Lampiran

Source Code Pembuatan Aplikasi Sistem Metadata Forensik

```
import java.io.File;
import java.io.FileFilter;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.attribute.BasicFileAttributes;
import java.nio.file.attribute.FileOwnerAttributeView;
import java.nio.file.attribute.UserPrincipal;
import java.security.MessageDigest;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
/**
 * @author Amikom
 */
public class Home extends javax.swing.JFrame {
    /**
     * Creates new form Home
     */
    public Home() {
        initComponents();
        setIconImage(new javax.swing.ImageIcon(this.getClass().getResource("/tag.png")).getImage());
        tabelMeta.setModel(tModel);
        tabelHasil.setModel(tModelHasil);
        disablekorelasi(false);
        setLocationRelativeTo(null);
        jPanel12.setVisible(false);
    }
    private void initComponents() {
        jFileChooser1 = new javax.swing.JFileChooser();
        buttonGroup1 = new javax.swing.ButtonGroup();
        jLabel5 = new javax.swing.JLabel();
        jFileChooser2 = new javax.swing.JFileChooser();
        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        txtNamaFile = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jTabbedPane2 = new javax.swing.JTabbedPane();
        jPanel2 = new javax.swing.JPanel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel5 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        txtLokasi = new javax.swing.JTextField();
        txtNamaFile2 = new javax.swing.JTextField();
        txtTypeFile = new javax.swing.JTextField();
        txtAuthors = new javax.swing.JTextField();
        txtComputer = new javax.swing.JTextField();
        txtOwner = new javax.swing.JTextField();
        jPanel6 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        tabelMeta = new javax.swing.JTable();
        jPanel7 = new javax.swing.JPanel();
        jScrollPane3 = new javax.swing.JScrollPane();
        txtSUM = new javax.swing.JTextArea();
        jPanel3 = new javax.swing.JPanel();
        jButton2 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        txtPath = new javax.swing.JTextField();
```

```
jButton3 = new javax.swing.JButton();
jPanel8 = new javax.swing.JPanel();
jLabel10 = new javax.swing.JLabel();
txtKorelasi = new javax.swing.JTextField();
cmbKorelasi = new javax.swing.JComboBox();
jPanel11 = new javax.swing.JPanel();
rbantara = new javax.swing.JRadioButton();
jLabel11 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();
txtAntaraAwal = new javax.swing.JFormattedTextField();
jLabel14 = new javax.swing.JLabel();
txtAntaraAkhir = new javax.swing.JFormattedTextField();
jLabel15 = new javax.swing.JLabel();
jPanel12 = new javax.swing.JPanel();
rbantara2 = new javax.swing.JRadioButton();
jLabel16 = new javax.swing.JLabel();
tglAntaraAwal = new com.toedter.calendar.JDateChooser();
jLabel17 = new javax.swing.JLabel();
tglAnataraAkhir = new com.toedter.calendar.JDateChooser();
jPanel13 = new javax.swing.JPanel();
rbSama = new javax.swing.JRadioButton();
rbSamaKecil = new javax.swing.JRadioButton();
rbSamaBesar = new javax.swing.JRadioButton();
rbBesar = new javax.swing.JRadioButton();
rbKecil = new javax.swing.JRadioButton();
jPanel9 = new javax.swing.JPanel();
chkAuthor = new javax.swing.JCheckBox();
chkFiletype = new javax.swing.JCheckBox();
chkOwner = new javax.swing.JCheckBox();
txtOwnerManual = new javax.swing.JTextField();
jPanel4 = new javax.swing.JPanel();
jScrollPane2 = new javax.swing.JScrollPane();
tabelHasil = new javax.swing.JTable();
jPanel10 = new javax.swing.JPanel();
jLabel13 = new javax.swing.JLabel();
jLabel5.setText("jLabel5");
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Korelasi Metadata");
 jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.cr
 eateEtchedBorder(), "File Induk"));
jLabel1.setText("Cari File Utama");
txtNamaFile.setEditable(false);
jButton1.setText("Browse");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(txtNamaFile)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jButton1)
        .addContainerGap())
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel1)
            .addComponent(txtNamaFile,               javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jButton1))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jLabel3.setText("Lokasi File ");
jLabel6.setText("Nama File");
jLabel4.setText("Extensi / Type File");
jLabel7.setText("Owner");
```

```
  jLabel8.setText("Computer");
  txtLokasi.setEditable(false);
  txtLokasi.setBackground(new java.awt.Color(204, 255, 204));
  txtNamaFile2.setEditable(false);
  txtNamaFile2.setBackground(new java.awt.Color(204, 255, 204));
  txtTypeFile.setEditable(false);
  txtTypeFile.setBackground(new java.awt.Color(204, 255, 204));
  txtAuthors.setEditable(false);
  txtAuthors.setBackground(new java.awt.Color(204, 255, 204));
  txtComputer.setEditable(false);
  txtComputer.setBackground(new java.awt.Color(204, 255, 204));
  txtOwner.setEditable(false);
  txtOwner.setBackground(new java.awt.Color(204, 255, 204));
  javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
  jPanel5.setLayout(jPanel5Layout);
  jPanel5Layout.setHorizontalGroup(
 Panel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel5Layout.createSequentialGroup()
          .addContainerGap()
   .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addComponent(jLabel3)
              .addComponent(jLabel6)
              .addComponent(jLabel4)
              .addComponent(jLabel7)
              .addComponent(jLabel8))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
   )
              .addComponent(txtLokasi,       javax.swing.GroupLayout.DEFAULT_SIZE,       723,
   Short.MAX_VALUE)
              .addComponent(txtNamaFile2)
              .addComponent(txtTypeFile)
              .addComponent(txtAuthors)
              .addComponent(txtComputer)
              .addComponent(txtOwner))
         .addContainerGap())
  );
  jPanel5Layout.setVerticalGroup(
jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel5Layout.createSequentialGroup()
          .addContainerGap()

   .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(jLabel3)
              .addComponent(txtLokasi,                    javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
   E)
              .addComponent(jLabel6)
              .addComponent(txtNamaFile2,                    javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
   E)
              .addComponent(jLabel4)
              .addComponent(txtTypeFile,                javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
        )
              .addComponent(jLabel7)
              .addComponent(txtAuthors,                javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
         .addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
   E)
              .addComponent(jLabel8)
              .addComponent(txtComputer,                javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
          .addComponent(txtOwner,                    javax.swing.GroupLayout.PREFERRED_SIZE,
   javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
         .addContainerGap(191, Short.MAX_VALUE))
  );
  jTabbedPane1.addTab("General", jPanel5);
  tabelMeta.setModel(new javax.swing.table.DefaultTableModel(
```

```
        new Object [][] {
            {null, null, null},
            {null, null, null},
            {null, null, null},
            {null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3"
        }
    ));
    jScrollPane1.setViewportView(tabelMeta);
    javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);
    jPanel6.setLayout(jPanel6Layout);
    jPanel6Layout.setHorizontalGroup(
jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel6Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1,        javax.swing.GroupLayout.DEFAULT_SIZE,        815,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jPanel6Layout.setVerticalGroup(
        jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel6Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1,        javax.swing.GroupLayout.DEFAULT_SIZE,        330,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jTabbedPane1.addTab("Detail", jPanel6);
    txtSUM.setEditable(false);
    txtSUM.setColumns(20);
    txtSUM.setRows(5);
    txtSUM.setAutoscrolls(false);
    jScrollPane3.setViewportView(txtSUM);
    javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);
    jPanel7.setLayout(jPanel7Layout);
    jPanel7Layout.setHorizontalGroup(
     jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel7Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane3,        javax.swing.GroupLayout.DEFAULT_SIZE,        815,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jPanel7Layout.setVerticalGroup(
  jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel7Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane3,        javax.swing.GroupLayout.DEFAULT_SIZE,        330,
     Short.MAX_VALUE)
            .addContainerGap())
    );
    jTabbedPane1.addTab("CheckSum", jPanel7);
    javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jTabbedPane1)
            .addContainerGap())
    );
    jPanel2Layout.setVerticalGroup(
     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jTabbedPane1)
            .addContainerGap())
    );
    jTabbedPane2.addTab("Detail Meta Data", jPanel2);
    jButton2.setText("Korelasi File");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });
```

```
jLabel2.setText("Tentukan Lokasi Korelasi");
txtPath.setEditable(false);
jButton3.setText("Browse");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});


 jPanel8.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.cr
 eateEtchedBorder(),                                "Kriteria                                Korelasi",
 javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
 javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Arial", 1, 10)));
jLabel10.setText("Korelasi Berdasarkan");
txtKorelasi.setEditable(false);
txtKorelasi.setBackground(new java.awt.Color(204, 255, 204));
cmbKorelasi.setModel(new javax.swing.DefaultComboBoxModel(new String[] { " ", "Creation Time",
 "Last Modified Time", "Last Access Time", "Size" }));
cmbKorelasi.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        cmbKorelasiActionPerformed(evt);
    }
});
buttonGroup1.add(rbantara);
rbantara.setText("Antara");
rbantara.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        rbantaraActionPerformed(evt);
    }
});
jLabel11.setText("Batas Awal");
jLabel12.setText("Batas Akhir");
txtAntaraAwal.setFormatterFactory(new          javax.swing.text.DefaultFormatterFactory(new
 javax.swing.text.NumberFormatter(new java.text.DecimalFormat("#0"))));
jLabel14.setText("B");
txtAntaraAkhir.setFormatterFactory(new          javax.swing.text.DefaultFormatterFactory(new
 javax.swing.text.NumberFormatter(new java.text.DecimalFormat("#0"))));
jLabel15.setText("B");
javax.swing.GroupLayout jPanel11Layout = new javax.swing.GroupLayout(jPanel11);
jPanel11.setLayout(jPanel11Layout);
jPanel11Layout.setHorizontalGroup(
 jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel11Layout.createSequentialGroup()
        .addContainerGap(
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
 false)
                .addComponent(rbantara,              javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanel11Layout.createSequentialGroup()
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel11)
                    .addComponent(jLabel12))
                .addGap(98, 98, 98)))
            .addGroup(jPanel11Layout.createSequentialGroup()
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 false)
                .addComponent(txtAntaraAkhir,  javax.swing.GroupLayout.Alignment.LEADING,
 javax.swing.GroupLayout.DEFAULT_SIZE, 123, Short.MAX_VALUE)
                .addComponent(txtAntaraAwal,  javax.swing.GroupLayout.Alignment.LEADING))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel14)
                .addComponent(jLabel15))))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jPanel11Layout.setVerticalGroup(
    jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel11Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(rbantara)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel11)
      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
            .addComponent(txtAntaraAwal,            javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jLabel14))
        .addGap(3, 3, 3)
        .addComponent(jLabel12)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(txtAntaraAkhir,           javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jLabel15))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
buttonGroup1.add(rbantara2);
rbantara2.setText("Antara");
jLabel16.setText("Dari Tanggal/jam");
tglAntaraAwal.setDate(new Date());
tglAntaraAwal.setDateFormatString("dd MM yyyy hh:mm:ss");
jLabel17.setText("Sampai Tangal/Jam");
tglAnataraAkhir.setDate(new Date());
tglAnataraAkhir.setDateFormatString("dd MM yyyy hh:mm:ss");
javax.swing.GroupLayout jPanel12Layout = new javax.swing.GroupLayout(jPanel12);
jPanel12.setLayout(jPanel12Layout);
jPanel12Layout.setHorizontalGroup(
 jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel12Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(tglAntaraAwal,     javax.swing.GroupLayout.DEFAULT_SIZE,     159,
 Short.MAX_VALUE)
            .addGroup(jPanel12Layout.createSequentialGroup()
 .addGroup(jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(rbantara2)
                    .addComponent(jLabel16)
                    .addComponent(jLabel17))
                .addGap(0, 0, Short.MAX_VALUE))
            .addComponent(tglAnataraAkhir,           javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addContainerGap())
);
jPanel12Layout.setVerticalGroup(
 jPanel12Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel12Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(rbantara2)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel16)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(tglAntaraAwal,                javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,              javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel17)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(tglAnataraAkhir,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
buttonGroup1.add(rbSama);
rbSama.setSelected(true);
rbSama.setText("Sama Dengan");
buttonGroup1.add(rbSamaKecil);
rbSamaKecil.setText("Lebih Kecil Sama Dengan");
buttonGroup1.add(rbSamaBesar);
rbSamaBesar.setText("Lebih Besar Sama Dengan");
buttonGroup1.add(rbBesar);
rbBesar.setText("Lebih Besar");
buttonGroup1.add(rbKecil);
rbKecil.setText("Lebih Kecil");
javax.swing.GroupLayout jPanel13Layout = new javax.swing.GroupLayout(jPanel13);
jPanel13.setLayout(jPanel13Layout);
jPanel13Layout.setHorizontalGroup(
 jPanel13Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel13Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel13Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(rbSama)
            .addComponent(rbKecil)
            .addComponent(rbBesar)
```

```
            .addComponent(rbSamaBesar)
            .addComponent(rbSamaKecil))
        .addContainerGap(40, Short.MAX_VALUE))
);
jPanel13Layout.setVerticalGroup(
 jPanel13Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel13Layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(rbSama)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(rbSamaKecil)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(rbSamaBesar)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(rbBesar)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(rbKecil)
        .addContainerGap())
);
javax.swing.GroupLayout jPanel8Layout = new javax.swing.GroupLayout(jPanel8);
jPanel8.setLayout(jPanel8Layout);
jPanel8Layout.setHorizontalGroup(
 jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel8Layout.createSequentialGroup()
        .addGap(10,                                    10,                                    10)
 .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
 jPanel8Layout.createSequentialGroup()
                .addComponent(jPanel13,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,               javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jPanel11,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,               javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanel12,              javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
 jPanel8Layout.createSequentialGroup()
                .addComponent(jLabel10)
                .addGap(18, 18, 18)
                .addComponent(cmbKorelasi,    javax.swing.GroupLayout.PREFERRED_SIZE,    148,
 javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtKorelasi,    javax.swing.GroupLayout.PREFERRED_SIZE,    244,
 javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(260, Short.MAX_VALUE))
);
jPanel8Layout.setVerticalGroup(
 jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel8Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel10)
            .addComponent(txtKorelasi,               javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(cmbKorelasi,               javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,              javax.swing.GroupLayout.PREFERRED_SIZE))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
 .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel13,                    javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel11,                    javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel12,                    javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addContainerGap())
);
 jPanel9.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.cr
 eateEtchedBorder(),                          "Kriteria                              Tambahan",
 javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
 javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Arial", 1, 10)));
chkAuthor.setText("Owner");
chkAuthor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        chkAuthorActionPerformed(evt);
    }
```

```
});
chkFiletype.setText("File Type");
chkOwner.setText("Cari Nama Pemilik File Disini:");
chkOwner.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        chkOwnerMouseClicked(evt);
    }
    public void mouseReleased(java.awt.event.MouseEvent evt) {
        chkOwnerMouseReleased(evt);
    }
});
javax.swing.GroupLayout jPanel9Layout = new javax.swing.GroupLayout(jPanel9);
jPanel9.setLayout(jPanel9Layout);
jPanel9Layout.setHorizontalGroup(
 jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel9Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel9Layout.createSequentialGroup()
                .addComponent(chkAuthor)
                .addGap(40, 40, 40)
                .addComponent(chkOwner)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(txtOwnerManual,  javax.swing.GroupLayout.PREFERRED_SIZE,  132,
 javax.swing.GroupLayout.PREFERRED_SIZE))
            .addComponent(chkFiletype))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jPanel9Layout.setVerticalGroup(
 jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel9Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel9Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(chkAuthor)
            .addComponent(chkOwner)
            .addComponent(txtOwnerManual,           javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,              javax.swing.GroupLayout.PREFERRED_SIZE))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(chkFiletype)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel3Layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel8,                  javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addComponent(txtPath)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton3))
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addComponent(jLabel2)
                .addGap(0, 0, Short.MAX_VALUE))
            .addComponent(jPanel9,                  javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel3Layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(jButton2)))
        .addContainerGap())
);
jPanel3Layout.setVerticalGroup(
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel3Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel2)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(txtPath,                  javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jButton3))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
        .addComponent(jPanel8,                          javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,                  javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jPanel9,                          javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE,                  javax.swing.GroupLayout.PREFERRED_SIZE)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jButton2)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
jTabbedPane2.addTab("Korelasi Meta Data", jPanel3);
tabelHasil.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null}
    },
    new String [] {
        "Title 1", "Title 2", "Title 3", "Title 4"
    }
));
jScrollPane2.setViewportView(tabelHasil);
javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
jPanel4.setLayout(jPanel4Layout);
jPanel4Layout.setHorizontalGroup(
 jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel4Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jScrollPane2,        javax.swing.GroupLayout.DEFAULT_SIZE,        840,
 Short.MAX_VALUE)
        .addContainerGap())
);
jPanel4Layout.setVerticalGroup(
 jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel4Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jScrollPane2,        javax.swing.GroupLayout.DEFAULT_SIZE,        380,
 Short.MAX_VALUE)
        .addContainerGap())
);
jTabbedPane2.addTab("Hasil Korelasi", jPanel4);
jPanel10.setBorder(javax.swing.BorderFactory.createEtchedBorder());
jLabel13.setText("Copy Right @ 2017 By Subli");
javax.swing.GroupLayout jPanel10Layout = new javax.swing.GroupLayout(jPanel10);
jPanel10.setLayout(jPanel10Layout);
jPanel10Layout.setHorizontalGroup(
    jPanel10Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 jPanel10Layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel13)
        .addContainerGap())
);
jPanel10Layout.setVerticalGroup(
 jPanel10Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jLabel13)
);
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
        .addContainerGap()
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jPanel1,                      javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jTabbedPane2)))
    .addComponent(jPanel10,                             javax.swing.GroupLayout.DEFAULT_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jPanel1,                      javax.swing.GroupLayout.PREFERRED_SIZE,
 javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addComponent(jTabbedPane2)
          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jPanel10,                    javax.swing.GroupLayout.PREFERRED_SIZE,
     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
    );
    pack();
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    kosong();
    if (jFileChooser1 == null) {
        jFileChooser1 = new JFileChooser();
        jFileChooser1.addChoosableFileFilter(new MyCostumFilter());
        jFileChooser1.setAcceptAllFileFilterUsed(false);
    }
    int a = jFileChooser1.showOpenDialog(this);
    if (a == JFileChooser.APPROVE_OPTION) {
        File file = jFileChooser1.getSelectedFile();
        txtNamaFile.setText(file.getAbsolutePath());
        getMetaData(file, file.getName());
    }
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    if (txtPath.getText().isEmpty()) {
        JOptionPane.showMessageDialog(rootPane, "Pilih Lokasi Korelasi terlebih dahulu");
    } else {
        if (rbSama.isSelected()) {
            perbandingan = 1;
        } else if (rbBesar.isSelected()) {
            perbandingan = 3;
        } else if (rbKecil.isSelected()) {
            perbandingan = 2;
        } else if (rbSamaBesar.isSelected()) {
            perbandingan = 5;
        } else if (rbantara.isSelected()) {
            perbandingan = 6;
            try {
                sizeawal = Integer.parseInt(txtAntaraAwal.getText());
                sizeakhir = Integer.parseInt(txtAntaraAkhir.getText());
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(null, "Input Size Awal atau Size akhir salah");
            }
        } else if (rbantara2.isSelected()) {
            perbandingan = 7;
        } else {
            perbandingan = 4;
        }
        if(rbantara.isSelected()        &&        (txtAntaraAwal.getText().isEmpty()        ||
    txtAntaraAkhir.getText().isEmpty())){
            JOptionPane.showMessageDialog(null,  "Isi  Size  awal  dan  atau  size  akhir  terlebih
    dahulu");
        }else
        if (chkOwner.isSelected() && txtOwnerManual.getText().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Inputkan Nama Terlebih Dahulu");
        } else if (cmbKorelasi.getSelectedIndex() == 2) {
            jTabbedPane2.setSelectedIndex(2);
            if (chkAuthor.isSelected()) {
                if (chkFiletype.isSelected()) {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), tglauthorextensiownerFilter, true);
                        walk(txtPath.getText(), tglauthorextensiownerFilter);
                    } else {
                        getHasil(txtPath.getText(), tglauthorextensiFilter, true);
                        walk(txtPath.getText(), tglauthorextensiFilter);
                    }
                } else {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), tglauthorownerFilter, true);
                        walk(txtPath.getText(), tglauthorownerFilter);
                    } else {
                        getHasil(txtPath.getText(), tglauthorFilter, true);
                        walk(txtPath.getText(), tglauthorFilter);
                    }
                }
            } else {
                if (chkFiletype.isSelected()) {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), tglextensiownerFilter, true);
```

```
                walk(txtPath.getText(), tglextensiownerFilter);
            } else {
                getHasil(txtPath.getText(), tglextensiFilter, true);
                walk(txtPath.getText(), tglextensiFilter);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), tglownerFilter, true);
                walk(txtPath.getText(), tglownerFilter);
            } else {
                getHasil(txtPath.getText(), tglFilter, true);
                walk(txtPath.getText(), tglFilter);
            }
        }
    }
    if(tabelHasil.getRowCount()<=0){
        JOptionPane.showMessageDialog(null, "Tidak ada file yang ditemukan");
    }
} else if (cmbKorelasi.getSelectedIndex() == 4) {
    jTabbedPane2.setSelectedIndex(2);
    if (chkAuthor.isSelected()) {
        if (chkFiletype.isSelected()) {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeauthorextensiownerFilter, true);
                walk(txtPath.getText(), sizeauthorextensiownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeauthorextensiFilter, true);
                walk(txtPath.getText(), sizeauthorextensiFilter);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeauthorownerFilter, true);
                walk(txtPath.getText(), sizeauthorownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeauthorFilter, true);
                walk(txtPath.getText(), sizeauthorFilter);
            }
        }
    } else {
        if (chkFiletype.isSelected()) {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeextensiownerFilter, true);
                walk(txtPath.getText(), sizeextensiownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeextensiFilter, true);
                walk(txtPath.getText(), sizeextensiFilter);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), sizeownerFilter, true);
                walk(txtPath.getText(), sizeownerFilter);
            } else {
                getHasil(txtPath.getText(), sizeFilter, true);
                walk(txtPath.getText(), sizeFilter);
            }
        }
    }
     if(tabelHasil.getRowCount()<=0){
        JOptionPane.showMessageDialog(null, "Tidak ada file yang ditemukan");
    }
} else {
    jTabbedPane2.setSelectedIndex(2);
    if (chkAuthor.isSelected()) {
        if (chkFiletype.isSelected()) {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), authorextensiowner, true);
                walk(txtPath.getText(), authorextensiowner);
            } else {
                getHasil(txtPath.getText(), authorextensi, true);
                walk(txtPath.getText(), authorextensi);
            }
        } else {
            if (chkOwner.isSelected()) {
                getHasil(txtPath.getText(), authorowner, true);
                walk(txtPath.getText(), authorowner);
            } else {
                getHasil(txtPath.getText(), author, true);
```

```
                        walk(txtPath.getText(), author);
                    }
                }
            } else {
                if (chkFiletype.isSelected()) {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), extensiowner, true);
                        walk(txtPath.getText(), extensiowner);
                    } else {
                        getHasil(txtPath.getText(), extensi, true);
                        walk(txtPath.getText(), extensi);
                    }
                } else {
                    if (chkOwner.isSelected()) {
                        getHasil(txtPath.getText(), owner, true);
                        walk(txtPath.getText(), owner);
                    } else {
                        jTabbedPane2.setSelectedIndex(1);
                        JOptionPane.showMessageDialog(rootPane, "Pilih Jenis Korelasi terlebih
     dahulu"
                                + "Pada ComboBox diatas");
                    }
                }
            }
             if(tabelHasil.getRowCount()<=0){
                JOptionPane.showMessageDialog(null, "Tidak ada file yang ditemukan");
            }
        }
    }
}
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    jFileChooser2 = new JFileChooser();
    jFileChooser2.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    jFileChooser2.setAcceptAllFileFilterUsed(false);
    int a = jFileChooser2.showOpenDialog(this);
    if (a == JFileChooser.APPROVE_OPTION) {
        File file = jFileChooser2.getSelectedFile();
        txtPath.setText(file.getAbsolutePath());
    }
}
private void chkOwnerMouseClicked(java.awt.event.MouseEvent evt) {
    if (chkOwner.isSelected()) {
        txtOwnerManual.setEnabled(true);
    } else {
        txtOwnerManual.setEnabled(false);
    }
}
private void chkOwnerMouseReleased(java.awt.event.MouseEvent evt) {
    if (chkOwner.isSelected()) {
        txtOwnerManual.setEnabled(true);
    } else {
        txtOwnerManual.setEnabled(false);
    }
}
private void cmbKorelasiActionPerformed(java.awt.event.ActionEvent evt) {
    if (cmbKorelasi.getSelectedIndex() == 0) {
        disablekorelasi(false);
    } else {
        disablekorelasi(true);
        if (cmbKorelasi.getSelectedIndex() == 1) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(0, 2).toString());
            jPanel11.setVisible(false);
            jPanel12.setVisible(true);
        } else if (cmbKorelasi.getSelectedIndex() == 2) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(1, 2).toString());
            jPanel11.setVisible(false);
            jPanel12.setVisible(true);
        } else if (cmbKorelasi.getSelectedIndex() == 3) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(2, 2).toString());
            jPanel11.setVisible(false);
            jPanel12.setVisible(true);
        } else if (cmbKorelasi.getSelectedIndex() == 4) {
            txtKorelasi.setText(cmbKorelasi.getSelectedItem().toString()     +     ":     "     +
     tabelMeta.getModel().getValueAt(7, 2).toString());
```

```
            jPanel11.setVisible(true);
            jPanel12.setVisible(false);
        }
    }
}
private void rbantaraActionPerformed(java.awt.event.ActionEvent evt) {
}
private void chkAuthorActionPerformed(java.awt.event.ActionEvent evt) {
}
public static void main(String args[]) {
    try {
        for              (javax.swing.UIManager.LookAndFeelInfo              info              :
     javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Windows".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }               catch            (InstantiationException              ex)               {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }               catch              (IllegalAccessException              ex)               {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }          catch          (javax.swing.UnsupportedLookAndFeelException          ex)          {
     java.util.logging.Logger.getLogger(Home.class.getName()).log(java.util.logging.Level.SEVERE
     , null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Home().setVisible(true);
        }
    });
}
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JCheckBox chkAuthor;
private javax.swing.JCheckBox chkFiletype;
private javax.swing.JCheckBox chkOwner;
private javax.swing.JComboBox cmbKorelasi;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JFileChooser jFileChooser1;
private javax.swing.JFileChooser jFileChooser2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel10;
private javax.swing.JPanel jPanel11;
private javax.swing.JPanel jPanel12;
private javax.swing.JPanel jPanel13;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
private javax.swing.JPanel jPanel8;
private javax.swing.JPanel jPanel9;
private javax.swing.JScrollPane jScrollPane1;
```

```
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTabbedPane jTabbedPane2;
private javax.swing.JRadioButton rbBesar;
private javax.swing.JRadioButton rbKecil;
private javax.swing.JRadioButton rbSama;
private javax.swing.JRadioButton rbSamaBesar;
private javax.swing.JRadioButton rbSamaKecil;
private javax.swing.JRadioButton rbantara;
private javax.swing.JRadioButton rbantara2;
private javax.swing.JTable tabelHasil;
private javax.swing.JTable tabelMeta;
private com.toedter.calendar.JDateChooser tglAnataraAkhir;
private com.toedter.calendar.JDateChooser tglAntaraAwal;
private javax.swing.JFormattedTextField txtAntaraAkhir;
private javax.swing.JFormattedTextField txtAntaraAwal;
private javax.swing.JTextField txtAuthors;
private javax.swing.JTextField txtComputer;
private javax.swing.JTextField txtKorelasi;
private javax.swing.JTextField txtLokasi;
private javax.swing.JTextField txtNamaFile;
private javax.swing.JTextField txtNamaFile2;
private javax.swing.JTextField txtOwner;
private javax.swing.JTextField txtOwnerManual;
private javax.swing.JTextField txtPath;
private javax.swing.JTextArea txtSUM;
private javax.swing.JTextField txtTypeFile;
String header[] = {"NO", "JENIS META", "VALUE"};
DefaultTableModel tModel = new DefaultTableModel(header, 0);
long size = 0, lastmodifide = 0, lastaccess = 0, creationtime = 0, sizeawal = 0, sizeakhir = 0,
        tglawal = 0, tglakhir = 0;
public void getMetaData(File f, String namafile) {
    tModel = new DefaultTableModel(header, 0);
    Path file = Paths.get(f.getAbsolutePath());
    try {
        FileInfo filein = new FileInfo(f);
        try {
            txtOwner.setText(filein.getOwner());
        } catch (ParseException ex) {
        }
        txtLokasi.setText(f.getPath());
        txtNamaFile2.setText(namafile);
        txtTypeFile.setText(getFileExtension(f));
        txtComputer.setText(getComputerName());
        FileOwnerAttributeView   ownerAttributeView   =   Files.getFileAttributeView(f.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = ownerAttributeView.getOwner();
     txtAuthors.setText(owner.getName().substring(owner.getName().lastIndexOf("\\") + 1));
        String checksum = "MD5 :\n" + getChecksum(f.getPath(), "MD5")
                + "\nSHA-256 :\n" + getChecksum(f.getPath(), "SHA-256");
        txtSUM.setText(checksum);
        BasicFileAttributes attr = Files.readAttributes(file, BasicFileAttributes.class);
        String data1[] = {"1", "creationTime: ", attr.creationTime().toString()};
        tModel.addRow(data1);
        creationtime = attr.creationTime().toMillis();
        String data8[] = {"2", "lastAccessTime: ", attr.lastAccessTime().toString()};
        tModel.addRow(data8);
        lastaccess = attr.lastAccessTime().toMillis();
        String data2[] = {"3", "lastModifiedTime: ", attr.lastModifiedTime().toString()};
        tModel.addRow(data2);
        lastmodifide = attr.lastModifiedTime().toMillis();
        String data3[] = {"4", "isDirectory: ", String.valueOf(attr.isDirectory())};
        tModel.addRow(data3);
        String data4[] = {"5", "isOther: ", String.valueOf(attr.isOther())};
        tModel.addRow(data4);
        String data5[] = {"6", "isRegularFile: ", String.valueOf(attr.isRegularFile())};
        tModel.addRow(data5);
        String data6[] = {"7", "isSymbolicLink: ", String.valueOf(attr.isSymbolicLink())};
        tModel.addRow(data6);
        String data7[] = {"8", "size: ", String.valueOf(attr.size())};
        size = attr.size();
        tModel.addRow(data7);
    } catch (IOException ex) {
    }
    tabelMeta.setModel(tModel);
    AturLebarKolom alk = new AturLebarKolom(tabelMeta);
```

```
    alk.adjustColumns();
}
String headerHasil[] = {"NAME FILE", "SIZE", "DATE", "FOLDER PATH"};
DefaultTableModel tModelHasil = new DefaultTableModel(headerHasil, 0);
SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM YYYY");
public void walk(String path, FileFilter dd) {
    File root = new File(path);
    File[] list = root.listFiles();
    if (list == null) {
        return;
    }
    for (File f : list) {
        if (f.isDirectory()) {
            walk(f.getAbsolutePath(), dd);
            getHasil(f.getAbsolutePath(), dd, false);
        }
    }
}
public void getHasil(String dirPath, FileFilter ff, boolean cek) {
    if (cek) {
        tModelHasil = new DefaultTableModel(headerHasil, 0);
    }
    File dir = new File(dirPath);
    File[] files = dir.listFiles(ff);
    try {
        for (File aFile : files) {
            String data[] = {aFile.getName(), String.valueOf(aFile.length()),
                sdf.format(new Date(aFile.lastModified())), aFile.getPath()};
            tModelHasil.addRow(data);
        }
    } catch (Exception ex) {
        System.out.println("Folder : " + dirPath + " Hasil = " + files);
    }
    tabelHasil.setModel(tModelHasil);
    AturLebarKolom alk = new AturLebarKolom(tabelHasil);
    alk.adjustColumns();
}
int perbandingan = 1;
FileFilter sizeFilter = new FileFilter() {
    public boolean accept(File file) {
        if (file.isFile()) {
            if (file.length() == size && perbandingan == 1) {
                return true;
            } else if (file.length() < size && perbandingan == 2) {
                return true;
            } else if (file.length() > size && perbandingan == 3) {
                return true;
            } else if (file.length() <= size && perbandingan == 4) {
                return true;
            } else if (file.length() >= size && perbandingan == 5) {
                return true;
            } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) && perbandingan
     == 6) {
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    }
};
FileFilter tglFilter = new FileFilter() {
    public boolean accept(File file) {
        if (file.isFile()) {
            if (file.lastModified() == lastmodifide && perbandingan == 1) {
                return true;
            } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                return true;
            } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                return true;
            } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                return true;
            } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                return true;
            } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir) &&
     perbandingan == 7) {
```

```
                return true;
            } else {
                return false;
            }

        } else {
            return false;
        }
    }
};
FileFilter extensi = new FileFilter() {
    @Override
    public boolean accept(File file) {
        if (file.isFile()) {
            return getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
        } else {
            return false;
        }
    }
};
FileFilter author = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    return true;
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter owner = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    return true;
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter sizeauthorFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (file.length() == size && perbandingan == 1) {
                        return true;
```

```
                    } else if (file.length() < size && perbandingan == 2) {
                        return true;
                    } else if (file.length() > size && perbandingan == 3) {
                        return true;
                    } else if (file.length() <= size && perbandingan == 4) {
                        return true;
                    } else if (file.length() >= size && perbandingan == 5) {
                        return true;
                    } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter sizeauthorextensiFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (file.length() == size && perbandingan == 1) {
                            return true;
                        } else if (file.length() < size && perbandingan == 2) {
                            return true;
                        } else if (file.length() > size && perbandingan == 3) {
                            return true;
                        } else if (file.length() <= size && perbandingan == 4) {
                            return true;
                        } else if (file.length() >= size && perbandingan == 5) {
                            return true;
                        } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter sizeauthorextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
```

```
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                            if (file.length() == size && perbandingan == 1) {
                                return true;
                            } else if (file.length() < size && perbandingan == 2) {
                                return true;
                            } else if (file.length() > size && perbandingan == 3) {
                                return true;
                            } else if (file.length() <= size && perbandingan == 4) {
                                return true;
                            } else if (file.length() >= size && perbandingan == 5) {
                                return true;
                            } else if ((file.length() >= sizeawal && file.length() <= sizeakhir)
     && perbandingan == 6) {
                                return true;
                            } else {
                                return false;
                            }
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } catch (IOException ex) {
                return false;
            }

        }
};
FileFilter sizeauthorownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if (file.length() == size && perbandingan == 1) {
                            return true;
                        } else if (file.length() < size && perbandingan == 2) {
                            return true;
                        } else if (file.length() > size && perbandingan == 3) {
                            return true;
                        } else if (file.length() <= size && perbandingan == 4) {
                            return true;
                        } else if (file.length() >= size && perbandingan == 5) {
                            return true;
                        } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
```

```
    }
};
FileFilter sizeextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if (file.length() == size && perbandingan == 1) {
                            return true;
                        } else if (file.length() < size && perbandingan == 2) {
                            return true;
                        } else if (file.length() > size && perbandingan == 3) {
                            return true;
                        } else if (file.length() <= size && perbandingan == 4) {
                            return true;
                        } else if (file.length() >= size && perbandingan == 5) {
                            return true;
                        } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }

    }
};
FileFilter sizeextensiFilter = new FileFilter() {
    @Override
    public boolean accept(File file) {
        if (file.isFile()) {
            if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                if (file.length() == size && perbandingan == 1) {
                    return true;
                } else if (file.length() < size && perbandingan == 2) {
                    return true;
                } else if (file.length() > size && perbandingan == 3) {
                    return true;
                } else if (file.length() <= size && perbandingan == 4) {
                    return true;
                } else if (file.length() >= size && perbandingan == 5) {
                    return true;
                }  else  if  ((file.length()  >=  sizeawal  &&  file.length()  <=  sizeakhir)  &&
     perbandingan == 6) {
                    return true;
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } else {
            return false;
        }
    }
};
FileFilter sizeownerFilter = new FileFilter() {
    public boolean accept(File file) {
```

```
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {

                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    if (file.length() == size && perbandingan == 1) {
                        return true;
                    } else if (file.length() < size && perbandingan == 2) {
                        return true;
                    } else if (file.length() > size && perbandingan == 3) {
                        return true;
                    } else if (file.length() <= size && perbandingan == 4) {
                        return true;
                    } else if (file.length() >= size && perbandingan == 5) {
                        return true;
                    } else if ((file.length() >= sizeawal && file.length() <= sizeakhir) &&
     perbandingan == 6) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }

        } catch (IOException ex) {
            return false;
        }

    }
};
FileFilter tglauthorFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (file.lastModified() == lastmodifide && perbandingan == 1) {
                        return true;
                    } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                        return true;
                    } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                        return true;
                    } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                        return true;
                    } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                        return true;
                    } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir)
     && perbandingan == 7) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglauthorextensiFilter = new FileFilter() {
```

```
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (file.lastModified() == lastmodifide && perbandingan == 1) {
                            return true;
                        } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                            return true;
                        } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                            return true;
                        } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                            return true;
                        } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                            return true;
                        } else if ((file.lastModified() >= tglawal && file.lastModified() <=
     tglakhir) && perbandingan == 7) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglauthorextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if
     (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                            if (file.lastModified() == lastmodifide && perbandingan == 1) {
                                return true;
                            } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                                return true;
                            } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                                return true;
                            } else if (file.lastModified() <= lastmodifide && perbandingan == 4)
     {
                                return true;
                            } else if (file.lastModified() >= lastmodifide && perbandingan == 5)
     {
                                return true;
                            } else if ((file.lastModified() >= tglawal && file.lastModified() <=
     tglakhir) && perbandingan == 7) {
                                return true;
                            } else {
                                return false;
                            }
                        } else {
                            return false;
                        }
                    } else {
                        return false;
```

```
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglextensiownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {

                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                        if (file.lastModified() == lastmodifide && perbandingan == 1) {
                            return true;
                        } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                            return true;
                        } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                            return true;
                        } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                            return true;
                        } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                            return true;
                        } else if ((file.lastModified() >= tglawal && file.lastModified() <=
     tglakhir) && perbandingan == 7) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }

                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglauthorownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                        if (file.lastModified() == lastmodifide && perbandingan == 1) {
                            return true;
                        } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                            return true;
                        } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                            return true;
                        } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                            return true;
                        } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
```

```
                            return true;
                        } else if ((file.lastModified() >= tglawal && file.lastModified() <=
    tglakhir) && perbandingan == 7) {
                            return true;
                        } else {
                            return false;
                        }
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }

            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglownerFilter = new FileFilter() {
    public boolean accept(File file) {
        FileOwnerAttributeView ownerAttributeView = Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtOwnerManual.getText().trim())) {
                    if (file.lastModified() == lastmodifide && perbandingan == 1) {
                        return true;
                    } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                        return true;
                    } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                        return true;
                    } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                        return true;
                    } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                        return true;
                    } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir)
     && perbandingan == 7) {
                        return true;
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter tglextensiFilter = new FileFilter() {
    @Override
    public boolean accept(File file) {
        if (file.isFile()) {
            if (getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim())) {
                if (file.lastModified() == lastmodifide && perbandingan == 1) {
                    return true;
                } else if (file.lastModified() < lastmodifide && perbandingan == 2) {
                    return true;
                } else if (file.lastModified() > lastmodifide && perbandingan == 3) {
                    return true;
                } else if (file.lastModified() <= lastmodifide && perbandingan == 4) {
                    return true;
                } else if (file.lastModified() >= lastmodifide && perbandingan == 5) {
                    return true;
                } else if ((file.lastModified() >= tglawal && file.lastModified() <= tglakhir) &&
     perbandingan == 7) {
                    return true;
```

```
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } else {
            return false;
        }
    }
};
FileFilter authorextensi = new FileFilter() {
    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    return
     getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter authorextensiowner = new FileFilter() {
    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwner.getText().trim())) {
                        return
     getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter authorowner = new FileFilter() {

    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtAuthors.getText().trim())) {
                    if (dd.equalsIgnoreCase(txtOwner.getText().trim())) {
                        return true;
```

```
                    } else {
                        return false;
                    }
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
FileFilter extensiowner = new FileFilter() {
    @Override
    public boolean accept(File file) {
        FileOwnerAttributeView  ownerAttributeView  =  Files.getFileAttributeView(file.toPath(),
     FileOwnerAttributeView.class);
        UserPrincipal owner = null;
        try {
            owner = ownerAttributeView.getOwner();
            String dd = owner.getName().substring(owner.getName().lastIndexOf("\\") + 1);
            if (file.isFile()) {
                if (dd.equalsIgnoreCase(txtOwner.getText().trim())) {
                    return
     getFileExtension(file).equalsIgnoreCase(txtTypeFile.getText().trim());
                } else {
                    return false;
                }
            } else {
                return false;
            }
        } catch (IOException ex) {
            return false;
        }
    }
};
private String getFileExtension(File file) {
    String name = file.getName();
    try {
        return name.substring(name.lastIndexOf(".") + 1);
    } catch (Exception e) {
        return "";
    }
}
public byte[] createChecksum(String filename, String type) throws Exception {
    InputStream fis = new FileInputStream(filename);
    byte[] buffer = new byte[1024];
    MessageDigest complete = MessageDigest.getInstance(type);
    int numRead;
    do {
        numRead = fis.read(buffer);
        if (numRead > 0) {
            complete.update(buffer, 0, numRead);
        }
    } while (numRead != -1);

    fis.close();
    return complete.digest();
}
public String getChecksum(String filename, String type) {
    String result = "";
    byte[] b;
    try {
        b = createChecksum(filename, type);
        for (int i = 0; i < b.length; i++) {
            result += Integer.toString((b[i] & 0xff) + 0x100, 16).substring(1);
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "GAGAL cek CheckSum");
    }
    return result;
}
private String getComputerName() {
    Map<String, String> env = System.getenv();
    if (env.containsKey("COMPUTERNAME")) {
```

```
            return env.get("COMPUTERNAME");
        } else if (env.containsKey("HOSTNAME")) {
            return env.get("HOSTNAME");
        } else {
            return "Unknown Computer";
        }
    }
    public void kosong() {
        txtAuthors.setText(null);
        txtComputer.setText(null);
        txtLokasi.setText(null);
        txtNamaFile2.setText(null);
        txtOwner.setText(null);
        txtPath.setText(null);
        txtSUM.setText(null);
        txtTypeFile.setText(null);
        txtAntaraAkhir.setText(null);
        txtAntaraAwal.setText(null);
        tModel = new DefaultTableModel(header, 0);
        tabelMeta.setModel(tModel);
        tModelHasil = new DefaultTableModel(headerHasil, 0);
        tabelHasil.setModel(tModelHasil);
    }
    private void disablekorelasi(boolean status) {
        txtKorelasi.setText("");
        txtKorelasi.setEnabled(status);
        rbSama.setEnabled(status);
        rbBesar.setEnabled(status);
        rbKecil.setEnabled(status);
        rbSamaBesar.setEnabled(status);
        rbSamaKecil.setEnabled(status);
        rbantara.setEnabled(status);
        rbantara2.setEnabled(status);
        txtAntaraAwal.setEnabled(status);
        txtAntaraAkhir.setEnabled(status);
        tglAnataraAkhir.setDate(new Date());
        tglAntaraAwal.setDate(new Date());
        rbSama.setSelected(status);
    }
}
```