MODUL WORKSHOP

DASAR PEMROGRAMAN MOBILE DENGAN JAVA MOBILE EDITION (J2ME)

DISUSUN OLEH :

Y.YOHAKIM MARWANTA, S.KOM

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

AKAKOM

YOGYAKARTA

2008

KATA PENGANTAR

Puji dan Syukur kepada Tuhan Yang Maha Kuasa atas selesainya pembuatan modul ini. Modul workshop dengan topik bahasan Java Mobile Edition (J2ME) ini disusun untuk membantu pelaksanaan workshop di STMIK AKAKOM. Modul ini memberikan penjelasan tentang bagaimana membangun aplikasi berbasis mobile untuk pemula. Dengan adanya modul ini diharapkan mahasiswa nantinya dapat memahami dasar-dasar pemrograman J2ME dan akhirnya diharapkan dapat mengembangkan lebih lanjut.

Tidak lupa penulis mengucapkan banyak terimakasih pada semua pihak yang telah membantu dalam penyusunan modul ini. Akhirnya penyusun berharap semoga modul ini bermanfaat bagi pembaca semuanya, segala kritik dan saran yang membangun demi sempurnanya modul ini dengan terbuka penyusun terima.

Yogyakarta Mei 2008

Penyusun

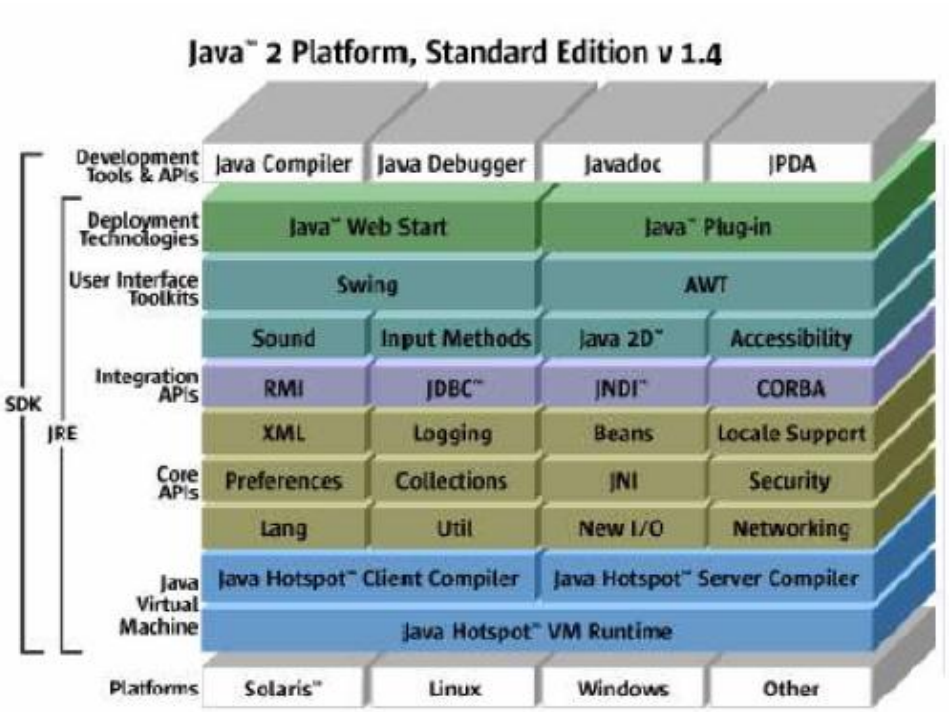
MODUL 1
PENGENALAN JAVA MOBILE EDITION (J2ME)

1. Teori

1.1 Java 2 Platform

Java adalah bahasa yang dapat dijalankan dimanapun dan di sembarang *platform* apapun, di beragam lingkungan: *Internet*, *intranets*, *consumer electronic products*, dan *computer applications*. Untuk beragam aplikasi yang dibuat dengan bahasa Java, Java dipaketkan dalam edisi-edisi berikut:

1. *Java 2 Standar Edition* (J2SE), J2SE menyediakan lingkungan pengembangan yang kaya fitur, stabil, aman, dan *cross-platform*. Edisi ini mendukung konektivitas basis data, rancangan *user interface*, masukkan/ keluaran (*input/ output*), dan pemrograman jaringan (*network programming*), dan termasuk sebagai paket-paket dasar bahasa Java.



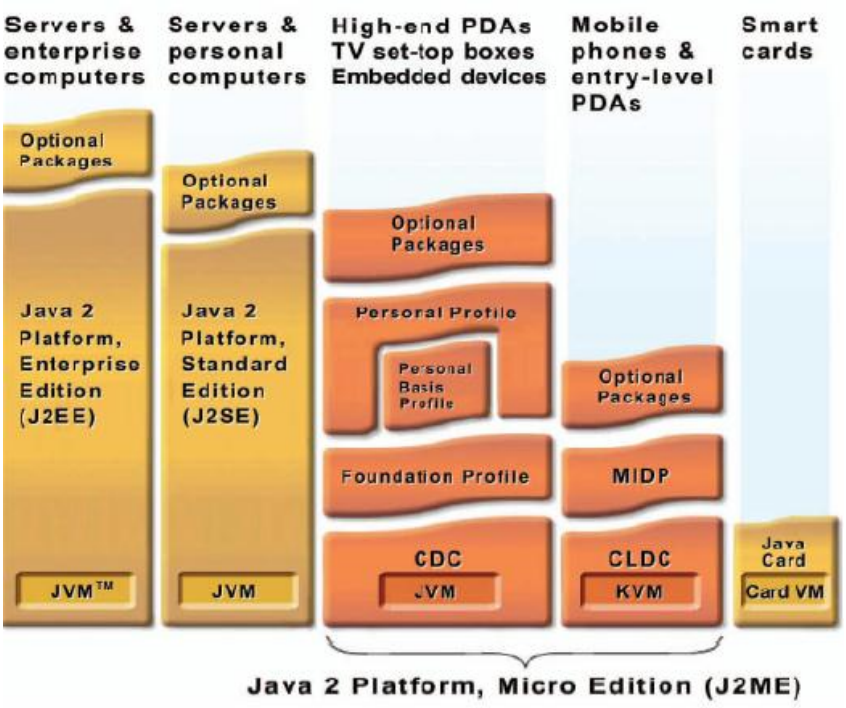
Gambar 1 Lingkungan Java Standard Edition

2. *Java 2 Enterpise Edition* (J2EE), J2EE menyediakan tempat untuk membangun dan menjalankan *multitier enterprise editions*. J2EE berisi paket-paket di J2SE ditambah

paket-paket untuk mendukung pengembangan *Enterprise JavaBeans, Java Servlets, JavaServer Pages, XML*, dan kendali transaksi yang fleksibel.

3. *Java 2 Micro Edition* (J2ME), J2ME selain menyedikan bahasa Java yang sama, unggul dalam portabilitas (kemampuan dapat dijalankan dimanapun), *safe network delivery*, seperti J2SE dan J2EE. Aplikasi-aplikasi dapat diskalakan (dimampukan) agar dapat bekerja dengan J2SE dan J2EE. J2ME adalah untuk beragam *consumer electronic product*, seperti *pager, smart card, cell phone, handheld* PDA, dan *set-top box*.

1.2 J2ME



Gambar 2 Lingkungan J2ME

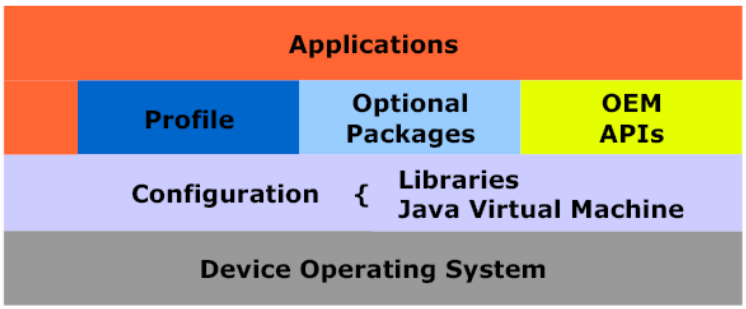
Paparan singkat di atas adalah penjelasan singkat mengenai Java dan sedikit gambaran dimana paket J2ME digunakan. Sebenarnya masih panjang penjelasan tentang Java dan paket J2ME, tetapi tidak dibahas disini.

Komponen-komponen J2ME terdiri dari Java Virtual Machine (JVM) yang digunakan untuk menjalankan aplikasi Java pada emulator atau *handheld device*, Java API (*Aplication Programming Interface*) dan *tools* lain untuk pengembangan aplikasi Java semacam emulator *Java Phone*, emulator Motorolla dari J2ME *wireless toolkit*.

J2ME adalah satu set spesifikasi dan teknologi yang fokus kepada perangkat konsumen. Perangkat ini memiliki jumlah memori yang terbatas, menghabiskan sedikit daya dari baterei,layar yang kecil dan bandwith jaringan yang rendah.

Program J2ME, seperti semua program JAVA adalah diterjemahkan oleh VM. Program-program tersebut dikompile ke dalam bytecode dan diterjemahkan denga Java Virtual Machine(JVM).Ini berarti bahwa program-program tersebut tidak berhubungan langsung dengan perangkat.

J2ME menyediakan suatu interface yang sesuai dengan perangkat. Aplikasi-aplikasi tersebut tidak harus dikompile ulang supaya mampu dijalankan pada mesin yang berbeda. Inti dari J2ME terletak pada configuration dan profile-profile. Suatu configuration menggambarkan lingkungan runtime dasar dari suatu sistem J2ME. Ia menggambarkan core library, virtual machine, fitur keamanan dan jaringan.



Gambar 3 Arsitektur J2ME

Dalam pengembangan aplikasi wireless dengan Java, J2ME dibagi menjadi dua buah bagian diantaranya ialah bagian *configuration* dan *profile*.

1.2.1 Profile

Sebuah profile memberikan library tambahan untuk suatu kelas tertentu pada sebuah perangkat. profile-profile menyediakan user interface(UI) API,persistence, messaging library, dan sebagainya.

Satu set library tambahan atau package tambahan menyediakan kemampuan program tambahan. Pemasukan package ini ke dalam perangkat J2ME dapat berubah-ubah karena tergantung pada kemampuan sebuah perangkat. Sebagai contoh, beberapa perangkat MIDP tidak memiliki Bluetooth built-in, sehingga Bluetooth API tidak disediakan dalam perangkat ini.

J2ME mempunyai beberapa profil antara lain :

1. *MOBILE INFORMATION DEVICE PROFILE* (MIDP)
2. *Foundation Profile* (FP)
3. *Personal Profile*
4. *Personal Digital Assistance* (PDA)

1.2.2 Configuration

Suatu configuration menggambarkan fitur minimal dari lingkungan lengkap Java runtime. Untuk menjamin kemampuan portabilitas dan interoperabilitas optimal diantara berbagai macam perangkat yang dibatasi sumber dayanya(memory, prosesor, koneksi yang dibatasi), configuration tidak menggambarkan fitur tambahan. Suatu configuration J2ME menggambarkan suatu komplemen yang minimum dari teknologi JAVA. Adalah merupakan tugas profile-profile untuk menggambarkan tambahan library untuk suatu kategori perangkat tertentu.

configuration menggambarkan:

- Subset bahasa pemrograman JAVA
- Kemampuan Java Virtual Machine(JVM)
- Core platform libraries
- Fitur sekuriti dan jaringan

J2ME mempunyai dua konfigurasi yaitu `*Connected Limited Device Configuration (CLDC)* dan *Connected Device Configuration* (CDC).

1.3 CLDC

The Connected Limited Device Configuration (CLDC) menggambarkan dan menunjuk pada area berikut ini:

- Fitur Bahasa Java dan Virtual Machine(VM)
- Library dasar(java.lang.*,java.util.*)
- Input/Output(java.io.*)
- Kemanan
- Jaringan
- Internationalization

CLDC tidak menggambarkan instalasi dan daur hidup sebuah aplikasi, antarmuka(UI) dan penanganan peristiwa(event handling). Adalah merupakan tugas profile yang berada di bawah CLDC untuk menggambarkan area ini. Secara khusus, spesifikasi MIDP

menggambarkan daur hidup aplikasi MIDP (MIDlet), library UI dan event handling(javax.microedition.lcdui.*).

1.4 MIDP

The Mobile Information Device Profile (MIDP) berada di atas dari CLDC. Anda tidak bisa menulis aplikasi mobile hanya dengan menggunakan CLDC API. Anda harus tetap memanfaatkan MIDP yang mendefinisikan UI.

Spesifikasi MIDP, kebanyakan seperti CLDC dan API lainnya sudah digambarkan melalui Java Community Process (JCP). JCP melibatkan sebuah kelompok ahli berasal dari lebih dari 50 perusahaan, yang terdiri atas pabrik perangkat mobile, pengembang software. MIDP terus berkembang, dengan versi-versi masa depan yang telah lulus dari proses ketat JCP.

Spesifikasi MIDP menggambarkan suatu perangkat MID yang memiliki karakteristik-karateristik ini sebagai batas minimum:

· **Tampilan:**

- · Ukuran Layar: 96x54
- · kedalaman tampilan: 1-bit
- · Ketajaman pixel: sekitar 1:1

· **Masukan:**

- · Satu atau lebih mekanisme user-input: satu keybboard, dua keyboard, atau touch screen

· **Memory:**

- · 256 kilobytes of non-volatile memory untuk implementasi MIDP.
- · 8 kilobytes of non-volatile memory for application-created persistent data
- · 128 kilobytes of volatile memory for the Java runtime (e.g., the Java heap)

· **Jaringan:**

- · dua jalur, wireless, bandwidth terbatas

· **Sound:**

- · Kemampuan untuk memainkan nada-nada

MIDP menggambarkan model aplikasi, UI API, penyimpanan dan jaringan yang kuat, permainan dan media API, kebijakan keamanan, penyebaran aplikasi dan ketetapan over-theair.

1.4 Midlet

Aplikasi yang berjalan pada sebuah perangkat yang mendukung MIDP disebut dengan MIDlets, atau lebih singkatnya MIDlet merupakan aplikasi yang dibuat menggunakan Java 2 Micro Edition dengan profile *Mobile Information Device Profile* (MIDP).

MIDP dikhususkan untuk digunakan pada *handset* dengan kemampuan CPU, memori, *keyboard* dan *layer* yang terbatas, seperti *handphone, pager,* PDA dan sebagainya.

Perangkat application management software (AMS) berinteraksi langsung dengan MIDlet dengan method MIDlet **create, start, pause, dan destroy**.

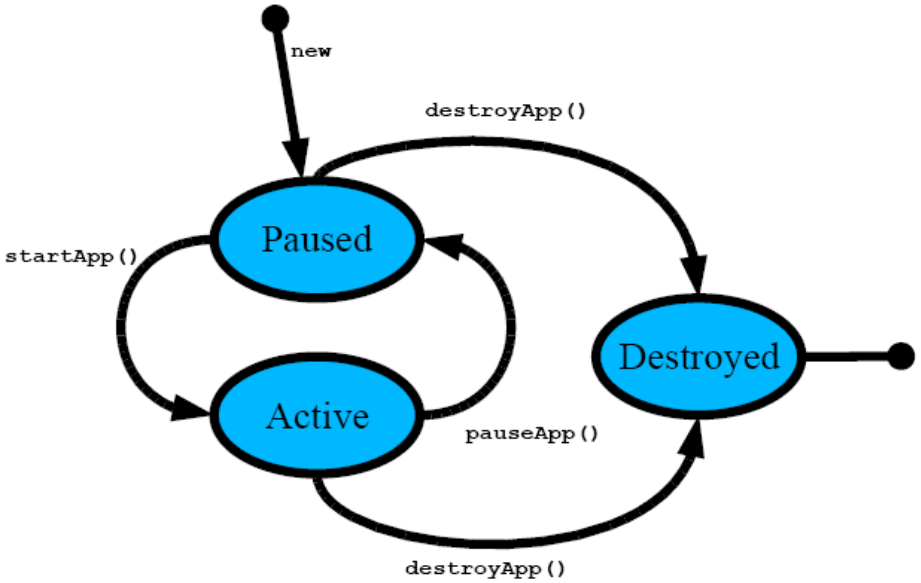
MIDlet adalah bagian dari package **javax.microedition.midlet**. Sebuah MIDlet harus **di-extend** dengan class MIDlet. Dan dapat meminta parameter dari AMS seperti dirumuskan dalam application descriptor (JAD). Suatu MIDlet **tidak harus memiliki** (dan memang harus tidak mempunyai) sebuah **method public static void main(String[] argv)**.Method tersebut tidak akan dikenal lagi oleh AMS sebagai titik awal sebuah program.

1.4.1 Siklus MIDlet

Kehidupan MIDlet dimulai ketika di-instantiate oleh AMS. MIDlet pada awalnya masuk status "Pause" setelah perintah baru dibuat. AMS memanggil constructor public tanpa argumen dari MIDlet. Jika sebuah exception terjadi dalam constructor, MIDlet memasuki status "Destroyed" dan membuangnya segera.

- MIDlet masuk ke dalam status "Active" atas pemanggilan method **startUp()** oleh AMS.
- MIDlet masuk ke dalam status "Destroyed" ketika AMS memanggil method **destroyApp().**

Status ini juga kembali diakses ketika method **notifyDestroyed()** kembali dengan sukses kepada aplikasi. Dengan catatan bahwa MIDlet hanya bisa memasuki status "Destroyed" sekali dalam masa hidupnya.



Gambar 4 Daur Hidup Midlet

1.5 Perangkat untuk Midlet

Untuk penjalankan Midlet tentunya diperlukan perangkat keras (device) yang mendukung Java artinya perangkat tersebut harus memiliki Java Virtual Machine untuk menjalankan Midlet. Sekarang tidak susah lagi untuk menemui perangkat yang bisa menjalankan Midlet terutama untuk jenis ponsel. Hampir setiap ponsel keluaran terbaru telah menyertakan dukungan akan teknologi Java.

1.4 Emulator Ponsel Java

Untuk menjalankan Midlet, programmer tidak perlu memiliki dan mencobanya pada ponsel. Cukup dengan emulator dari ponsel yang dapat dijalan pada PC. Berikut adalah contoh emulator :

Gambar 5 Emulator Ponsel Java

2. Hal-Hal Dasar :

Susunan Kode Sumber .java

Dalam mengembangkan program Java, Anda menulis class dan class. Class ini ditulis dalam kode sumber yang disimpan sebagai file teks biasa berekstension .java.

Dalam file .java, dapat dideklarasikan :

1. package
2. import
3. satu atau lebih class

Deklarasi Class

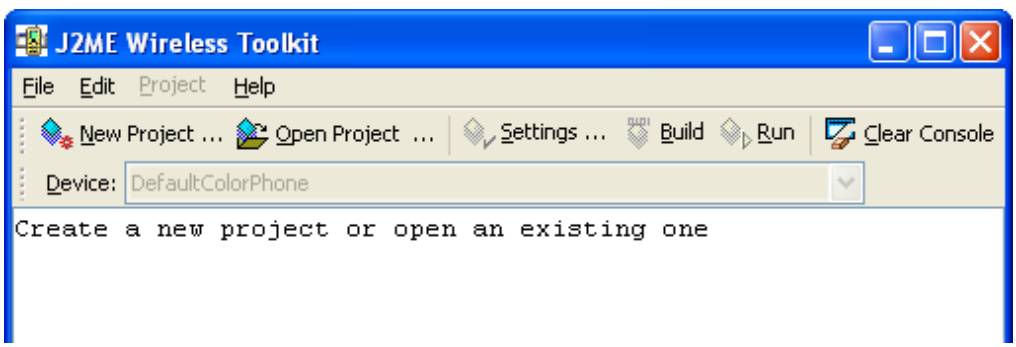
Deklarasi class merupakan kandungan utama sebuah file berekstension .java. Deklarasi class terutama memuat :

1. nama class, bisa dilengkapi dengan kendali akses, deklarasi extends maupun deklarasi implements. Pola yang lumrah adalah :
2. public class ClassName
3. extends SuperClassName
4. implements Interface1Name, Interface2Name, Interface3Name
5. constructor, yang dipanggil pada saat dibuat instans dari class.
6. deklarasi variabel-variabel
7. deklarasi prosedur-prosedur, yang di dalam Java disebut method.

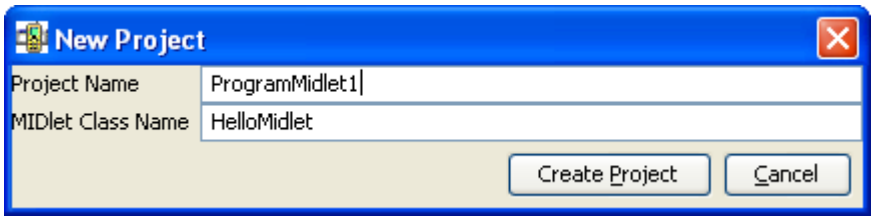
3. PRAKTIK .

Dalam praktik ini menggunakan emulator Wireless Toolkit untuk membangun aplikasi. Langkah yang dipraktikkan adalah sebagai berikut :

1. dari J2ME Wireless Toolkit 2.2 Beta pilih Ktoolbar, akan muncul tampilan berikut :

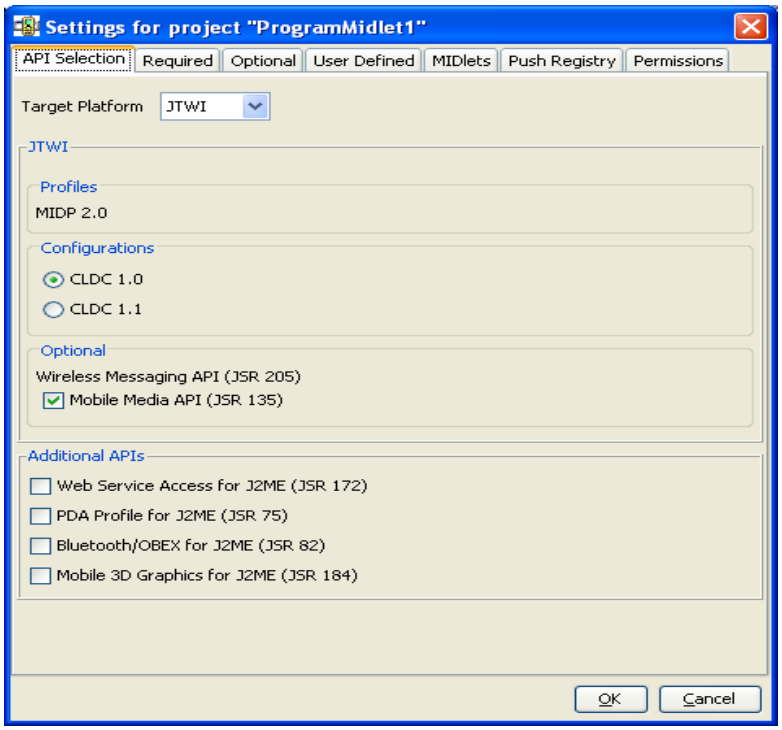


2. untuk membuat project baru pilihlah New Project …, akan muncul tampilan berikut :

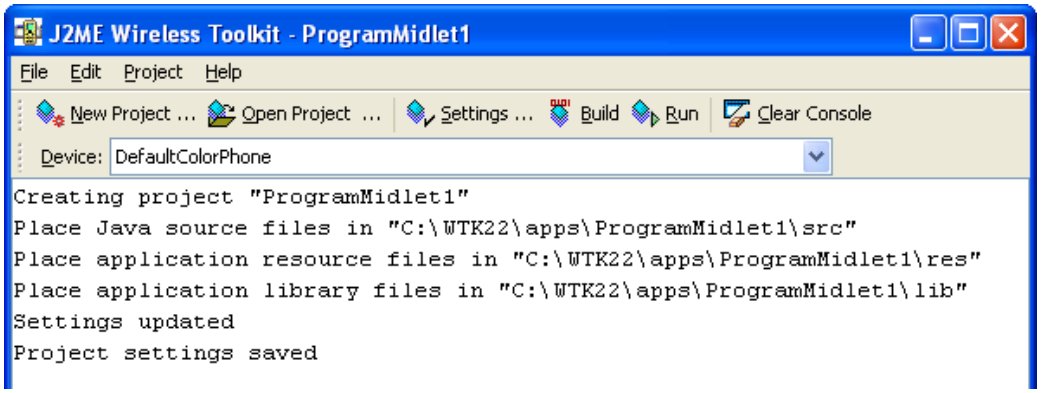


- Pada project name isikan dengan nama project yang akan dibuat, usahakan nama project mencerminkan apa yang sedang dikerjakan.
- **MIDlet Class Name** : diisi dengan nama class yang akan dibuat (nama class ini harus benar-benar sama dengan nama clas pada source code . java)

3. pilih **Create Project** sehingga nampak dilayar sebagai berikut :



4. Pilih OK, maka akan muncul tampilan berikut :



Perhatian pada tampilan diatas :

Java Source (file java) nanti harus di letakkan (disimpan) di directori

C:\WTK22\apps\ProgramMidlet1\src

Setelah lanngkah 1 s.d 5 diatas dikerjakan selanjutnya dengan menggunakan Text Editor (dalam praktik ini menggunakan TextPad) ketikkan program berikut dan simpan di directori **C:\WTK22\apps\ProgramMidlet1\src** dengan nama file **HelloMidlet.java** :

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloMidlet extends MIDlet
    implements CommandListener {

    Display display;
    Command exitCommand =
        new Command("Exit", Command.EXIT, 1);
    Alert helloAlert;

public HelloMidlet(){
        helloAlert = new Alert(
            "Hello MIDlet", "Hallo, dunia!",
            null, AlertType.INFO
        );

     helloAlert.setTimeout(Alert.FOREVER);
     helloAlert.addCommand(exitCommand);
     helloAlert.setCommandListener(this);
}

public void startApp() {
 if (display == null){
  display =           Display.getDisplay(this);
 }
      display.setCurrent(helloAlert);

 }
public void pauseApp() {
 }

 public void destroyApp(boolean unconditional) {
 }

public void commandAction(Command c, Displayable d)
{
      if (c==exitCommand)
      {
            destroyApp(true);
            notifyDestroyed();
      }
}
}
```

Beberapa hal penting dari kode program diatas dapat dijelaskan sebagai berikut :

```
public class HelloMidlet extends MIDlet implements CommandListener {
```

Seperti yang sudah kita katakan sebelumnya, kita harus membuat subclass dari MIDlet untuk membuat MIDP program. Pada line ini, kita sudah membuat subclass dari MIDlet dengan memberikan turunan kelas induk dan menamakannya HelloMIDlet.

```
    Display display;
    Command exitCommand = new Command("Exit", Command.EXIT, 1);
    Alert helloAlert;
```

Line diatas ini adalah variabel properties dari MIDlet. Kita membutuhkan **object Display** (hanya ada satu diplay per MIDlet) untuk melakukan fungsi **menggambar pada layar**.

exitCommand adalah perintah yang akan kita taruh pada layar agar kita dapat keluar dari program. Jika kita tidak memiliki perintah keluar, maka kita tidak memiliki cara untuk keluar dari MIDlet dengan benar.

```
public HelloMidlet(){
    helloAlert = new Alert(
    "Hello MIDlet", "Hello, world!", null, AlertType.INFO);
    helloAlert.setTimeout(Alert.FOREVER);
    helloAlert.addCommand(exitCommand);
    helloAlert.setCommandListener(this);
}
```

Consturctor melakukan inisialisasi dari object Alert. Kita akan mempelajari lebih lanjut dari Alert class pada bab berikutnya. Method addCommand() pada object Alert memberikan perintah "Exit" pada layar. Method setCommandListener() memberikan informasi kepada sistem untuk memberikan semua command events ke MIDlet.

Latihan : Buatlah program MIDlet yang lain.

MODUL 2

MIDlets menggunakan komponen high-level UI

A. Teori

1. MIDP User Interface

MDIP user interface didesain untuk peralatan mobile. Aplikasi MDIP ditunjukan pada area limited screen. Peralatan memory juga menjadi faktor penting jika perlengkapan mobile hanya memiliki kapasitas memory yang kecil.

Dengan berbagai macam peralatan mobile, dari berbagai model mobile phones sampai PDAs, MIDP user interface telah didesain untuk lebih fleksibel dan mudah digunakan dalam berbagai macam peralatan ini.

MIDP mempunyai class yang dapat menangani fungsi high-level dan low-level user interface. High-level UI interfaces didesain secara fleksibel. Penampilan dari komponen ini tidak didefinisikan secara spesifik. Penampilan screen yang sebenarnya dari berbagai macam komponen ini digunakan dari satu peralatan ke peralatan yang lain. Tetapi para programmer telah teryakinkan oleh kegunaan dari high-level komponen UI interfaces memiliki persamaan dalam berbagai spesifikasi-pengimplementasi secara keseluruhan.

High Level UI	***Low-Level UI***
highly portable across devices	Memungkinkan semua peralatan
look dan feel sama dengan peralatannya	Spesifik aplikasi look and feel
Memiliki interaksi seperti scrolling yang dienkapsulasi	Pengimplementasiannya harus dengan petunjuk sendiri
Penampilannya tidak dapat digambarkan secara aktual	Penampilannya tidak dapat digambarkan dalam satuan pixel
High Level UI	***Low-Level UI***
Tidak memiliki akses untuk peralatan dengan feature yang spesifik	Mengakses masukkan low-level hanya dengan menekan

Kapan menggunakan High-Level UI

- Saat membangun aplikasi text-based yang mudah
- Saat Anda ingin aplikasi Anda dapat dengan mudah dipertukarkan dengan berbagai macam peralatan (Portabilitas)
- Saat Anda ingin aplikasi Anda memiliki tampilan yang sama dengan komponen UI yang lain dari berbagai peralatan
- Saat Anda ingin kode Anda dapat menjadi sesedikit mungkin, ketika sebuah interaksi ditangani oleh API

Kapan menggunakan Low-Level UI

- Saat Anda memerlukan sebuah high-level untuk mengkontrol tampilan dari suatu aplikasi
- Saat aplikasi Anda membutuhkan tempat yang tepat dari elemen-elemen yang ada pada screen
- Saat membuat game secara grafik; meskipun Anda tetap dapat menggunakan highlevel UI pada menu game, hal tersebut lebih disarankan untuk membuat menu UI Anda sendiri untuk menghindari **seamless atmosphere** bagi para user
- Saat sebuah aplikasi membutuhkan akses ke low-level yang memiliki inputan seperti key presses
- Jika aplikasi Anda akan diimplementasikan pada layar navigasi Anda sendiri

1.1 Display

Inti dari MIDP user interfaces adalah display. Yang merupakan satu-satunya kemudahan dari Display per MIDlet. MIDlet dapat mendapatkan referensi Display object dengan menggunakan method static Display.getDisplay(), melewatkan referensi tersebut ke MIDlet instance.

MIDlet dijamin dengan display object tidak akan berubah dengan adanya eksistensi instance MIDlet. Hal ini berarti bahwa variebel dikembalikan (returned) ketika Anda memanggil getDisplay() dan tidak akan berpengaruh jika anda memenggilnya dengan startApp() atau destroyApp() (Lihat pada gambar Midlet Life Cycle).

1.2 Displayable

Hanya satu displayable yang ditampilkan pada satu waktu. Secara langsung, displayable tidak ditampilkan pada layar. Seuabh displayable dapat ditampilkan dengan memanggil method setCurrent() dari Display instance. Method setCurrent() harus dipanggil pada saat memulai aplikasi, dengan kata lain sebuah screen kosong akan ditampilkan atau aplikasi tersebut tidak akan dijalankan.

Method startApp dari MIDlet merupakan suatu tempat dimana Anda dapat menaruh method pemanggil setCurrent(). Tetapi Anda harus mempertimbangkan bahwa dalam MIDlet startApp() dapat dipanggil lebih dari satu kali. Untuk memberhentikan MIDlet sementara waktu dapat dipause dengan memanggil fungsi pauseApp(), dengan adanya incoming call, memungkinkan startApp() dipanggil lagi (setelah ada telepon masuk).

Maka dengan memanggil setCurrent() pada method pada startApp(), dan ada kemungkinan layar akan menjadi gelap (blank) pada screen displayed yang sebelumnya, sampai adanya penghentian sementara (pause by the phone call).
Sebuah displayable dapat memiliki nama, beberapa perintah(command), commandListener dan Ticker.

1.3 Title

Sebuah Displayable memiliki title yang berhubungan dengan dirinya sendiri. Posisi dan penampilan dari title tersebut merupakan piranti spesifik yang hanya dapat ditentukan oleh peralatan dari aplikasi yang sedang dijalankan. Sebuah title ditampilkan pada Displayable dengan memanggil setTitle().Dengan memanggil method ini maka seketika akan meng-update title pada Displayable. Jika pada saat Displayable ditampilkan pada layar, MIDP specification states menyebutkan bahwa title harus dirubah dengan implementasi "Memungkinkan untuk dilakukan dengan cepat".

Memberi parameter null pada setTitle() berarti menghapus title pada Displayable. Merubah atau menghapus sebuah title dari Displayable dapat mempengaruhi ukuran
area untuk isi dari Displayable tersebut.Jika terjadi perubahan ukuran area terjadi, MIDlet akan diberitahu dengan memanggil kembali method sizeChanged().

1.4 Command

Dengan adanya kekurangan ukuran pada screen, MIDP tidak menggambarkan sebuah menu bar. Untuk menggantikan menu bar, MIDlet memiliki Commands. Biasanya Command diimplementasikan sebagai soft key atau item dalam sebuah menu. Object Command hanya berisi informasi tentang action yang harus dikerjakan pada saat Command diaktifkan. Dia tidak berisikan kode yang akan dieksekusi pada saat Command tersebut dipilih.
Properti CommandListener dari Displayable berisi action yang akan dieksekusi saat
Command diaktifkan. CommandListener merupakan interface yang spesifik pada single method :

public void commandAction(Command command, Displayable displayable)

Mapping dari Commands pada peralatan bergantung pada nomer yang telah ditetapkan atau programable button pada peralatan. Jika nomer dari Command tidak benar pada semua button, maka memungkinkan peralatan menaruh beberapa atau semua Command pada menu dan peta pada menu dan button akan diberi label "Menu".

Command memiliki sebuah short label, long label, tipe dan prioritas.

- **Command Label**

Diasumsikan bahwa screen yang berukuran kecil dari target sebuah peralatan, selalu menjadi faktor ketika membangun aplikasi MIDP. Asumsi ini juga dapat diterapkan untuk Command label. Command label harus singkat, namun deskriptif, maka hal itu harus benar pada screen dan tetap dapat dipahami oleh user.

Ketika long label ditentukan, hal tersebut akan ditampilkan kapan saja pada saat sebuah implementasi sistem dilihat secara sesuai. Tidak ada pemanggilan API yang menetapkan label yang akan ditampilkan. Hal tersebut juga memungkinkan bahwa sebuah Command akan menampilkan short label pada saat Command lain pada screen yang sama menampilkan long labels.

- **Command Type**

Sebuah Command yang diperkenalkan pada peralatan sering disebut device-dependent. Seorang programer dapat mengetahui spesifikasi tipe dari Command. Tipe ini akan ditampilkan sebagai hint pada tempat Command diletakkan. Berbagai macam tipe Command:

Command.OK, Command.BACK,
Command.CANCEL, Command.EXIT,
Command.HELP, Command.ITEM,
Command.SCREEN, Command.STOP

- **Command Priority**

Aplikasi dapat menetapkan spesifikasi Command yang penting pada priority property. Hal ini merupakan integer property dan nilai rendah yang sangat penting. Priority property juga hanya sebuah hint pada tempat dimana seharusnya Command ditempatkan. Biasanya implementasi menentukan posisi dari Command oleh tipenya.

Jika terdapat lebih dari satu Command dari tipe yang sama, secara normal priority akan mempertimbangkan penempatan Command.

1.5 CommandListener

CommandListener merupakan interface dengan single method:

void commandAction(Command command, Displayable displayable)

Method commandAction() akan dipanggil jika Command dipilih. Variabel Command merupakan referensi Command yang telah dipilih. Tampilan merupakan Displayable (atau screen) dimana Command ditempatkan dan saat action “pilih” terjadi.

CommandAction() harus dikembalikan dengan seketika, jika tidak maka pengeksekusian aplikasi akan diblock. Hal ini dikarenakan, spesifikasi MIDP tidak memerlukan implementasi untuk membuat sebuah pembatas untuk pengiriman event.

1.6 Ticker

Ticker adalah sebuah baris dari text yang dapat discrolling secara terus-menerus pada display. Method konstruktor dari ticker menerima text string untuk ditampilkan. Hal tersebut hanya memiliki dua method lain, yaitu getter dan setter untuk text ini: String getString() dan void setString(String text). Tidak ada cara lain pada sebuah aplikasiuntuk mengontrol kecepatan dan arah dari scrolling text. Scrolling tidak dapat dipause atau distop. Jika spasi diletakkan pada text, hal tersebut tidak akan ditampilkan pada layar. Semua baris text akan ditampilkan sebagai single line dari scrolling text.

Sebuah ticker dapat dipasang pada Displayable dengan memanggil setTicker(). Jika ticker telah ada pada Displayable, maka akan diganti oleh ticker yang baru yang terdapat dalam parameter.

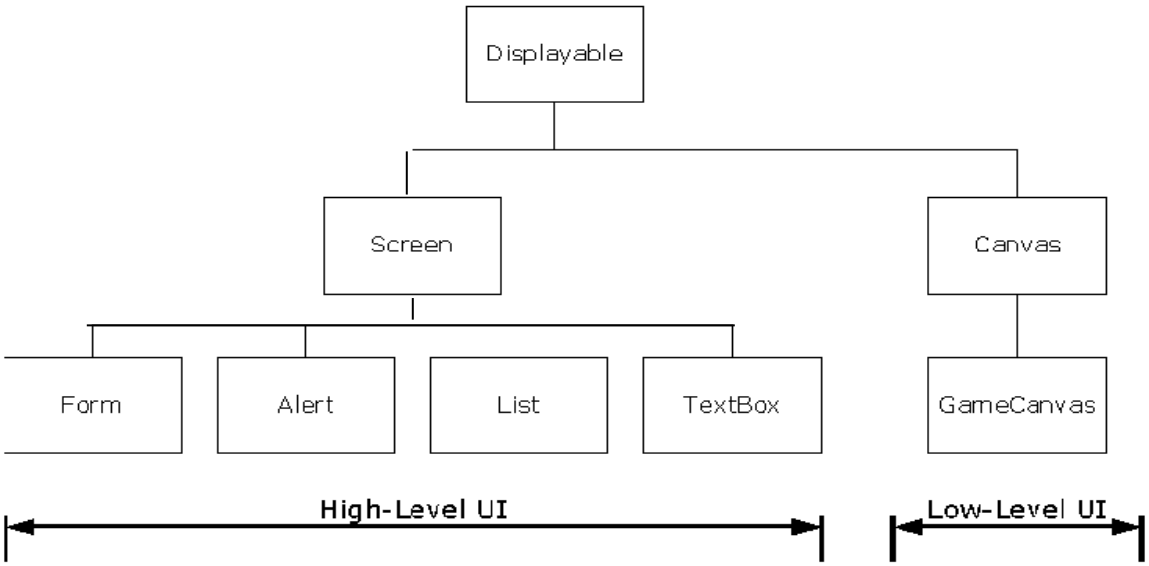
Memberi parameter null pada setTicker akan mengganti semua ticker yang telah dimasukkan pada Displayable. Menghapus ticker dari Displayable dapat menyebabkan perubahan ukuran area dari isi Displayable tersebut. Jika perubahan ukuran area terjadi, maka MIDlet akan memanggil sebuah ukuran dengan method sizeChanged().

Pada ticker object Displayable boleh berbagi suatu kejadian(action).

1.7 Screen

Screen merupakan inti abstrak class yang digunakan untuk high-level UI ketika canvas merupakan Displayable abstrak class untuk low-level UI.

Berikut ini empat subclasses dari abstract class screen : Form, TextBox, List dan Alert.

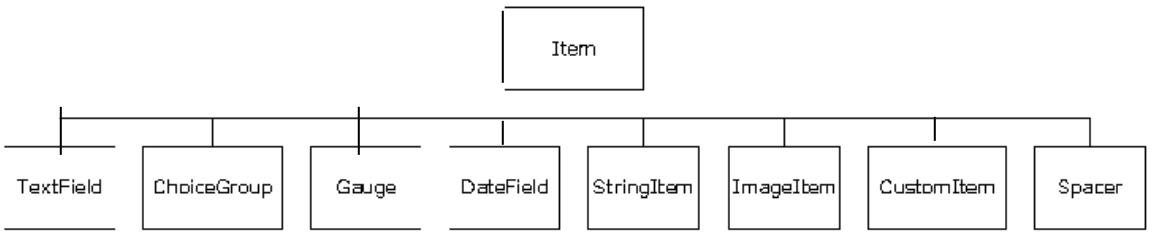


Gambar 7 Displayable Class Hirarcy

1.8 Item

Items merupakan komponen yang dapat diletakan kedalam container, seperti Form atau Alert. Sebuah item dapat memiliki property seperti dibawah ini:

Property	***Default Value***
Label	Dikelompokan pada subclass konstruktor
Commands	-
defaultCommand	null
ItemCommandListener	null
Layout directive	LAYOUT_DEFAULT
Preferred width and height	-1 (unlocked)



Gambar 8 Item Class Hirarcy

Spesifikasi layout dari item dengan Form. Direktif layout dapat dikombinasikan menggunkan bitwise atau operasi (|). Bagaimanapun juga, beberapa direktif bersifat mutually exclusive. Berikut ini direktif horizontal alignment yang mutually exclusive:

LAYOUT_LEFT
LAYOUT_RIGHT
LAYOUT_CENTER

Berikut ini direktif vertical alignment yang juga mutually exclusive:

LAYOUT_TOP
LAYOUT_BOTTOM
LAYOUT_VCENTER

Berikut ini layout yang lain dari direktif (tidak mutually exclusive):

LAYOUT_NEWLINE_BEFORE
LAYOUT_NEWLINE_AFTER
LAYOUT_SHRINK
LAYOUT_VSHRINK
LAYOUT_EXPAND
LAYOUT_VEXPAND
LAYOUT_2

2. Alert

Alert merupakan sebuah screen yang dapat menampilkan text dan gambar. Alert merupakan komponen untuk menampilkan error dan warning, display text dan informasi gambar atau untuk mendapatkan informasi dari user.

Alert ditampilkan untuk spesifikasi peeriode dari waktu. Waktu di-set menggunakan method setTimeout() dan method tersebut dispesifikasikan dalam unit millliseconds. Hal tersebut dapat dibuat untuk ditampilkan hingga user mengaktifkan perintah ("Done") dengan menspesifikasikan spesial timeout dari Alert.FOREVER.

Alert juga dapat menampilkan komponen Gauge (Lihat pada Gauge item) sebagai indikator. Ketika alert berisi text yang tidak sesuai dengan screenful dan harus discroll, maka secara otomatis alert menge-set ke modal(timeout di set kepada Alert.FOREVER).

3. List

List merupakan subclass dari screen yang berisi sebuah daftar dari suatu pilihan. Sebuah list dapat dibagi menjadi tiga tipe: IMPLICIT, EXCLUSIVE atau MULTIPLE.
Jika List bertipe IMPLICIT dan user mengeksekusi tombol "select", commandAction() dari list commandListener akan dipanggil. Default perintahnya adalah **List.SELECT_COMMAND.**

Untuk tipe IMPLICIT dan EXCLUSIVE, GetSelectedIndex() mengembalikan index dari element yang dipilih. Untuk tipe MULTIPLE, getSelectedFlags() mengembalikan sebuah array dari boolean yang berisi state dari elemen-elemen. isSelected(int index) mengembalikan state dari elemen dalam pemberian posisi index.

4. Text Box

TextBox merupakan sub-class dari screen yang dapat digunakan untuk mendapatkan input text dari user. Hal ini memperbolehkan user untuk memasukan dan mengedit text. TextBox hampir

sama dengan TextField(Lihat pada item TextField) karena dia dapat memiliki input constraint dan input modes. Perbedaannya dengan TextField adalah user dapat memasukan garis baru(ketika input constraint di-set untuk semua "ANY").

Isi dari TextBox dapat diambil kembali dengan menggunakan method **getString().**

5. Form

Form merupakan subclass dari Screen. Form merupakan conntainer untuk item subclass, seperti TextField, StringItem, ImageItem, DateField dan ChoiceGroup. Dia menghandle layout untuk komponen ini. Dan juga menghandle traversal antar komponen-komponen dan scrolling dari Screen.

Item ditambahkan dan dimasukkan ke dalam sebuah Form menggunakan method

append() dan insert(), berturut-turut.

Item direferensikan menggunakan index zero-based.

6. ChoiceGroup

Item Choicegroup merupakan group dari selectable choice. Sebuah choice boleh berisi sebuah text, gambar atau kedua-duanya. Choice boleh EXCLUSIVE (hanya satu pilihan yang dapat dipilih) atau MULTIPLE (banyak pilihan yang dapat dipilih pada suatu waktu). Jika ChoiceGroup bertipe POPUP, hanya satu choice yang ditampilkan. Popup selection akan ditampilkan ketika item ini dipilih.

Dari popup seleksi ini, user diperbolehkan memilih pilihannya. Choice yang ditampilkan selalu choice yang dipilih.

GetSelectedIndex() mengembalikan nilai index pada element dari ChoiceGroup yang dipilih. GetSelectedFlags() mengembalikan sebuah array dariboolean yang merespon elemen dari **Choicegroup. isSelected(int index)** mengembalikan state dari elemen yang diberikan oleh posisi index.

7. Image Item

ImageItem merupakan Image sederhana yang dapat dimasukan kedalam komponen, seperti Form.

ImageItem menerima item layout sebagai parameter (Lihat pada bagian Item):

```
public ImageItem( String label,
                  Image img,
                  int layout,
                  String altText)
```

Konstruktor yang lain menerima tampilan mode yang bertipe Item.PLAIN, Item.HYPERLINK atau Item.BUTTON (Lihat pada bagian StringItem):

```
public ImageItem(String label,
Image image,
int layout,
String altText,
int appearanceMode)
imageForm = new Form("ImageItem");
imageForm.addCommand(backCommand);
imageForm.setCommandListener(this);
try {
Image img = Image.createImage("/java.png");
ImageItem image =
new ImageItem("JENI", img, Item.LAYOUT_CENTER, "jeni logo");
imageForm.append(image);
} catch (Exception e){e.printStackTrace();}
```

File “java.png” sangat penting untuk dimasukan kedalam project dengan menggunakan operating system's manager dan menaruh image tersebut kedalam direktori project dibawah subdirektori “src”. Kemudian project direfresh dengan mengklik kanan nama project dan pilih “Refresh Folders”.

B. PRAKTIK

MEMBUAT APLIKASI INTERAKSI USER

Sebuah aplikasi tentunya dirancang untuk interaksi antara user dengan aplikasi tersebut. Untuk merancang interaksi dapat disajikan dalam bentuk menu-menu. Dengan memilih menu tertentu user akan diarahkan menuju link yang di sebutkan pada menu tersebut. Dalam J2ME untuk menangani user interface seperti tersebut diatas, dapat menggunakan API MIDP yang mempunyai komponen yang dapat digunakan untuk membangun user interface seperti list, radio button, selection box dan lain-lain.

Praktik 1
1. Membuat Menu Dengan List

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class MenuMidlet01 extends MIDlet {
private List mainMenu = null;
private Display display;

public MenuMidlet01() {
     mainMenu = new List("Menu Midlet", List.IMPLICIT);
     mainMenu.append("Menu 1", null);
```

```
        mainMenu.append("Menu 2", null);
        mainMenu.append("Keluar", null);
}
public void startApp(){
        display = Display.getDisplay(this);
        display.setCurrent(mainMenu);
}
public void pauseApp() {
}
public void destroyApp(boolean b) {
}
}
```

Build dan Jalankan aplikasi diatas dan amati hasilnya.

2. Menambahkan Aksi pada menu

Dari aplikasi yang saudara bangun pada praktik 1, list menu yang ada pada saat di pilih/ditekan tidak akan memberikan efek atau reaksi apapun, hal ini dikarenakan masing-masing-masing list tersebut belum di beri perintah (action) apapun. Untuk itu ketik dan amati program berikut, kemudian jalankan dan analisalah hasilnya.

Praktik 2

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class MenuMidlet02 extends MIDlet implements CommandListener {
private List mainMenu = null;
private Display display;
private Alert alert;

public MenuMidlet02() {
            mainMenu = new List("Menu Midlet", List.IMPLICIT);
            mainMenu.append("Menu 1", null);
            mainMenu.append("Menu 2", null);
            mainMenu.append("Keluar", null);
            mainMenu.setCommandListener(this);
      }
public void startApp(){
            display = Display.getDisplay(this);
            display.setCurrent(mainMenu);
      }

public void pauseApp() {
}

public void destroyApp(boolean b) {
}

//memberikan aksi pada masing-masing list
public void commandAction(Command c, Displayable d) {
```

```
if (d == mainMenu) {
      //jika list pada indek ke -1 ( indeks pada list
                dimulai dari 0)dipilih
      if (mainMenu.isSelected(0)) {
            alert = new Alert("Perhatian!!");
            alert.setString("Menu 1 ditekan, silakan tunggu..");
            display.setCurrent(alert);
            }
      // jika list ke -2 yang di pilih
      else if (mainMenu.isSelected(1)) {
            alert = new Alert("Perhatian!!");
            alert.setString("Menu 2 ditekan, silakan tunggu..");
            display.setCurrent(alert);
            }
      // jika list terakhir yang(ke -3) yang dipilih
      else if (mainMenu.isSelected(2)) {
            destroyApp(false);
            notifyDestroyed();
            }
      }
  }
}
```

3. Menambahkan Command

Objek yang dibentuk dari class Command dapat ditambahkan pada banyak objek, misalnya objek yang dibuat dari class Form, class TextBox, class Canvas dan lain-lain. Berikut ini objek-objek yang dibentuk dari class Command akan ditambahkan ke dalam objek yang dibentuk dari class Form.

Praktik 3

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class CommandMidlet01 extends MIDlet implements
CommandListener {
private Command cmdKeluar;
private Command cmdAlert;
private Display display;
private Form form;
private Alert alert;

public CommandMidlet01() {
form = new Form("Aplikasi Pertama"); // membuat form baru
cmdKeluar = new Command("Keluar", Command.EXIT, 1);//
cmdAlert = new Command("Peringatan", Command.SCREEN, 2);
form.addCommand(cmdAlert); // menambahkan Command Alert ke form
form.addCommand(cmdKeluar);// menambahkan Command Keluar ke form
form.setCommandListener(this);
}
```

```
public void startApp() {
      display = Display.getDisplay(this);
      display.setCurrent(form);
      }

public void pauseApp() {
      }

public void destroyApp(boolean unconditional) {
      }

public void commandAction(Command c, Displayable d) {
      if (d == form) {
      //perintah pada Command Keluar
      if (c == cmdKeluar) {
      destroyApp(false);
      notifyDestroyed();
      }
      //Aksi yang dikerjakan jika Command Peringatan Yang ditekan
      else {
            alert = new Alert("Peringatan..!!!");
            alert.setString("Tombol Peringatan ditekan...");
            display.setCurrent(alert);
            }
      }
      }
}
```

Jalankan Dan Amati Hasilnya.

4. Memadukan List dan Command

Praktik 4

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class CommandMidlet02 extends MIDlet implements
CommandListener {
private Display display;
private Command cmdMenu;
private Command cmdKeluar;
private TextBox textBox;
private List mainMenu;

public CommandMidlet02() {
      textBox = new TextBox("Aplikasi Kedua", "Ini Command Midlet
                Kedua", 256, 0);
      cmdMenu = new Command("Menu", Command.SCREEN, 1);
      cmdKeluar = new Command("Keluar", Command.SCREEN, 1);
      textBox.addCommand(cmdMenu);
      textBox.addCommand(cmdKeluar);
      textBox.setCommandListener(this);
      }
```

```
public void startApp() {
      display = Display.getDisplay(this);
      display.setCurrent(textBox);
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
public void commandAction (Command c, Displayable d) {
      if (d == textBox) {
      if (c == cmdKeluar) {
      destroyApp(false);
      notifyDestroyed();
      }
      else if (c == cmdMenu) {
            mainMenu = new List("Menu Aplikasi Kedua", List.IMPLICIT);
            mainMenu.append("Tulis Pesan", null);
            mainMenu.append("Baca Pesan", null);
            mainMenu.append("Kembali", null);
            mainMenu.setCommandListener(this);
            display.setCurrent(mainMenu);
         }
      }
      else if (d == mainMenu) {
            if (mainMenu.isSelected(0)) {
            // Ketik operasi yang diinginkan
            // ketika tombol untuk Menu 1 ditekan
            }
      else if (mainMenu.isSelected(1)) {
      // Ketik operasi yang diinginkan
      // ketika tombol untuk Menu 2 ditekan
      }
      else if (mainMenu.isSelected(2)) {
            display.setCurrent(textBox);
            }
      }
      }
}
```

5. Membuat Form dan Membuat Kolom Isian

Pada saat membuat aplikasi kadang kita memerlukan suatu masukan yang diisi oleh pengguna untuk diproses. Untuk maksud tersebut diperlukan form yang didalamnya terdapat kolom

isian.

Praktik 5

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public    class    FormIsianMidlet    extends    MIDlet    implements
CommandListener{
private Form form;
```

```
private TextField textField;
private Display display;
private Command cmdKeluar;
private Command cmdAksi;
private Alert alert;
public String strNama;

public FormIsianMidlet() {
      form = new Form("Form UI");
      textField = new TextField("Nama :", "", 10, 0);
      cmdKeluar = new Command("Keluar", Command.EXIT, 1);
      cmdAksi = new Command("Aksi", Command.SCREEN, 2);
      form.addCommand(cmdKeluar);
      form.addCommand(cmdAksi);
      form.append(textField);
      form.setCommandListener(this);
}

public void startApp() {
      display = Display.getDisplay(this);
      display.setCurrent(form);
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d) {
      if (d == form) {
            if (c == cmdKeluar) {
                  destroyApp(false);
                  notifyDestroyed();
                  }
            else if (c == cmdAksi) {
                  strNama = textField.getString();
                  alert = new Alert("Pesan");
                  alert.setString("Hallo "+strNama);
                  display.setCurrent(alert);
                  }
            }
      }
}
```

6. Praktik Ticker

Bab sebelumnya pernah dibahas mengenai pembuatan form dan menambahkan objek textfield. Selain textfield masih banyak objek lain yang bisa ditambahkan pada form. Pada tulisan ini akan diberikan contoh aplikasi yang menggunakan objek ticker pada form.

Apabila sedang menyaksikan tayangan pada televisi akan sering terlihat berita pada bagian bawah layar yang berupa tulisan berjalan. Atau pada acara yang berhubungan dengan bursa efek akan bisa dilihat juga berupa tulisan berjalan yang memberikan informasi mengenai keadaan saham pada saat itu. Untuk membuat “tayangan” seperti itu pada layar ponsel dapat digunakan class Ticker.

Praktik 6

Membuat Ticker Sederhana

Berikut adalah contoh penambahkan objek ticker pada form.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class TickerMidlet01 extends MIDlet implements CommandListener
{
private Form form;
private Display display;
private Ticker ticker;
private Command cmdKeluar;
public TickerMidlet01() {
cmdKeluar = new Command("Keluar", Command.EXIT, 1);
ticker = new Ticker("Belajar Menggunakan Ticker");
form = new Form("Ticker");
form.setTicker(ticker);
form.addCommand(cmdKeluar);
form.setCommandListener(this); }
public void startApp() {
display = Display.getDisplay(this);
display.setCurrent(form); }
public void pauseApp() { }
public void destroyApp(boolean unconditional) { }
public void commandAction(Command c, Displayable d) {
if (c == cmdKeluar) { destroyApp(false);
notifyDestroyed();
}
}
}
```

7. Setting Teks pada Ticker

Pada class Ticker terdapat dua method yaitu getString() dan setString(String str). Berikut adalah contoh untuk aplikasi yang memberikan fasilitas bagi user untuk mengubah teks yang ditampilkan oleh objek ticker. Berikut adalah contoh dari aplikasi tersebut.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class TickerMidlet02 extends MIDlet implements CommandListener
{
private Form form;
private Display display;
private Ticker ticker;
private Command cmdKeluar;
private Command cmdTicker;
private TextField textField;

public TickerMidlet02() {
      cmdKeluar = new Command("Keluar", Command.EXIT, 1);
      cmdTicker = new Command("Ticker", Command.SCREEN, 2);
      textField = new TextField("Teks : ", "", 50, TextField.ANY);
      ticker = new Ticker(""); form = new Form("Ticker");
```

```
      form.append(textField); form.addCommand(cmdKeluar);
      form.addCommand(cmdTicker); form.setCommandListener(this); }
public void startApp() {
      display = Display.getDisplay(this);
      display.setCurrent(form); }
public void pauseApp() { }
public void destroyApp(boolean unconditional) { }
public void commandAction(Command c, Displayable d) {
      if (c == cmdKeluar) {
      destroyApp(false);
      notifyDestroyed(); }
      if (c == cmdTicker) {
      form.setTicker(ticker);
      ticker.setString(textField.getString());
      }
      }
}
```

C. Tugas

Dari kode program **`CommandMidlet02`** diatas kembangkan program sehingga pada waktu menu **Tulis Pesan** `di pilih,` muncul text box yang bisa digunakan untuk menuliskan pesan.

MODUL 3
CANVAS

A. TEORI

1. Canvas

Canvas, sesuai namanya guna objek ini untuk mengambar. Selain itu sifanya hampir sama dengan objek lain yang bisa ditambahkan objek-objek lain kedalamnya seperti penambahan objek command. Objek canvas sering digunakan untuk membuat aplikasi game. Pada bagian ini akan diberikan contoh bagaimana menggambar di 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().

1.1 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.

Praktikkan program dibawah ini, analisislah hasilnya:

```
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, Dunia!");
      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();
      }
      }
}
```

Beberapa hal yang dapat dijelaskan dari program diatas adalah :

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, Dunia!” 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);
```

1.2 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 sama :

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){
// perintah yang harus dikerjakan
}
}
```

1.3 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).

1.4 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.

2. Grafik

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

2.1 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
`public int getColor()`	**Mengembalikan warna terbaru dalam bentuk integer.**
	warna terbaru
`public int getGrayScale()`	Mengembalikan nilai abu-abu sebagai warna terbaru
`public void setGrayScale(int value)`	Memilih nilai abu-abu untuk mengganti operasi menggambar

2.2 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()
```

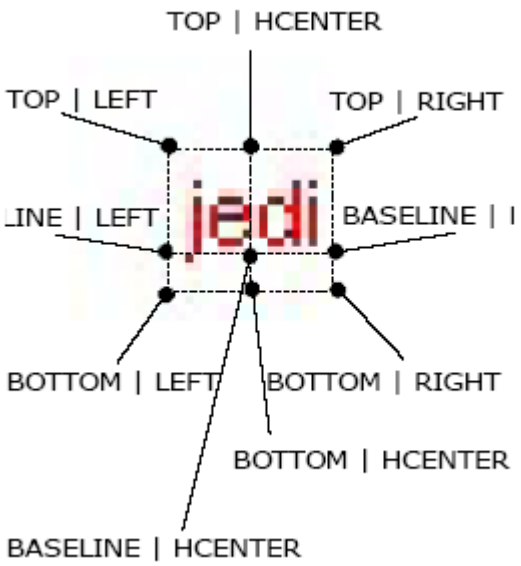
```
public boolean isItalic()
public boolean isUnderlined()
```

2.3 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.



2.4 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 getBaselinePosition()`	Mengembalikan jarak dalam pixel antara TOP dan BASELINE pada teks, berdasarkan pada huruf ini

2.5 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
```

2.6 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-byone" 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);
// 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);
```

2.7 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);
```

PRAKTIK

Membuat dan Menambahkan Canvas

Dari teori yang ada diatas, selanjtunya akan dipraktikkan teori-teori tersebut. Yang dilakukan terlebih dahulu adalah membuat class Canvas yang nantinya akan dipanggil sebagai objek. Berikut adalah contoh kodenya.

Praktik 1

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Canvas01 extends MIDlet implements CommandListener {
private Display display;
private Canvas canvas;
private Command cmdKeluar;

public Canvas01() {
        cmdKeluar = new Command("Keluar", Command.SCREEN, 1);
```

```
        canvas = new CanvasGUIMidlet01();
        canvas.addCommand(cmdKeluar);
        }

public void startApp() {
        display = Display.getDisplay(this);
        display.setCurrent(canvas);
        canvas.setCommandListener(this);
}

public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
public void commandAction(Command c, Displayable d) {
        if (d == canvas) {
        if (c == cmdKeluar) {
        destroyApp(false);
        notifyDestroyed();
        }
        }
        }
}

class CanvasGUIMidlet01 extends Canvas {
public void paint(Graphics g) {
}
}
```

Praktik 2

Menambahkan Tulisan Pada Canvas

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class CanvasText extends MIDlet implements CommandListener {
private Display display;
private Canvas canvas;
private Command cmdKeluar;

public CanvasText() {
      cmdKeluar = new Command("Keluar", Command.SCREEN, 1);
      canvas = new CanvasCanvasText();
      canvas.addCommand(cmdKeluar);
      }

public void startApp() {
      display = Display.getDisplay(this);
      display.setCurrent(canvas);
      canvas.setCommandListener(this);
      }

public void pauseApp() {
}
```

```
public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d) {
      if (d == canvas) {
      if (c == cmdKeluar) {
      destroyApp(false);
      notifyDestroyed();
      }
      }
      }
}

class CanvasCanvasText extends Canvas {
public void paint(Graphics g) {
      g.setFont(Font.getFont(Font.FACE_SYSTEM,Font.STYLE_BOLD,
      Font.SIZE_LARGE));
      g.drawString("Menulis Di Canvas!", 0, 0, g.TOP|g.LEFT);
      g.setFont(Font.getFont(Font.FACE_PROPORTIONAL,Font.STYLE_UNDERL
      INED,Font.SIZE_LARGE));
      g.drawString("Menulis Di Kanvas!", getWidth()/2, getHeight()/2,
      g.TOP|g.HCENTER);
      g.setFont(Font.getFont(Font.FACE_SYSTEM,Font.STYLE_ITALIC,
      Font.SIZE_MEDIUM));
      g.drawString("Menulis  Di  Kanvas!",  getWidth(),  getHeight(),
      g.BOTTOM|g.RIGHT);
   }
}
```

Pada kode di atas terdapat baris sebagai berikut :

```
g.setFont(Font.getFont(Font.FACE_SYSTEM,Font.STYLE_ITALIC, Font.SIZE_MEDIUM));
```

setFont adalah method yang dimiliki oleh objek g yang berguna untuk melakukan setting font. Untuk jenis font, style font dan ukuran font digunakan objek Font dan method getFont. Method ini berguna untuk mengambil nilai jenis, style dan ukuran font. Untuk masing-masing nilai telah disediakan oleh objek Font dalam fieldnya. Berikut adalah nilai-nilai dari field-field tersebut.

Kelompok jenis (face) font :

- FACE_MONOSPACE.
- FACE_PROPOTIONAL.
- FACE_SYSTEM.

Kelompok style font :

- STYLE_BOLD.
- STYLE_ITALIC.
- STYLE_PLAIN.
- STYLE_UNDERLINED.

Kelompok ukuran font :

- SIZE_LARGE.
- SIZE_MEDIUM.
- SIZE_SMALL.

Pada kode di atas terdapat baris sebagai berikut :

g.drawString("Menulis Di Kanvas!", ", getWidth(), getHeight(), g.BOTTOM|g.RIGHT);

Method ini adalah cara untuk melukis string pada canvas. Yang perlu diperhatikan bagian `g.BOTTOM|g.RIGHT`, bagian ini merupakan titik pusat dari objek yang akan digambar.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class CanvasGambar extends MIDlet implements CommandListener{
private Command cmdKeluar;
private Canvas canvas;
private Display display;

public CanvasGambar() {
cmdKeluar = new Command("Keluar", Command.SCREEN, 0);
canvas = new CanvasCanvasGambar();
canvas.addCommand(cmdKeluar);
      }
public void startApp() {
      display = Display.getDisplay(this);
      display.setCurrent(canvas);
      canvas.setCommandListener(this);
      }
public void pauseApp() {
      }
public void destroyApp(boolean unconditional) {
      }
public void commandAction(Command c, Displayable d) {
      if (d == canvas) {
      if (c == cmdKeluar) {
      destroyApp(false);
      notifyDestroyed();
      }
      }
      }
}
class CanvasCanvasGambar extends Canvas {
public void paint(Graphics g) {
      /* background color luar - COKLAT */
      g.setColor(210, 135, 28);
      g.fillRect(0, 0, getWidth(), getHeight());
      /* background color dalam - PUTIH */
      g.setColor(255, 255, 255);
      g.fillRect(3, 20, getWidth()-7, getHeight()-27);
      /* warna border - HITAM */
      g.setColor(0, 0, 0);
```

```
        /* border luar */
        g.drawRect(0, 1, getWidth()-1, getHeight()-3);
        /* border dalam */
        g.setStrokeStyle(g.DOTTED);
        g.drawRect(3, 20, getWidth()-8, getHeight()-27);
        g.drawRect(4, 21, getWidth()-10, getHeight()-29);
        /* judul */
        g.setColor(255, 255, 255);
        g.setFont(Font.getFont(Font.FACE_SYSTEM,Font.STYLE_BOLD,
        Font.SIZE_LARGE));
        g.drawString("Gambar       Canvas",       getWidth()/2,       1,
        g.TOP|g.HCENTER);
        /* layar dan tombol */
        g.setStrokeStyle(g.SOLID);
        g.setColor(0, 0, 0);
        g.drawRoundRect(8, 25, getWidth()-40, getHeight()-37, 3, 3);
        g.drawRoundRect(getWidth()-28, 25, 19, getHeight()-57, 3, 3);
        g.drawArc(getWidth()-26, getHeight()-28, 16, 16, 0, 360);
        }
}
```

C. Tugas

Buatlah aplikasi dengan mengunakan kanvas untuk menggambar sebuah mobil. Gunakan Teori-teori diatas.

MODUL 4
CANVAS (LANJUTAN)

A. TEORI

1. Game 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 offscreen 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.

Praktik :

Ketikkan program berikut simpan didalam 1 project (1 ptoject 2 class)

Program 1 sebagai kelas utama simpan dengan nama **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();
      }
}
```

Kemudian tambahkan kelas dibawah ini dan simpan dalam satu project dengan class diatas, beri nama class **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("-----@", currentX, currentY,
      Graphics.TOP|Graphics.LEFT);
      flushGraphics();
      }
}
```

Tugas :

- Gantilah program diatas, agar yang digerakkan bukan gambar/string ”-----@” tetapi gunakan image (bebas)
- Petunjuk : image harus berekstensi .png

MODUL 5
GAME DENGAN SPRITE

A.TEORI

1. 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.

Salah satu faktor yang membuat keberhasilan game adalah grafis yang unggul. Sebagian besar objek pada sebuah game dikategorikan sebagai grafis khusus yang disebut sprites. Sebuah sprite dapat berupa bullet, monster, karakter utama, musuh, kekuatan spesial, kunci dan pintu. Pada umumnya Sprite adalah grafis animasi. Grafis animasi dibuat dari sprite yang sama namun berbeda penampakannya. Kumpulan sprite biasanya mengacu sebagai sebuah kumpulan frame. Frame ini dapat dibuat secara terurut ataupun tidak terurut. Umumnya sprites ditampilkan secara terurut untuk memudahkan pengkodean.

B. PRAKTIK

Ketik program dibawah ini, program terdiri dari 2 class yaitu MidletSpriteMove.java dan SpriteMoveCanvas.java.

Gunakan gambar yang ada pada komputer saudara (berekstensi.png) dan letakkan di directori res pada project saudara. Jika tidak ada gambar yang berekstensi .png, dapat dibuat dengan menggunakan macromedia flash atau yang lain.

1. MidletSpriteMove.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class MidletSpriteMove extends MIDlet {
private Display display;
public void startApp() {
try {
display = Display.getDisplay(this);
// Memanggil class SpriteMoveCanvas dengan membuat obyek spriteCanvas
SpriteMoveCanvas spriteCanvas = new SpriteMoveCanvas();
spriteCanvas.start();
display.setCurrent(spriteCanvas);
```

```
} catch (Exception ex) {System.out.println(ex); }
}

public Display getDisplay() {
return display;
}
public void pauseApp() {}
public void destroyApp(boolean unconditional) { }
public void exit() {
destroyApp(false);
notifyDestroyed();
}
}
```

2. SpriteMoveCanvas.java

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class SpriteMoveCanvas extends GameCanvas implements Runnable
{
private boolean isPlay; // Game berjalan berulang-ulang ketikan isPlay bernilai true
private long delay; // memberikan nilai thread secara konsisten
private int currentX, currentY; // Membuat posisi awal 'X' dan 'Y'
private int width; // untuk lebar layar/screen
private int height; // untuk lebar layar/screen
private Image image, imageTemp; //Membuat obyek Image
private Sprite sprite; //Membuat sprite transparan
private Sprite nonTransparentSprite; //Membuat sprite tidak transparan
// Constructor dan initialisasi

public SpriteMoveCanvas() throws Exception {
      super(true);
      width = getWidth();
      height = getHeight();
      currentX = width / 2;
      currentY = height / 2;
      delay = 20;
      try {
      // Load Images untuk Sprites
            image = Image.createImage("/transparent.png");
            imageTemp = Image.createImage("/credits.png");
      } catch (Exception ioex) { System.out.println(ioex); }
            sprite = new Sprite(image,32,32);
            nonTransparentSprite = new Sprite(imageTemp,32,32);
      }
// membuat otomatis start thread untuk game yang berulang-ulang(loop)
public void start() {
      isPlay = true;
      Thread t = new Thread(this);
      t.start();
}
public void stop() {
      isPlay = false;
}
// Main Game Loop
public void run() {
      Graphics g = getGraphics();
```

```
      while (isPlay == true) {
      input();
      drawScreen(g);
      try { Thread.sleep(delay); }
      catch (InterruptedException ie) {}
      }
}
// Method untuk menghandle inputan dari keypad
private void input() {
      int keyStates = getKeyStates();
      // Ke Kiri
      if ((keyStates ==4)) {
      currentX = currentX - 1;
      sprite.setFrame(1);
      }
// Ke Kanan
      else if ((keyStates == 32)) {
      if ( currentX + 5 < width) {
      currentX = currentX + 1;
      sprite.setFrame(3);
      }
      }
// Ke Atas
      if ((keyStates == 2)) {
      currentY = currentY - 1;
      sprite.setFrame(2);
      }
// Ke Bawah
      else if ((keyStates == 64)) {
      if ( currentY + 10 < height) {
            currentY = currentY + 1;
            sprite.setFrame(4);
            }
}
}
// Method untuk Display Graphics
      private void drawScreen(Graphics g) {
      g.setColor(228,77,3);
      g.fillRect(0, 0, getWidth(), getHeight());
      g.setColor(0x0000ff);
// display untuk sprites
      sprite.setPosition(currentX,currentY);
      sprite.paint(g);
      nonTransparentSprite.paint(g);
      flushGraphics();
      }
}
```

Praktik 2

File : ExampleGameSpriteMidlet.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExampleGameSpriteMidlet extends MIDlet {
  private Display display;

  public void startApp() {
    try {
      display = Display.getDisplay(this);
      ContohCanvas gameCanvas = new ContohCanvas();
      gameCanvas.start();
      display.setCurrent(gameCanvas);
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }

  public Display getDisplay() {
    return display;
  }

  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
    exit();
  }

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

file : ContohCanvas.java

```
import java.util.*;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

//tambahan Joehakim

public class ContohCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int currentX, currentY;  // To hold current position of
  private int width;        // To hold screen width
  private int height;       // To hold screen height

  // Sprites to be used
  private Sprite sprite;
  private Sprite nonTransparentSprite;
  private Sprite gambarkuSprite;

  private Timer timer;
  private TimerTask task;
  private Random          random;
//tambahan
```

```
  private int posisiX;
  private int posisiY;
  private int skor=0;
  private boolean tertabrak=false;

private class TestTimerTask extends TimerTask
  {
    public final void run()
    {
      if(posisiY == getHeight())
                {posisiY=0;
                Random random = new Random( );
            int randomNumber = Math.abs( random.nextInt() % (getWidth()-32
));
            posisiX=randomNumber;
            tertabrak=false;}
      posisiY+=1;
    }
  }

  // Constructor and initialization
  public ContohCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
    currentX = width / 2;
    currentY = height / 2;
    delay =2;

    // Load Images to Sprites
    Image image = Image.createImage("/transparent.png");
    sprite = new Sprite (image,32,32);

    Image imageTemp = Image.createImage("/transparent.png");
    nonTransparentSprite = new Sprite (imageTemp,32,32);

  }

  // Automatically start thread for game loop
  public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
  }

  public void stop() { isPlay = false; }

  // Main Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {

      input();
      Menggambar(g);
      try { Thread.sleep(delay); }
      catch (InterruptedException ie) {}
    }
  }

  // Method to Handle User Inputs
  private void input() {
        Graphics graf = getGraphics();
    int keyStates = getKeyStates();
```

```
    sprite.setFrame(0);
        // Left
    if ((keyStates & LEFT_PRESSED) != 0) {
      currentX = Math.max(0, currentX - 1);
      sprite.setFrame(1);
}
    // Right
    if ((keyStates & RIGHT_PRESSED) !=0 )
      if ( currentX + 5 < width) {
        currentX = Math.min(width, currentX + 1);
        sprite.setFrame(3);
      }

    // Up
    if ((keyStates & UP_PRESSED) != 0) {
      currentY = Math.max(0, currentY - 1);
      sprite.setFrame(2);
    }

    // Down
    if ((keyStates & DOWN_PRESSED) !=0)
      if ( currentY + 10 < height) {
        currentY = Math.min(height, currentY + 1);
        sprite.setFrame(4);
      }
  }

  // Method to Display Graphics
  private void Menggambar(Graphics g) {
    //g.setColor(0xffffff);
    g.setColor(0xFF0000);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0xff00ff);
    // display sprites
    sprite.setPosition(currentX,currentY);
    sprite.paint(g);
    timer = new Timer();
    task = new TestTimerTask();
    timer.schedule(task,500);

    nonTransparentSprite.setPosition(posisiX,posisiY);
    nonTransparentSprite.paint(g);
      if(nonTransparentSprite.collidesWith(sprite, true))
         {
         if(tertabrak==false){skor+=1;tertabrak=true;};
         }
         g.setColor(0xff00ff);
              g.fillRect(0, getHeight()-20, getWidth(), getHeight());
              g.setColor(0x0000ff);
              g.drawString("JML     KETABRAK     :     "+skor,0,getHeight()-
20,Graphics.TOP|Graphics.LEFT);
          flushGraphics();
         flushGraphics();
  }
}
```

C. TUGAS

Kembangkan program diatas, buatlah sebuah permainan dengan menggunakan sprite. Misal : membuat permainan ular, setiap kali menabrak obyek, diberi skor.