[Group 27]

[PORTFOLIO I]

Claude Cullen & Jered Hoff

[Group 27 Enterprises]

Table of Contents

Introduction

The purpose of this portfolio was to demonstrate our understanding of Java Swing and to present a challenge to us to expand on what we learned in class to develop a useful piece of software. With both members of the group having no background experience in Java Swing, there was a lot to learn. We decided that a Java Swing application to help organize a student's homework schedule would be an adequate way to demonstrate a sufficient understanding of the material covered in class, as well as give us an opportunity to exhibit the aspects of Java Swing we came across in our own personal research. We created two new data models to organize the user provided information in a way that could be saved and read from a .txt file to ensure no data was lost when the user closes the application. With the data manipulation working well, we were able to expand on our user experience through simplifying the user interface. We implemented open-source libraries we found online to eliminate formatting inconsistencies and to improve the flow and look of the application.

Interaction with Materials

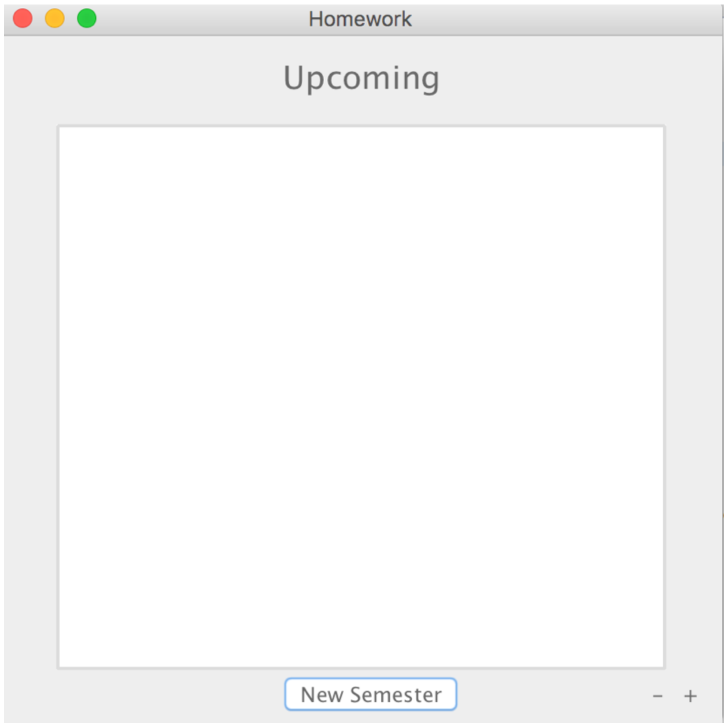
Insights & Questions

- How do we store data about homework assignments (i.e. due date, class assignment is for, assignment name, etc.)?
- How do we receive multiple inputs from a dialog window?
- When creating a new semester, how do we allow the user to see the class list they are creating without changing the current class list incase a mistake is made and the user wants to return to the current class list?
- How do we ensure that the user enters the due date of a new assignment in the correct format?
- How do we sort the list of assignments based on their due dates?

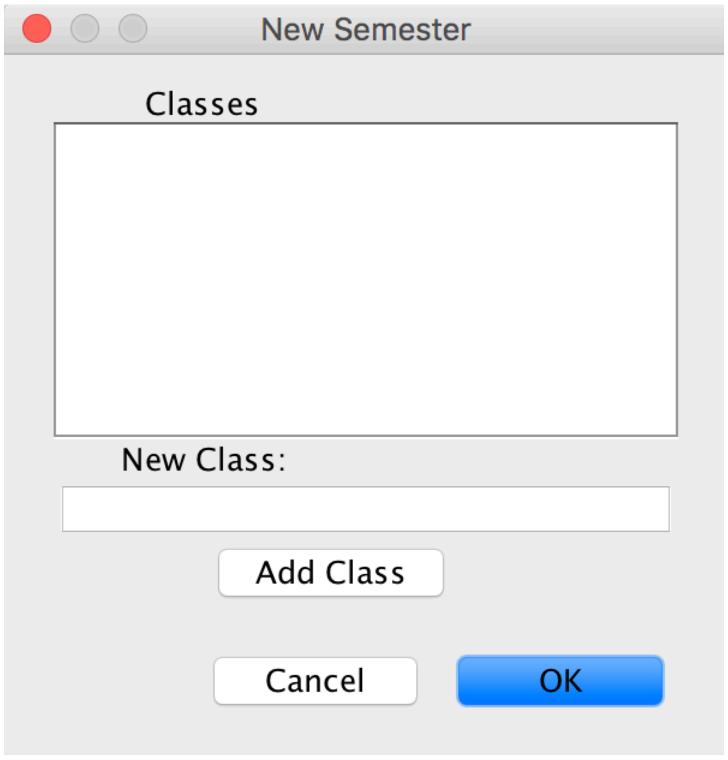
User Interface Elements

First Use

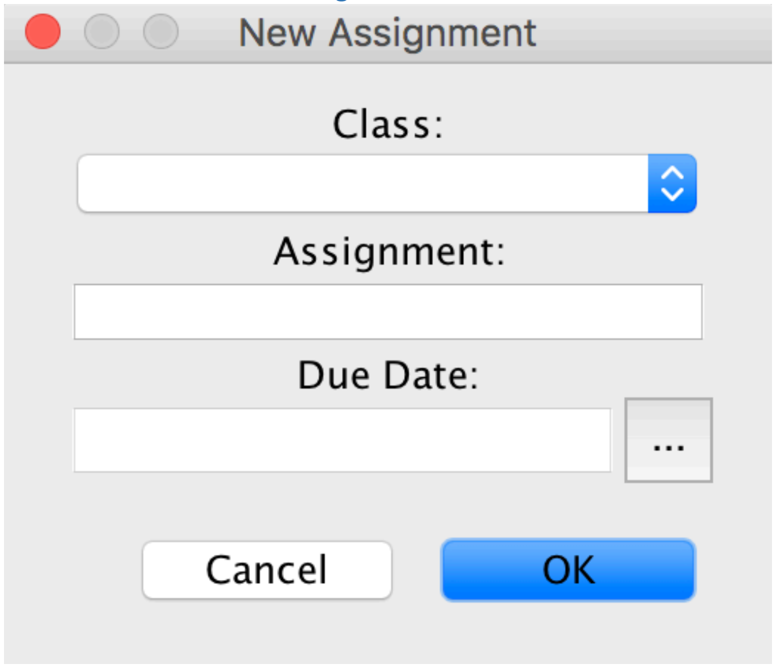
Homework List Interface



New Semester Interface

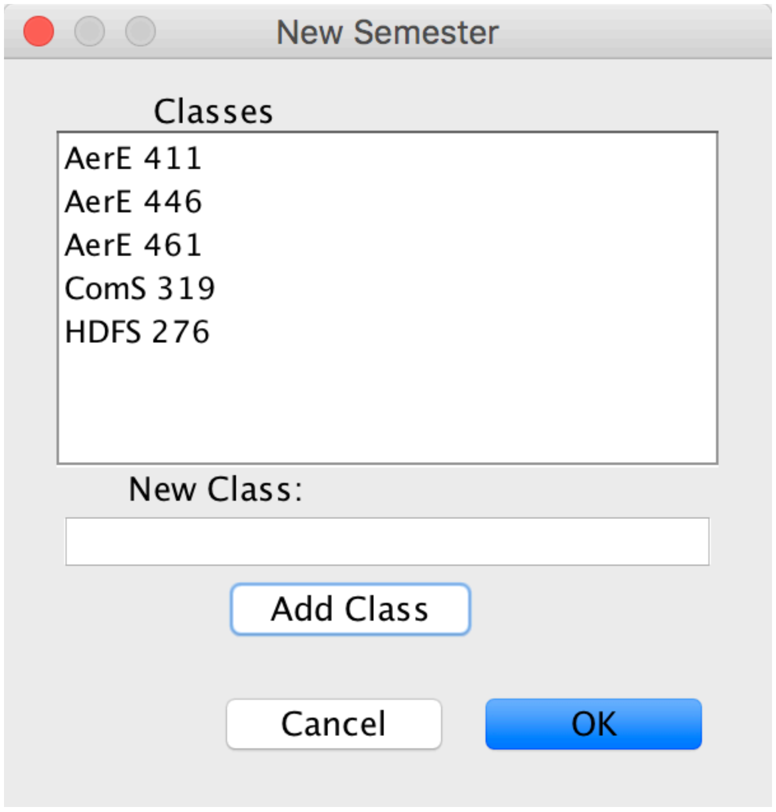


New Assignment Interface



Creating a New Semester

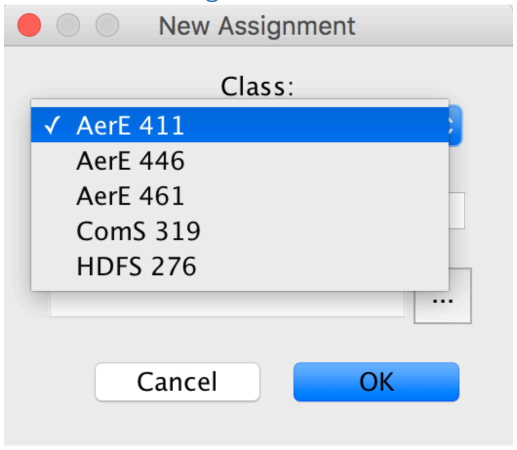
New Semester Interface

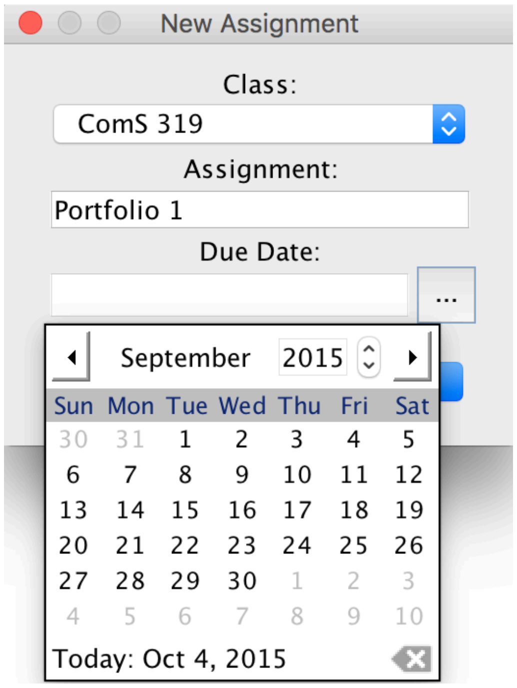


AerE 411
AerE 446
AerE 461
ComS 319
HDFS 276

Creating a New Assignment

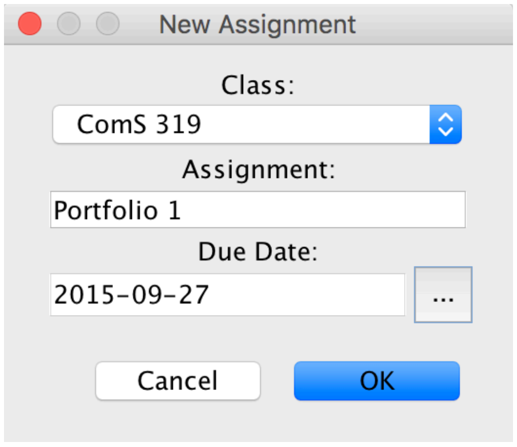
New Assignment Interface





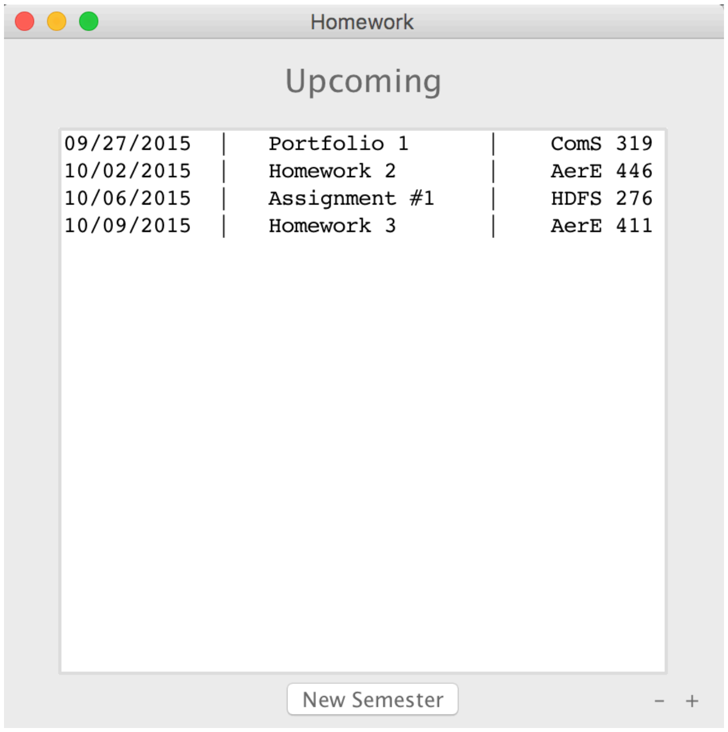
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

Today: Oct 4, 2015



Updated Homework List

Homework List Interface



```
09/27/2015  |    Portfolio 1          |     ComS 319
10/02/2015  |    Homework 2           |     AerE 446
10/06/2015  |    Assignment #1        |     HDFS 276
10/09/2015  |    Homework 3           |     AerE 411
```

User Interface Functionality

Homework List Interface

The Homework List Interface consists of JLabels, JButtons, and a JScrollPane which is populated from a JList of a custom type, assignment. The JList sorts itself based off of the due dates of the assignments it is populated with. This allows the JScrollPane to display the assignments in order of when they will be due, with the assignment with the closest due date to be displayed at the top.

New Semester Interface

The New Semester Interface contains JLabels, JButtons, a JTextField, and a JScrollPane. The user populates the JScrollPane with classes for the new semester by entering the course name in the JTextField and pressing the “Add Class” JButton. The class list is automatically sorted alphabetically by class name, and then by course number if it cannot be successfully sorted alphabetically. This adds the class to a temporary semester, but does not make any permanent changes to the current semester. Once the user is satisfied with the course list displayed in the JScrollPane, they hit the “OK” JButton and the old semester is discarded and the new semester is created. The code bellow is the action listener for the “New Semester” JButton. It has an additional action listener inside of it to allow the user to edit the temporary semester without committing permeant changes to their current semester.

```
addClass.addActionListener(new ActionListener() {

                              public void actionPerformed(ActionEvent e)
                              {
                                   if(!newClassName.getText().isEmpty()){
                                           String newClass = newClassName.getText();
                                         try {
                                                                    newClassDataModel.add(newClass);
                                                                    newClassName.setText("");
                                                            } catch (IOException e1) {
                                                                    e1.printStackTrace();
                                                            }
                                   }

                              }

                          });
                                   classListPanel.add(addClass);

                                   int result = JOptionPane.showConfirmDialog(null, classListPanel, "New Semester",
                                                   JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

                                   if (result == 0 && !newClassDataModel.classList.isEmpty()) {
                                           classDataModel.classList.clear();
                                           for(int i = 0; i < newClassDataModel.classList.size(); i++){
                                                   try {
                                                           classDataModel.add(newClassDataModel.classList.get(i));
                                                   } catch (IOException e1) {
                                                           e1.printStackTrace();
                                                   }
                                           }
                                           try {
                                                   classDataModel.write();
                                           } catch (IOException e1) {
                                                   e1.printStackTrace();
                                           }
                                           newClassDataModel.classList.clear();
                                   } else {
                                           newClassDataModel.classList.clear();
                                   }
                           }

                   });
```

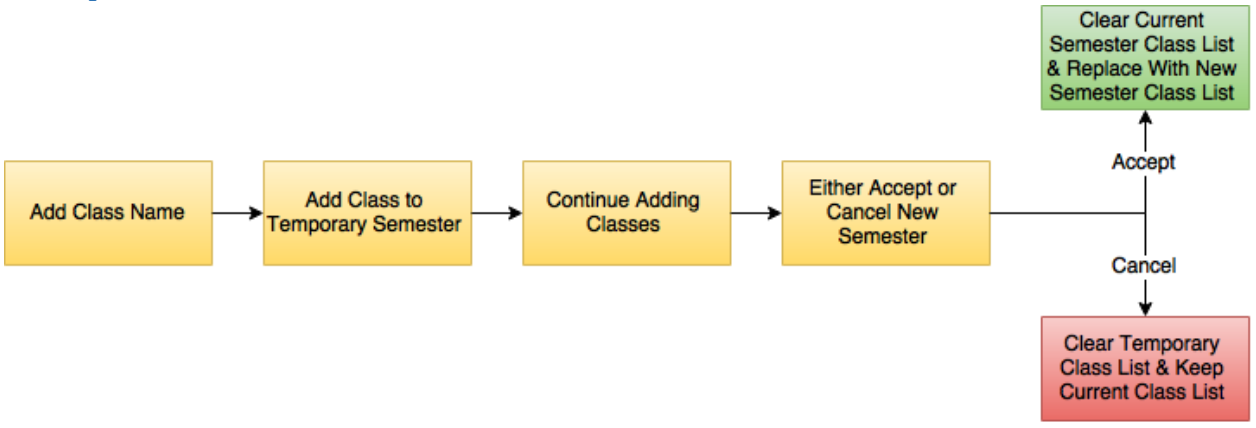
New Assignment Interface

The New Assignment Interface is made up of JLabels, JButtons, a JTextField, a JComboBox, and a JDatePicker. The JComboBox is populated from the current semester list of classes that the user has previously entered. The user cannot change the information inside of this JComboBox through the New Assignment Interface; they must perform any desired changes through the New Semester Interface. The user enters the name of the new Assignment in the "Assignment:" JTextField. To add complexity and originality to our project, we decided to use an open-source library called JDatePicker. The JDatePicker creates a small dropdown calendar that the user can use to chose their desired date, which in our case is the due date of the new assignment that they are creating. This provided a way to ensure that there would be no inconsistencies with the format that the user entered the due date in. The source for the JDatePicker we used can be found here: http://sourceforge.net/projects/jdatepicker/. The code below is the action listener for the New Assignment Interface.

```java
  btnNewButton.addActionListener(new java.awt.event.ActionListener() {
  @Override
  public void actionPerformed(java.awt.event.ActionEvent evt) {

            // Assignment Name JTextField
            JTextField assignmentNameTextField = new JTextField(5);

            // Class Selection JComboBox
            JComboBox classesComboBox = new JComboBox(classDataModel.classList.toArray());

            // Setup for drop down calendar
            UtilDateModel model = new UtilDateModel();
            Properties p = new Properties();
            p.put("text.today", "Today");
            JDatePanelImpl datePanel = new JDatePanelImpl(model,p);
            JDatePickerImpl datePicker = new JDatePickerImpl(datePanel, new DateLabelFormatter());

            // New Assignment JLabels
            JLabel classJLabel = new JLabel("Class:", SwingConstants.CENTER);
            classJLabel.setAlignmentX(CENTER_ALIGNMENT);
            JLabel assignmentJLabel = new JLabel("Assignment:", SwingConstants.CENTER);
            assignmentJLabel.setAlignmentX(CENTER_ALIGNMENT);
            JLabel dueDateJLabel = new JLabel("Due Date:", SwingConstants.CENTER);
            dueDateJLabel.setAlignmentX(CENTER_ALIGNMENT);

            // New Assignment Panel
            JPanel newAssignmentPanel = new JPanel();
            newAssignmentPanel.setLayout(new BoxLayout(newAssignmentPanel, BoxLayout.Y_AXIS));
            newAssignmentPanel.add(classJLabel);
            newAssignmentPanel.add(classesComboBox);
            newAssignmentPanel.add(assignmentJLabel);
            newAssignmentPanel.add(assignmentNameTextField);
            newAssignmentPanel.add(dueDateJLabel);
            newAssignmentPanel.add(datePicker);

            int result = JOptionPane.showConfirmDialog(null, newAssignmentPanel, "New Assignment",
                                  JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

            if (result == 0) {

                        if(classesComboBox.getSelectedItem() == null) {
                                  JOptionPane.showMessageDialog(newAssignmentPanel, "Please create a new semester.");
                        } else if(assignmentNameTextField.getText().equals("")) {
                                  JOptionPane.showMessageDialog(newAssignmentPanel, "Please enter an assignment name.");
                        } else if (datePicker.getModel().getValue() == null) {
                                  JOptionPane.showMessageDialog(newAssignmentPanel, "Please enter a due date.");
                        } else {
                                  String tempAssignmentClass = classesComboBox.getSelectedItem().toString();
                                  String tempAssignmentName = assignmentNameTextField.getText();
                                  Date tempAssignmentDueDate = (Date) datePicker.getModel().getValue();

                                  Assignment tempAssignment = new Assignment(tempAssignmentClass, tempAssignmentName, tempAssignmentDueDate);

                                  try {
                                            assignmentDataModel.add(tempAssignment);
                                  } catch (IOException e) {
                                            e.printStackTrace();
                                  }
                        }
            }
  }
});
```

Complex Issues

Adding Classes to a New Semester


Add Class Name
Add Class to Temporary Semester
Continue Adding Classes
Either Accept or Cancel New Semester
Accept
Clear Current Semester Class List & Replace With New Semester Class List
Cancel
Clear Temporary Class List & Keep Current Class List

Due Date Input Formatting

To overcome the issue of having inconsistent due date formats submitted by the user, the JDatePicker was implemented. Although this solved the user end issues of saving due dates, it did not solve our issue of saving and filtering assignments by date. We had to create a method in the assignment class that would format the date in a specific way so that it could easily be saved and read back into the program on shutdown and restart. Below is the code we used to format each assignment to a string that could be saved to a .txt file:

```
public String toStringFile() {
		DateFormat df = new SimpleDateFormat("MM/dd/yyyy");
		String tempDate = df.format(dueDate);

		return tempDate + "^" + assignmentName + "^" + className;
}
```

Below is the code we used to write each assignment to a .txt file:

```
public void write() throws IOException {
	File file = new File("homework.txt");
	FileWriter fileWriter = new FileWriter(file);
	BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
	for(int i = 0; i < assignmentList.size(); i++){
		bufferedWriter.write(assignmentList.get(i).toStringFile() + "\n");
	}
	bufferedWriter.close();
	fileWriter.close();
}
```

Below is the code we used to read each assignment from a .txt file:

```java
public void read() throws ParseException, FileNotFoundException {
	File file = new File("homework.txt");
	Scanner scanner = new Scanner(file);
	while(scanner.hasNextLine()){
		String temp = scanner.nextLine();
		String[] holder = temp.split("\\^", 3);
		DateFormat df = new SimpleDateFormat("MM/dd/yyyy");
		Date tempDate = df.parse(holder[0]);
		Assignment tempAssignment = new Assignment(holder[2], holder[1], tempDate);
		assignmentList.add(tempAssignment);
	}
	scanner.close();
}
```

Filtering Assignments by Due Date

The point of this application is to provide a user a quick and easy way to see what homework assignments they have coming up. So intuition would tell you that the assignment that is the closest to being due should show up at the top of your to-do list. So to filter the assignments by their respective due dates, we had to override the assignment compareTo method. The code below shows how we did that:

```java
@Override
public int compareTo(Assignment o) {
	int temp = this.dueDate.compareTo(o.dueDate);
	return temp;
}
```

Bloom's Taxonomy

Analysis

In this portfolio, analyzing how information could be manipulated and stored was very important for our success. We wanted this application to be very easy for the user to use, since students want to spend their time working on their homework rather than organizing their reminders list. How we implemented a ClassDataModel to keep track of the classes the user was registered for in the current semester is an example of how we analyzed the user's experience. This data model allowed the user to quickly choose which class the new assignment is for, which streamlined the process of creating a new assignment to add to their reminders list. We decided that a student would appreciate only having to fill out class information once a semester rather than every time they created a new assignment.

Evaluation

When we were evaluating our portfolio, we decided that the user experience would be improved if their homework list was sorted based on due dates, rather than assignment names like we originally had it doing. We decided that a list of upcoming assignments really wasn't that useful if the assignments weren't sorted in chronological order based on their due dates. So we decided to override the compareTo method in the Assignment class. It really was a simple change that really enhanced the value of our application.

Another aspect we decided was very important to the user experience was the look of the application. We not only wanted it to be very easy to use, but we also wanted it to look very easy to use. From the simplicity of the main Homework List Interface, to the minimal use of buttons and user provided data, the look of the application gives the user a sense of reassurance that it will be easy to use. A great example of how we improved the user experience by improving the look of the application was through our use of the JDatePicker in the New Assignment Interface. The drop-down calendar looks better, and it works better. The user doesn't need to worry about providing the application with data in the correct format. The user doesn't even need to know the exact date the assignment is due. The student might know that the assignment they are creating is due "next Friday." With the JDatePicker highlighting the current day in red, and displaying the current date at the bottom of the graphical interface, the user can easily find "next Friday" on the calendar and populate the due date field with very little thought. This evaluation really helped us streamline the creation process and really improve the user's experience.

Creation

- Our application is a completely new Java Swing project that was built from scratch and implemented an open-source library to create a better user experience.
- Created two types of data models to store information to be used by the Java Swing components.
- Designed user interface to offer simplicity and a small initial learning curve of how to use the software.
- Created methods of sorting user data to optimize application usefulness.