A

PROJECT

ON

MP3 SPLITTER & JOINER

By :
Baldeep Singh Handa
Computer Science & Engineering
00913202712

Submitted to:
Gaurav Sandhu
Bhavneet Kaur

Department of Computer Science & Engineering
Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University
Dwarka, New Delhi
Year 2012-2016

ACKNOWLEDGEMENT

With the immense pleasure, I extend my heart full thankful to all those who helped me in the completion of this project.

I would like to express my gratitude towards our supervisor, **Ms. Gagan Bansal, Senior Technical Consultant, CMC** who has given me support and suggestions. Without her help I could not have presented this dissertation up to the present standard. I would also like to thank others who gave me support for the project or in other aspects of my study at **CMC Private Limited, Pitampura.**

Date: 16/10/2014

Baldeep Singh Handa

(009/CSE1/2012-2016)

handa_baldeepsingh@yahoo.com

ABSTRACT

This project is based on ‘MP3 Splitter & Joiner’. MP3 Splitter & Joiner is a very useful MP3 audio editor. It builds MP3 Splitter(MP3 Cutter and MP3 Joiner in one, you can split, cut, trim a large MP3 file into multiple smaller pieces or join, merge multiple MP3 files to a larger one. Split and merge MP3 files directly without MP3 re-encoding, it is fast and keeps exactly the same sound quality of original MP3 files. This application is practical, simple, easy-to-use and very user-friendly.

CONTENTS

INTRODUCTION

Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation). The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The language was initially called Oak after an oak tree that stood outside Gosling's office and was later renamed Java, from Java coffee, said to be consumed in large quantities by the language's creators. Gosling aimed to implement a virtual machine and a language that had a familiar C/C++ style of notation.

There were five primary goals in the creation of the Java language:

1. It should be "simple, object-oriented and familiar"
2. It should be "robust and secure"
3. It should be "architecture-neutral and portable"
4. It should execute with "high performance"
5. It should be "interpreted, threaded, and dynamic"

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executable would. Just-in-Time (JIT) compilers were introduced from an early stage that compiles bytecode to machine code during runtime.

GUI Applications in Java: GUI applications in Java are event driven. The model, which is now used for handling these events in Java, is known as the Delegation Event Model. There are two ways to create Graphic user interface in java, i.e. AWT and Swing. Out of these, AWT is the older version and is not commonly used. Reason being that it uses native operating system components and look and feel could vary across various platforms, which violates the platform independence feature of Java. So nowadays, Swing is mostly used. When swing was first added in Java, it was included as a part of Java Foundation Classes (J.F.C) with few other new packages. But by Java 1.3, Swing components were fully integrated as a separate package into Java Development Kit.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

Model View Controller Architecture in Java Swing:

MVC is a software architecture pattern which separates the representation of information from the user's interaction with it. In addition to dividing the application into three kinds of components, the Model–view–controller (MVC) design defines the interactions between them.

- A controller can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document).
- A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A *passive* implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.
- A view requests information from the model that it needs for generating an output representation to the user

MP3 Splitter and Joiner:

MP3 Splitter & Joiner is a very useful MP3 audio editor. It builds MP3 Splitter(MP3 Cutter and MP3 Joiner in one, you can split, cut, trim a large MP3 file into multiple smaller pieces or join, merge multiple MP3 files to a larger one. Split and merge MP3 files directly without MP3 re-encoding, it is fast and keeps exactly the same sound quality of original MP3 files. This application is practical, simple, easy-to-use and very user-friendly.

Objectives:

- To easily edit audio MP3 files.
- To split, cut, trim large MP3 file into multiple smaller pieces.
- To merge multiple MP3 files to a single file.
- Splitter and Merger included in the same application.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Section 1: Introduction

1.1 Purpose

The purpose of this project is to how Java can be used to build GUI based standalone applications in Java using JDK. The user selects his/her source and destination MP3 files from a filechooser dialog box which in background are accessed through file handling. Once the selection is made the application shows the user the length and name of the clip and allows the user to set the name for the new file. Further an error is displayed if no file is selected for splitting or not more than one file is selected for merging. Further it also allows to remove the selected file.

1.2 Scope

In the future, additional features may be added, like

- Many different formats can also be included instead of MP3 formats only.
- Progress bar can be included to show the progress of splitting or merging.
- This application can be developed for different platforms like android and iOS because its extensive importance.

1.3 Definitions, acronyms, and Abbreviations

JVM- Java Virtual Machine is a virtual machine that can execute Java byte code. It is the code execution component of the Java platform. Sun Microsystems has stated that there are over 5.5 billion JVM enabled devices.

JDK- Java Development Kit is an implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation in the form of binary product aimed at Java developers on Solaris, Linux, Mac OS X or Windows.

JRE- Java Runtime Environment is also known as Java runtime. It is a part of JDK, a set of programming tools for developing Java applications. The JRE provides the minimum requirement for executing a Java application; it consists of JVM, core classes and supporting files.

Frames and Panels:

A **Frame** in Java represents an area of predefined size which has minimize, maximize and close button. In addition, it acts as a top level container, and allows other components to be added on to it.

A **Panel** sits over a frame or any other top level container and holds the components added to it. add() method is used to add components to a panel. Moreover, it doesn't have 3 buttons specified in a panel.

1.4 Overview

This introduction is intended to give a brief overview of a desktop application that can be made using Java. The following section will give background information that is necessary to fully understand the functional and non-functional requirements of the system. All of the requirements of the system will be stated, and each requirement must be testable.

1.5 Platform used

Java applications are platforms independent so can execute on any hardware or operating system. The kit used for development is JDK 1.8 .

Section 2: Overall Description

2.1 Product Perspective

This Java Swing based desktop application can run on any computer having JRE 1.4 or above. MP3 format files can only be opened.

2.1.1 System Interfaces

The system is intended to interact with one single user at a time. The user is able to interact with the system using a graphical user interface. The system required to execute this application is any OS with JRE 1.4 installed. The screen resolution required to execute is 400 x 400 or above.

2.1.2 User Interfaces

The user interface must provide the user an understandable and effective way for using the application. Java Swing will be used to create the graphical user interface for the system.

- Tabbed panes for splitting and merging are separately provided, user can choose any of the two as per his requirement.
- User adds a file to split into maximum of 5 parts or adds more than one file to be merged.
- Selected file can also be removed by remove button.
- New file name and destination can be set, user can also use the target option to directly jump to a particular directory or he can browse instead.
- ToolTip is displayed whenever user points to any of the buttons for ease of access.

2.1.3 Hardware Requirements

- Operating System: Any OS with JRE 1.4 or above installed.
- Processor : Pentium 3, 1 GHz or higher
- RAM : 256 MB or higher
- Hard Drive: 10 GB or higher.

Section 3: Specific Requirements for Proposed System

3.1 Overview

These requirements will allow the user to gain knowledge in how he should design the overall system so that it functions as per the requirements stated below

3.2 Functional Requirements

3.2.1 Tabbed Pane

Two panels- Splitter panel and Joiner panel are tabbed into a single tabbed pane allowing the user to select any as per the requirement.

3.2.3 Splitter Panel

Here user will be required to add source file, specify the number of parts to split the file into and provide new file name and destination. Further a file can split into a maximum of 5 parts and by default it is 2.

3.2.5 Joiner Panel

Here the user will be required to add more than one file to be merged and provide new file name and destination. Further not more than 5 files can be merged so an alert box shows the respective error. Also any of the selected file can be removed.

3.3 Nonfunctional Requirements

3.3.1 Security Requirements

User must have the admin access to add and modify(split or join) files from the system otherwise an error is prompted and no file is opened.

3.3.2 Software Quality Attributes

The Quality of the System is maintained in such a way so that it can be very user friendly to all the users.

The software quality attributes are assumed as under:

- Accurate and hence reliable.
- User Friendly GUI.
- Easy to understand.
- Fast speed operation.

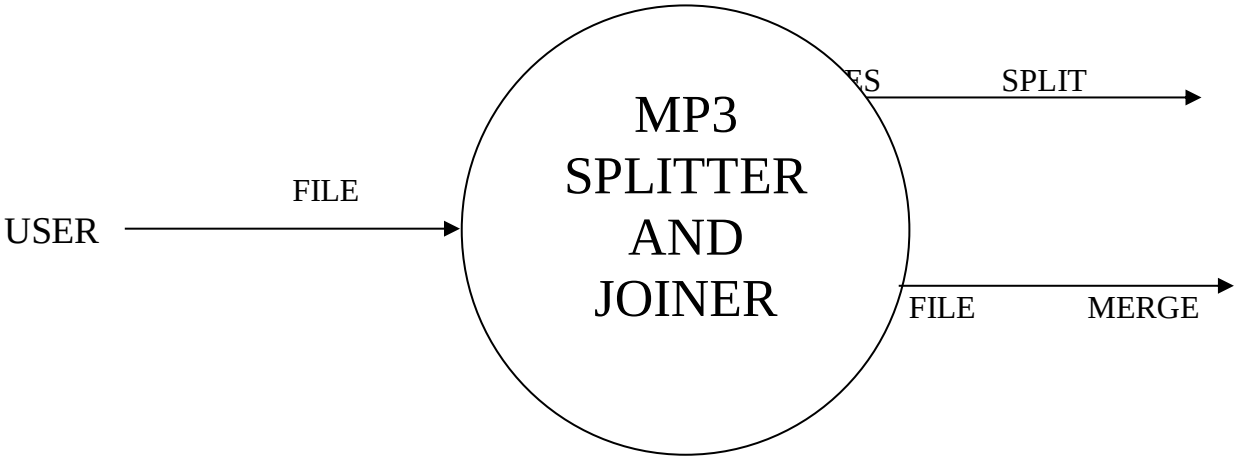
3.3.3 Other Requirements:

Requires Computers with the specifications given above with same number if a number of users want to use this application at the same time.

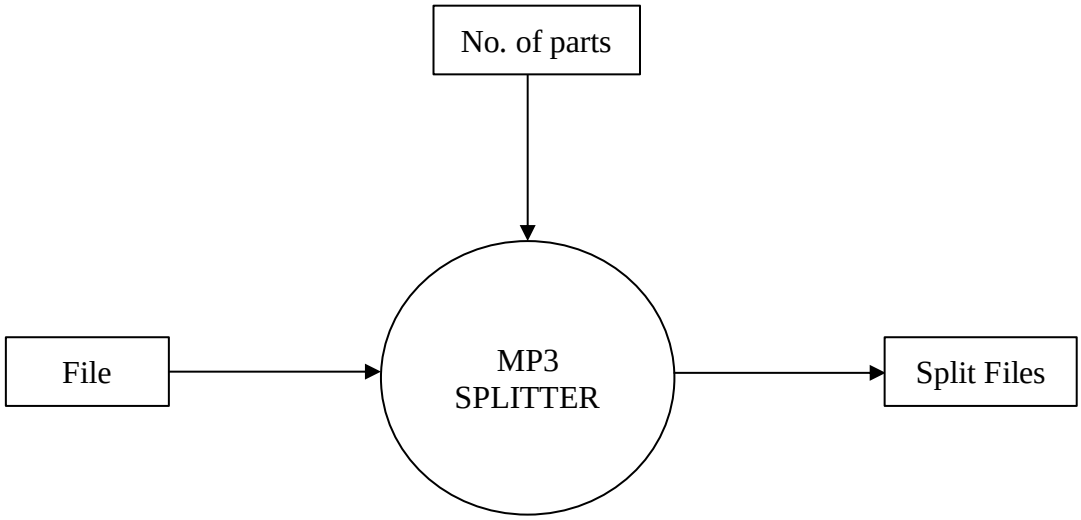
SYSTEM DESIGN

1. DFD (DATA FLOW DIAGRAM):

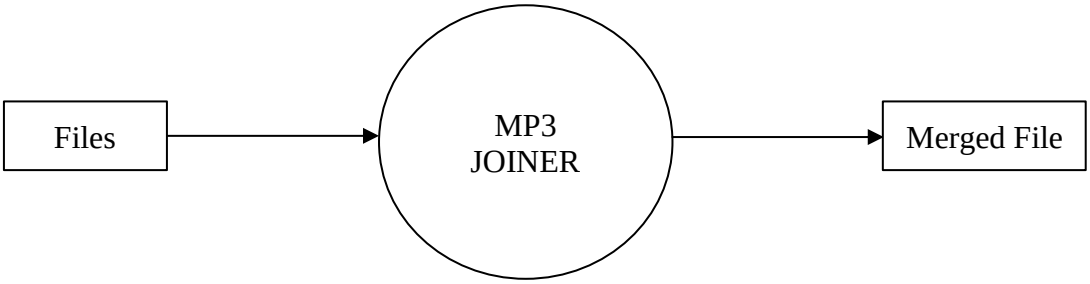
0 Level DFD:



1 Level DFD for Splitter:



1 Level DFD for Joiner:



SOURCE CODE

```
import java.awt.Color;
import java.awt.Container;
import java.awt.Image;
import java.awt.Toolkit;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileFilter;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author BaldeepSingh
 */
public class SplitterJoiner extends javax.swing.JFrame {

    /**
     * Creates new form SplitterJoiner
     */
    JFileChooser fc = null;

    File f = null;
    File[] fo=null;
    File f1 = null;
    File f2 = null;
    File f3 = null;
    File f4 = null;
    File f5 = null;
    File f6 = null;
    String oldf="";
    String[] newf;
    long start=0,end=0,interval=0;
    String fpath1 = "";
    String fpath2 = "";
    String fpath3 = "";
    String fpath4 = "";
    String fpath5 = "";
    String fpath6 = "";
    String newfilename="";
    String newfilepath="";
    String fname1 = "";
    String fname2 = "";
    String fname3 = "";
    String fname4 = "";
```

```
String fname5 = "";
String fname6 = "";
String fpath = "";
int a = 0, i = 0, j = 0, r = 0,nop=0;
double d = 0, c = 0;
FileInputStream fis = null;
FileOutputStream fos = null;
FileOutputStream[] foso=null;
byte[] b = null;
Image icon;

public SplitterJoiner() {

    initComponents();
    Container ctr = this.getContentPane();
    ctr.setBackground(Color.LIGHT_GRAY);
    icon = Toolkit.getDefaultToolkit().getImage("close.png");
    setIconImage(icon);
    setTitle("MP3 Splitter & Joiner");
    BAdd.setToolTipText("Click to add a file");
    BRemove.setToolTipText("Click to remove selected file");
    BSplitJoin.setToolTipText("Click to Split\\Merge");
    BBrowse.setToolTipText("Set output directory");
    BFindTarget.setToolTipText("Find directory");
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
private void initComponents() {

    TB = new javax.swing.JTabbedPane();
    PSplitter = new javax.swing.JPanel();
    LClipLengthS = new javax.swing.JLabel();
    LLengthS = new javax.swing.JLabel();
    LPartsS = new javax.swing.JLabel();
    CBPartsS = new javax.swing.JComboBox();
    LClipMBS = new javax.swing.JLabel();
    LFileNameS = new javax.swing.JLabel();
    TFFileNameS = new javax.swing.JTextField();
    LFileSizeS = new javax.swing.JLabel();
    LSizeS = new javax.swing.JLabel();
    LMBS = new javax.swing.JLabel();
    LClipNameS = new javax.swing.JLabel();
    LNameS = new javax.swing.JLabel();
    PJoiner = new javax.swing.JPanel();
    SPJ = new javax.swing.JScrollPane();
    TableJ = new javax.swing.JTable();
```

```
        TFNewFileJ = new javax.swing.JTextField();
        LNewFileSizeJ = new javax.swing.JLabel();
        LMBJ = new javax.swing.JLabel();
        LNewFileJ = new javax.swing.JLabel();
        LNewSizeJ = new javax.swing.JLabel();
        BAdd = new javax.swing.JButton();
        BRemove = new javax.swing.JButton();
        LOutput = new javax.swing.JLabel();
        TFOutput = new javax.swing.JTextField();
        BBrowse = new javax.swing.JButton();
        BFindTarget = new javax.swing.JButton();
        BSplitJoin = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        LClipLengthS.setText("Clip length:");

        LLengthS.setText("0.00");

        LPartsS.setText("No. of parts:");

        CBPartsS.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "2", "3", "4", "5" }));
        CBPartsS.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                CBPartsSActionPerformed(evt);
            }
        });

        LClipMBS.setText("mb");

        LFileNameS.setText("New file's name:");

        TFFileNameS.setText("new");

        LFileSizeS.setText("New file's size:");

        LSizeS.setText("0.00");

        LMBS.setText("mb");

        LClipNameS.setText("Clip name:");

        javax.swing.GroupLayout PSplitterLayout = new javax.swing.GroupLayout(PSplitter);
        PSplitter.setLayout(PSplitterLayout);
        PSplitterLayout.setHorizontalGroup(
            PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(PSplitterLayout.createSequentialGroup()
                .addGap(36, 36, 36)
                .addGroup(PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(LClipNameS, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                .addComponent(LFileSizeS, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(LFileNameS, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(LPartsS, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(LClipLengthS, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(TFFileNameS, javax.swing.GroupLayout.DEFAULT_SIZE, 151,
Short.MAX_VALUE)
                .addGroup(PSplitterLayout.createSequentialGroup()
                    .addComponent(LSizeS)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(LMBS))
                .addComponent(CBPartsS, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(PSplitterLayout.createSequentialGroup()
                    .addComponent(LLengthS)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(LClipMBS))
                .addComponent(LNameS, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap(127, Short.MAX_VALUE))
        );
        PSplitterLayout.setVerticalGroup(
            PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(PSplitterLayout.createSequentialGroup()
                .addGap(26, 26, 26)
                .addGroup(PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(LClipNameS, javax.swing.GroupLayout.DEFAULT_SIZE, 26,
Short.MAX_VALUE)
                .addComponent(LNameS, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(LClipLengthS, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(LLengthS, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(LClipMBS, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(LPartsS, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(CBPartsS, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
            .addGroup(PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(LFileNameS, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(TFFileNameS, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(PSplitterLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(LFileSizeS, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(LSizeS, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(LMBS, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(30, Short.MAX_VALUE))
        );

        TB.addTab("Splitter", PSplitter);

        TableJ.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null},
                {null, null},
                {null, null},
                {null, null},
                {null, null}
            },
            new String [] {
                "Source", "Name"
            }
        ));
        SPJ.setViewportView(TableJ);

        TFNewFileJ.setText("new");

        LNewFileSizeJ.setText("0.00");

        LMBJ.setText("mb");

        LNewFileJ.setText("New file name:");

        LNewSizeJ.setText("New file size:");

        javax.swing.GroupLayout PJoinerLayout = new javax.swing.GroupLayout(PJoiner);
        PJoiner.setLayout(PJoinerLayout);
        PJoinerLayout.setHorizontalGroup(
            PJoinerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(PJoinerLayout.createSequentialGroup()
                .addComponent(SPJ, javax.swing.GroupLayout.PREFERRED_SIZE, 399,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
            .addGroup(PJoinerLayout.createSequentialGroup()
                .addContainerGap()
```

```
            .addGroup(PJoinerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(LNewSizeJ, javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(LNewFileJ, javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(PJoinerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(TFNewFileJ, javax.swing.GroupLayout.PREFERRED_SIZE, 184,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(PJoinerLayout.createSequentialGroup()
                    .addComponent(LNewFileSizeJ, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(LMBJ, javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        PJoinerLayout.setVerticalGroup(
            PJoinerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(PJoinerLayout.createSequentialGroup()
                .addComponent(SPJ, javax.swing.GroupLayout.PREFERRED_SIZE, 107,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(31, 31, 31)
                .addGroup(PJoinerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(TFNewFileJ, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(LNewFileJ, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(PJoinerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(LNewFileSizeJ, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(LMBJ, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(LNewSizeJ, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(0, 25, Short.MAX_VALUE))
        );

        TB.addTab("Joiner", PJoiner);

        BAdd.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        BAdd.setForeground(new java.awt.Color(255, 255, 255));
        BAdd.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/add.png"))); // NOI18N
        BAdd.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        BAdd.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                BAddMouseEntered(evt);
            }
        });
        BAdd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            BAddActionPerformed(evt);
        }
    });

    BRemove.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/remove.png"))); //
NOI18N
    BRemove.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            BRemoveActionPerformed(evt);
        }
    });

    LOutput.setFont(new java.awt.Font("Tahoma", 0, 13)); // NOI18N
    LOutput.setText("Output");

    TFOutput.setText("C:\\Users");

    BBrowse.setText("Browse");
    BBrowse.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            BBrowseActionPerformed(evt);
        }
    });

    BFindTarget.setText("Find Target");
    BFindTarget.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            BFindTargetActionPerformed(evt);
        }
    });

    BSplitJoin.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/5.jpg"))); // NOI18N
    BSplitJoin.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            BSplitJoinActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(TB, javax.swing.GroupLayout.PREFERRED_SIZE, 403,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(LOutput, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(BBrowse)
                        .addGap(18, 18, 18)
                        .addComponent(BFindTarget)
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addComponent(TFOutput))))
            .addGap(34, 34, 34)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(BSplitJoin, javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(BAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(BRemove, javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(63, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(TB, javax.swing.GroupLayout.PREFERRED_SIZE, 256,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(23, 23, 23)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(TFOutput, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(LOutput, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(BBrowse)
                            .addComponent(BFindTarget))
                        .addContainerGap(77, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(BAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(36, 36, 36)
                        .addComponent(BRemove, javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(85, 85, 85)
                        .addComponent(BSplitJoin, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(65, 65, 65))))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void BAddActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BAddActionPerformed
```

```
// TODO add your handling code here:

try {
    fc = new JFileChooser("C:\\Users");
    fc.setFileFilter(new FileFilter() {

        @Override
        public boolean accept(File f) {
            return f.getPath().endsWith(".mp3");
        }

        @Override
        public String getDescription() {
            return "Music(mp3) files only";
        }
    });
    a = fc.showOpenDialog(this);
    if (a == JFileChooser.APPROVE_OPTION && PSplitter.isVisible() == true) {
        f1 = fc.getSelectedFile();
        fpath1 = f1.getPath();
        fname1 = f1.getName();
        d = (f1.length()) / (1024 * 1024);
        LNameS.setText(fname1);
        LLengthS.setText(Double.toString(d));
        c = d / (double) (CBPartsS.getSelectedIndex() + 2);
        LSizeS.setText(Double.toString(c));

    } else if (a == JFileChooser.APPROVE_OPTION && PJoiner.isVisible() == true) {
        if (i == 0 || TableJ.getValueAt(0, 0) == "") {
            f1 = fc.getSelectedFile();
            fpath1 = f1.getPath();
            fname1 = f1.getName();
            TableJ.setValueAt(fpath1, 0, 0);
            TableJ.setValueAt(fname1, 0, 1);
            i++;
            d = d + (f1.length()) / (1024 * 1024);
            LNewFileSizeJ.setText(Double.toString(d));
        } else if (i == 1 || TableJ.getValueAt(1, 0) == "") {
            f2 = fc.getSelectedFile();
            fpath2 = f2.getPath();
            fname2 = f2.getName();
            TableJ.setValueAt(fpath2, 1, 0);
            TableJ.setValueAt(fname2, 1, 1);
            i++;
            d = d + (f2.length()) / (1024 * 1024);
            LNewFileSizeJ.setText(Double.toString(d));
        } else if (i == 2 || TableJ.getValueAt(2, 0) == "") {
            f3 = fc.getSelectedFile();
            fpath3 = f3.getPath();
            fname3 = f3.getName();
            TableJ.setValueAt(fpath3, 2, 0);
            TableJ.setValueAt(fname3, 2, 1);
```

```
                i++;
                d = d + (f3.length()) / (1024 * 1024);
                LNewFileSizeJ.setText(Double.toString(d));
            } else if (i == 3 || TableJ.getValueAt(3, 0) == "") {
                f4 = fc.getSelectedFile();
                fpath4 = f4.getPath();
                fname4 = f4.getName();
                TableJ.setValueAt(fpath4, 3, 0);
                TableJ.setValueAt(fname4, 3, 1);
                i++;
                d = d + (f4.length()) / (1024 * 1024);
                LNewFileSizeJ.setText(Double.toString(d));
            } else if (i == 4 || TableJ.getValueAt(4, 0) == "") {
                f5 = fc.getSelectedFile();
                fpath5 = f5.getPath();
                fname5 = f5.getName();
                TableJ.setValueAt(fpath5, 4, 0);
                TableJ.setValueAt(fname5, 4, 1);
                i++;
                d = d + (f5.length()) / (1024 * 1024);
                LNewFileSizeJ.setText(Double.toString(d));
            } else {
                JOptionPane.showMessageDialog(this, "Not more then 5 files can be joined");
            }

        }
    } catch (Exception e) {
        System.out.println("Error");
    }


}//GEN-LAST:event_BAddActionPerformed

private void BBrowseActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BBrowseActionPerformed
    // TODO add your handling code here:
    fc = new JFileChooser("C:\\Users");
    fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    a = fc.showOpenDialog(this);
    if (a == JFileChooser.APPROVE_OPTION) {
        f = fc.getSelectedFile();
        fpath = f.getPath();
        TFOutput.setText(fpath);
    }
}//GEN-LAST:event_BBrowseActionPerformed

private void BFindTargetActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BFindTargetActionPerformed
    // TODO add your handling code here:
    String target = TFOutput.getText();
    fc = new JFileChooser(target);
    fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
```

```
        a = fc.showOpenDialog(this);
        if (a == JFileChooser.APPROVE_OPTION) {
            f = fc.getSelectedFile();
            fpath = f.getPath();
            TFOutput.setText(fpath);
        }
    }//GEN-LAST:event_BFindTargetActionPerformed

    private void BSplitJoinActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BSplitJoinActionPerformed
        // TODO add your handling code here:

        try {
            if (PSplitter.isVisible() == true) {
                nop= Integer.parseInt(CBPartsS.getSelectedItem().toString());
                fis=new FileInputStream(f1);
                foso=new FileOutputStream[nop];
                fo=new File[nop];
                newf=new String[nop];
                newfilename=TFFileNameS.getText();
                newfilepath=TFOutput.getText();
                interval=f1.length()/nop;
                start=0;
                end=interval;
                b=new byte[1];
                for (int k = 0; k < nop; k++) {
                    newf[k]=newfilepath+"\\"+newfilename+k+".mp3";
                    fo[k]=new File(newf[k]);
                    foso[k]=new FileOutputStream(fo[k]);
                    for (long l =  start; l < end; l++) {
                        fis.read(b);
                        foso[k].write(b);
                    }
                    foso[k].flush();
                    foso[k].close();
                    start=end;
                    end=end+interval;
                }
                JOptionPane.showMessageDialog(this,"File Splitted into "+ nop +" parts");
            }
            if (PJoiner.isVisible() == true) {
                fname6 = TFNewFileJ.getText();
                fpath6 = TFOutput.getText();
                f6 = new File(fpath6+"\\"+fname6+".mp3");
                if (i == 2) {
                    fis = new FileInputStream(f1);
                    fos = new FileOutputStream(f6, true);
                    b = new byte[1000];
                    j = 0;
                    while ((j = fis.read(b)) != -1) {
                        fos.write(b);
                    }
```

```
            fis.close();
            fis = new FileInputStream(f2);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis.close();
            fos.flush();
            fos.close();
            JOptionPane.showMessageDialog(this, "2 files joined succesfully");
        } else if (i == 3) {
            fis = new FileInputStream(f1);
            fos = new FileOutputStream(f6, true);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis.close();
            fis = new FileInputStream(f2);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis = new FileInputStream(f3);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis.close();
            fos.flush();
            fos.close();
            JOptionPane.showMessageDialog(this, "3 files joined succesfully");
        } else if (i == 4) {
            fis = new FileInputStream(f1);
            fos = new FileOutputStream(f6, true);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis.close();
            fis = new FileInputStream(f2);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis = new FileInputStream(f3);
```

```
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis = new FileInputStream(f4);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis.close();
            fos.flush();
            fos.close();
            JOptionPane.showMessageDialog(this, "4 files joined succesfully");
        } else if (i == 5) {
            fis = new FileInputStream(f1);
            fos = new FileOutputStream(f6, true);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis.close();
            fis = new FileInputStream(f2);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis = new FileInputStream(f3);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis = new FileInputStream(f4);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis = new FileInputStream(f5);
            b = new byte[1000];
            j = 0;
            while ((j = fis.read(b)) != -1) {
                fos.write(b);
            }
            fis.close();
            fos.flush();
            fos.close();
            JOptionPane.showMessageDialog(this, "5 files joined succesfully");
```

```
            } else {
                JOptionPane.showMessageDialog(this, "2 or more files must be added", "Alert",
JOptionPane.ERROR_MESSAGE);
            }
          }
       } catch (Exception e) {
          System.out.println(e);
          System.out.println(e.getMessage());
       }
    }//GEN-LAST:event_BSplitJoinActionPerformed

    private void BRemoveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BRemoveActionPerformed
       // TODO add your handling code here:
       if (PSplitter.isVisible() == true) {
          LNameS.setText("");
          LLengthS.setText("0.00");
          TFFileNameS.setText("new");
          LSizeS.setText("0.00");
       } else if (PJoiner.isVisible() == true) {
          r = TableJ.getSelectedRow();
          TableJ.setValueAt("", r, 0);
          TableJ.setValueAt("", r, 1);
          i--;
          if (r == 0) {
             d = d - (f1.length()) / (1024 * 1024);
             LNewFileSizeJ.setText(Double.toString(d));
          } else if (r == 1) {
             d = d - (f2.length()) / (1024 * 1024);
             LNewFileSizeJ.setText(Double.toString(d));
          } else if (r == 2) {
             d = d - (f3.length()) / (1024 * 1024);
             LNewFileSizeJ.setText(Double.toString(d));
          } else if (r == 3) {
             d = d - (f4.length()) / (1024 * 1024);
             LNewFileSizeJ.setText(Double.toString(d));
          } else if (r == 4) {
             d = d - (f5.length()) / (1024 * 1024);
             LNewFileSizeJ.setText(Double.toString(d));
          }
       }
    }//GEN-LAST:event_BRemoveActionPerformed

    private void BAddMouseEntered(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_BAddMouseEntered
       // TODO add your handling code here:

    }//GEN-LAST:event_BAddMouseEntered

    private void CBPartsSActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_CBPartsSActionPerformed
       // TODO add your handling code here:
       c = d / (double) (CBPartsS.getSelectedIndex() + 2);
```

```
        LSizeS.setText(Double.toString(c));
    }//GEN-LAST:event_CBPartsSActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(SplitterJoiner.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(SplitterJoiner.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(SplitterJoiner.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(SplitterJoiner.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new SplitterJoiner().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton BAdd;
    private javax.swing.JButton BBrowse;
    private javax.swing.JButton BFindTarget;
    private javax.swing.JButton BRemove;
    private javax.swing.JButton BSplitJoin;
    private javax.swing.JComboBox CBPartsS;
    private javax.swing.JLabel LClipLengthS;
    private javax.swing.JLabel LClipMBS;
```

```
    private javax.swing.JLabel LClipNameS;
    private javax.swing.JLabel LFileNameS;
    private javax.swing.JLabel LFileSizeS;
    private javax.swing.JLabel LLengthS;
    private javax.swing.JLabel LMBJ;
    private javax.swing.JLabel LMBS;
    private javax.swing.JLabel LNameS;
    private javax.swing.JLabel LNewFileJ;
    private javax.swing.JLabel LNewFileSizeJ;
    private javax.swing.JLabel LNewSizeJ;
    private javax.swing.JLabel LOutput;
    private javax.swing.JLabel LPartsS;
    private javax.swing.JLabel LSizeS;
    private javax.swing.JPanel PJoiner;
    private javax.swing.JPanel PSplitter;
    private javax.swing.JScrollPane SPJ;
    private javax.swing.JTabbedPane TB;
    private javax.swing.JTextField TFFileNameS;
    private javax.swing.JTextField TFNewFileJ;
    private javax.swing.JTextField TFOutput;
    private javax.swing.JTable TableJ;
    // End of variables declaration//GEN-END:variables
}
```

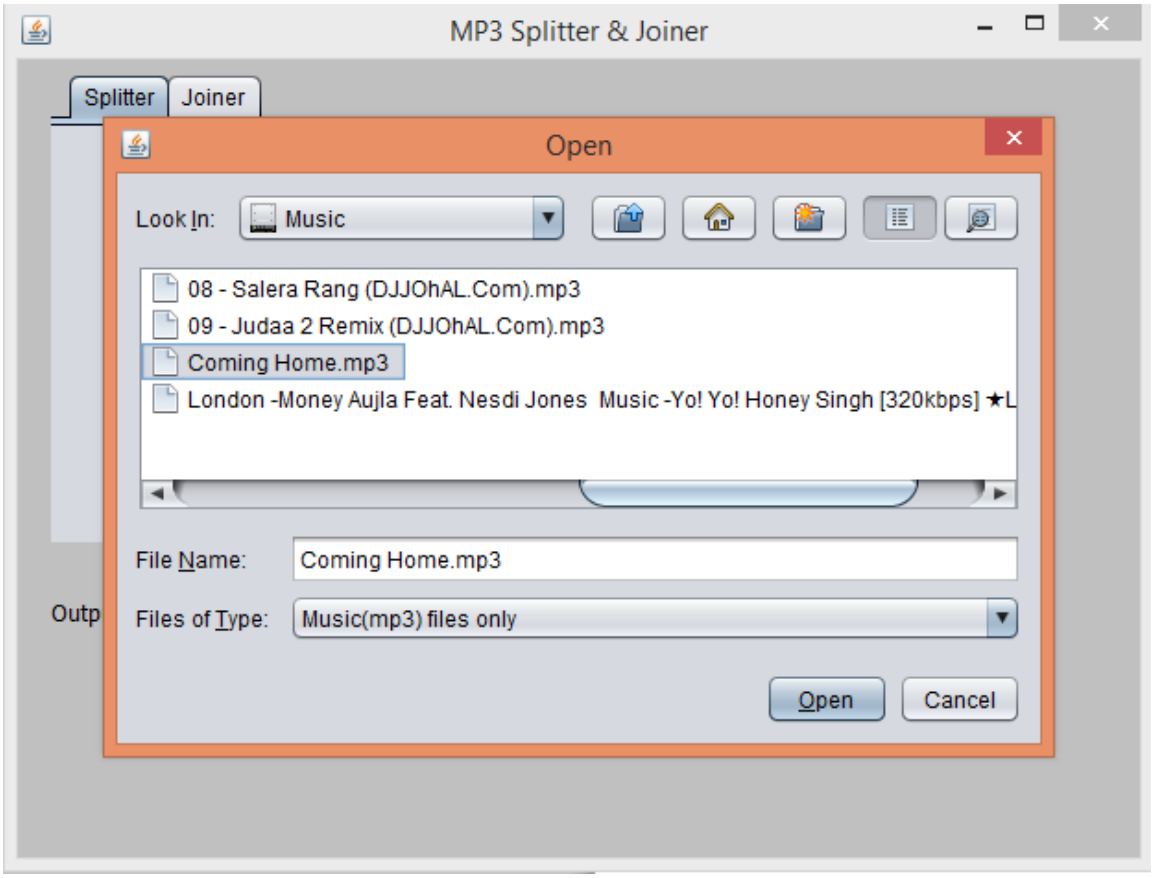
Results and Conclusions

After Completion of the project and successful implementation of the software, we were able to split, trim, cut a large mp3 file into a number of smaller files or merge more than one mp3 file into a single mp3 audio. The sound quality is unchanged.
The User was also able to do the required task very fast, with ease without any problems.
Also it can be concluded that the application is eco-friendly. We were also able to implement the file handling correctly.

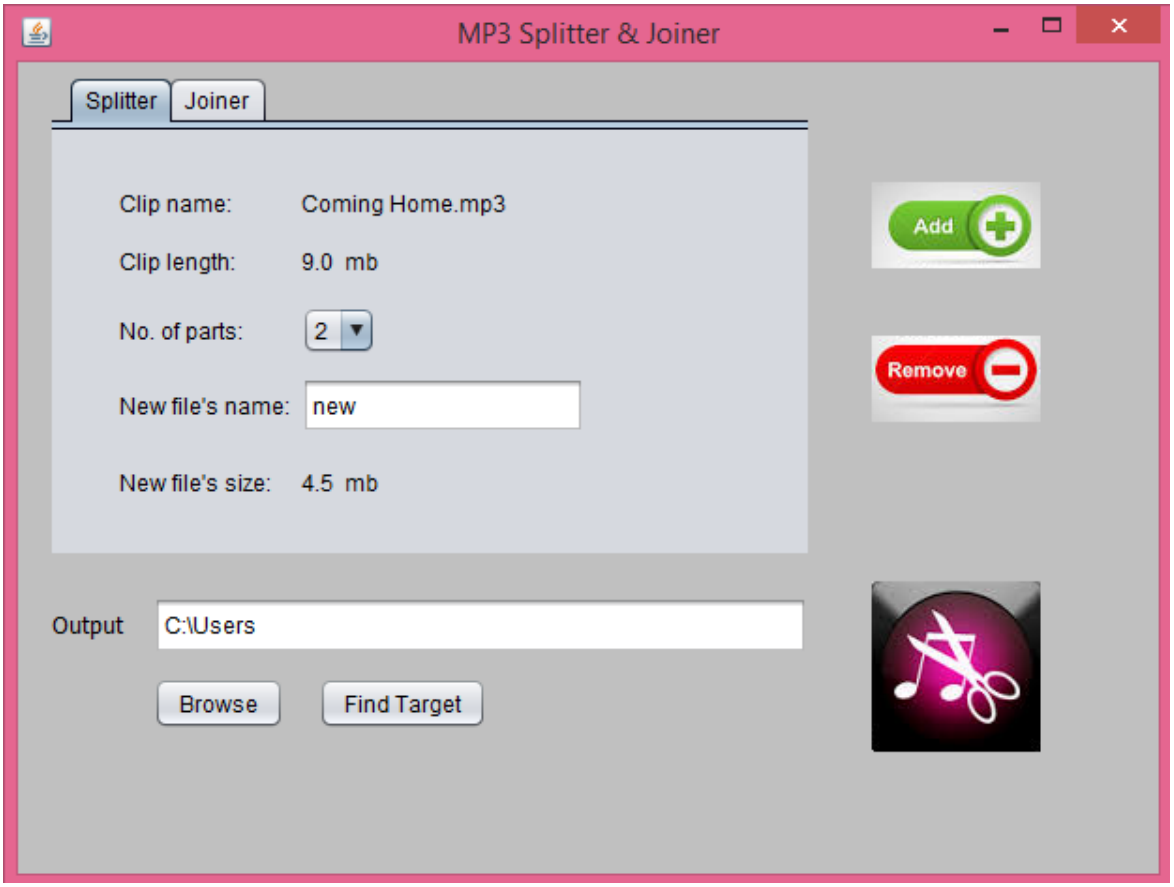
Below are the screenshots of the different phases of the project:

Splitter Panel with no file added:

File Chooser Dialog Box:

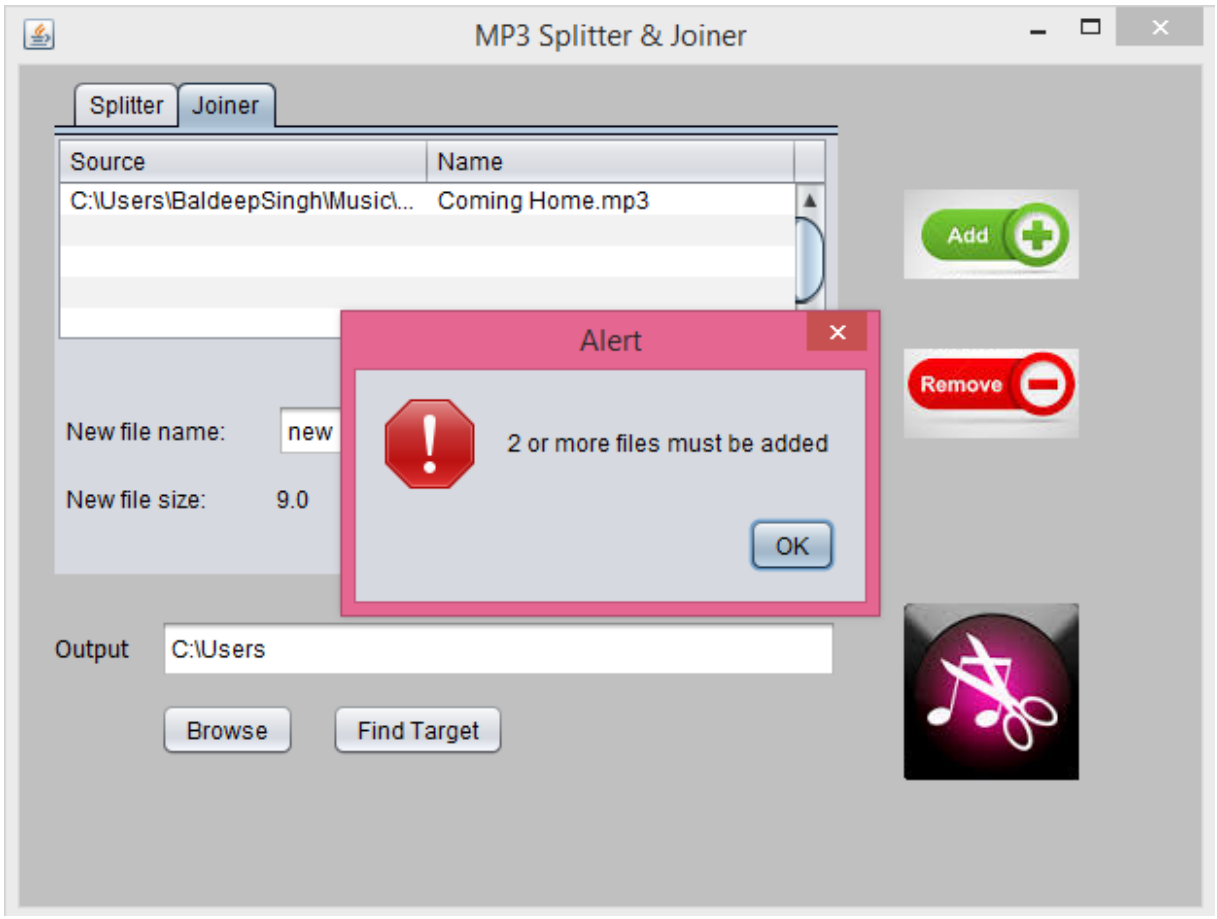


Splitter Panel after adding a file:

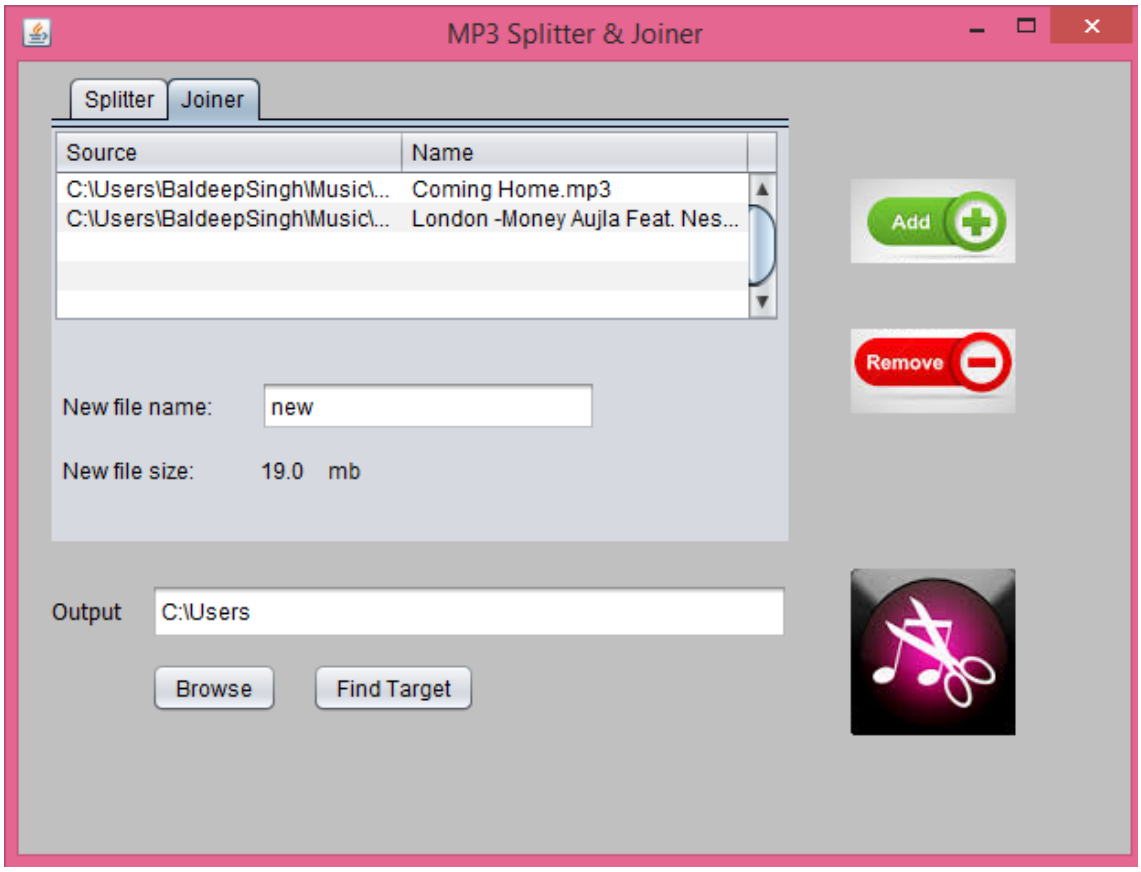


Joiner Panel before selecting a file:

Error displayed in an alert box:



Joiner Panel after selecting more than one file:



References

Books:

- Java the Complete Reference by Herbert Schildt
- Java Swing by Manning

Websites:

- www.java2s.com
- www.stackoverflow.com
- www.google.co.in
- www.coderanch.com