TRABALHO 1
TOCADOR MIDI

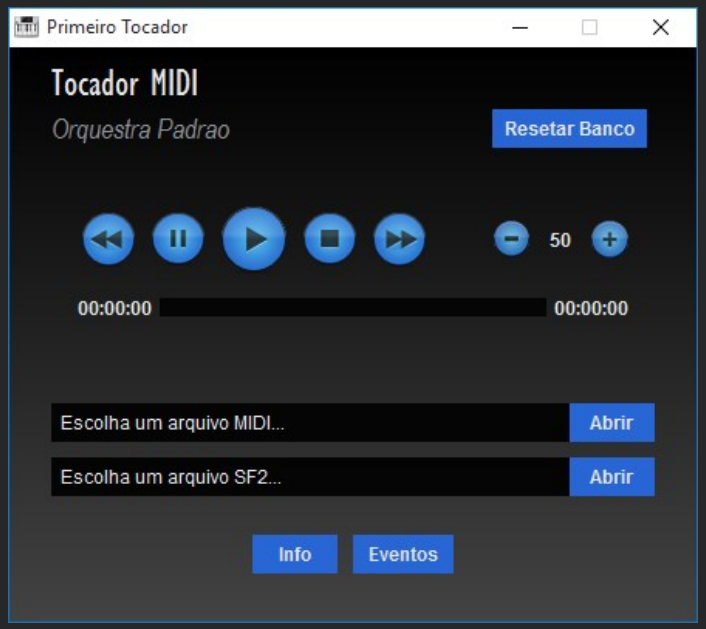
Allisson Barros 12/0055619
Cássio Pantoja 11/0147413
Francisco Anderson 11/0117964

Introdução à Computação Sônica 2017-1

O TOCADOR

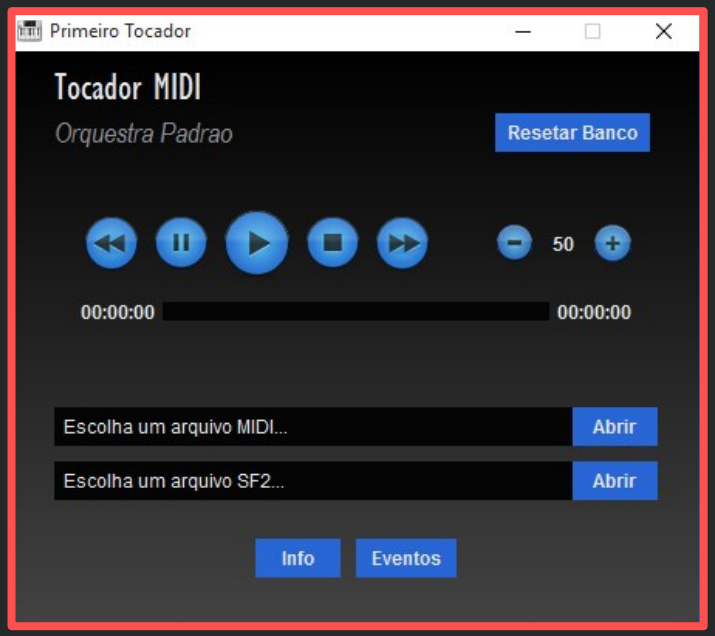

Primeiro Tocador
Tocador MIDI
Orquestra Padrao
Resetar Banco
50
00:00:00
00:00:00
Escolha um arquivo MIDI...
Abrir
Escolha um arquivo SF2...
Abrir
Info
Eventos

INTERFACE




```
JFrame frame = new JFrame("Primeiro Tocador");

PanelPersonalizado panelConteudo = new PanelPersonalizado();
GroupLayout layout = new GroupLayout(panelConteudo);

JPanel panelTopo      = new JPanel();
JPanel panelBotoes    = new JPanel();
JPanel panelProgresso = new JPanel();
JPanel panelArquivos  = new JPanel();
JPanel panelRodape    = new JPanel();
```

FRAME




```
//------------------
// Configuração do Frame
//------------------
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setPreferredSize(new Dimension (450, 400));
frame.setResizable(false);
frame.setContentPane(panelConteudo);
frame.setIconImage(frameIcon.getImage());
frame.paintComponents(this.getGraphics());
frame.pack();
frame.setVisible(true);
```

LAYOUT

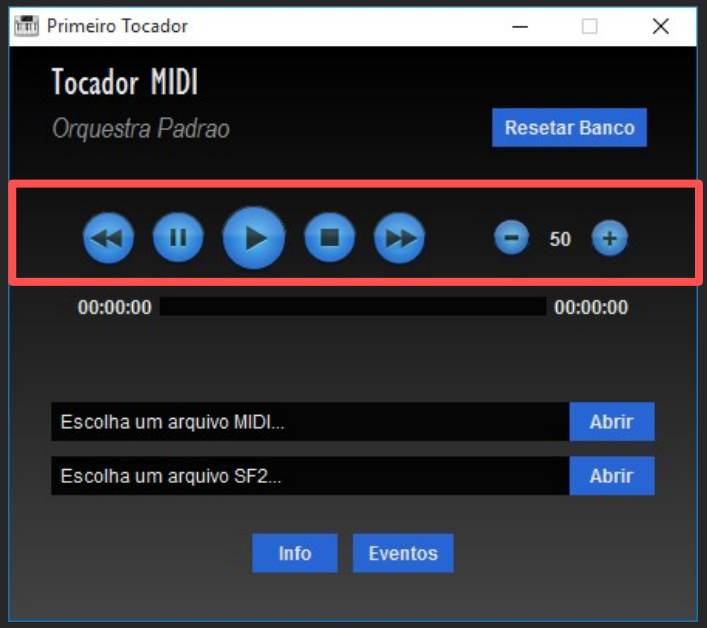

```java
public class BotaoSimples extends JButton{

    public BotaoSimples(String label) {
        setText(label);
        setContentAreaFilled(false);
        setBorder(BorderFactory.createEmptyBorder());
        setOpaque(true);
        setBackground(corAzul);
        setForeground(corTexto);
        setFocusable(false);
        setCursor(new Cursor(Cursor.HAND_CURSOR));
    }
}
```

```java
//---------
// Layout
//---------
layout.setHorizontalGroup(layout.createSequentialGroup()
        .addComponent(panelTopo)
        .addComponent(panelCentro)
        .addComponent(panelProgresso)
        .addComponent(panelArquivos)
        .addComponent(panelRodape)
);
```

PAINEL TOPO

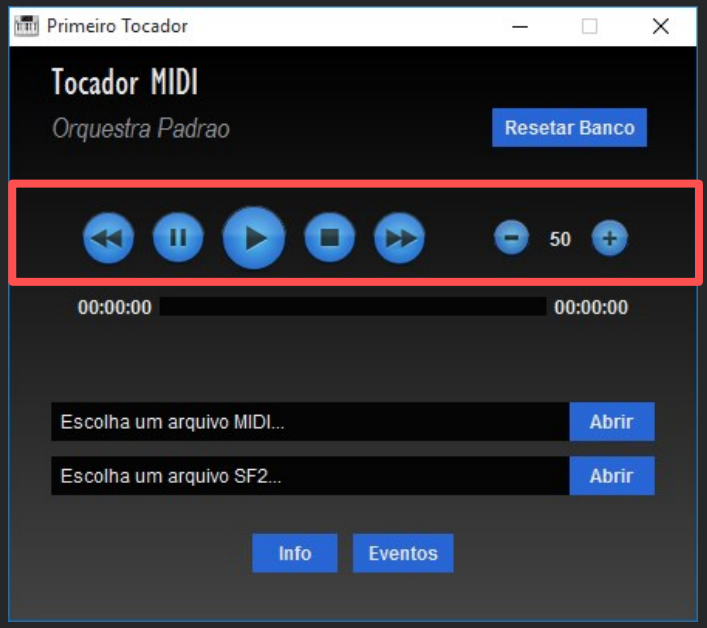

```
JLabel nomeArquivo    = new JLabel("Tocador MIDI");
JLabel nomeArquivo2   = new JLabel("Orquestra Padrao");


BotaoSimples botaoPadrao  = new BotaoSimples("Resetar Banco");
```

PAINEL CENTRO



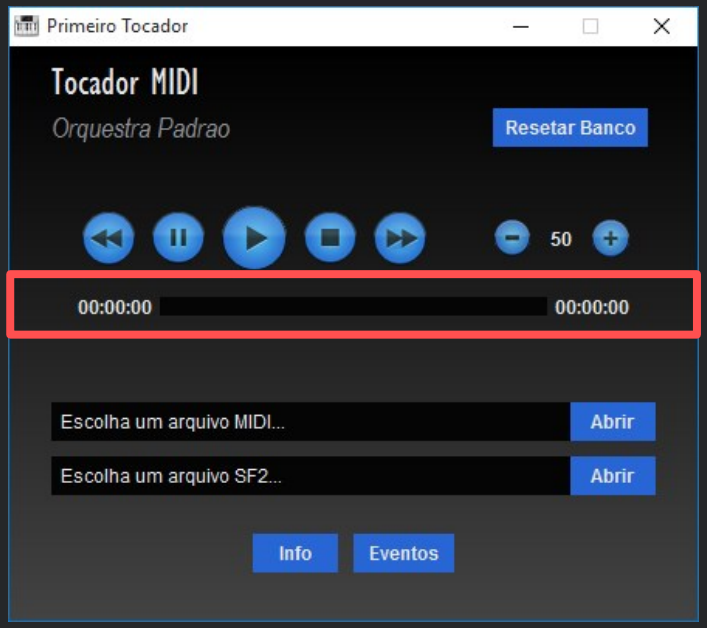

```java
public class BotaoIcone extends JButton{

    public BotaoIcone(ImageIcon botaoNormal){
        setContentAreaFilled(false);
        setBorderPainted(false);
        setPreferredSize(new Dimension (40, 40));
        setFocusable(false);
        setOpaque(false);
        setIcon(botaoNormal);
        setCursor(new Cursor(Cursor.HAND_CURSOR));
    }

    public void efeitoHover(ImageIcon botaoNormal, ImageIcon botaoHover){

        addMouseListener(new MouseAdapter(){
            public void mouseEntered(MouseEvent evt) {
                setIcon(botaoHover);
            }
            public void mouseExited(MouseEvent evt) {
                setIcon(botaoNormal);
            }
        });
    }
}
```

PAINEL CENTRO



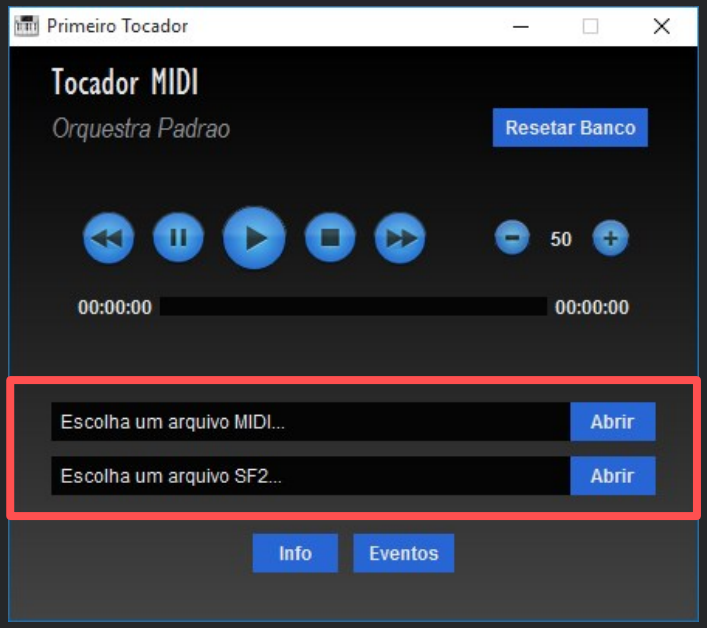

```
BotaoIcone botaoStop     = new BotaoIcone(stopIcon);
BotaoIcone botaoPlay     = new BotaoIcone(playIcon);
BotaoIcone botaoPause    = new BotaoIcone(pauseIcon);
BotaoIcone botaoFoward   = new BotaoIcone(fowardIcon);
BotaoIcone botaoBackward = new BotaoIcone(backwardIcon);
BotaoIcone botaoMenos    = new BotaoIcone(menosIcon);
BotaoIcone botaoMais     = new BotaoIcone(maisIcon);
```

PAINEL PROGRESSO




```
JLabel tempoTotal    = new JLabel("00:00:00");
JLabel tempoCorrente = new JLabel("00:00:00");

JProgressBar progresso = new JProgressBar(0);


progresso.setPreferredSize(new Dimension(250,12));
progresso.setBackground(corFundo);
progresso.setForeground(corAzul);
progresso.setBorderPainted(false);
progresso.setBorder(BorderFactory.createEmptyBorder());
progresso.setStringPainted(false);
```

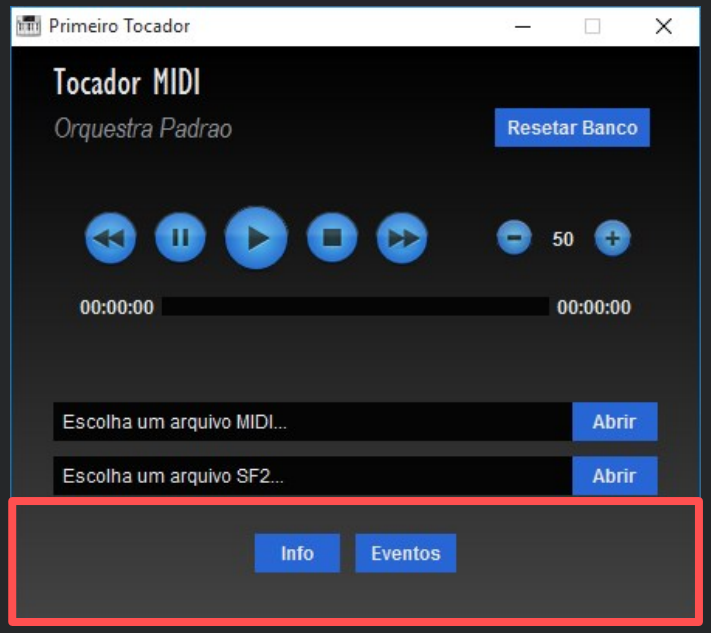
PAINEL ARQUIVOS




```
BotaoSimples botaoAbrir    = new BotaoSimples("Abrir");
BotaoSimples botaoAbrir2   = new BotaoSimples("Abrir");
```

```
JTextField caminhoTextField = new JTextField("  Escolha um arquivo MIDI...");
JTextField caminhoTextField2 = new JTextField("  Escolha um arquivo SF2...");

caminhoTextField.setEditable(false);
caminhoTextField.setBackground(corFundo);
caminhoTextField.setForeground(corTexto);
caminhoTextField.setBorder(BorderFactory.createEmptyBorder());
caminhoTextField.setPreferredSize(new Dimension(335, 25));
```

PAINEL RODAPÉ




```java
public class BotaoSimples extends JButton{

    public BotaoSimples(String label) {
        setText(label);
        setContentAreaFilled(false);
        setBorder(BorderFactory.createEmptyBorder());
        setOpaque(true);
        setBackground(corAzul);
        setForeground(corTexto);
        setFocusable(false);
        setCursor(new Cursor(Cursor.HAND_CURSOR));
    }
}
```

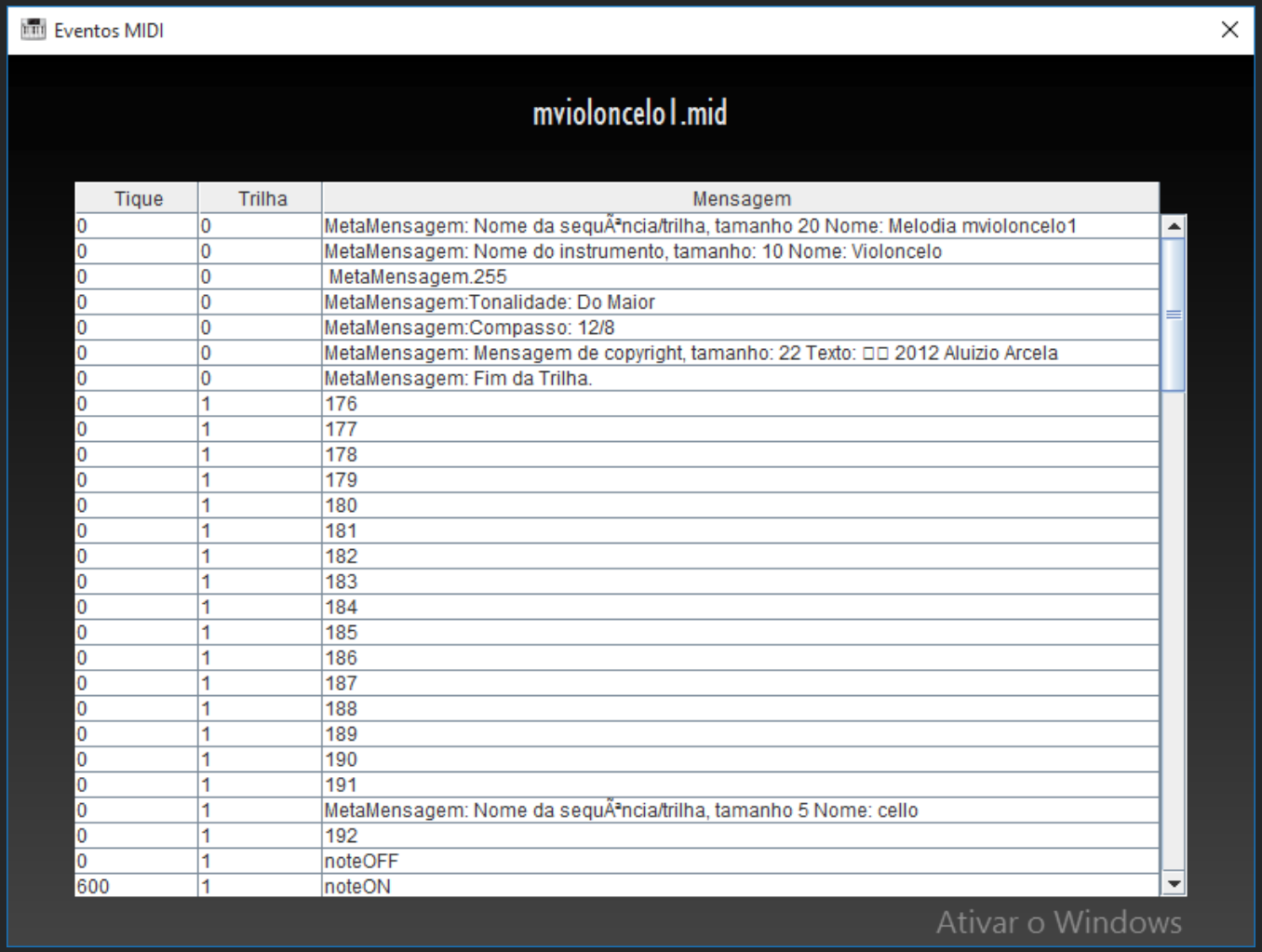
```java
BotaoSimples botaoInfo     = new BotaoSimples("Info");
BotaoSimples botaoEventos  = new BotaoSimples("Eventos");
```

```java
botaoEventos.setPreferredSize(new Dimension(65, 25));
botaoEventos.setToolTipText("Visualizar eventos MIDI");
```

DIALOG EVENTOS

```
public void mostraEventos(JFrame frame){

    JLabel nome = new JLabel(nomeArquivo.getText());
    nome.setFont(font1);
    nome.setForeground(corTexto);

    String[] colunas = {"Tique", "Trilha", "Mensagem"};

    Object[][] dados = obterMatrizDeEventosMidi();

    JTable tabela = new JTable(dados, colunas);
    TableColumnModel tcm = tabela.getColumnModel();

    ...


    JScrollPane panelTabela = new JScrollPane(tabela);
    tabela.setFillsViewportHeight(true);

    PanelPersonalizado contentPanel = new PanelPersonalizado();
    contentPanel.setOpaque(false);

    JPanel nomePanel = new JPanel();

    ...

    dialogEventos.setContentPane(contentPanel);

    //Show it.
    dialogEventos.setPreferredSize(new Dimension(800, 600));
    dialogEventos.setMinimumSize(new Dimension(800, 600));
    dialogEventos.setIconImage(frameIcon.getImage());
    dialogEventos.setLocation(450, 0);
    dialogEventos.setVisible(true);
}
```

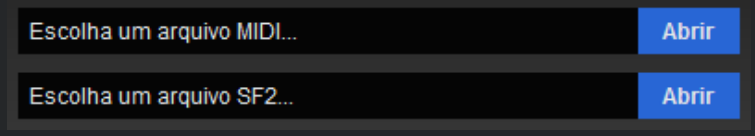
DIALOG EVENTOS



DIALOG INFO

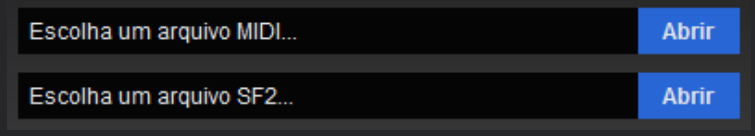

```
JLabel  nome = new JLabel(nomeArquivo.getText());
nome.setFont(font1);
nome.setForeground(corTexto);

JLabel informacoes = new JLabel(
    "<html><table border='0'>"
        + "<tr><td> Resolucao:           </td><td align='right'>" + resolucao       + "</td></tr><br>"
        + "<tr><td> Duracao:             </td><td align='right'>" + duracao         + "</td></tr><br>"
        + "<tr><td> Numero de Tiques:    </td><td align='right'>" + totalTitques    + "</td></tr><br>"
        + "<tr><td> Duracao do Tique:    </td><td align='right'>" + duracaoTique    + "</td></tr><br>"
        + "<tr><td> Duracao da Seminima: </td><td align='right'>" + duracaoSeminima + "</td></tr><br>"
        + "<tr><td> Total de Seminimas:  </td><td align='right'>" + totalSeminimas  + "</td></tr><br>"
        + "<tr><td> Andamento:           </td><td align='right'>" + Math.round(bpm) + "</td></tr><br>"
    + "</table></html>"
);
```

ARQUIVOS



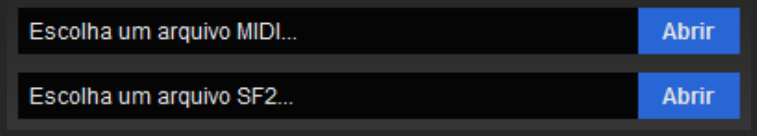

```
public void abrirMIDI(){
    JFileChooser arqMIDI = new JFileChooser(".");
    arqMIDI.setFileSelectionMode(JFileChooser.FILES_ONLY);
    arqMIDI.setFileFilter(new FileFilter(){...
    });

    arqMIDI.showOpenDialog(this);
    caminhoMIDI = arqMIDI.getSelectedFile().toString();
    caminhoTextField.setText(caminhoMIDI);

    File arqSeqNovo = arqMIDI.getSelectedFile();

    try {
        if(sequenciador!=null && sequenciador.isRunning()){
            sequenciador.stop();
            sequenciador.close();
            sequenciador = null;
        }

        sequenciaNova = MidiSystem.getSequence(arqSeqNovo);
        double duracao = sequenciaNova.getMicrosecondLength()/1000000.0d;

        ...

        File arquivoMIDI = new File(nomeArquivo.getText());
        sequencia    = MidiSystem.getSequence(arquivoMIDI);
        sequenciador = MidiSystem.getSequencer();

        sequenciador.setSequence(sequencia);
        sequenciador.open();

    }
    catch(InvalidMidiDataException e2) { System.out.println(e2+" : Erro nos dados midi.");}
    catch(IOException              e3) { System.out.println(e3+" : O arquivo midi nao foi encontrado.");}
    catch(Throwable e1) { System.out.println("Erro ao carregar arquivo Midi: "+ e1.toString()); }
}
```

ARQUIVOS



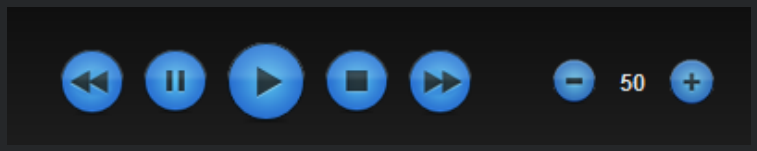

```
public void abrirSF2(){

    JFileChooser arqSF2 = new JFileChooser(".");
    arqSF2.setFileSelectionMode(JFileChooser.FILES_ONLY);
    arqSF2.setFileFilter(new FileFilter(){

        public boolean accept(File f){

            if (!f.isFile()) return true;
            String name = f.getName().toLowerCase();
            if (name.endsWith(".sf2"))  return true;
            return false;
        }

        public String getDescription(){
            return "Arquivo SF2 (*.sf2)";
        }
    });

    arqSF2.showOpenDialog(this);
    caminhoSF2 = arqSF2.getSelectedFile().toString();
    caminhoTextField2.setText(caminhoSF2);
    nomeArquivo2.setText(arqSF2.getSelectedFile().getName());

    try {

        carregarBanco(caminhoSF2);

    }catch (Throwable e1) {
        System.out.println("Erro em carregaArquivoSF2: "+ e1.toString());
    }
}
```

ARQUIVOS



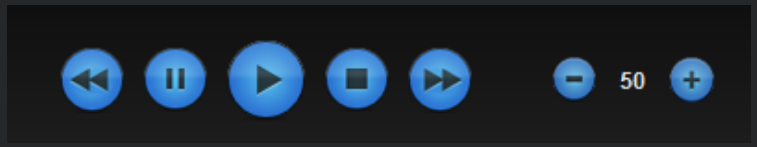

```
public void carregarBanco(String caminhoSF2) {

    try {
        sintetizador = MidiSystem.getSynthesizer();
        sintetizador.open();
    } catch (Exception e) {
        System.out.println("Erro em MidiSystem.getSynthesizer(): " + e);
        return;
    }

    bancoDefault = sintetizador.getDefaultSoundbank();
    if(bancoDefault != null){
        sintetizador.unloadAllInstruments(bancoDefault);
    }

    File arquivoSF2 = new File(caminhoSF2);

    try {
        bancoSelecionado = MidiSystem.getSoundbank(arquivoSF2);
    } catch (Exception e) { e.printStackTrace(); }

    sintetizador.loadAllInstruments(bancoSelecionado);

    try{
        sequenciador.getTransmitter().setReceiver(sintetizador.getReceiver());
        botaoPadrao.setEnabled(true);
        botaoPadrao.setBackground(corAzul);
    }catch (Exception e) { System.out.println("Erro no carregamento do banco: "+ e); }
}
```

TOCAR




```
public void tocar(String caminho, long inicio) {
    try {

        sequenciador.start();

        receptor = sequenciador.getTransmitters().iterator().next().getReceiver();
        sequenciador.getTransmitter().setReceiver(receptor);

        long duracao  = sequencia.getMicrosecondLength()/1000000;
        tempoCorrente.setText(formataInstante(0));

        sequenciador.setMicrosecondPosition(inicioAudio);

        if (sequenciador.isRunning()){
            duracao = sequenciador.getMicrosecondLength();
            soando = true;
        } else {
            soando = false;
            sequenciador.stop();
            sequenciador.close();
            inicio = 0L;
            duracao = 0;
        }

        ...

    }
    catch(MidiUnavailableException e1) { System.out.println(e1+" : Dispositivo midi nao disponivel.");}
    catch(Exception e)                 { System.out.println(e.toString());}
}
```

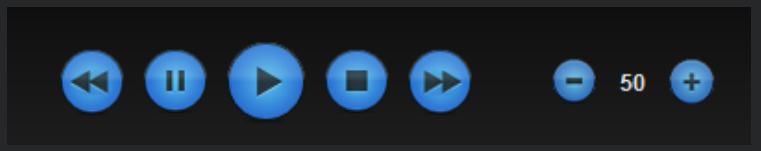
PAUSAR E PARAR




```
public void pausar(){

    soando = false;
    sequenciador.stop();

    botaoPlay.setEnabled(true);
    botaoPause.setEnabled(false);
}
```

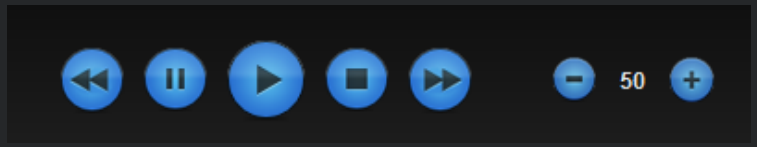
```
public void parar(){

    soando = false;
    sequenciador.stop();
    inicioAudio = 0L;

     ...

    volumeLabel.setText("50");
    volumeAtual = (50*100/127);
    progresso.setValue(0);
    tempoCorrente.setText(formataInstante(0));
}
```

VOLUME




```
public void diminuiVolume(int duploClique){

    if((volumeAtual > 9) && (duploClique == 1)){
        volumeAtual-=10;
    } else if (volumeAtual < 9){
        volumeAtual = 0;
    }
    for(int i=0; i<16; i++){
        try {
            mudancaVolume.setMessage(ShortMessage.CONTROL_CHANGE, i, 7, volumeAtual);
            receptor.send(mudancaVolume, -1);
        } catch (InvalidMidiDataException e1) {}
    }
    if(volumeAtual < 2){
        botaoMenos.setEnabled(false);
    }
    if(volumeAtual < 127){
        botaoMais.setEnabled(true);
    }

    if((volumeAtual > 0) && (duploClique == 0)){
        volumeAtual--;
        for(int i=0; i<16; i++){
            try {
                mudancaVolume.setMessage(ShortMessage.CONTROL_CHANGE, i, 7, volumeAtual);
                receptor.send(mudancaVolume, -1);
            } catch (InvalidMidiDataException e1) {}
        }
        if(volumeAtual < 2){
            botaoMenos.setEnabled(false);
        }
        if(volumeAtual < 127){
            botaoMais.setEnabled(true);
        }
    }
    volumeLabel.setText("" + ((volumeAtual*100)/127) + "");
}
```

AVANÇAR E RETROCEDER




```java
public void retroceder(){

    long novaPosicao = 0;

    novaPosicao = sequenciador.getMicrosecondPosition();
    if (novaPosicao >= 1000000) {
        novaPosicao -= 1000000;
        sequenciador.setMicrosecondPosition(novaPosicao);
        retardo(200);
    }
}
```

```java
public void avancar(){

    long novaPosicao = 0;
    long duracao     = 0;

    novaPosicao = sequenciador.getMicrosecondPosition();
    duracao  = sequenciador.getMicrosecondLength();

    if (novaPosicao <= duracao) {
        novaPosicao += 1000000;
        sequenciador.setMicrosecondPosition(novaPosicao);
        retardo(200);
    }
}
```

PREPARAÇÃO

```
public void preparaTocador(){

    try{
        botaoAbrir.setEnabled(true);

        botaoBackward.efeitoHover(backwardIcon, backwardHover);
        botaoPause.efeitoHover(pauseIcon, pauseHover);
        botaoPlay.efeitoHover(playIcon, playHover);
        botaoStop.efeitoHover(stopIcon, stopHover);
        botaoFoward.efeitoHover(fowardIcon, fowardHover);

        botaoBackward.setEnabled(false);
        botaoPause.setEnabled(false);
        botaoPlay.setEnabled(false);
        botaoStop.setEnabled(false);
        botaoFoward.setEnabled(false);
        botaoMenos.setEnabled(false);
        botaoMais.setEnabled(false);

        botaoPadrao.setEnabled(false);
        botaoPadrao.setBackground(corFundo2);
        botaoAbrir2.setEnabled(false);
        botaoAbrir2.setBackground(corFundo2);
        botaoInfo.setEnabled(false);
        botaoInfo.setBackground(corFundo2);
        botaoParam.setEnabled(false);
        botaoParam.setBackground(corFundo2);
        botaoEventos.setEnabled(false);
        botaoEventos.setBackground(corFundo2);
```

PREPARAÇÃO

```
botaoAbrir.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        if (dialogInfo.isVisible() {
            dialogInfo.setVisible(false);
            dialogInfo.dispose();
        }
        if (dialogParam.isVisible() {
            dialogParam.setVisible(false);
            dialogParam.dispose();
        }
        if (dialogEventos.isVisible() {
            dialogEventos.setVisible(false);
            dialogEventos.dispose();
        }
        abrirMIDI();
    }
});
```

```
botaoAbrir2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        abrirSF2();
    }
});
```

PREPARAÇÃO

```
botaoMais.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        aumentaVolume(0);
    }
});

botaoMais.addMouseListener(new MouseAdapter(){
    public void mousePressed(MouseEvent evt) {
        if(evt.getClickCount() >= 2){
            aumentaVolume(1);
        }
    }
});

botaoInfo.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        mostraInfo(frame);
    }
});


botaoPlay.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        tocar(nomeArquivo.getText(),inicioAudio);
    }
});
```

EXECUÇÃO

```
public void run(){

    double duracao;
    double tempo;
    int    posicao = 0;

    while(true) {
        if (soando) {
            duracao = sequenciador.getMicrosecondLength()/1000000;
            tempo   = sequenciador.getMicrosecondPosition()/1000000;
            posicao = (int) ((tempo*100)/duracao);
            try {
                progresso.setValue(posicao);
                tempoCorrente.setText(formataInstante(tempo));
                retardo(1000);
                if(tempo >= duracao){
                    progresso.setValue(0);
                    tempoCorrente.setText(formataInstante(0));

                    botaoAbrir.setEnabled(true);
                    botaoAbrir.setBackground(corAzul);
                    botaoAbrir2.setEnabled(true);
                    botaoAbrir2.setBackground(corAzul);
                    botaoPlay.setEnabled(true);
                    botaoPause.setEnabled(false);
                    botaoStop.setEnabled(false);
                    botaoFoward.setEnabled(false);
                }
            } catch(Exception e) { System.out.println(e.getMessage());}
        } else {
            try {
                retardo(1000);
            } catch(Exception e) { System.out.println(e.getMessage());}
        }
    }
}
```

OBRIGADO