MATCH GAME

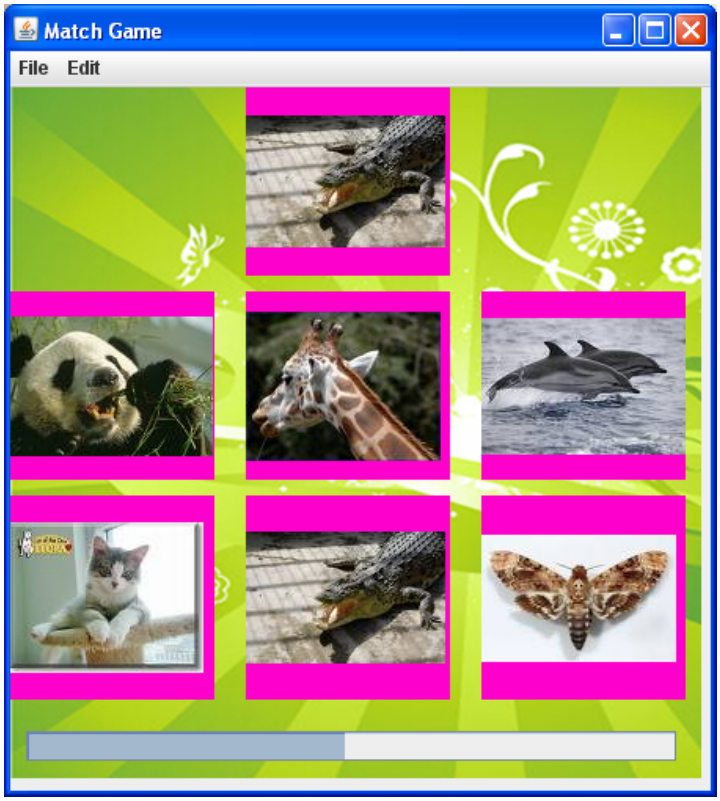
Target Praktikum :

- Mahasiswa dapat membuat aplikasi pembacaan file.
- Mahasiswa dapat menggunakan ImageIcon, Menu Bar, ProgressBar.
- Mahasiswa dapat menggunakan JDialog, membuka dan menutup JDialog.
- Mahasiswa dapat menggunakan Java Media Framework.

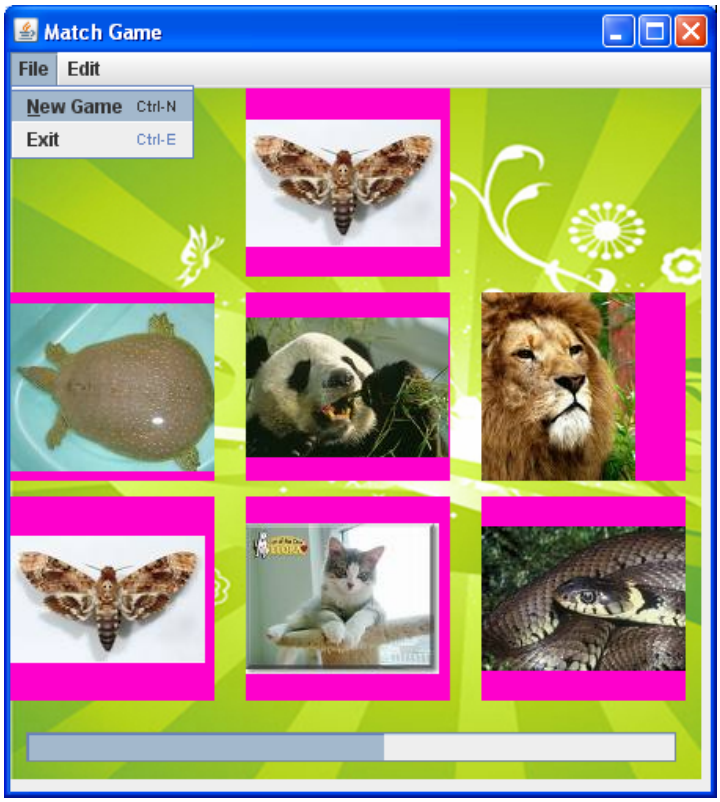
Pada praktikum ini mahasiswa membuat aplikasi tentang Match Game. User harus mencocokkan gambar acuan dengan 6 gambar yang ada dan harus memilih mana yang sama. Jika user benar dalam melakukan pencocokan maka tampilkan suara ”Kamu Hebat” (suara sesuai keinginan), jika user salah maka tampilkan suara ”Masih Salah Coba Lagi” (suara sesuai keinginan).

Kategori gambar dan data gambar tergantung dari keinginan user (bisa ditambahkan maupun dikurangi). Kategori gambar dan nama gambar tersimpan dalam file text.
Pada aplikasi terdapat menu bar File dan Edit. Isi menu File adalah : New Game dan Exit, sedangkan isi menu Edit adalah Kategori.

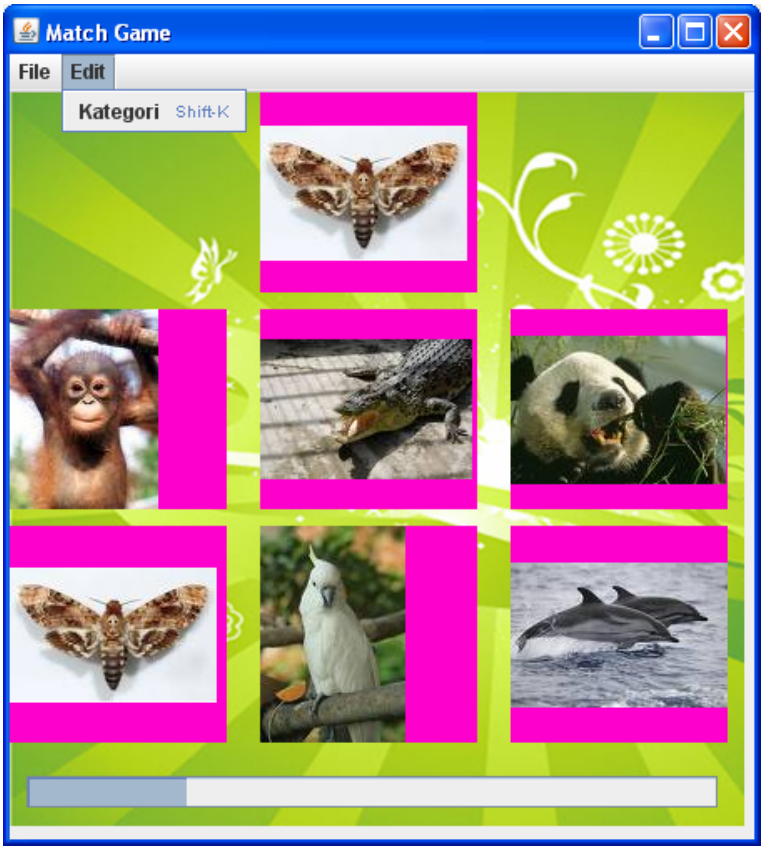
Tampilan utama ditunjukkan pada gambar 1, sedangkan isi dari menu File dan Edit ditunjukkan pada gambar 2 dan 3. Pada menu Edit, jika kita pilih menu item kategori untuk mengubah kategori dari gambar maka muncul seperti gambar 4. Hasil ditunjukkan pada gambar 5.



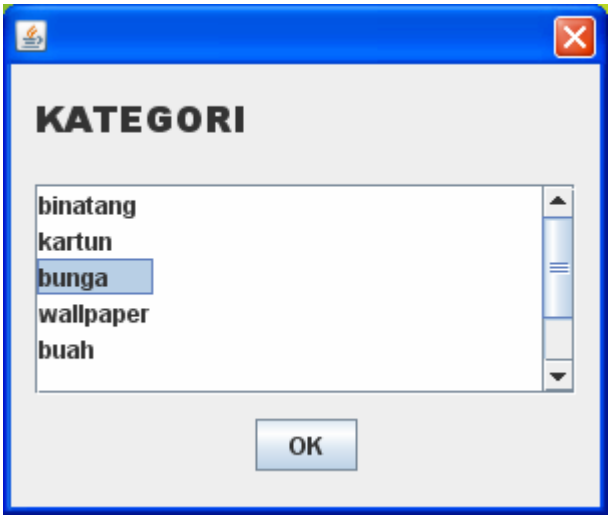
Gambar 1



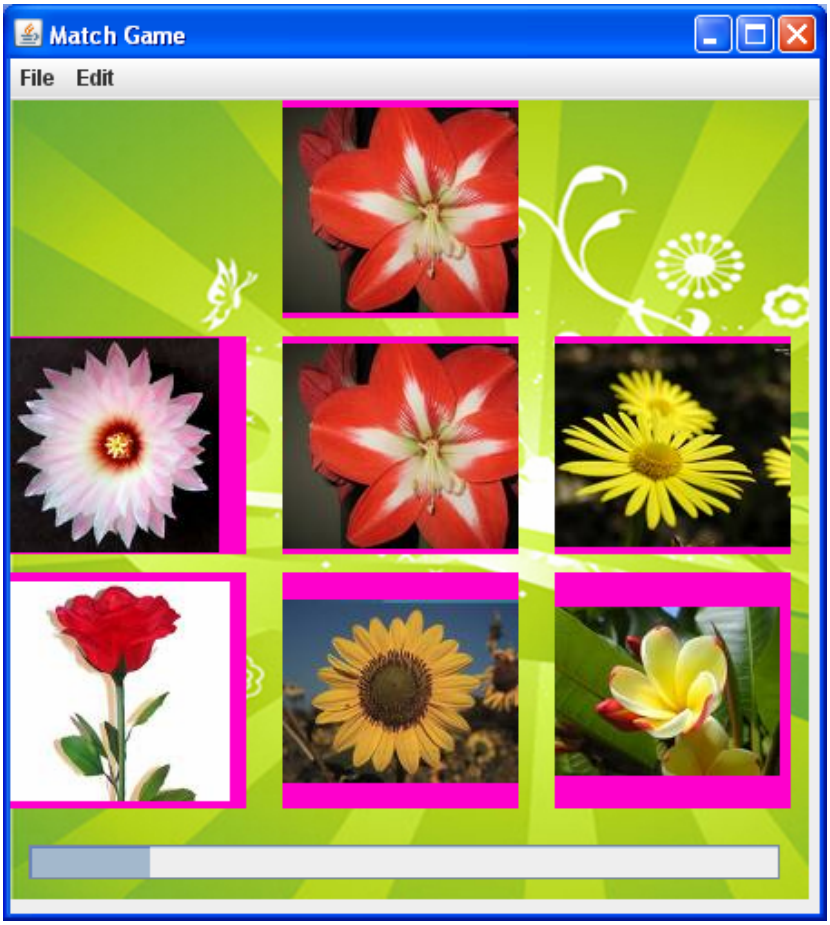
Gambar 2



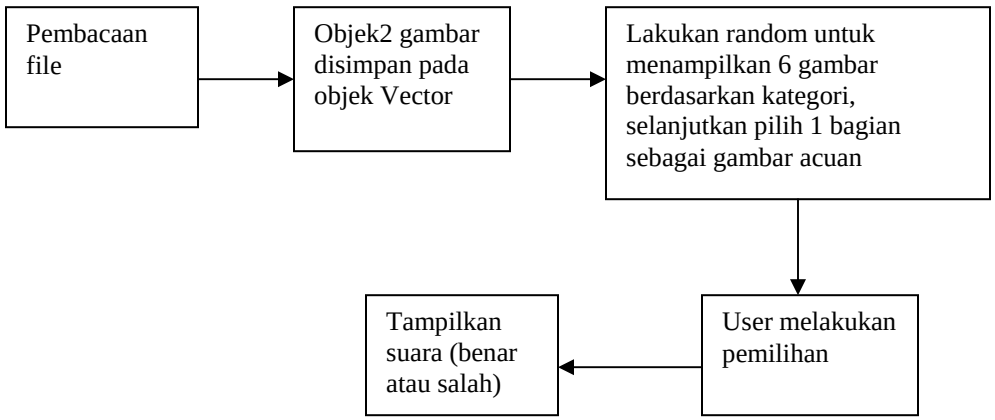
Gambar 3



Gambar 4



Gambar 5

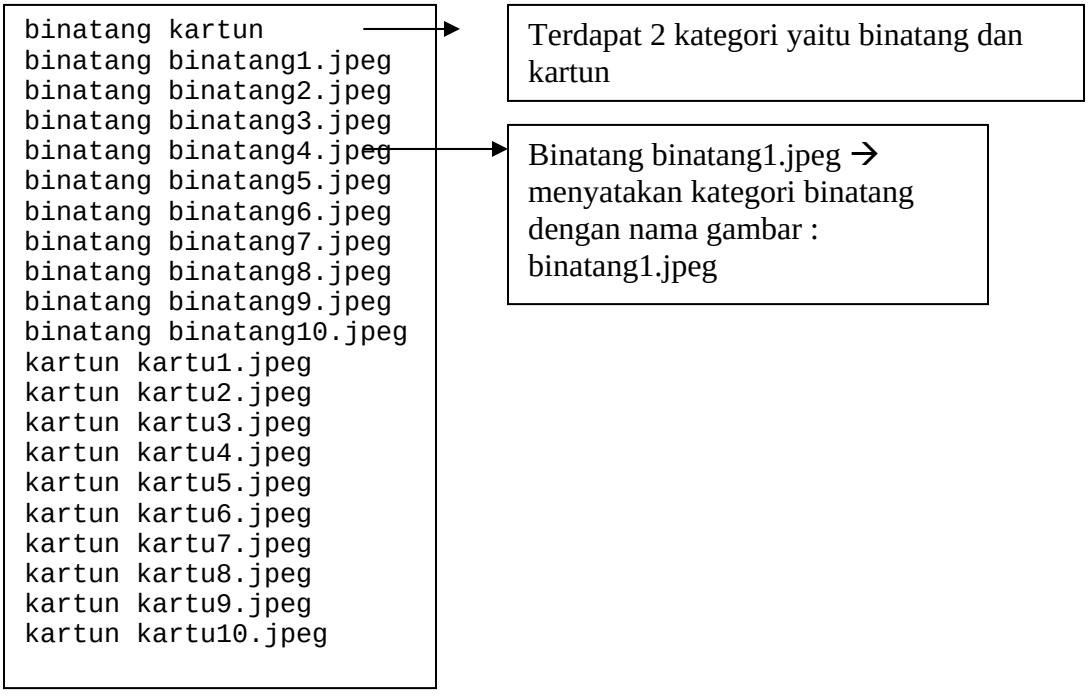


- Dalam pembuatan aplikasi "Match Game", variabel-variabel penting adalah :
  - Vector dataGambar : objek dataGambar berupa Vector. Objek dataGambar ini digunakan untuk menyimpan objek-objek Gambar, pada saat pembacaan file txt pada baris kedua dan baris selanjutnya.
  - Vector kategori : objek kategori bertipe Vector, digunakan untuk menyimpan kategori gambar. Isi dari objek kategori berupa String. Penyimpanan ini dilakukan pada saat pembacaan file txt, data pada baris pertama.

- Gambar kategoriMain : objek kategoriMain berupa objek Gambar, untuk menyimpan kategori dari gambar yang akan dibangkitkan (contoh kategori misal : bunga, binatang, kendaraan). Yang dipentingkan pada objek kategoriMain ini adalah kategorinya, sehingga namaGambarnya bisa diisi dengan sembarang nilai. Contoh :
  Gambar kategoriMain = new Gambar("binatang","abc");
- int indexGambar[] : pada aplikasi akan menampilkan gambar sebanyak 6. Misal objek gambar dengan kategori kartun tersimpan pada index 10 – 19. Maka indexGambar ini akan menyimpan 6 data yaitu bilangan antara 10 – 19. Dengan asumsi bilangan tidak boleh sama. Contoh : int indexGambar[] = {11,18,17,13,15}
- int noAcuan : dari data index yang tersimpan di indexGambar, kita ambil salah satu, kita simpan pada variabel noAcuan. Contoh int noAcuan = 17 ;
- int value =0 : untuk memberikan nilai awal pada progress bar. Contoh pBar.setValue(value);

- **Pembacaan File.**

Penyimpanan nama gambar menggunakan file teks dengan format seperti di bawah ini. Baris pertama menyatakan kategori yang ada pada aplikasi Match Game, sedangkan baris berikutnya menyatakan kategori dan nama gambar.



- **Objek-objek gambar disimpan pada objek Vector**

Baris pertama pada file teks menyatakan kategori, disimpan pada Vector kategori.

Vector kategori

0	new String(binatang)
1	new String(kartun)

Setiap baris disimpan sebagai objek Gambar, selanjutnya objek-objek tersebut disimpan pada Vector dataGambar. Buatlah class Gambar seperti di bawah ini :

```
public class Gambar {
    private String kategori ;
    private String namaGambar ;

    public Gambar(String k, String nm){
        this.kategori = k ;
        this.namaGambar = nm ;
    }

    public void setKategori(String kategori){
        this.kategori = kategori ;
    }

    public void setNamaGambar(String nm){
        this.namaGambar = nm ;
    }

    public String getKategori(){
        return kategori ;
    }

    public String getNamaGambar(){
        return namaGambar ;
    }

    public String toString(){
        return kategori + " " + namaGambar ;
    }
}
```

Sehingga isi dari Vector dataGambar sbb :

0	new Gambar(binatang, binatang1.jpeg);
1	new Gambar(binatang, binatang2.jpeg);
2	new Gambar(binatang, binatang3.jpeg);
3	new Gambar(binatang, binatang4.jpeg);
4	new Gambar(binatang, binatang5.jpeg);
5	new Gambar(binatang, binatang6.jpeg);
6	new Gambar(binatang, binatang7.jpeg);
7	new Gambar(binatang, binatang8.jpeg);
8	new Gambar(binatang, binatang9.jpeg);
9	new Gambar(binatang, binatang10.jpeg);
10	new Gambar(kartun,kartun1.jpeg);
11	new Gambar(kartun,kartun2.jpeg);
12	new Gambar(kartun,kartun3.jpeg);
13	new Gambar(kartun,kartun4.jpeg);
14	new Gambar(kartun,kartun5.jpeg);
15	new Gambar(kartun,kartun6.jpeg);

16	new Gambar(kartun,kartun7.jpeg);
17	new Gambar(kartun,kartun8.jpeg);
18	new Gambar(kartun,kartun9.jpeg);
19	new Gambar(kartun,kartun10.jpeg);

- **Lakukan random untuk menampilkan 6 gambar berdasarkan kategori, selanjutkan pilih 1 bagian sebagai gambar acuan**

Pertama kali tentukan kategori dari data gambar yang akan ditampilkan. Kategori berupa objek Gambar, misal kategoriMain = new Gambar("binatang","binatang1.jpeg");

Sebagai bantuan, terdapat 3 fungsi yaitu :

- private int[] searchGambar(){} : untuk mendapatkan 6 data gambar yang akan ditampilkan untuk dipilih oleh user. Yang disimpan adalah no indeks dari data tsb.
- private int gambarAcuan(int temp[]){} : dari 6 data gambar yang diperoleh dari fungsi searchGambar(), diambil satu data gambarAcuan.
- private void TampilGambar(int temp[]){} : fungsi ini bertugas untuk menampilkan ke objek Jlabel.

```
private int[] searchGambar(){
        int awal = dataGambar.indexOf(kategoriMain);
        int akhir = dataGambar.lastIndexOf(kategoriMain);
        . . .
}
```

Fungsi indexOf untuk mendapatkan index yang pertama kali pada objek Vector dataGambar yang sesuai dengan objek kategoriMain. Fungsi lastIndexOf untuk mendapatkan index yang terakhir kali yang sesuai dengan objek kategoriMain.

Supaya fungsi ini bisa digunakan maka lakukan overriding pada fungsi equals() pada class Gambar. Fungsi equals() ini, menghasilkan nilai true jika dua objek dari class Gambar mempunyai kategori yang sama, bernilai false jika dua objek dari class Gambar tidak mempunyai `kategori yang sama.`

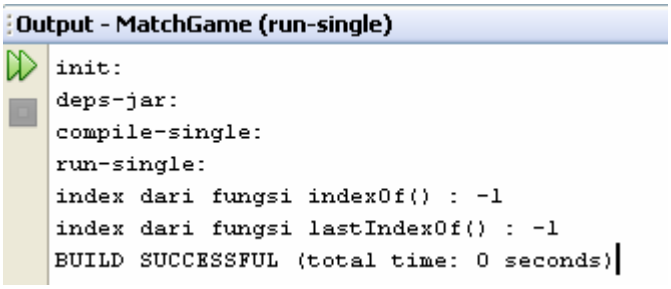
```
public class Gambar {
    private String kategori ;
    private String namaGambar ;
    public boolean equals(Object o){
       // isi fungsi
    }
}
```

Jika objek kategoriMain mempunyai kategori binatang maka variabel awal = 0, akhir = 9. Jika objek kategoriMain mempunyai kategori kartun maka awal = 10, akhir = 19. Sehingga 6 data gambar yang disimpan adalah indeks diantara awal – akhir.

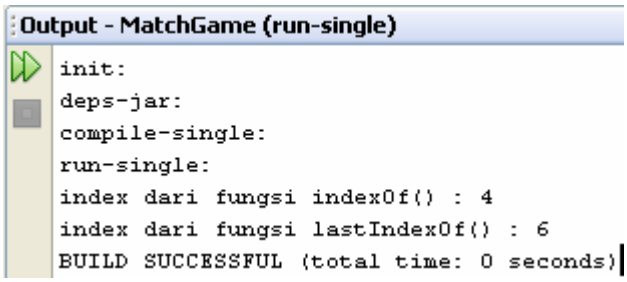
Dibawah ini contoh penggunaan method indexOf() dan lastIndexOf () tanpa melakukan overriding terhadap method equals(boolean) (isi dari fungsi equals menyatakan dua buah objek dikatakan sama jika mempunyai kategori yang sama) , hasil dari fungsi indexOf()

adalah -1 yang artinya tidak menemukan objek yang sesuai dengan objek dengan kriteria yang dimaksud. Hasil sama seperti lastIndexOf () yaitu -1.

```
import java.util.Vector;

public class Test {
   public static void main(String args[]){
       Vector v = new Vector();
       Gambar gambarAcuan = new Gambar("kartun","k2.jpeg");
       v.add(new Gambar("binatang","b1.jpeg"));
       v.add(new Gambar("binatang","b2.jpeg"));
       v.add(new Gambar("binatang","b3.jpeg"));
       v.add(new Gambar("binatang","b4.jpeg"));
       v.add(new Gambar("kartun","k1.jpeg"));
       v.add(new Gambar("kartun","k2.jpeg"));
       v.add(new Gambar("kartun","k3.jpeg"));
       v.add(new Gambar("bunga","bg1.jpeg"));
       v.add(new Gambar("bunga","bg2.jpeg"));
       v.add(new Gambar("bunga","bg3.jpeg"));
       v.add(new Gambar("bunga","bg4.jpeg"));
       v.add(new Gambar("bunga","bg5.jpeg"));

       int index1 = v.indexOf(gambarAcuan);
       System.out.print("index dari fungsi indexOf() : " + index1 +
"\n");
       int index2 = v.lastIndexOf(gambarAcuan);
       System.out.print("index dari fungsi lastIndexOf() : " + index2);
   }
}
```



Selanjutnya lakukan overriding terhadap fungsi equals(). Hasilnya akan tampak sebagai berikut:



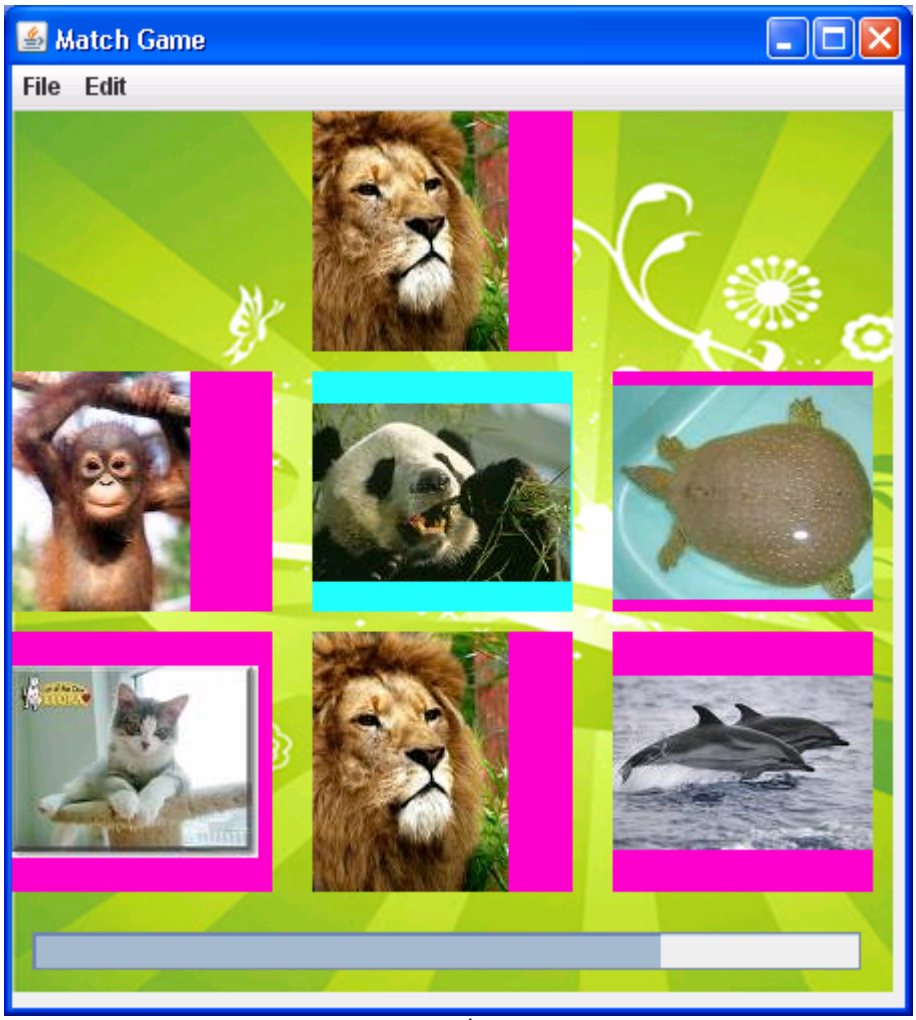
`private void TampilGambar(int temp[]){}` : fungsi ini bertugas untuk menampilkan ke objek Jlabel. Caranya : (s2 berupa string)

```
jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource(s2)));
```

- User melakukan pemilihan

Jika kursor user mengenai sebuah label maka background dari label menjadi berwarna biru (default berwarna pink – warna sesuai selera user). Untuk melakukan hal tersebut kita menggunakan event MouseListener.
Klik kanan pada label pilih Event → pilih Mouse → pilih MouseEntered(), isi dengan program di bawah ini, lakukan hal yang sama untuk fungsi MouseEntered dengan warna yang berbeda.

```
private void gb5MouseEntered(java.awt.event.MouseEvent evt) {
        gb5.setBackground(new Color(32,255,255));
}
```



Gambar 6

Gambar 7

- **Tampilkan suara (benar atau salah)**

Pada praktikum ini mahasiswa membuat aplikasi tentang Match Game. User harus mencocokkan gambar acuan dengan 6 gambar yang ada dan harus memilih mana yang sama. Jika user benar dalam melakukan pencocokan maka tampilkan suara ”Kamu Hebat” (suara sesuai keinginan), jika user salah maka tampilkan suara ”Masih Salah Coba Lagi” (suara sesuai keinginan). Untuk menampilkan suara ini kita menggunakan Java Media Framework.

JMF yang menawarkan beberapa mekanisme untuk memutar media. Mekanisme yang sederhana adalah dengan menggunakan obyek yang mengimplementasikan interface Player yang dideklarasikan dalam package javax.media. Paket javax.media dan subpackage berisi kelas-kelas yang mendukung Java Media Framework. Untuk memutar media clip, Anda harus terlebih dahulu membuat sebuah URL yang mengacu ke obyek tersebut. Selanjutnya URL tersebut sebagai parameter dari method static createRealizedPlayer dari class Manager untuk mendapatkan Player untuk media clip. Class Manager mempunyai method-method untuk mengakses system resource untuk memainkan dan memanipulasi media.

Buatlah fungsi seperti di bawah ini dan panggilah fungsi dengan cara : play("backSound.wav");

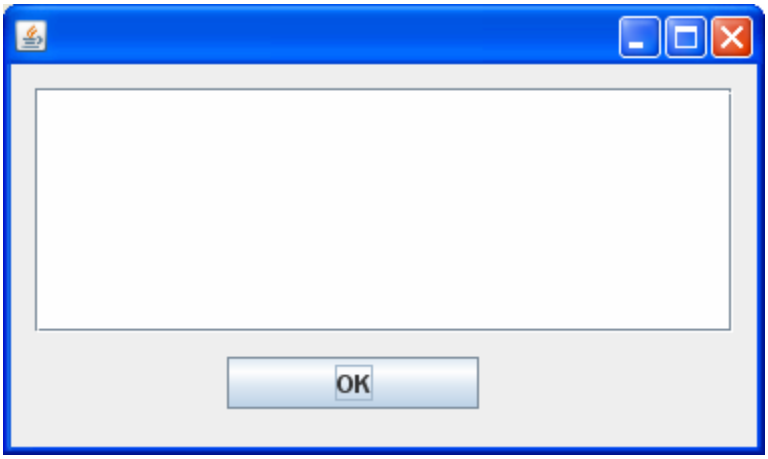
```
public void play(String namafile){
   try{
     File f = new File(namafile);

     // create a player to play the media specified in the URL
```

```
      Player mediaPlayer = Manager.createRealizedPlayer(f.toURL());
      mediaPlayer.start(); // start playing the media clip
   } // end try
   catch ( NoPlayerException noPlayerException )
   {
         System.err.println( "No media player found" );
   } // end catch
   catch ( CannotRealizeException cannotRealizeException )
   {
         System.err.println( "Could not realize media player" );
   } // end catch
   catch ( IOException iOException )
   {
         System.err.println( "Error reading from the source" );
   } // end catch
}
```

Persiapan Praktikum :

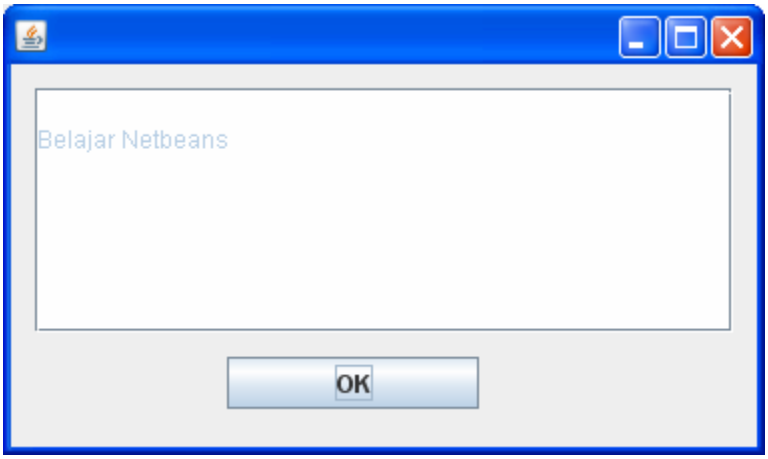
Persiapan 1

Buatlah aplikasi dimana terdapat form utama yang terdiri dari text area dan sebuah button(gambar 19), beri nama classnya dengan Frame1.java. Jika button ok ditekan maka akan muncul form seperti gambar 20 yang terdiri sebuah lable, sebuah textfield dan sebuah button (beri nama classnya Dialog1.java). Kita masukkan string dan tekan button ok (gambar 20), hasil akan tampak pada gambar 21, ulangi lagi dengan langkah yang sama maka hasil akan tampak pada gambar 21.



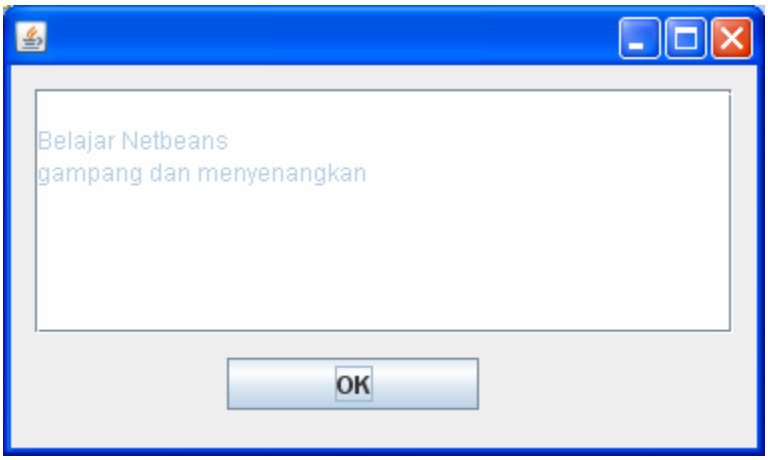
Gambar 19

Gambar 20



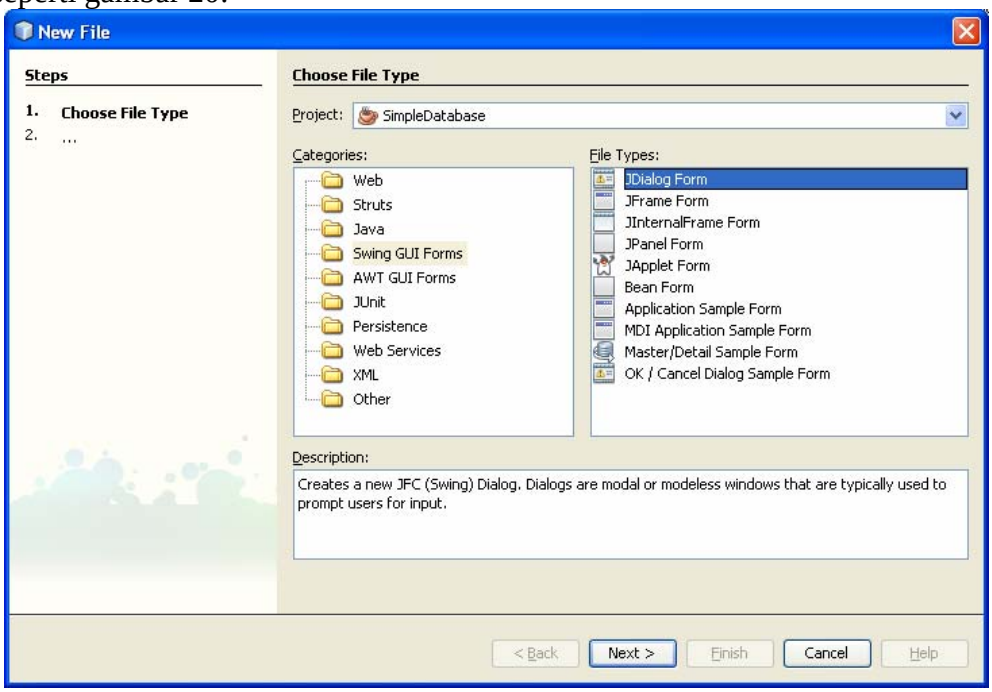
Gambar 21

Gambar 22



Gambar 23

Untuk membuat form seperti gambar 20, buatlah file baru pilih kategori: Swing GUI Forms dengan type file JDialog Form (gambar 24). Class yang dibuat merupakan class yang extends class JDialog. Masukkan sebuah lable, sebuah textfield dan sebuah button, form seperti gambar 20.



Gambar 24

Pada class Dialog1, buat variabel f1 dari class Frame1, jangan lupa deklarasikan private (memenuhi aturan enkapsulasi). Pada konstruktur class Dialog1 terdapat 2 parameter yaitu Frame dan boolean, tambahkan :

```
f1 = (Frame1) parent;
```

Kemudian pada saat button OK ditekan pada form Dialog1, isi dari text field akan ditulis ke text area pada form Frame1. Buatlah fungsi dengan parameter String pada form Frame1 untuk menuliskan ke textArea. Sehingga fungsi ini dapat dipanggil dari form Dialog1 (deklarasikan public) cara memanggil **`f1.isiText(tf.getText());`**Untuk memanggil class Dialog1 **`new Dialog1(this,true).setVisible(true);`**

```
//Form Utama
public class Frame1 extends javax.swing.JFrame {

    /** Creates new form Frame1 */
    public Frame1() {
        initComponents();
    }

    public void isiText(String str){
        ta.append("\n"+str);
    }
```

```
    private void bOKActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new Dialog1(this,true).setVisible(true);
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Frame1().setVisible(true);
            }
        });
    }
}


public class Dialog1 extends javax.swing.JDialog {
    private Frame1 f1 ;

    /** Creates new form Dialog1 */
    public Dialog1(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        f1 = (Frame1) parent ;
        initComponents();
    }

    private void bOKActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        f1.isiText(tf.getText());
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                Dialog1 dialog = new Dialog1(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }
}
```

Persiapan 2

Java Media Framework

JMF API merupakan arsitektur yang menggabungkan protokol dan pemrograman *interface* untuk merekam, mentransmisi, dan *playback* media. Pada JMF versi 2.1.1, Sun's sebagai perusahaan pengembang bahasa pemrograman java berinisiatif untuk membawa pemrosesan time-base media kedalam bahasa pemrograman Java. Time-base media adalah mengubah data yang diterima dengan berdasarkan waktu, termasuk

didalamnya seperti audio dan video klip, MIDI, dan animasi[1,2, 3,4]. Beberapa dari fungsi JMF, yaitu :
a. Dapat digunakan untuk berbagai file multimedia pada Java Applet atau aplikasi. Format yang mendukung antara lain AU, AVI, MIDI, MPEG, QUICKTIME dan WAV.
b. Play media *streaming* dari internet
c. *Capture* audio dan video dengan mikropon dan kamera video kemudian menyimpan data tersebut kedalam format yang mendukungnya.
d. Mengirimkan audio dan video secara *realtime* ke dalam jaringan internet atau intranet.
e. Dapat digunakan untuk pemrograman penyiaran radio atau televisi secara langsung.

Dasar Pemrograman JMF
Dikembangkan atas kerjasama Sun Microsystem dan IBM Haifa pada tahun 1998 dengan keunggulan dapat digunakan sebagai Media *capture*dan pemrograman untuk media *playback* . JMF 2.1.1 API merupakan rilis terbaru dari Sun's. *Interface* dan *Class* untuk koneksi *high-level* API dalam pemrograman JMF adalah sebagai berikut:

a. DataSource
`DataSource` berfungsi untuk mengatur transfer dari media. Pada JMF DataSource diidentifikasi dengan `MediaLocator` .

b. Capture Device
Capture device merupakan perangkat keras yang digunakan untuk memperoleh data, seperti mikropon, kamera biasa, atau video kamera. *Capture* media data akan menjadi input untuk *Player* agar dapat ditampilkan.

c. Player
Player memperoleh input *stream* data audio dan video kemudian mengirimnya ke speaker atau layar. Player merupakan *interface* yang akan mempersiapkan suatu DataSource untuk dipresentasikan.

Terdapat enam tahapan pada JMF *Player*:
a. `Unrealized` : pada tahapan ini *Player* belum mengenali media yang akan digunakan.
b. `Realizing` : pada tahapan ini *Player* akan menentukan materi atau media yang akan dipakai.
c. `Realized` : pada tahapan ini *Player* mengetahui materi atau media yang digunakan dan memiliki informasi tentang tipe media yang akan ditampilkan.
d. `Prefetching` : *Player* akan mempersiapkan untuk menampilkan media. Selama tahapan ini, *Player* akan *preload* media data untuk memperoleh *resource* yang digunakan dan apa saja yang dibutuhkan untuk mulai memainkan
media data.
e. `Prefetched` : *Player* telah selesai `prefetching` media data, dan siap untuk start *Player*.
f. `Started` : Langkah ini merupakan hasil ketika memanggil `start()`. *Player* bisa untuk menampilkan media data sekarang.

d. Processor

Processor merupakan jenis dari *Player*. Dalam JMF API, *Processor* adalah *interface* dengan *extends Player*. Seperti halnya *Player*, *Processor* juga mendukung untuk kontrol menampilkan media data. Disamping enam langkah *Player* seperti yang dibahas sebelumnya, *Processor* memasukkan dua langkah sebelum proses ke `Realizing` tetapi setelah `Unrealized.`

a. `Configuring` : *Processor* masuk ke tahapan `configuring` dari `Unrealized` ketika metode `configure ()` dipanggil. *Processor* berada di `Configuring` setelah terhubung ke *DataSource*.

b. `Configured` : setelah `Configuring`, *Processor* pindah ke `Configured` ketika *Processor* telah terhubung ke *DataSource*, dan data telah memiliki format yang telah ditentukan .

e. DataSink

DataSink adalah *interface* dasar untuk objek yang membaca isi media yang dikirimkan oleh suatu *DataSource* dan mengirimkan media tersebut ke beberapa tujuan

f. Format

Format merupakan *class* yang akan menempatkan suatu objek ke suatu format media yang tepat.

g. Manager

Manager adalah *interface* yang berfungsi sebagai penghubung objek, mengintegrasikan implementasi *interface* yang digunakan dengan kelas-kelas yang ada. Misalnya dengan *Manager*dapat dibuat *Player* dari *DataSource.*

Sumber :

Agung Budi Prasetijo, Aghus Sofwan, Hery Oktafiandi, **APLIKASI WEBCAM DENGAN JAVA MEDIA FRAMEWORK**