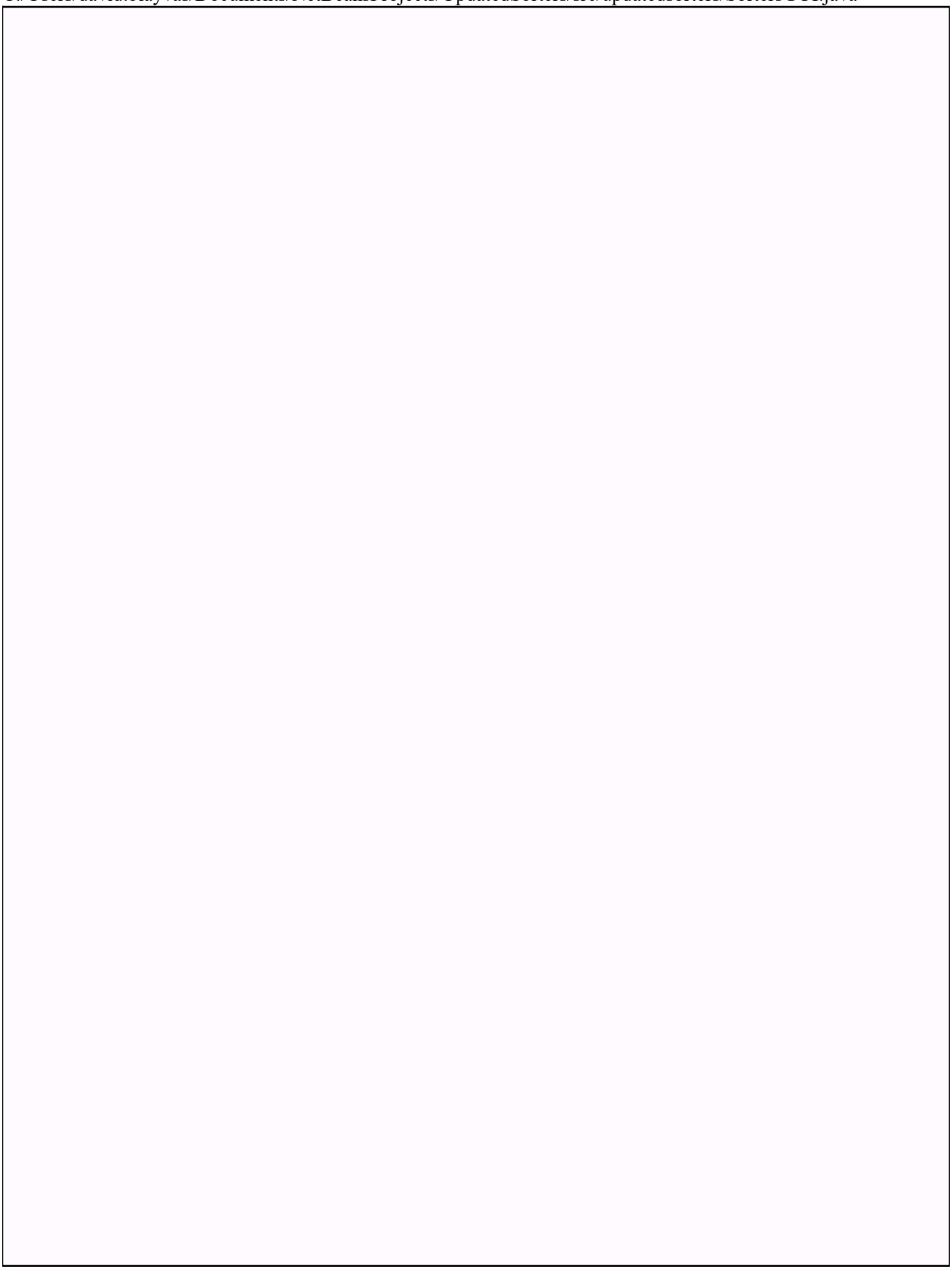
```
/*
 * David Blayvas
 * 8/16/2018
 * List Sorting Algorithms Evaluator
 * Mr. Odell
 */
package updatedsorters;

import java.util.Random;

/**
 *
 * @author david.blayvas
 */
public class SortersGUI extends javax.swing.JFrame {

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

//--------------------------------------------------------------------------------//
//                                                                                //
//                                David Blayvas                                   //
//                                                                                //
//                              Main part of code                                 //
//                                                                                //
//--------------------------------------------------------------------------------//

    public static int size;

    public static long compares;
    public static long swaps;

    public static int minInt;
    public static float minFlt;
    public static int maxInt;
    public static float maxFlt;


    public static Random rd = new Random();
```



```
    public boolean ErrorCheck()
    {
        boolean stop = false;
        if (IntButton.isSelected())
        {
            try
            {
                Integer.parseInt( MinText.getText() );
                MinError.setText("");
            }
            catch(NumberFormatException | NullPointerException e)
            {
                MinError.setText("*");
                stop = true;
            }
            try
            {
                Integer.parseInt( MaxText.getText() );
                MaxError.setText("");
            }
            catch(NumberFormatException | NullPointerException e)
            {
                MaxError.setText("*");
                stop = true;
            }
        }
        if (FltButton.isSelected())
        {
            try
            {
                Float.parseFloat( MinText.getText() );
                MinError.setText("");
            }
            catch(NumberFormatException | NullPointerException e)
            {
                MinError.setText("*");
                return true;
            }
            try

            {
                Float.parseFloat( MaxText.getText() );
                MaxError.setText("");
            }
```

```
            catch(NumberFormatException | NullPointerException e)
            {
                MaxError.setText("*");
                stop = true;
            }
        }
        try
        {
            Integer.parseInt( SizeText.getText() );
            SizeError.setText("");
        }
        catch(NumberFormatException | NullPointerException e)
        {
            SizeError.setText("*");
            stop = true;
        }
        return stop;
    }


    public int[] Fill(int[] list)
    {
        for (int i = 0; i < size; i++)
        {
            list[i] = rd.nextInt(maxInt - minInt + 1) + minInt;
        }
        return list;
    }
    public float[] Fill(float[] list)
    {
        for (int i = 0; i < size; i++)
        {
            list[i] = rd.nextFloat()*(maxFlt - minFlt) + minFlt;
        }
        return list;
    }



    public void DisplayArray(int list[])
    {
        int counter = 0;
        Swaps.setText(String.valueOf(swaps));
        Compares.setText(String.valueOf(compares));
        ArrayArea.setText("");
```

```
        ArrayArea.setText("");
        for (int i = 0; i < size; i++)
        {
            if (counter != 0)
            {
                ArrayArea.append(",  ");
            }
            ArrayArea.append( String.valueOf( list[i] ) );
            counter++;
            if (counter >= 10)
            {
                ArrayArea.append("\n");
                counter = 0;
            }
        }
    }
    public void DisplayArray(float[] list)
    {
        int counter = 0;
        Swaps.setText( String.valueOf(swaps) );
        Compares.setText( String.valueOf(compares) );
        ArrayArea.setText("");
        for (int i = 0; i < size; i++)
        {
            if (counter != 0)
            {
                ArrayArea.append(",  ");
            }
            ArrayArea.append( String.valueOf( list[i] ) );
            counter++;
            if (counter >= 1)
            {
                ArrayArea.append("\n");
                counter = 0;
            }

        }
    }



    public int[] Selection(int old[])
    {
        int high = 0;
        int[] sorted = new int[size];
        for (int i = size-1; i >= 0; i--)
```

```
        {
            for (int j = 0; j < size; j++)
            {
                if (old[j] > old[high])
                {
                    high = j;
                    swaps++;
                }
                compares++;
            }
            sorted[i] = old[high];
            old[high] = 0;
        }
        return sorted;
    }
    public float[] Selection(float[] old)
    {
        int high = 0;
        float[] sorted = new float[size];
        for (int i = size-1; i >= 0; i--)
        {
            for (int j = 0; j < size; j++)
            {
                if (old[j] > old[high])
                {
                    high = j;
                    swaps++;
                }
                compares++;
            }
            sorted[i] = old[high];
            old[high] = 0;

        }
        return sorted;
    }


    public int[] Insertion(int[] list)
    {
        for (int i = 1; i < size; i++)
        {
            int temp = list[i];
            for (int j = i-1; j >= 0; j--)
```

```
            {
                if (temp < list[j])
                {
                    list[j+1] = list[j];
                    swaps++;
                }
                compares++;
                if (j < 0 || list[j] > temp)
                {
                    list[j] = temp;
                    swaps++;
                }
                compares++;
            }
        }
        return list;
    }
    public float[] Insertion(float[] list)
    {
        for (int i = 1; i < size; i++)
        {
            float temp = list[i];
            for (int j = i-1; j >= 0; j--)
            {
                if (temp < list[j])
                {
                    list[j+1] = list[j];
                    swaps++;
                }
                compares++;
                if (j < 0 || list[j] > temp)
                {
                    list[j] = temp;
                    swaps++;
                }
                compares++;
            }
        }
        return list;
    }


    public int[] Bubble(int[] list)
```

```
    {
        for (int i = size-1; i > 0; i--)
        {
            for (int j = 0; j < i; j++)
            {
                if (list[j] > list[j+1])
                {
                    int temp = list[j+1];
                    list[j+1] = list[j];
                    list[j] = temp;
                    swaps++;
                }
                compares++;
            }
        }
        return list;
    }
    public float[] Bubble(float[] list)
    {
        for (int i = size-1; i > 0; i--)
        {
            for (int j = 0; j < i; j++)
            {
                if (list[j] > list[j+1])
                {
                    float temp = list[j+1];
                    list[j+1] = list[j];
                    list[j] = temp;

                    swaps++;
                }
                compares++;
            }
        }
        return list;
    }


    public float[] Quick(float[] numbers)
    {
        doQuickSort(numbers, 0, numbers.length-1);
        return numbers;
    }
    public void doQuickSort(float[] numbers, int start, int end)
```

```
    {
        if (start < end)
        {
            int middle = partition(numbers, start, end);
            doQuickSort(numbers, start, middle);
            doQuickSort(numbers, middle+1, end);
        }
    }
    public int partition (float[] numbers, int start, int end)
    {
        float pivot = numbers[start];
        int i = start - 1;
        int j = end + 1;
        while (true)
        {
            i++;
            while (numbers[i] < pivot) { i++; }
            j--;
            while (numbers[j] > pivot) { j--; }
            compares++;
            if (i < j)
            {
                float tmp = numbers[i];
                numbers[i] = numbers[j];
                numbers[j] = tmp;
                swaps++;

            }
            else return j;
        }
    }

    public int[] Quick(int[] numbers)
    {
        doQuickSort(numbers, 0, numbers.length-1);
        return numbers;
    }
    public void doQuickSort(int[] numbers, int start, int end)
    {
        if (start < end)
        {
            int middle = partition(numbers, start, end);
            doQuickSort(numbers, start, middle);
            doQuickSort(numbers, middle+1, end);
```

```
        }
    }
    public int partition (int[] numbers, int start, int end)
    {
        int pivot = numbers[(start+end)/2];
        int i = start - 1;
        int j = end + 1;
        while (true)
        {
            i++;
            while (numbers[i] < pivot) { i++; }
            j--;
            while (numbers[j] > pivot) { j--; }
            compares++;
            if (i < j)
            {
                int tmp = numbers[i];
                numbers[i] = numbers[j];
                numbers[j] = tmp;
                swaps++;
            }
            else return j;
        }
    }



    public int[] MergeSort(int[] numbers)
    {
        doMergeSort(numbers, 0, numbers.length-1);
        return numbers;
    }
    public void doMergeSort(int[] numbers, int start, int end)
    {
        if (start < end)
        {
            int middle = (start+end) / 2;
            doMergeSort(numbers, start, middle);
            doMergeSort(numbers, middle+1, end);
            merge(numbers, start, middle, end);
        }
    }
    public void merge(int[] numbers, int start, int middle, int end)
    {
        int[] tmp = new int[end-start+1];
```

```
        int[] tmp = new int[end-start+1];

        int index1 = start;
        int index2 = middle + 1;
        int indexTmp = 0;

        while (index1 <= middle && index2 <= end)
        {
            compares++;
            if (numbers[index1] < numbers[index2])
            {
                tmp[indexTmp] = numbers[index1];
                index1++;
                swaps++;
            }
            else
            {
                tmp[indexTmp] = numbers[index2];
                index2++;
                swaps++;
            }
            indexTmp++;

        }
        while (index1 <= middle)
        {
            tmp[indexTmp] = numbers[index1];
            index1++;
            indexTmp++;
        }
        while (index2 <= end)
        {
            tmp[indexTmp] = numbers[index2];
            index2++;
            indexTmp++;
        }
        for (indexTmp = 0; indexTmp < tmp.length; indexTmp++)
        {
            numbers[start + indexTmp] = tmp[indexTmp];
        }
    }

    public float[] MergeSort(float[] numbers)
    {
        doMergeSort(numbers, 0, numbers.length-1);
```

```
        return numbers;
    }
    public void doMergeSort(float[] numbers, int start, int end)
    {
        if (start < end)
        {
            int middle = (start+end) / 2;
            doMergeSort(numbers, start, middle);
            doMergeSort(numbers, middle+1, end);
            merge(numbers, start, middle, end);
        }
    }
    public void merge(float[] numbers, int start, int middle, int end)
    {
        float[] tmp = new float[end-start+1];

        int index1 = start;
        int index2 = middle + 1;
        int indexTmp = 0;


        while (index1 <= middle && index2 <= end)
        {
            compares++;
            if (numbers[index1] < numbers[index2])
            {
                tmp[indexTmp] = numbers[index1];
                index1++;
                swaps++;
            }
            else
            {
                tmp[indexTmp] = numbers[index2];
                index2++;
                swaps++;
            }
            indexTmp++;
        }
        while (index1 <= middle)
        {
            tmp[indexTmp] = numbers[index1];
            index1++;
            indexTmp++;
        }
```

```
        while (index2 <= end)
        {
            tmp[indexTmp] = numbers[index2];
            index2++;
            indexTmp++;
        }
        for (indexTmp = 0; indexTmp < tmp.length; indexTmp++)
        {
            numbers[start + indexTmp] = tmp[indexTmp];
        }
    }

    private int[] Heap(int[] numbers)
    {
        int length = numbers.length;

        HeapBuild(numbers, length);

        for (int i = length-1; i > 0; i--)
        {
            int temp = numbers[0];
            numbers[0] = numbers[i];
            numbers[i] = temp;
            Heapify(numbers, 1, i);
            swaps++;
        }

        return numbers;
    }
    private static void HeapBuild(int[] numbers, int size)
    {
        for (int i = size/2; i > 0; i--)
        {
            Heapify(numbers, i, size);
        }
    }
    private static void Heapify(int[] numbers, int index, int size)
    {
        int m = 2 * index,
            s = m + 1,
            max;

        if (m <= size && numbers[m-1] > numbers[index-1])
        {
```

```
            max = m;
        }
        else
        {
            max = index;
        }
        compares++;

        if(s <= size && numbers[s-1] > numbers[max-1])
        {
            max = s;
        }
        compares++;

        if (max != index)
        {
            int temp = numbers[index-1];
            numbers[index-1] = numbers[max-1];
            numbers[max-1] = temp;
            Heapify(numbers, max, size);
            swaps++;
        }
    }

    private float[] Heap(float[] numbers)
    {
        int length = numbers.length;

        HeapBuild(numbers, length);
        for (int i = length-1; i > 0; i--)
        {
            float temp = numbers[0];
            numbers[0] = numbers[i];
            numbers[i] = temp;
            Heapify(numbers, 1, i);
            swaps++;
        }

        return numbers;
    }
    private static void HeapBuild(float[] numbers, int size)
    {
        for (int i = size/2; i > 0; i--)
```

```
        {
            Heapify(numbers, i, size);
        }
    }
    private static void Heapify(float[] numbers, int index, int size)
    {
        int m = 2 * index,
            s = m + 1,
            max;

        if (m <= size && numbers[m-1] > numbers[index-1])
        {
            max = m;

        }
        else
        {
            max = index;
        }
        compares++;

        if(s <= size && numbers[s-1] > numbers[max-1])
        {
            max = s;
        }
        compares++;

        if (max != index)
        {
            float temp = numbers[index-1];
            numbers[index-1] = numbers[max-1];
            numbers[max-1] = temp;
            Heapify(numbers, max, size);
            swaps++;
        }
    }

//-----------------------------------------------------------------------------//
//                                                                             //
//                                David Blayvas                                //
//                                                                             //
//                             End main part of code                           //
//                                                                             //
//-----------------------------------------------------------------------------//
```

```
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        ArrayDisplay = new javax.swing.JFrame();
        jPanel2 = new javax.swing.JPanel();
        HideArray = new javax.swing.JButton();
        jScrollPane2 = new javax.swing.JScrollPane();
        jScrollPane1 = new javax.swing.JScrollPane();
        ArrayArea = new javax.swing.JTextArea();
        PrimaryPanel = new javax.swing.JPanel();
        MinText = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        MaxText = new javax.swing.JTextField();
        Choice = new javax.swing.JComboBox<>();
        Go = new javax.swing.JButton();
        Reset = new javax.swing.JButton();
        Exit = new javax.swing.JButton();
        DispArr = new javax.swing.JButton();
        Swaps = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        FltButton = new javax.swing.JRadioButton();
        IntButton = new javax.swing.JRadioButton();
        jLabel4 = new javax.swing.JLabel();
        SizeText = new javax.swing.JTextField();
        jLabel8 = new javax.swing.JLabel();
        Compares = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
        MinError = new javax.swing.JLabel();
        MaxError = new javax.swing.JLabel();
        SizeError = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        TimeField = new javax.swing.JTextField();

        HideArray.setText("Ok");
        HideArray.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                HideArrayActionPerformed(evt);
            }
        });
```

```
        ArrayArea.setEditable(false);
        ArrayArea.setColumns(20);
        ArrayArea.setRows(5);
        jScrollPane1.setViewportView(ArrayArea);

        jScrollPane2.setViewportView(jScrollPane1);

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayou
                    .addComponent(jScrollPane2, javax.swing.GroupLayout.Alignment.
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2L
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(HideArray)))
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE,
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELA
                .addComponent(HideArray)
                .addContainerGap())
        );

        javax.swing.GroupLayout ArrayDisplayLayout = new javax.swing.GroupLayout(A
        ArrayDisplay.getContentPane().setLayout(ArrayDisplayLayout);
        ArrayDisplayLayout.setHorizontalGroup(
            ArrayDisplayLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
            .addGroup(ArrayDisplayLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax
                .addContainerGap())
        );
        ArrayDisplayLayout.setVerticalGroup(
            ArrayDisplayLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
            .addGroup(ArrayDisplayLayout.createSequentialGroup()
                .addContainerGap()
```

```
2);


ADING)


t.Alignment.LEADING)
TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 360, Short.MAX_VALUE)
ayout.createSequentialGroup()




ADING)


192, Short.MAX_VALUE)
TED)



rrayDisplay.getContentPane());


nt.LEADING)


.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)


nt.LEADING)
```

```
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax
                .addContainerGap())
        );

        ArrayDisplay.pack();


        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        MinText.setText("0");

        jLabel1.setText("Min");

        jLabel2.setText("Max");

        MaxText.setText("100");

        Choice.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Ran

        Go.setText("Go");
        Go.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                GoActionPerformed(evt);
            }
        });

        Reset.setText("Reset");
        Reset.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ResetActionPerformed(evt);
            }
        });

        Exit.setText("Exit");
        Exit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ExitActionPerformed(evt);
            }
        });

        DispArr.setText("Display array");
        DispArr.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                DispArrActionPerformed(evt);
```

```
.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
dom", "Selection", "Insertion", "Bubble", "Quick", "Merge", "Heap" }));
```

```
            }
        });

        Swaps.setEditable(false);


        jLabel3.setText("Swaps");

        buttonGroup1.add(FltButton);
        FltButton.setText("Float");

        buttonGroup1.add(IntButton);
        IntButton.setSelected(true);
        IntButton.setText("Integer");

        jLabel4.setText("Size");

        SizeText.setText("50");

        jLabel8.setText("Algorithm");

        Compares.setEditable(false);

        jLabel5.setText("Comparisons");

        MinError.setForeground(new java.awt.Color(255, 0, 0));
        MinError.setText(" ");

        MaxError.setForeground(new java.awt.Color(255, 0, 0));
        MaxError.setText(" ");

        SizeError.setForeground(new java.awt.Color(255, 0, 0));
        SizeError.setText(" ");

        jLabel6.setText("Elapsed time (ms)");

        TimeField.setEditable(false);
        TimeField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                TimeFieldActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout PrimaryPanelLayout = new javax.swing.GroupLayout(P
```

```
rimaryPanel);
```

```
        PrimaryPanel.setLayout(PrimaryPanelLayout);
        PrimaryPanelLayout.setHorizontalGroup(

            PrimaryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
            .addGroup(PrimaryPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swing.Group
                    .addComponent(Go, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                    .addGroup(PrimaryPanelLayout.createSequentialGroup()
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addGroup(PrimaryPanelLayout.createSequentialGroup()
                                .addComponent(jLabel1)
                                .addPreferredGap(javax.swing.LayoutStyle.Component
                                .addComponent(MinError, javax.swing.GroupLayout.PR
                            .addComponent(MinText, javax.swing.GroupLayout.PREFERR
                        .addGap(18, 18, 18)
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addComponent(MaxText, javax.swing.GroupLayout.PREFERR
                            .addGroup(PrimaryPanelLayout.createSequentialGroup()
                                .addComponent(jLabel2)
                                .addPreferredGap(javax.swing.LayoutStyle.Component
                                .addComponent(MaxError, javax.swing.GroupLayout.PR
                        .addGap(18, 18, 18)
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addComponent(SizeText, javax.swing.GroupLayout.PREFER
                            .addGroup(PrimaryPanelLayout.createSequentialGroup()
                                .addComponent(jLabel4)
                                .addPreferredGap(javax.swing.LayoutStyle.Component
                                .addComponent(SizeError, javax.swing.GroupLayout.P
                        .addGap(18, 18, 18)
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addComponent(Choice, javax.swing.GroupLayout.PREFERRE
                            .addComponent(jLabel8))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacemen
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addComponent(IntButton)
                            .addComponent(FltButton)))
                    .addGroup(PrimaryPanelLayout.createSequentialGroup()
                        .addComponent(DispArr, javax.swing.GroupLayout.PREFERRED_S
                        .addGap(18, 18, 18)
                        .addComponent(Reset)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacemen
                        .addComponent(Exit))
                    .addGroup(PrimaryPanelLayout.createSequentialGroup()
```

```
nt.LEADING)


Layout.Alignment.LEADING)
swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

ng.GroupLayout.Alignment.LEADING)


Placement.RELATED)
EFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE))
ED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE))

ng.GroupLayout.Alignment.LEADING)
ED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE)


Placement.RELATED)
EFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)))

ng.GroupLayout.Alignment.LEADING)
RED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE)


Placement.RELATED)
REFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)))

ng.GroupLayout.Alignment.LEADING)
D_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SI

t.RELATED, 11, Short.MAX_VALUE)
ng.GroupLayout.Alignment.LEADING)



IZE, 118, javax.swing.GroupLayout.PREFERRED_SIZE)


t.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
ZE)
```

```
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addGroup(javax.swing.GroupLayout.Alignment.LEADING, P
                                .addComponent(jLabel6)
                                .addGap(18, 18, 18)
                                .addComponent(TimeField))
                            .addGroup(PrimaryPanelLayout.createSequentialGroup()
                                .addComponent(jLabel5)
                                .addPreferredGap(javax.swing.LayoutStyle.Component
                                .addComponent(Compares, javax.swing.GroupLayout.PR
                                .addGap(18, 18, 18)
                                .addComponent(jLabel3)
                                .addPreferredGap(javax.swing.LayoutStyle.Component
                                .addComponent(Swaps, javax.swing.GroupLayout.PREFE
                        .addGap(0, 0, Short.MAX_VALUE)))
                .addContainerGap())
        );
        PrimaryPanelLayout.setVerticalGroup(
            PrimaryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
            .addGroup(PrimaryPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swing.Group
                    .addGroup(PrimaryPanelLayout.createSequentialGroup()
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addGroup(PrimaryPanelLayout.createParallelGroup(javax
                                .addComponent(jLabel1)
                                .addComponent(jLabel2)
                                .addComponent(jLabel4)
                                .addComponent(MinError)
                                .addComponent(MaxError)
                                .addComponent(SizeError))
                            .addComponent(jLabel8))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacemen
                        .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swi
                            .addComponent(MinText, javax.swing.GroupLayout.PREFERR
                            .addComponent(MaxText, javax.swing.GroupLayout.PREFERR
                            .addComponent(SizeText, javax.swing.GroupLayout.PREFER
                            .addComponent(Choice, javax.swing.GroupLayout.PREFERRE
                    .addGroup(PrimaryPanelLayout.createSequentialGroup()
                        .addComponent(IntButton)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacemen
                        .addComponent(FltButton)))
                .addGap(24, 24, 24)
                .addComponent(Go)
```

```
ng.GroupLayout.Alignment.TRAILING, false)
rimaryPanelLayout.createSequentialGroup()




Placement.UNRELATED)
EFERRED_SIZE, 130, javax.swing.GroupLayout.PREFERRED_SIZE)


Placement.UNRELATED)
RRED_SIZE, 130, javax.swing.GroupLayout.PREFERRED_SIZE)))



nt.LEADING)


Layout.Alignment.TRAILING)

ng.GroupLayout.Alignment.TRAILING, false)
.swing.GroupLayout.Alignment.BASELINE)







t.RELATED)
ng.GroupLayout.Alignment.BASELINE)
ED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_S
ED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_S
RED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_
D_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SI


t.RELATED)
```

```
IZE)
IZE)
SIZE)
ZE)))
```

```
                .addGap(18, 18, 18)
                .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swing.Group
                    .addComponent(Swaps, javax.swing.GroupLayout.PREFERRED_SIZE, j
                    .addComponent(jLabel3)
                    .addComponent(Compares, javax.swing.GroupLayout.PREFERRED_SIZE
                    .addComponent(jLabel5))
                .addGap(18, 18, 18)
                .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swing.Group
                    .addComponent(jLabel6)
                    .addComponent(TimeField, javax.swing.GroupLayout.PREFERRED_SIZ
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
                .addGroup(PrimaryPanelLayout.createParallelGroup(javax.swing.Group
                    .addComponent(DispArr)
                    .addComponent(Reset)
                    .addComponent(Exit))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPan
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(PrimaryPanel, javax.swing.GroupLayout.PREFERRED_SIZE
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_V
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(PrimaryPanel, javax.swing.GroupLayout.PREFERRED_SIZE
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_V
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>


//--------------------------------------------------------------------------//
//                                                                          //
//                               David Blayvas                              //
//                                                                          //
```

```
Layout.Alignment.BASELINE)
avax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)


Layout.Alignment.BASELINE)

E, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
D, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
Layout.Alignment.BASELINE)









e());









, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
ALUE))




, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
ALUE))
```

```
//                                Main part of code                                //
//                                                                                 //
//---------------------------------------------------------------------------------//

    private void ExitActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void DispArrActionPerformed(java.awt.event.ActionEvent evt) {
        if (ArrayDisplay.isVisible())
        {
            ArrayDisplay.setVisible(false);
        }
        else
        {
            ArrayDisplay.setVisible(true);
        }
    }

    private void HideArrayActionPerformed(java.awt.event.ActionEvent evt) {
        ArrayDisplay.setVisible(false);
    }

    private void GoActionPerformed(java.awt.event.ActionEvent evt) {

        double startTime = 0, elapsedTime = 0;

        if ( ErrorCheck() )
        {
            return;
        }

        size = Integer.parseInt( SizeText.getText() );
        swaps = 0;
        compares = 0;

        DispArr.setEnabled(true);


        if (IntButton.isSelected())
        {
            maxInt = Integer.parseInt( MaxText.getText() );
            minInt = Integer.parseInt( MinText.getText() );
            int[] list = new int[size];
            list = Fill(list);
```

```
            list = Fill(list);

            switch (Choice.getSelectedIndex()) {
                case 1:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Selection(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 2:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Insertion(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 3:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Bubble(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 4:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Quick(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 5:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = MergeSort(list);

                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 6:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Heap(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
```

```
                    break;
                default:
                    break;
            }

            DisplayArray(list);
        }


        if (FltButton.isSelected())
        {
            maxFlt = Float.parseFloat( MaxText.getText() );
            minFlt = Float.parseFloat( MinText.getText() );
            float[] list = new float[size];
            list = Fill(list);

            switch (Choice.getSelectedIndex()) {
                case 1:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Selection(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 2:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Insertion(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 3:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Bubble(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 4:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Quick(list);
                    elapsedTime = System.nanoTime()- startTime;
```

```
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 5:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = MergeSort(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                case 6:
                    startTime = System.nanoTime();
                    elapsedTime = 0;
                    list = Heap(list);
                    elapsedTime = System.nanoTime()- startTime;
                    TimeField.setText("" + elapsedTime/1000000);
                    break;
                default:
                    break;
            }

            DisplayArray(list);
        }


    }

    private void ResetActionPerformed(java.awt.event.ActionEvent evt) {
        MinText.setText("0");
        MinError.setText("");
        MaxText.setText("100");

        MaxError.setText("");
        SizeText.setText("50");
        SizeError.setText("");
        Choice.setSelectedIndex(0);
        IntButton.setSelected(true);
        Compares.setText("");
        Swaps.setText("");
        size = 0;
        compares = 0;
        swaps = 0;
        minInt = 0;
        minFlt = 0;
        maxInt = 0;
```

```
        maxFlt = 0;
    }

//--------------------------------------------------------------------------//
//                                                                          //
//                                David Blayvas                             //
//                                                                          //
//                            End main part of code                         //
//                                                                          //
//--------------------------------------------------------------------------//

    private void TimeFieldActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @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
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the def
         * For details see http://download.oracle.com/javase/tutorial/uiswing/look
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManage
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(SortersGUI.class.getName()).log(jav
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(SortersGUI.class.getName()).log(jav
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(SortersGUI.class.getName()).log(jav
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(SortersGUI.class.getName()).log(jav
        }
        //</editor-fold>

        /* Create and display the form */
```

```
(optional) ">
ault look and feel.
andfeel/plaf.html


r.getInstalledLookAndFeels()) {









a.util.logging.Level.SEVERE, null, ex);

a.util.logging.Level.SEVERE, null, ex);

a.util.logging.Level.SEVERE, null, ex);

a.util.logging.Level.SEVERE, null, ex);
```

```
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new SortersGUI().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private static javax.swing.JTextArea ArrayArea;
    private static javax.swing.JFrame ArrayDisplay;
    private javax.swing.JComboBox<String> Choice;
    private static javax.swing.JTextField Compares;
    private javax.swing.JButton DispArr;
    private javax.swing.JButton Exit;
    private javax.swing.JRadioButton FltButton;
    private javax.swing.JButton Go;
    private javax.swing.JButton HideArray;
    private javax.swing.JRadioButton IntButton;
    private javax.swing.JLabel MaxError;
    private static javax.swing.JTextField MaxText;
    private javax.swing.JLabel MinError;
    private static javax.swing.JTextField MinText;
    private javax.swing.JPanel PrimaryPanel;
    private javax.swing.JButton Reset;
    private javax.swing.JLabel SizeError;
    private static javax.swing.JTextField SizeText;
    private static javax.swing.JTextField Swaps;
    private javax.swing.JTextField TimeField;
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JLabel jLabel1;
    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 jLabel8;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    // End of variables declaration
}
```