**CS230 Assignment 4**

**Author:** Diana Eastman

**Date Submitted:** March 4, 2013

**Collaborators:** None.

**Notes:**

1. No known bugs.

2. The programs work as indicated in the comments.

3. To demonstrate that the intended functionality has been achieved, I have included a copy of testing Screenshots.

4. GradSchools.java also included because additional methods created.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// GradSchoolsGUI.java

// Created by: Diana Eastman

// Created: March 4, 2013

// Purpose: Creates a Tabbed Pane GUI with 3 Tabs to allow a user to

// add data about graduate schools to a collection and then use

// components (sliders and radio buttons) to dynamically rank the

// school in the collection based on Academics, Publication Impact,

// Research or Overall Rating. This GUI uses GradSchools.java and

// Schools.java to facilitate the creation of School and GradSchool

// Objects and perform any backend operations

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import javax.swing.\*;

import java.awt.Container;

public class GradSchoolsGUI

{

public GradSchools myGradschools;

//-----------------------------------------------------------------

// Sets up a frame containing a tabbed pane with three tabs

// Calls the GradSchools constructor and then passes

// myGradSchools as a parameter to AddSchools and EvalSchools

// Adds three schools to the collection for testing purposes

//-----------------------------------------------------------------

public static void main (String[] args)

{

JFrame myframe = new JFrame ("Graduate School Picker");

myframe.setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);

GradSchools gsCollection = new GradSchools(20);

gsCollection.addSchool("Research U", 3, 5, 1); //Add schools for testing purposes

gsCollection.addSchool("Publications U", 1, 3, 5);

gsCollection.addSchool("Academics U", 5, 3, 1);

JTabbedPane mytpane = new JTabbedPane();

mytpane.addTab ("About", new AboutPanel());

mytpane.addTab ("Add School", new AddSchoolsPanel(gsCollection));

mytpane.addTab ("Evaluate", new EvalSchoolsPanel(gsCollection));

myframe.getContentPane().add(mytpane);

myframe.pack();

myframe.setVisible(true);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// About.java

// Created by: Diana Eastman

// Created: March 4, 2013

// Purpose: Represents the About Panel for GradSchoolsGUI.java

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.awt.\*;

import javax.swing.\*;

import javax.swing.text.SimpleAttributeSet;

import javax.swing.text.StyleConstants;

import javax.swing.text.AttributeSet;

import javax.swing.border.Border;

import javax.swing.BorderFactory;

public class AboutPanel extends JPanel

{

public AboutPanel()

{

setLayout (new BoxLayout(this, BoxLayout.Y\_AXIS));

setBackground (new Color (240, 248, 255));

JLabel headerText = new JLabel("Find the right grad school; it's easy.");

JLabel instructionText = new JLabel("Use the Add School tab to create your personal collection of grad schools.");

JLabel instructionText2 = new JLabel("Then, use the Evaluate tab to order your collection by the factors you care about!");

Border border = BorderFactory.createLineBorder(new Color (240, 248, 255)); //Add a black border to the outside of the JTextArea

headerText.setBorder(BorderFactory.createCompoundBorder(border,

BorderFactory.createEmptyBorder(20, 20, 20, 20))); //Empty border creates a margin btwn JTextArea and panel edge

instructionText.setBorder(BorderFactory.createCompoundBorder(border,

BorderFactory.createEmptyBorder(10, 50, 5, 50))); //Empty border creates a margin btwn JTextArea and panel edge

instructionText2.setBorder(BorderFactory.createCompoundBorder(border,

BorderFactory.createEmptyBorder(5, 50, 10, 50))); //Empty border creates a margin btwn JTextArea and panel edge

headerText.setFont(new Font("Sans-serif", Font.BOLD, 40));

instructionText.setFont(new Font("Sans-serif", Font.BOLD, 18));

instructionText2.setFont(new Font("Sans-serif", Font.BOLD, 18));

add(headerText);

add(instructionText);

add(instructionText2);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// AddSchoolsPanel.java

// Created by: Diana Eastman

// Created: March 4, 2013

// Purpose: Represents the AddSchools Panel for GradSchoolsGUI.java

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.BoxLayout;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JSlider;

import javax.swing.JTextField;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

import javax.swing.border.Border;

import javax.swing.BorderFactory;

public class AddSchoolsPanel extends JPanel {

//Declare instance variables

private GradSchools gsCollection;

private JPanel outerPanel, instructions, controls, newSchool;

private JLabel schoolLabel, instructionsLabel, academicsLabel, researchLabel, publicationsLabel;

private JComboBox gsAcademics, gsResearch, gsPublications;

private JTextField gsNameInput;

private JButton addSchool;

private JTextArea txtArea;

private JScrollPane scroll;

private int aSelection;

private int rSelection;

private int pSelection;

/\*Constructor\*/

public AddSchoolsPanel(GradSchools gsCollection){

this.gsCollection = gsCollection;

String[] rank = {"0", "1", "2", "3", "4", "5"}; //Array of Strings, passed to each ComboBox

/\*BorderLayout used for this tab\*/

setLayout(new BorderLayout());

/\*Declare Labels\*/

instructionsLabel = new JLabel("Fill in the information to add a school, then click \"Add School\".");

schoolLabel = new JLabel("School Name"); //Label for JTextField

academicsLabel = new JLabel("Academics"); //Labels for each of the ComboBoxes

researchLabel = new JLabel("Research");

publicationsLabel = new JLabel("Publications");

/\*Initalize all 3 ComboBoxes and the school name JTextField and set alignment of each to left\*/

gsAcademics = new JComboBox(rank); gsAcademics.setAlignmentX (Component.LEFT\_ALIGNMENT);

gsResearch = new JComboBox(rank); gsResearch.setAlignmentX (Component.LEFT\_ALIGNMENT);

gsPublications = new JComboBox(rank); gsPublications.setAlignmentX (Component.LEFT\_ALIGNMENT);

addSchool = new JButton("Add School"); addSchool.setAlignmentX (Component.LEFT\_ALIGNMENT);

gsNameInput = new JTextField(10); gsNameInput.setAlignmentX (Component.LEFT\_ALIGNMENT);

/\*Set up ComboBox listeners\*/

gsAcademics.addActionListener (new ComboListener());

gsResearch.addActionListener (new ComboListener());

gsPublications.addActionListener (new ComboListener());

/\*Set up Add School Button listener\*/

addSchool.addActionListener(new ButtonListener());

/\*Create a new JTextArea and populate it with informational text\*/

txtArea = new JTextArea("Your new school will be added to those in the database and appear here.", 200, 400);

txtArea.setAlignmentX(Component.LEFT\_ALIGNMENT);

txtArea.setLineWrap(true); //Wrap text so JTextArea remains within preferred dimensions; otherwise no line breaks

txtArea.setWrapStyleWord(true);

Border border = BorderFactory.createLineBorder(Color.BLACK); //Add a black border to the outside of the JTextArea

txtArea.setBorder(BorderFactory.createCompoundBorder(border,

BorderFactory.createEmptyBorder(10, 10, 10, 10))); //Empty border creates a margin btwn JTextArea and panel edge

txtArea.setEditable(false); //Set textArea to non-editable

scroll = new JScrollPane(txtArea); //Add scrollbars to JTextArea to allow text to scroll if beyond dimensions

scroll.setPreferredSize(new Dimension(450, 400));

scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS);

/\*Create a new JPanel to hold the text instructions for using this tab; set its bg color to ghost white\*/

instructions = new JPanel();

instructions.setBackground(new Color (248, 248, 255));

/\*Create a new JPanel to hold the ComboBoxes, JTextField for school name, and Add School button; set bg color to light blue\*/

controls = new JPanel();

controls.setLayout(new BoxLayout(controls, BoxLayout.Y\_AXIS));

controls.setBorder(BorderFactory.createCompoundBorder(border,

BorderFactory.createEmptyBorder(10, 10, 10, 10))); //Empty border creates a margin btwn JPanel and edge

controls.setBackground(new Color(240, 248, 255));

/\*Create a new JPanel to hold the JTextArea for schools listing\*/

newSchool = new JPanel();

newSchool.setBackground(new Color (119, 136, 153));

newSchool.setBorder(BorderFactory.createCompoundBorder(border,

BorderFactory.createEmptyBorder(10, 10, 10, 10)));

/\*Add components to JPanels\*/

instructions.add(instructionsLabel);

controls.add (Box.createRigidArea (new Dimension (0, 20))); //RigidAreas create white space between components

controls.add(schoolLabel);

controls.add(gsNameInput);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add(academicsLabel);

controls.add(gsAcademics);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add(researchLabel);

controls.add(gsResearch);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add(publicationsLabel);

controls.add(gsPublications);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add(addSchool);

controls.add (Box.createRigidArea (new Dimension (0, 300))); //Large RigidArea between Add School button and bottom of controls JPanel

newSchool.add (scroll);

/\*Add Panels to BorderLayout\*/

add(instructions, BorderLayout.NORTH);

add(newSchool, BorderLayout.CENTER);

add(controls, BorderLayout.WEST);

}

public void updateTextArea(){

//Gets the new collection of schools and stores in temp array for clarity

School temp[] = gsCollection.getCollection();

txtArea.setText("");

/\* Loops through temp array from last position (displaying school last added to collection first)

\* Appends each school in collection to JTextArea

\* Used in lieu of toString(), which wasn't displaying satisfactorily in GUI\*/

for (int i=gsCollection.count-1; i > -1; i--){

txtArea.append("School Name: " + temp[i].getName());

txtArea.append(System.getProperty("line.separator")); //Line break

txtArea.append("Academics: " + temp[i].getAcademicRating()

+ " " + "Publications: " + temp[i].getPubImpactRating()

+ " " + "Research: " + " " +

temp[i].getResearchRating());

txtArea.append(System.getProperty("line.separator")); //Line break

txtArea.append(System.getProperty("line.separator")); //Line break

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Represents the listener for all three ComboBoxes

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private class ComboListener implements ActionListener{

private ComboListener(){

}

/\*Gets the valueOf ComboBox indices on button press and used in addSchools() method below\*/

public void actionPerformed (ActionEvent event){

aSelection = gsAcademics.getSelectedIndex();

pSelection = gsPublications.getSelectedIndex();

rSelection = gsResearch.getSelectedIndex();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Represents the listener for the Add School button

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private class ButtonListener implements ActionListener {

private ButtonListener(){

}

public void actionPerformed (ActionEvent event){

if (event.getSource() == addSchool) {

String name = gsNameInput.getText();

/\*If a school name is not entered, show dialog box; addSchool() not called\*/

if (name.equals("")) {

JOptionPane.showMessageDialog(null,

"You must specify a school name.",

"Missing Information",JOptionPane.WARNING\_MESSAGE);

}

else {

String a = String.valueOf(aSelection);

String r = String.valueOf(rSelection);

String p = String.valueOf(pSelection);

/\*Call method addSchool(), which has been modified to return false if school name exists in collection,

\* to determine if the school already exists in the collection.

\* If school exists, dialog box shown with appropriate warning message; school is not added

\* Otherwise, the school is added to the collection and updateTextArea() method called \*/

if (gsCollection.addSchool(name, Integer.parseInt(a), Integer.parseInt(r), Integer.parseInt(p))==false) {

JOptionPane.showMessageDialog(null,

"This school already exists in the database. Please add a unique school.",

"Duplicate Entry",JOptionPane.WARNING\_MESSAGE);

}

else {

updateTextArea();

}

gsNameInput.setText(""); //Clear school name JTextField after submission (whether school name is valid or not)

}

}

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// EvalSchoolsPanel.java

// Created by: Diana Eastman

// Created: March 4, 2013

// Purpose: Represents the EvalSchools Panel for GradSchoolsGUI.java

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.event.\*;

import java.util.\*;

import java.awt.event.ActionListener;

import javax.swing.BoxLayout;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JSlider;

import javax.swing.JTextField;

import javax.swing.JTextArea;

import javax.swing.border.Border;

import javax.swing.BorderFactory;

public class EvalSchoolsPanel extends JPanel{

/\*Instance variables\*/

private GradSchools gsCollection;

private JPanel controls;

private JLabel aLabel, pLabel, rLabel;

private JSlider aSlider, pSlider, rSlider;

private JRadioButton academics, publications, research, overall;

private JTextArea txtArea;

/\*Constructor\*/

public EvalSchoolsPanel(GradSchools gsCollection){

this.gsCollection = gsCollection;

//Set up Academics slider

aSlider = new JSlider (JSlider.HORIZONTAL, 0, 5, 0);

aSlider.setMajorTickSpacing (5); //Rating is from 0 - 5 (repeated for all three sliders)

aSlider.setMinorTickSpacing (1); //Rating scale increments by 1 (repeated for all three sliders)

aSlider.setPaintTicks (true);

aSlider.setPaintLabels (true);

aSlider.setAlignmentX (Component.LEFT\_ALIGNMENT);

//Set up Publications slider

pSlider = new JSlider (JSlider.HORIZONTAL, 0, 5, 0);

pSlider.setMajorTickSpacing (5);

pSlider.setMinorTickSpacing (1);

pSlider.setPaintTicks (true);

pSlider.setPaintLabels (true);

pSlider.setAlignmentX (Component.LEFT\_ALIGNMENT);

//Set up Research slider

rSlider = new JSlider (JSlider.HORIZONTAL, 0, 5, 0);

rSlider.setMajorTickSpacing (5);

rSlider.setMinorTickSpacing (1);

rSlider.setPaintTicks (true);

rSlider.setPaintLabels (true);

rSlider.setAlignmentX (Component.LEFT\_ALIGNMENT);

//Set up slider listeners

SliderListener listenerSlider = new SliderListener();

aSlider.addChangeListener (listenerSlider);

pSlider.addChangeListener (listenerSlider);

rSlider.addChangeListener (listenerSlider);

//Create labels for each of the three sliders

aLabel = new JLabel ("Academics: 0");

aLabel.setAlignmentX (Component.LEFT\_ALIGNMENT);

pLabel = new JLabel ("Publications: 0");

pLabel.setAlignmentX (Component.LEFT\_ALIGNMENT);

rLabel = new JLabel ("Research: 0");

rLabel.setAlignmentX (Component.LEFT\_ALIGNMENT);

//Set up radio buttons; will determine how the user would like to rank the schools in her/his colllection

academics = new JRadioButton("Academics", true);

publications = new JRadioButton("Publications", true);

research = new JRadioButton("Research", true);

overall = new JRadioButton("Overall", true);

//Add radio buttons to a button group

ButtonGroup group = new ButtonGroup();

group.add(academics);

group.add(publications);

group.add(research);

group.add(overall);

//Set up radio button listeners

CategoryListener listenerRadio = new CategoryListener();

academics.addActionListener(listenerRadio);

publications.addActionListener(listenerRadio);

research.addActionListener(listenerRadio);

overall.addActionListener(listenerRadio);

/\*Create a new JTextArea and populate it with informational text initially\*/

txtArea = new JTextArea("Use the sliders to change how highly you weigh each factor. " +

"Toggle the radio buttons to change the ranking order, based on factor. " +

"The top 3 schools will be printed below.");

txtArea.setAlignmentX(Component.LEFT\_ALIGNMENT);

txtArea.setLineWrap(true); //Wrap text so JTextArea remains within preferred dimensions; otherwise no line breaks

txtArea.setWrapStyleWord(true);

txtArea.setEditable(false); //Set textArea to non-editable

Border border = BorderFactory.createLineBorder(Color.BLACK); //Add a black border to the outside of the JTextArea

txtArea.setBorder(BorderFactory.createCompoundBorder(border, //Empty border creates a margin btwn JTextArea and panel edge

BorderFactory.createEmptyBorder(10, 10, 10, 10)));

/\*Create a new JPanel to hold the sliders and radio buttons; set layout to BoxLayout; add all components to panel\*/

controls = new JPanel();

controls.setPreferredSize(new Dimension(450, 550));

BoxLayout layout = new BoxLayout (controls, BoxLayout.Y\_AXIS);

controls.setLayout (layout);

controls.add (aLabel);

controls.add (aSlider);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add (pLabel);

controls.add (pSlider);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add (rLabel);

controls.add (rSlider);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add (academics);

controls.add (publications);

controls.add (research);

controls.add (overall);

controls.add (Box.createRigidArea (new Dimension (0, 20)));

controls.add (txtArea);

add (controls);

}

//--------------------------------------------------------------

// Gets the top three schools based on weighting (determined

// by sliders) and/or factor (determined by radio buttons)

// Replaces informational JTextArea with empty string

// before each call of method and appends the top three schools

// to JTextArea through each turn of the loop

//--------------------------------------------------------------

public void updateTextArea(){

School[] temp = gsCollection.getTopThree();

int count = temp.length;

txtArea.setText("");

for (int i=0; i < count; i++){

txtArea.append("School Name: " + temp[i].getName());

txtArea.append(System.getProperty("line.separator"));

txtArea.append("Academics: " + temp[i].getAcademicRating()

+ " " + "Publications: " + temp[i].getPubImpactRating()

+ " " + "Research: " + " " +

temp[i].getResearchRating() + " " + "Overall Rating: " + temp[i].getOverallRating());

txtArea.append(System.getProperty("line.separator"));

txtArea.append(System.getProperty("line.separator"));

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Represents the listener for all three sliders.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private class CategoryListener implements ActionListener{

public void actionPerformed (ActionEvent event){

Object source = event.getSource();

if (source == academics)

gsCollection.rankSchools("Academics");

else if (source == publications)

gsCollection.rankSchools("Publications");

else if (source == research)

gsCollection.rankSchools("Research");

else gsCollection.rankSchools("Overall");

updateTextArea(); //Calls updateTextArea to reorder print out, based on factor chosen

}

}

private class SliderListener implements ChangeListener{

private int aweight, pweight, rweight;

//--------------------------------------------------------------

// Gets the value of each slider, then updates the labels and

// computes the Overall Rating; calls updateTextArea

// based on weighting chosen

//--------------------------------------------------------------

public void stateChanged (ChangeEvent event)

{

aweight = aSlider.getValue();

pweight = pSlider.getValue();

rweight = rSlider.getValue();

aLabel.setText ("Academics: " + aweight);

pLabel.setText ("Publications: " + pweight);

rLabel.setText ("Research: " + rweight);

gsCollection.computeRating(aweight, pweight, rweight);

updateTextArea();

}

}

}

/\*\*

\* GradSchools.java

\* Purpose: Stores information about a collection of CS grad schools and provides methods to sort the schools

\* according to four, weighted factors (Academics, Publication Impact, Research, and Overall Rating)

\* Written by: Diana Eastman

\* Modified date: February 19, 2013

\*/

import java.util.\*;

public class GradSchools {

private School[] collection;

private School[] topThreeSchools;

public int count;

private int test;

public String[] schoolNames;

/\*\*

\* School Constructor

\* Creates a collection of schools

\* and initializes count variable to 0

\*/

public GradSchools (int numSchools)

{

collection = new School[numSchools]; //Set at 4 to show that the addSchool method works when 5th school added in main

count = 0;

}

/\*\*

\* Loops through command line parameters,

\* passing to isInRange helper method; returns false

\* when encounters integer in the array that is out of range

\* @param integer array of weights arr

\* @return boolean

\*/

public static boolean allInRange(int[] arr)

{

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

{

if (!isInRange(arr[i])) return false;

}

return true;

}

/\*\*

\* Tests if integer is between 1 and 5

\* @param integer i

\* @return boolean

\*/

private static boolean isInRange(int i)

{

return i > 0 && i < 6;

}

/\*\*

\* Calls isDuplicate() method, which returns a boolean

\* Adds a school to the collection array and increments count variable if school to be added is not

\* a duplicate (i.e., isDuplicate() == false)

\* Method itself returns true after adding a school to the collection

\* @param name, aRating (Academic rating), rRating (research rating),

\* @returns boolean

\* pRating (Publication Impact rating)

\*/

public boolean addSchool (String name, int aRating, int rRating, int pRating)

{

if (isDuplicate(name)==false) {

if (count==collection.length) //Calls increaseSize() if count=current size of collection array so another school can be added

increaseSize();

collection[count] = new School(name, aRating, rRating, pRating);

count ++;

return true;

}

else

return false;

}

/\*\*

\* Loops through all school names in collection and determines if school to be added from AddSchoolsPane.java already exists

\* Returns true if duplicate name is found

\* @param name

\* @returns boolean

\*/

public boolean isDuplicate(String name)

{

for (int i=0; i<count; i++) {

if (name.toLowerCase().equals(collection[i].getName().toLowerCase()))

return true;

}

return false;

}

/\*\*

\* Doubles the size of the collection array by creating a temp array twice as large

\* and copying the collection array into the new temp array

\* from Java Foundations

\*/

private void increaseSize ()

{

School[] temp = new School[collection.length\*2];

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

{

temp[i]=collection[i];

}

collection = temp;

}

/\*\*

\* Computes the weighted rating of each school, based on command line

\* parameters

\* @param, aweight (weight for Academics), rweight (weight for Research), pweight (weight for Pub. Impact)

\*/

public void computeRating (int aweight, int rweight, int pweight)

{

int i =0;

for (i = 0; i < count; i++)

{

collection[i].computeRating(aweight, rweight, pweight);

}

}

/\*\*

\* Loops through the Schools in the collection and stores the appropriate factor (based on input parameter)

\* into the rank variable; calls sortArray() method

\* @param rankbyElement (determines which factor to sort on)

\*/

public void rankSchools (String rankbyElement)

{

int i =0;

for (i = 0; i < count; i ++)

{

if (rankbyElement.equals("Academics"))

{

collection[i].rank = collection[i].getAcademicRating();

}

else if (rankbyElement.equals("Research"))

{

collection[i].rank = collection[i].getResearchRating();

}

else if (rankbyElement.equals("Publications"))

{

collection[i].rank = collection[i].getPubImpactRating();

}

else if (rankbyElement.equals("Overall"))

{

collection[i].rank = collection[i].getOverallRating();

}

}

sortArray();

}

/\*

\* Sorts the integers in the input array in descending order and calls printOut()

\* Explanation (Task 1):

\* Works by dividing the array to be sorted into two parts: the part that is already sorted

\* and the part that isn't (initially, the sorted portion is empty).

\* At each step in the algorithm, the sorted portion of the array is extended by one element.

\* The inner for loop finds the largest element in the unsorted portion of the array and

\* swaps it with the element with the index as the first counter (at first pass through

\* elements at [0] and [1]) Array size - 1 passes through the loop are required to sort the entire array.

\* Modified from Sort.java

\*/

public void sortArray ()

{

int maxNum; // maximum integer so far

int maxIndex; // index of maximum integer

int i, j;

for (j = count - 1; j > 0; j--)

{

maxIndex = 0;

maxNum = collection[0].rank;

for (i = 1; i <= j; i++)

if (collection[i].rank < maxNum) //descending order

{

maxNum = collection[i].rank;

maxIndex = i;

}

swap(collection, maxIndex, j);

}

printOut(collection);

}

/\*\*

\* exchanges the contents of locations i and j in the input array

\*/

private void swap (School[] collection, int i, int j)

{

School temp = collection[i];

collection[i] = collection[j];

collection[j] = temp;

}

/\*\*

\* Prints the name of each School in the collection in descending order after sorting

\*/

private void printOut (School[] collection)

{

int i=0;

for (i = 0; i<count; i++)

{

System.out.println(collection[i].getName());

}

}

/\*\*

\* toString method to print out GradSchool object

\* @return String

\*/

final public String toString()

{

String result ="";

result += "There are " + count + " schools in the database" +"\n";

for (int i = 0; i < count; i++) {

result += collection[i].toString();

}

return result;

}

public School[] getTopThree()

{

int c = 0;

if (count > 2)

c = 3;

else if (count < 3 && count > 1)

c = 2;

else if (count == 1)

c = 1;

topThreeSchools = new School[c];

for (int i=0; i<c; i++){

topThreeSchools[i]=collection[i];

}

return topThreeSchools;

}

public School[] getCollection()

{

return collection;

}

//Testing method - takes in three parameters from the command line

public static void main(String args[])

{

if ((args.length != 3)) //Exit if three weights not provided

{

System.out.println("Please provide 3 weights between 1 and 5 for Academics, Research and Publications");

System.exit(0);

}

try {

int aweight = Integer.parseInt(args[0]);

int rweight = Integer.parseInt(args[1]);

int pweight = Integer.parseInt(args[2]);

int [] arr = {aweight, rweight, pweight};

//Calls allInRage method and proceeds if true

if (allInRange(arr))

{

/\*Instantiate GradSchools object and add schools to database\*/

/\*myGradSchools.addSchool("MIT", 10, 10, 7);

myGradSchools.addSchool("Stanford", 8, 10, 9);

myGradSchools.addSchool("CMU", 7, 8, 6);

myGradSchools.addSchool("UC Berkeley", 9, 9, 9);

myGradSchools.addSchool("Cornell", 8, 9, 6); //Add 5th school to show that increaseSize() method functions properly

myGradSchools.computeRating(aweight, rweight, pweight);\*/

/\*Print output\*/

/\*System.out.println(myschools);

System.out.println("\n" + "Ranking of schools from highest to lowest using Academics as a factor:");

myschools.rankSchools("Academics");

System.out.println("\n" + "Ranking of schools from highest to lowest using Publication Impact as a factor:");

myschools.rankSchools("Publications");

System.out.println("\n" + "Ranking of schools from highest to lowest using Research as a factor:");

myschools.rankSchools("Research");

System.out.println( "\n" + "Ranking of schools from highest to lowest using Overall as a factor:");

myschools.rankSchools("Overall");\*/

}

else

{

System.out.println("One or more values are out of range.");

}

}

catch (NumberFormatException e) {

System.out.println("Sorry, one of your inputs is not an integer.");

System.out.println("Please provide 3 weights between 1 and 5, for Academics Research and Publications");

}

}

} // End GradSchool Class