Pertemuan XV
APPLET

15.1. Membentuk *Applet*

Sebuah *applet* adalah tipe yang spesial dari program java yang dieksekusi melalui internet. Secara khusus berjalan pada suatu *web browser* seperti *Netscape Navigator*, *Mozilla*, atau *Microsoft Internet Explorer*. Bagaimanapun juga, jika dibandingkan dengan aplikasi Java yang normal, tidak diperbolehkan mengakses applet pada komputer yang dijalankan untuk alasan keamanan. *Applet* ini cukup terbatas jika dibandingkan dengan aplikasi Java.

Class Applet adalah sebuah *subclass* dari *class Panel* yang didefinisikan dalam AWT. Cara terbaik untuk memahami bagaimana membuat *applet* adalah dengan contoh. Jadi, berikut ini adalah contoh *applet* sederhana yang menampilkan "Belajar Applet!".

```
import java.awt.*;
import java.applet.*;

public class BelajarApplet extends Applet {
  public void paint(Graphics g) {
    g.drawString("Belajar Applet!", 80, 25);
  }
}
```

Ingat bahwa *applet* adalah aplikasi java yang spesial. Mereka tidak dieksekusi menggunakan perintah java. Bahkan *applet* berjalan pada *web browser* atau menggunakan *applet viewer*. Untuk membuka *applet* melalui sebuah *web browser*, secara sederhana buka dokumen HTML dimana applet terintegrasi ke dalamnya menggunakan *applet* HTML tag.

Jadi setelah berhasil melakukan proses kompilasi, buatlah file html yang isinya sebagai berikut :

```
<html>
  <title>Belajar Applet</title>

  <body>
    <applet code="BelajarApplet" width=300 height=100>
    </applet>
  </body>
</html>
```

Untuk menjalankan sebuah applet adalah melalui perintah *appletviewer*. Untuk lebih mudahnya ikuti *syntax* berikut ini :

```
appletviewer <NamaFileApplet.java>
```

atau

```
appletviewer <NamaFileHtml.html>
```

Untuk menjalankan contoh *applet* yang telah kita buat, menggunakan :

```
appletviewer BelajarApplet.java
```

atau

```
appletviewer BelajarApplet.html
```

Tag HTML pada contoh yang diberikan mengindikasikan bahwa sebuah *applet* dibuat dengan lebar 300 *pixel* dan tinggi 100 *pixel*. Kemudian, metode *drawstring* menggambar *string* "Belajar Applet!" pada posisi *pixel* (80,25) menghitung kebawah dari bagian kanan.

Gambar 15.1. Contoh tampilan *appletviewer*

Ketika membuat sebuah *applet*, diharuskan meng-*extend class Applet*. Sebagaimana yang disebutkan sebelumnya, *class*-nya dapat ditemukan dalam package *java.applet*. Oleh karena itu, meng-*import package java.applet* merupakan suatu keharusan. Juga, telah disebutkan sebelumnya bahwa *class Applet* adalah *subclass* dari *class Panel*. Hal ini mengimplikasikan bahwa beberapa metode dari *class applet* ditemukan dalam *class Panel*. Untuk mengakses metode atau *field* dalam *class Panel* atau *class-class* induk, diperlukan suatu aksi untuk *import package java.awt* .

15.2. Metode-Metode *Applet*

15.2.1. Siklus *Applet* (*The Applet Life Cycle*)

Bahkan untuk memulai eksekusi pada metode main seperti dalam aplikasi khas Java, *browser* atau *applet viewer* berhubungan dengan *applet* melalui metode-metode berikut :

a. `init()`
   init adalah metode yang dipanggil pertama kali. Yang sebenarnya berisi permintaan pertama ketika *applet* di *load*.

b. `start()`
   Setelah meminta metode *init*, mulai dengan metode yang dipangil selanjutnya. metode ini meminta dokumen HTML yang ditampilkan *applet* setiap waktu. Eksekusi ringkasan dengan metode ini dilakukan ketika *applet* ditampilkan kembali.

c. `stop()`
   Ketika *web browser* meninggalkan dokumen HTML *applet*, metode ini dipanggil untuk menginformasikan *applet* bahwa dia harus menghentikan proses eksekusinya.
d. `destroy()`
   Metode ini dipanggil ketika *applet* perlu dihapus dari memori. Metode *stop* selalu dipanggil sebelum metode ini diminta untuk dijalankan.

Ketika membuat *applet*, sedikitnya beberapa dari metode ini telah meng-override. Contoh *applet* berikut meng-*override* metode-metode tersebut.

```
import java.applet.*;
import java.awt.*;

public class LifeCycleDemo extends Applet {
  String msg ="";
  public void init() {
    msg += "initializing... ";
    repaint();
  }

  public void start() {
    msg += "starting... ";
    repaint();
  }

  public void stop() {
    msg += "stopping... ";
    repaint();
  }

  public void destroy() {
    msg += "preparing for unloading...";
    repaint();
  }

  public void paint(Graphics g) {
    g.drawString(msg, 15, 15);
  }
}
```

Berikut ini contoh file html untuk melengkapi *applet LifeCycleDemo*.

```
<html>
  <title>Applet Life Cycle Demo</title>

  <body>
    <applet code="LifeCycleDemo" width=300 height=100>
    </applet>
  </body>
</html>
```

15.2.2. Metode *Paint*

Metode lain yang tidak kalah penting adalah metode *paint*, yang mana *class Applet* menurunkannya dari *class* induknya yaitu *class Component*, yang meminta *output applet* setiap waktu yang diperlukan untuk dapat digambar kembali. Sebagai contoh dari setiap

instance adalah ketika sebuah *applet* tersembunyi oleh *window* lain dapat dibuat terlihat lagi. Metode ini selalu menolak ketika kita ingin membuat bagaimana *applet* yang kita buat harus terlihat seperti yang kita inginkan. Pada contoh Belajar Applet, *applet* memiliki *string* "Belajar Applet!" pada *background* setelah menolak metode *paint*.

15.2.3. Menggambar di *Applet*

Setelah kita dapat menjalankan program sederhana pada java applet, berikutnya kita belajar bagaimana menampilkan gambar di *applet*. Berikut ini contoh program untuk menggambar di *applet* :

```
import java.applet.*;
import java.awt.*;

public class GambarAppletWarna extends Applet{
  int x=250, y=100, r=50;

  public void paint(Graphics g){
    g.setColor(Color.blue);
    g.drawLine(3,300,200,10);
     g.setColor(Color.red);
     g.drawString("Garis",100,100);
     g.setColor(Color.black);
     g.drawOval(x-r,y-r,150,60);
     g.fillOval(x+30-r,y-r,60,60);
     y+=100;
     g.drawOval(x-r,y-r,200,100);
     g.drawString("Elips",265,200);
     g.drawRect(400,50,200,100);
     g.setColor(Color.magenta);
     g.fillRect(401,51,100,99);
     g.drawString("Kotak",550,100);
     g.setColor(Color.green);
     g.drawArc(400,200,200,100,90,180);
     g.setColor(Color.red);
     g.fillArc(500,200,100,99,90,270);
  }
}
```

Kode program untuk file HTML-nya adalah :

```
<html>
  <title>Menggambar Pada Applet</title>

  <body>
    <applet code="GambarAppletWarna" align="Center" width=800
height=300>
    </applet>
  </body>
</html>
```

15.2.4. Metode *ShowStatus*

Applet memiliki *window* status, dimana memberi informasi kepada kita tentang apa yang sebenarnya dilakukan *applet*. Jika kita ingin memberi *output* ke *window* status, secara sederhana memanggil metode *showStatus*.

Contoh berikut ini sama seperti contoh Belajar *Applet* tapi dengan pernyataan tambahan yang memodifikasi isi dari *window* status.

```
import java.awt.*;
import java.applet.*;

public class AppletShowStatus extends Applet {
  public void paint(Graphics g) {
    g.drawString("Belajar Applet!", 80, 25);
    showStatus("Ini penting untuk dimengerti.");
  }
}
```

File htmlnya :

```
<html>
  <title>Applet Show Status</title>

  <body>
    <applet code="AppletShowStatus" width=300 height=100>
    </applet>
  </body>
</html>
```

Berikut ini adalah contoh hasil outputnya:

Gambar 15.2. Contoh tampilan showStatus()

15.2.5. Memainkan Klip Audio

Applet juga menyediakan layanan melalui adanya suatu metode yang memungkinkan untuk memainkan file audio. Memainkan audio clips dalam sebuah applet melibatkan dua langkah dasar :

a. Dapatkan audio clip menggunakan metode getAudioClip.
b. Untuk memainkan audio clip, menggunakan metode *play* atau *loop* pada obyek *audio clip*. Metode *play* memungkinkan kita untuk memainkan audio satu kali, sedangkan metode loop untuk memainkan audio berulang-ulang sampai metode stop dipanggil.

Contoh berikut ini digunakan untuk memainkan file audio secara terus-menerus hingga metode stop.

```
 import java.awt.*;
 import java.applet.*;

 public class AudioApplet extends Applet {
   AudioClip ac;
   Image img;
   MediaTracker mt;
   String NamaFileAudio = "Secret.wav";

   public void init() {
     try {
       /*audio clip tersimpan dalam direktori yang sama dengan kode
javanya*/
       ac = getAudioClip(getCodeBase(), NamaFileAudio);
       ac.loop();
     }
      catch (Exception e) {
       System.out.println(e);
     }
   }

   public void stop() {
     ac.stop();
   }

   public void paint(Graphics g) {
     mt = new MediaTracker(this);

     g.setFont(new Font("TimesRoman", Font.BOLD, 16));
     g.setColor(Color.blue);
     g.drawString("Playing music!", 80, 25);

     img = getImage(getCodeBase(),"120Ayumi Hamasaki10488.jpg");
     mt.addImage(img,0);
     g.drawImage(img,70,45,this);

     img = getImage(getCodeBase(),"bailarinazc4.gif");
     mt.addImage(img,0);
     g.drawImage(img,70,75,this);


     showStatus("Playing : "+NamaFileAudio);
   }
 }
```

15.3. Tag-tag Applet HTML

Pada contoh sebelumnya, kita sudah melihat bagaimana *tag-tag applet* HTML digunakan dalam dokumen HTML atau *source code* java. Sekarang, kita akan dikenalkan pada versi *tag-tag applet* HTML yang lebih lengkap.

```
 <APPLET
 [CODEBASE = codebaseURL]
 CODE = appletFile
 [ATL = alternateText]
 [NAME = appletInstanceName]
 WIDTH = widthInPixels HEIGHT = heightInPixels
 [ALIGN = alignment]
 [VSPACE = vspaceInPixels] [HSPACE = hspaceInPixels]
 >
```

```
[<PARAM NAME = parameterName1 VALUE = parameterValue1>]
[<PARAM NAME = parameterName2 VALUE = parameterValue2>]
...
[<PARAM NAME = parameterNamen VALUE = parameterValuen>]
[HTML that will be displayed in the absence of Java]
</APPLET>
```

Kata kunci Applet HTML Tag antara lain :

a. CODEBASE
   Direktori dimana class applet diletakkan. Untuk dokumen HTML, direktori URL sesuai dengan setting awalnya/defaultnya.
b. CODE
   Nama file yang berisi kode applet. Dengan atau tanpa nama ekstensi .java atau .class .
c. ALT
   Text ditampilkan jika browser mengerti applet tags tapi applet tidak dapat dieksekusi secara langsung. Mungkin terjadi jika Javanya disabled.
d. NAME
   Nama dari applet. Digunakan untuk memungkinkan applet yang lain untuk berkomunikasi dengan applet ini dengan menunjukkan suatu applet berdasarkan namanya.
e. WIDTH, HEIGHT
   Width dan height digunakan untuk menentukan ukuran dari window applet. Dinyatakan dalam pixel.
f. ALIGN
   Alignment atau pengaturan posisi dari applet. Satu diantara "left", "right", "top", "bottom", "middle", "baseline", "texttop", "absmiddle", atau "absbottom". Peletakan posisi secara Default tergantung pada lingkungan.
   "top" – posisi atas dari applet diratakan dengan item tertinggi dalam baris yang ada.
   "bottom", baseline – posisi bawah dari applet diratakan dengan bawah dari content lain dalam baris yang sama.
   "middle" – tengah dari applet diratakan dengan bawah dari content yang lain dalam baris yang sama.
   "texttop" – posisi atas dari applet diratakan dengan posisi atas dari applet diratakan dengan posisi tertinggi dari posisi atas pada baris yang sama.
   "absmiddle" – tengah dari applet diratakan dengan vertical middle dari content lain pada baris yang sama.
   "absbottom" – posisi bawah dari applet diratakan dengan posisi bawah dari content lain dalam baris yang sama.
g. VSPACE, HSPACE
   Spasi diatas dan dibawah (VSPACE) dan pada sisi (HSPACE) dari applet..
h. PARAM NAME, VALUE
   Untuk mengelompokkan parameter yang dapat menampilkan applet; applet dapat meminta method getParameter(String paramName).

Contoh di bawah ini mendemokan bagaimana untuk mengakses parameter tertentu pada HTML tag.

```
import java.awt.*;
import java.applet.*;

public class AppletParameter extends Applet {
  public void paint(Graphics g) {
    g.drawString(getParameter("myParam"), 80, 25);
  }
}
```

File HTML-nya adalah sebagai berikut :

```
<html>
  <title>Belajar Applet Dengan Parameter</title>

  <body>
    <applet code="AppletParameter" width=300 height=100>
    <param name="myParam" value="Belajar Applet Dengan Parameter!">
    </applet>
  </body>
</html>
```

15.4. Menampilkan Jam Analog

Untuk membuat jam analog ketik kode program berikut ini :

```
import java.applet.*;
import java.awt.*;
import java.util.*;
import java.text.*;

public class AppletJamAnalog extends Applet implements Runnable {
  /*Mendefinisikan thread timer*/
  private volatile Thread timer;
  private Date WaktuSekarang;
  private SimpleDateFormat FormatWaktu;
  private String WaktuTerakhir;
  private boolean mulai;
  private int xhSblm, yhSblm, xmSblm, ymSblm, xsSblm, ysSblm;

  public void init() {
    FormatWaktu = new SimpleDateFormat ("HH:mm:ss EEEE, dd-MM-
yyyy",Locale.getDefault());
     WaktuSekarang = new Date();
    WaktuTerakhir = FormatWaktu.format(WaktuSekarang);
    mulai=false;
    xhSblm=0;
    yhSblm=0;
    xmSblm=0;
    ymSblm=0;
    xsSblm=0;
    ysSblm=0;
  }

  public void paint(Graphics g) {
    int jam = 0, menit = 0, detik = 0;
     int xh, yh, xm, ym, xs, ys;
```

```
      int xPusat = 100, yPusat = 80, xTeksWaktu=5, yTeksWaktu=175;
     String today;
      int i;

     WaktuSekarang = new Date();

      /*membaca jam, menit dan detik saat ini*/
     FormatWaktu.applyPattern("h");
     try {
       jam = Integer.parseInt(FormatWaktu.format(WaktuSekarang));
     } catch (NumberFormatException n) {
       jam = 0;
     }

      FormatWaktu.applyPattern("m");
     try {
       menit = Integer.parseInt(FormatWaktu.format(WaktuSekarang));
     } catch (NumberFormatException n) {
       menit = 0;
     }

     FormatWaktu.applyPattern("s");
     try {
       detik = Integer.parseInt(FormatWaktu.format(WaktuSekarang));
     } catch (NumberFormatException n) {
       detik = 0;
     }

      /*Jika jarum dan teks waktu sudah ditampilkan hapus dengan warna
background*/
      if (mulai) {
        g.setColor(getBackground());
        g.drawLine(xPusat, yPusat, xsSblm, ysSblm);
       g.drawLine(xPusat, yPusat-1, xmSblm, ymSblm);
       g.drawLine(xPusat-1, yPusat, xmSblm, ymSblm);
       g.drawLine(xPusat, yPusat-1, xhSblm, yhSblm);
       g.drawLine(xPusat-1, yPusat, xhSblm, yhSblm);
       g.drawString(WaktuTerakhir, xTeksWaktu, yTeksWaktu);
      }

      /*Gambar lingkaran jam*/
     g.setColor(Color.blue);
     g.drawArc(xPusat-50, yPusat-50, 100, 100, 0, 360);

      for (i = 1;i <= 12;i++){
       xs = (int) (60 * Math.sin((i * 30) * Math.PI/ 180));
       ys = (int) (60 * Math.cos((i * 30) * Math.PI/ 180));

       g.drawString(Integer.toString(i), (xPusat+xs)-5, (yPusat-ys)+5);
     }

      FormatWaktu.applyPattern("HH:mm:ss EEEE, dd-MM-yyyy");
     today = FormatWaktu.format(WaktuSekarang);

     g.setColor(Color.red);
     g.drawString(today, xTeksWaktu, yTeksWaktu);

      /*Menghitung koordinat ujung jarum */
      xs = (int) (Math.cos(detik * Math.PI / 30 - Math.PI / 2) * 45 +
xPusat);
```

```
     ys = (int) (Math.sin(detik * Math.PI / 30 - Math.PI / 2) * 45 +
yPusat);
     xm = (int) (Math.cos(menit * Math.PI / 30 - Math.PI / 2) * 40 +
xPusat);
     ym = (int) (Math.sin(menit * Math.PI / 30 - Math.PI / 2) * 40 +
yPusat);
     xh = (int) (Math.cos((jam*30 + menit / 2) * Math.PI / 180 - Math.PI
/ 2) * 30+ xPusat);
     yh = (int) (Math.sin((jam*30 + menit / 2) * Math.PI / 180 - Math.PI
/ 2) * 30+ yPusat);

      /*Mengubah warna menjadi hitam*/
     g.setColor(Color.black);
      /* Menampilkan teks waktu sekarang */
     g.drawString(today, xTeksWaktu, yTeksWaktu);
     /* Gambar jarum detik */
     g.drawLine(xPusat, yPusat, xs, ys);
      /*Mengubah warna menjadi biru*/
     g.setColor(Color.blue);
      /* Gambar jarum menit */
     g.drawLine(xPusat, yPusat-1, xm, ym);
     g.drawLine(xPusat-1, yPusat, xm, ym);
      /* Gambar jarum jam */
     g.drawLine(xPusat, yPusat-1, xh, yh);
     g.drawLine(xPusat-1, yPusat, xh, yh);

      /*simpan koordinat masing-masing jarum*/
     xhSblm=xh;
     yhSblm=yh;
     xmSblm=xm;
     ymSblm=ym;
     xsSblm=xs;
     ysSblm=ys;

     WaktuTerakhir = today;
     WaktuSekarang = null;
      mulai=true;
   }

   public void start() {
     timer = new Thread(this);
     timer.start();
   }

   public void stop() {
     timer = null;
   }

   public void run() {
     Thread me = Thread.currentThread();
     while (timer == me) {
       try {
         Thread.currentThread().sleep(100);
       } catch (InterruptedException e) {
       }
       repaint();
     }
   }

   public void update(Graphics g) {
     paint(g);
```

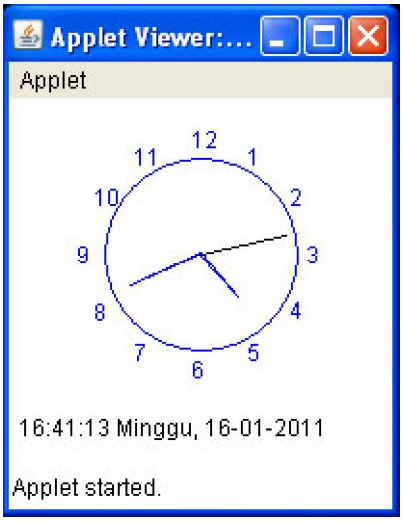
```
    }
}
```

Kode program untuk file HTML-nya adalah sebagai berikut :

```
<html>
  <title>Menggambar Pada Applet</title>

  <body>
    <applet code="AppletJamAnalog" align= "Center" width=200 height=190>
    </applet>
  </body>
</html>
```

Tampilan ketika dijalankan dengan appletviewer :



Gambar 15.3. Tampilan Jam Analog dengan Appletviewer

Referensi:

1. Hariyanto, Bambang, (2007), *Esensi-esensi Bahasa Pemrograman Java*, Edisi 2, Informatika Bandung, November 2007.
2. Utomo, Eko Priyo, (2009), *Panduan Mudah Mengenal Bahasa Java*, Yrama Widya, Juni 2009.
3. Tim Pengembang JENI, JENI 1-6, Depdiknas, 2007