesan tersebut kemudian akan dipaket sebagai banyak pesan didalam sebuah packet sebelum dikirimkan. Jika hal ini terjadi, maka akan terjadi keterlambatan pada koneksi. Jika aplikasi Anda menginginkan sesedikit mungkin terjadi keterlambatan, anda harus mengeset DELAY socket option ke nol (0). Atau jika aplikasi Anda dapat tetap berjalan dengan beberapa paket yang hilang atau tidak tersusun secara benar, Anda mungkin harus mencoba menggunakan UDP atau koneksi datagram. Koneksi UDP juga menggunakan sesedikit mungkin overhead packet.

```
SocketConnection conn =
(SocketConnection) Connector.open("socket://www.sun.com:80");
client.setSocketOption(DELAY, 0);
InputStream iStream = conn.openInputStream();
OutputStream oStream = conn.openOutputStream();
os.write("GET / HTTP/1.0\n\n".getBytes());
int c = 0;
while((c = is.read()) != -1) {
  // memproses data yang diterima
  ...
}
iStream.close(); oStream.close(); conn.close();
```

6.5 Server Sockets

Didalam model client-server, server akan secara terus menerus menunggu sebuah koneksi dari client atau dari port tertentu yang telah disetujui. Kita juga dapat menggunakan method Connector.open untuk menciptakan sebuah server socket. Sebuah URL akan memberikan sebuah format yang sama seperti pada TCP socket kepada method open(), dengan nama hostname yang dibiarkan kosong (misal socket://:8899).

```
ServerSocketConnection conn =
(ServerSocketConnection) Connector.open("socket://:8889");
// Dengarkan koneksi dari client
SocketConnection client = (SocketConnection) conn.acceptAndOpen();
client.setSocketOption(DELAY, 0);
InputStream iStream = client.openInputStream();
OutputStream oStream = client.openOutputStream();
// baca/tulis untuk input/output streams
...
is.close();
os.close();
client.close();
server.close();
```

6.6 Datagrams

Koneksi dari TCP socket adalah koneksi yang dapat dipercaya. Sebaliknya, tersampaikannya pesan dengan menggunakan packet UDP tidak dijamin. Tidak ada jaminan bahwa packet yang dikirimkan dengan menggunakan paket datagram akan diterima oleh pasangan. Susunan dari packet yang diterima juga tidak terpercaya. Susunan packet yang dikirimkan dimungkinkan untuk tidak sama dengan susunan packet yang diterima. UDP datagrams atau packet digunakan apabila aplikasi dapat tetap berjalan walaupun ada packet yang hilang atau packet tersebut tidak lagi memiliki susunan yang sama seperti yang dikirimkan.

```
String url;
try {
  if (isServer)
    // memulai sebagai server, mendengarkan port 8888
    url = "datagram://:8888";
  else
    //memulai sebagai client, koneksi dengan port 8888 sebagai localhost
    url = "datagram://localhost:8888";
  dc = (DatagramConnection) Connector.open(url);
  while (true) {
    Datagram dgram = dc.newDatagram(128);
```

```
    dc.receive(dgram);
    if (isServer)
      //memulai sebagai server, mendapatkan alamat koneksi
      //bagi pesan kita selama proses pengiriman
      url = dgram.getAddress();
    if (dgram.getLength() > 0)
      String mesg = new String(dgram.getData(), 0, dgram.getLength()));
  }
}
catch (IOException ioe) {}
catch (Exception e) {}
...
private void sendMesg(String line){
  try {
    byte[] bytes = line.getBytes();
    Datagram dgram = null;
    dgram = dc.newDatagram(bytes, bytes.length, url);
    dc.send(dgram);
  } catch (Exception ioe) {}
}
```