

FINAL PROJECT

ON

CS0007 - Algorithms (BSCSSE-N21)

MEMBERS:

Albert Josh Dizon

Mark Danhill Egana

Rajan Elio

Ramsel Fabian

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import java.awt.event.\*;

import java.beans.PropertyChangeEvent;

import java.beans.PropertyChangeListener;

import java.util.\*;

public class BruteForceFrame extends JFrame {

  private static final long serialVersionUID = 1L;

  private GridBagConstraints constraints;

  private JPanel sortPanel;

  private JPanel comboPanel;

  private JLabel sortLabel;

  private JComboBox<String> sortBox;

  private JPanel buttonPanel;

  private JButton inputButton;

  private JButton sortButton;

  private String inputStr = "";

  private ArrayList<Integer> inputArray;

  private String outputStr = "";

private JMenuBar menuBar;

private JMenu menu;

private JMenuItem aboutUs;

private JMenuItem aboutSys;

  private int inputSize;

  private int comparisonSize;

  private int swapSize;

  private static final String[] SORTALGO = {"Bubble Sort","Selection Sort"};

  public BruteForceFrame(){

    setTitle("Brute Force Algorithm");

    sortLabel = new JLabel("Sort Type: ");

    sortBox = new JComboBox<String>(SORTALGO);

    sortBox.setSelectedIndex(0);

    sortBox.setFocusable(false);

    inputButton = new JButton("Input");

    inputButton.setPreferredSize(new Dimension(75,25));

    inputButton.setFocusable(false);

    inputButton.addActionListener(new InputListener());

    sortButton = new JButton("Sort");

    sortButton.setPreferredSize(new Dimension(75,25));

    sortButton.setFocusable(false);

    sortButton.addActionListener(new SortListener());

    buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

    buttonPanel.add(inputButton);

    buttonPanel.add(sortButton);

    comboPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

    comboPanel.add(sortLabel);

    comboPanel.add(sortBox);

    constraints = new GridBagConstraints();

    sortPanel = new JPanel(new GridBagLayout());

    constraints.insets = new Insets(5,5,5,5);

    constraints.gridx = 0;

    constraints.gridy = 0;

    sortPanel.add(comboPanel,constraints);

    constraints.gridx = 0;

    constraints.gridy = 1;

    sortPanel.add(buttonPanel,constraints);

    add(sortPanel);

menuBar = new JMenuBar();

menu = new JMenu("About");

aboutUs = new JMenuItem("About us");

aboutSys = new JMenuItem("About system");

menu.add(aboutUs);

menu.add(aboutSys);

menuBar.add(menu);

super.setJMenuBar(menuBar);

aboutUs.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

if(e.getSource() == aboutUs)

{

            JOptionPane.showMessageDialog(rootPane, "Creators:\n" + "Rajan Emil Ceasar Elio\n"

            + "Mark Danhill Egana\n"

            + "Abraham Isreal Fabian\n"

            + "Albert Josh Dizon\n","About us",JOptionPane.PLAIN\_MESSAGE);

}

}

});

aboutSys.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent event)

{

            JOptionPane.showMessageDialog(rootPane,"Step 1: Click Input.\n"

            + "Step 2: Insert data. It reads left to right.\n"

            + "Step 3: Close the window and choose a sorting algorithm.\n"

            + "Step 4: Click Sort.","How to use the application",JOptionPane.PLAIN\_MESSAGE);

}

});

    pack();

    setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

    addWindowListener(new WindowAdapter(){

      @Override

      public void windowClosing(WindowEvent event){

        int response = JOptionPane.showConfirmDialog(rootPane, "Do you want to exit?",

        "Exit", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

        if (response == JOptionPane.YES\_OPTION){

          dispose();

        }

      }

    });

  }

  private class InputListener implements ActionListener {

    @Override

    public void actionPerformed(ActionEvent event){

      if (inputArray != null && inputArray.size() != 0){

        int response = JOptionPane.showConfirmDialog(rootPane, "Do you want to change the input?",

        "Input", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

        if (response == JOptionPane.YES\_OPTION){

          setInputArray();

        }

      }

      else {

        setInputArray();

      }

    }

  }

  private class SortListener implements ActionListener {

    @Override

    public void actionPerformed(ActionEvent event){

      if (inputArray == null || inputArray.size() == 0){

        JOptionPane.showMessageDialog(rootPane, "You must set the input first.",

        "Null Input", JOptionPane.WARNING\_MESSAGE);

      }

      else {

        setOutputArray();

      }

    }

  }

  private void setInputArray(){

    JTextArea inputArea = new JTextArea(inputStr);

    JScrollPane inputScroll = new JScrollPane(inputArea);

    inputScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

    inputScroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

    inputScroll.setPreferredSize(new Dimension(300,300));

    Object[] options = {"Set","Reset","Cancel"};

    JOptionPane optionPane = new JOptionPane(inputScroll, JOptionPane.PLAIN\_MESSAGE,

    JOptionPane.OK\_CANCEL\_OPTION, null, options);

    JDialog dialog = new JDialog(this, "Input", true);

    dialog.setContentPane(optionPane);

    optionPane.addPropertyChangeListener(new PropertyChangeListener(){

      @Override

      public void propertyChange(PropertyChangeEvent event) {

        if (JOptionPane.VALUE\_PROPERTY.equals(event.getPropertyName())){

          if (optionPane.getValue().equals(options[0])){

            optionPane.setValue(JOptionPane.UNINITIALIZED\_VALUE);

            try {

              ArrayList<Integer> tempArray = new ArrayList<Integer>();

              inputStr = inputArea.getText();

              String str = "";

              for (int i = 0; i < inputStr.length(); ++i){

                if (inputStr.charAt(i) != ' ' &&

                  inputStr.charAt(i) != '\n' &&

                  inputStr.charAt(i) != '\t'){

                  str = str + inputStr.charAt(i);

                }

                if (!(str.isEmpty()) && (inputStr.charAt(i) == ' ' || inputStr.charAt(i) == '\n' ||

                  inputStr.charAt(i) == '\t' || i == (inputStr.length() - 1))){

                  Integer input = Integer.valueOf(str);

                  for (int j = 0; j < tempArray.size(); ++j){

                    if (input == tempArray.get(j)){

                      throw new NumberFormatException("Duplicate Number Detected (" + input + ")");

                    }

                  }

                  tempArray.add(input);

                  str = "";

                }

              }

              inputArray = tempArray;

              dialog.dispose();

            }

            catch (NumberFormatException e){

              JOptionPane.showMessageDialog(rootPane, "NumberFormatException:\n"

              + e.getMessage(), "Number Format Exception", JOptionPane.ERROR\_MESSAGE);

            }

            catch (Exception e){

              e.printStackTrace();

              JOptionPane.showMessageDialog(rootPane, "Unimplemented Exception:\n"

              + e.getMessage(), "Exception", JOptionPane.ERROR\_MESSAGE);

            }

          }

          else if (optionPane.getValue().equals(options[1])){

            optionPane.setValue(JOptionPane.UNINITIALIZED\_VALUE);

            inputArea.setText("");

            inputStr = "";

            inputArray = null;

          }

          else if (optionPane.getValue().equals(options[2])){

            optionPane.setValue(JOptionPane.UNINITIALIZED\_VALUE);

            dialog.dispose();

          }

        }

      }

    });

    dialog.pack();

    dialog.setLocationRelativeTo(rootPane);

    dialog.setVisible(true);

  }

  private void setOutputArray(){

    sortArray();

    JTextArea outputPane = new JTextArea(outputStr);

    outputPane.setEditable(false);

    outputPane.setFocusable(false);

    JScrollPane outputScroll = new JScrollPane(outputPane);

    outputScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

    outputScroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

    outputScroll.setPreferredSize(new Dimension(500,500));

    Object[] options = {"Save","Summary","Close"};

    JOptionPane optionPane = new JOptionPane(outputScroll, JOptionPane.PLAIN\_MESSAGE,

    JOptionPane.OK\_CANCEL\_OPTION, null, options);

    JDialog dialog = new JDialog(this, SORTALGO[sortBox.getSelectedIndex()], true);

    dialog.setContentPane(optionPane);

    optionPane.addPropertyChangeListener(new PropertyChangeListener(){

      @Override

      public void propertyChange(PropertyChangeEvent event) {

        if (JOptionPane.VALUE\_PROPERTY.equals(event.getPropertyName())){

          if (optionPane.getValue().equals(options[0])){

            try {

              JFileChooser source = new JFileChooser();

              source.showSaveDialog(null);

              FileWriter fWriter = new FileWriter(source.getSelectedFile());

              BufferedWriter bWriter = new BufferedWriter(fWriter);

              bWriter.write(outputStr);

              bWriter.write("\ninput size: " + inputSize +

                     "\nNo. of comparison: " + comparisonSize +

                     "\nNo. of swap: " + swapSize);

              bWriter.close();

            }

            catch (IOException e){

              JOptionPane.showMessageDialog(rootPane, "IOException:\n"

              + e.getMessage(), "I/O Exception", JOptionPane.ERROR\_MESSAGE);

            }

            catch (NullPointerException e){

              // Expected Exception

            }

            catch (Exception e){

              e.printStackTrace();

              JOptionPane.showMessageDialog(rootPane, "Unimplemented Exception:\n"

              + e.getMessage(), "Exception", JOptionPane.ERROR\_MESSAGE);

            }

          }

          if (optionPane.getValue().equals(options[1])){

            JOptionPane.showMessageDialog(rootPane, "Input size: " + inputSize +

                                "\nNo. of comparison: " + comparisonSize +

                                "\nNo. of swap: " + swapSize,

                                "Summary", JOptionPane.INFORMATION\_MESSAGE);

          }

          else if (optionPane.getValue().equals(options[2])){

            optionPane.setValue(JOptionPane.UNINITIALIZED\_VALUE);

            dialog.dispose();

          }

        }

      }

    });

    dialog.pack();

    dialog.setLocationRelativeTo(rootPane);

    dialog.setVisible(true);

  }

  private void sortArray(){

    outputStr = "";

    int[] arr = new int[inputArray.size()];

    for (int i = 0; i < inputArray.size(); ++i){

      arr[i] = inputArray.get(i);

    }

    if (String.valueOf(sortBox.getSelectedItem()) == SORTALGO[0]){

      arr = bubbleSort(arr);

    }

    else if (String.valueOf(sortBox.getSelectedItem()) == SORTALGO[1]){

      arr = selectionSort(arr);

    }

    outputStr += "original array:\t";

    for (int i = 0; i < inputArray.size(); ++i){

      outputStr += inputArray.get(i) + " ";

    }

    outputStr += "\nsorted array:\t" + printArray(arr) + '\n';

  }

  private String printArray(int[] arr){

    String str = "";

    for (int i = 0; i < arr.length; ++i){

      str += arr[i] + " ";

    }

    return str;

  }

  private int[] bubbleSort(int[] arr){

    int comparisons = 0;

    int swaps = 0;

    outputStr += "Pass # 0\n";

    outputStr += printArray(arr) + "\n\n\n";

    for (int i = 0; i < (arr.length - 1); ++i){

      outputStr += "Pass # " + (i+1) + '\n';

      for (int j = (arr.length - 1); j > i; --j){

        int key = Integer.MAX\_VALUE;

        outputStr += printArray(arr) + '\n';

        outputStr += arr[j-1] + " > " + arr[j] + '\n';

        if (arr[j-1] > arr[j]){

          key = arr[j];

          arr[j] = arr[j-1];

          arr[j-1] = key;

        }

        if (key != Integer.MAX\_VALUE){

          outputStr += arr[j] + " <-> " + arr[j-1];

          ++swaps;

        }

        ++comparisons;

        outputStr += "\n\n";

      }

      outputStr += printArray(arr) + "\n\n\n";

    }

    inputSize = arr.length;

    comparisonSize = comparisons;

    swapSize = swaps;

    return arr;

  }

  private int[] selectionSort(int[] arr){

    int comparisons = 0;

    int swaps = 0;

    outputStr += "Pass # 0\n";

    outputStr += printArray(arr) + "\n\n\n";

    for (int i = 0; i < (arr.length-1); ++i){

      outputStr += "Pass # " + (i+1) + '\n';

      int key = arr[i];

      int index = i;

      for (int j = (i+1); j < arr.length; ++j){

        outputStr += printArray(arr) + '\n';

        outputStr += "key = " + key + '\n';

        outputStr += key + " > " + arr[j] +'\n';

        if (key > arr[j]){

          key = arr[j];

          index = j;

        }

        ++comparisons;

        outputStr += '\n';

      }

      arr[index] = arr[i];

      arr[i] = key;

      outputStr += printArray(arr) + '\n';

      outputStr += "key = " + key + '\n';

      if (index != i){

        outputStr += arr[index] + " <-> " + arr[i];

        ++swaps;

      }

      outputStr += "\n\n";

    }

    inputSize = arr.length;

    comparisonSize = comparisons;

    swapSize = swaps;

    return arr;

  }

  public static void main(String[] args) {

    BruteForceFrame bruteFrame = new BruteForceFrame();

    try {

      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

      bruteFrame.setSize(300,200);

      bruteFrame.setResizable(false);

      bruteFrame.setLocationRelativeTo(null);

      bruteFrame.setVisible(true);

    }

    catch (Exception e){

      e.printStackTrace();

      JOptionPane.showMessageDialog(bruteFrame, "Unimplemented Exception:\n"

      + e.getMessage(), "Exception", JOptionPane.ERROR\_MESSAGE);

    }

  }

}