Lab 6: More on Java ADTs and GUIs

SE2205: Data Structures and Algorithms using Java – Winter 2015

Discussion: March 12th **Due**: Sunday March 22

Modified and reviewed by Alexandra L'Heureux (alheure2@uwo.ca) and Vasundhara Sharma (vvasundh@uwo.ca) , for more information please send an email to Alex (TA, SE2205) or Vasu (TA.SE2205)

A. Rationale and Background

In this lab we will be discussing various Java *Abstract Data Types* (ADTs) you can use in future projects. In particular we will be taking a look at **LinkedList<K>.** You will be responsible for writing 2 small applications , as well as modifying existing code to use a different ADT. First, you will write a GUI-based application which will sort, reverse, or shuffle a LinkedList. You shall modify your existing Airline/Flight/Passenger classes in order to support Linked List rather than array as well as the direct comparison of passenger objects. Finally, you will modify an existing GUI program to get you more comfortable with GUIs created without the form utility.

B. Evaluation

You will get credit for your lab when you demonstrate it and get your TA's approval. Submitting your lab work online is required; failure to do so will result in a grade of 0%. Additionally, your TA will ask some questions about your coding during demonstration and you need to articulate your idea clearly. In place of questions, the TA may also ask you to modify your program in a small way - it is expected that you have sufficient understanding of the code to do so.

Your mark will be determined as follows:
Completion: Program Demonstration of Base Requirements (50%)
Understanding: Program Demonstration of Changes or Response to Questions (50%)

Please note that you may only demonstrate the code you have previously submitted to OWL.
Please ensure that the file naming conventions are being followed. You should be able to rename your classes once they are done by right clicking on them, selecting *refactor* and *rename*.

1. Submission Instructions

Number of files : 5
Files to be submitted : All Classes
Rename these files as : your_uwo_user_name_lab06_Airline.java,
your_uwo_user_name_lab06_Passenger.java,
your_uwo_user_name_lab06_Flight.java
your_uwo_user_name_lab06_GUIManager.java
your_uwo_user_name_lab06_LetterGrade.java

C. Lab Questions

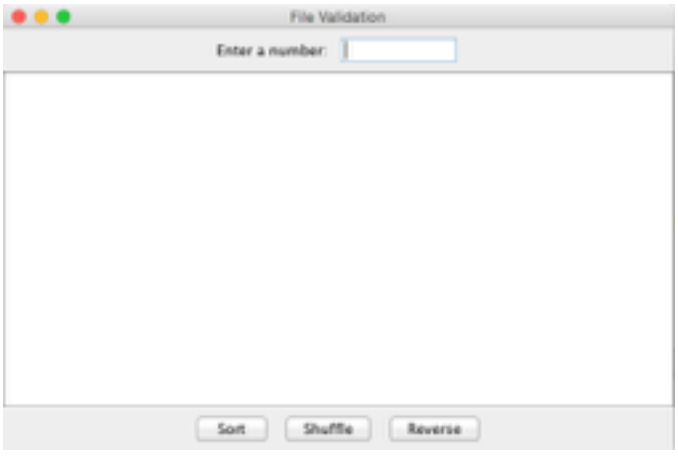
1. Sorting and Shuffling a LinkedList<K>

The first part of your lab will be to create a small application that sorts and shuffles a LinkedList of passengers (LinkedList<Passenger>). This will allow us to shuffle/reverse our list of passengers or to sort it in alphabetical order. The shuffling functionality will be given to you by the *Collections* class once you make some modifications to your existing classes. In order to facilitate the sorting and the use of the Collection utility, the following changes are required:

- You must modify your airline and flight such that instead of using arrays of passengers you use Linked List.
- You must ensure that some of your passenger class implement the Comparable interface.
    - The way you should rank passengers should be as follow:
        - In alphabetical order according to the name.
        - If the name is the same, you shall use the passenger number.
- You shall create a sort function within your GUI that takes in a Linked List of passengers, sorts them using SHELL SORT and prints the sorted result and each of the intermediate steps. The function shall return the sorted list, and the passenger list of the flight/airline should then be set to the new sorted values accordingly.
- You shall create a reverse function which reverse the order of the passenger, you should print the resulted reversed list.

a. Build your GUI

Like in previous labs, you should build a simple GUI. The GUI should give you the option of ranking and sorting passengers within a flight or within the airline. If a flight name is entered in the number field, only the passengers within that flight should be sorted, shuffled or placed in reverse order. However, if the keyword “all” is entered, all of the passengers within the airline should be classified. Below is a screenshot of what your GUI can/should look like. DO NOT use the form utility, you must write the GUI by hand.



Below is some code to get you started with the GUI. Please note that you are more than welcome to create your own GUI; this code is just to give you a base starting point

```
  public class anyClass extends JFrame {
       private LinkedList<Integer> list = new LinkedList<Integer>();
       private JTextField jtfNumber = new JTextField(8);
       private JTextArea jtaNumbers = new JTextArea();
       private JButton jbtSort = new JButton("Sort");
       private JButton jbtShuffle = new JButton("Shuffle");
       private JButton jbtReverse = new JButton("Reverse");

       public anyClass() {
         JPanel panel1 = new JPanel();
         panel1.add(new JLabel("Enter a number: "));
         panel1.add(jtfNumber);

         JScrollPane jsp = new JScrollPane(jtaNumbers);

         JPanel panel2 = new JPanel();
         panel2.add(jbtSort);
         panel2.add(jbtShuffle);
         panel2.add(jbtReverse);

         this.add(panel1, BorderLayout.NORTH);
         add(jsp, BorderLayout.CENTER);
         add(panel2, BorderLayout.SOUTH);

       jtfNumber.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
       }
     });

     jbtSort.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
       }
     });

     jbtShuffle.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
       }
     });

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

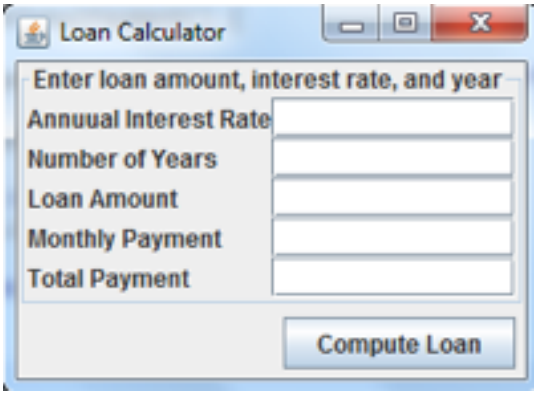
b. LinkedList<Passenger> and ListIterator<Passenger>

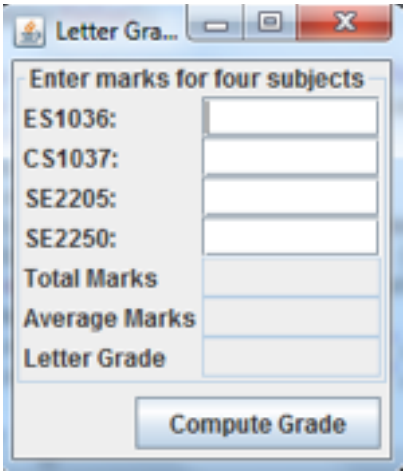
Once you have built your GUI, you need to fill out the actionListener methods. To do this, you need to make use of the LinkedList<Passenger> you created. The LinkedList<Passenger> should have the following functionality:

1. Get the appropriate passenger linked list when the user presses "enter" in the JTextField
2. Sort the passenger list when the user presses the sort button according to the sort function described above.
3. Call Collections.shuffle(list) when the user presses the shuffle button
4. Reverse the order of the passenger list when the user presses the reverse button according to the reverse function described above.
5. After all steps in 2-4, you need to display the List again in the JTextArea. To do this you need to use a ListIterator<Passenger>. `ListIterator <Passenger> iterator = list.listIterator();`

2. Modifying an existing GUI

We are going to give you a full code for a loan calculator application (see Section D), please note that this was not exhaustively error checked. You are then responsible for converting the Loan Calculator to the Letter Grade application shown below. You should then link the functionality to this application to a button on our main GUI named “Calculate Letter Grade”.





The functionality is simple: just take user input as JTextFields. The total marks, average marks, and letter grade JTextFields should be non-editable. When the user clicks Compute Grade, you should check that all the marks are filled in, then calculate the three non-editable JTextFields accordingly. This should only be seen as a calculator, therefore the inputs of this component are not required to be linked to content of the rest of the application. You can design the application to look exactly how the above picture looks; however you could also populate the labels from the course list. DO NOT use the form utility, you must write the GUI by hand.

D. Loan Calculator Source Code

```
package testpackage;
import javax.swing.*;
import javax.swing.border.TitledBorder;
```

```
import java.awt.*;
import java.awt.event.*;

public class anyClass extends JFrame {
	private JLabel jlblNumber1 = new JLabel("Annuual Interest Rate");
	private JLabel jlblNumber2 = new JLabel("Number of Years");
	private JLabel jlblNumber3 = new JLabel("Loan Amount");
	private JLabel jlblNumber4 = new JLabel("Monthly Payment");
	private JLabel jlblNumber5 = new JLabel("Total Payment");
	private JTextField jtfAnnualInterestRate = new JTextField();
	private JTextField jtfNumberOfYears = new JTextField();
	private JTextField jtfLoadAmount = new JTextField();
	private JTextField jtfMonthlyPayment = new JTextField();
	private JTextField jtfTotalPayment = new JTextField();

	private JButton computeLoan = new JButton("Compute Loan");

	anyClass() {
		JPanel panel = new JPanel(new GridLayout(5,2));
		panel.add(jlblNumber1);
		panel.add(jtfAnnualInterestRate);
		panel.add(jlblNumber2);
		panel.add(jtfNumberOfYears);
		panel.add(jlblNumber3);
		panel.add(jtfLoadAmount);
		panel.add(jlblNumber4);
		panel.add(jtfMonthlyPayment);
		panel.add(jlblNumber5);
		panel.add(jtfTotalPayment);

		panel.setBorder(new TitledBorder("Enter loan amount, interest rate,
and year"));
		JPanel panel3 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
		panel3.add(computeLoan);
		add(panel, BorderLayout.CENTER);
		add(panel3, BorderLayout.SOUTH);
		computeLoan.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				//get values from text fields
				double interest =
Double.parseDouble(jtfAnnualInterestRate.getText());
				int year = Integer.parseInt(jtfNumberOfYears.getText());
				double loanAmount =
Double.parseDouble(jtfLoadAmount.getText());
				//create a loan object
				Loan loan  = new Loan(interest, year, loanAmount);
				//display month payment and total payment
				jtfMonthlyPayment.setText(String.format("%.2f",
loan.getMonthlyPayment()));
				jtfTotalPayment.setText(String.format("%.2f",
loan.getTotalPayment()));
			}
		});
	}

	public static void main(String[] args) {
		anyClass frame = new anyClass();
		frame.pack();
		frame.setTitle("Loan Calculator");
		frame.setLocationRelativeTo(null);
```

```
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setVisible(true);
        }
}
class Loan {
        private double annualInterestRate;
        private int numberOfYears;
        private double loanAmount;
        private java.util.Date loanDate;
        /** Default constructor */
        public Loan() {
          this(2.5, 1, 1000);
        }
        /** Construct a loan with specified annual interest rate,
            number of years and loan amount */
        public Loan(double annualInterestRate, int numberOfYears,
            double loanAmount) {
          this.annualInterestRate = annualInterestRate;
          this.numberOfYears = numberOfYears;
          this.loanAmount = loanAmount;
          loanDate = new java.util.Date();
        }
        /** Return annualInterestRate */
        public double getAnnualInterestRate() {
          return annualInterestRate;
        }
        /** Set a new annualInterestRate */
        public void setAnnualInterestRate(double annualInterestRate) {
          this.annualInterestRate = annualInterestRate;
        }
        /** Return numberOfYears */
        public int getNumberOfYears() {
          return numberOfYears;
        }
        /** Set a new numberOfYears */
        public void setNumberOfYears(int numberOfYears) {
          this.numberOfYears = numberOfYears;
        }
        /** Return loanAmount */
        public double getLoanAmount() {
          return loanAmount;
        }
        /** Set a newloanAmount */
        public void setLoanAmount(double loanAmount) {
          this.loanAmount = loanAmount;
        }
        /** Find monthly payment */
        public double getMonthlyPayment() {
          double monthlyInterestRate = annualInterestRate / 1200;
          double monthlyPayment = loanAmount * monthlyInterestRate / (1 -
            (Math.pow(1 / (1 + monthlyInterestRate), numberOfYears * 12)));
          return monthlyPayment;
        }
        /** Find total payment */

        public double getTotalPayment() {

          double totalPayment = getMonthlyPayment() * numberOfYears * 12;

          return totalPayment;
```

```
  }
  /** Return loan date */
  public java.util.Date getLoanDate() {

    return loanDate;
  }
}
```

E. Learning Checkpoint

This lab should have introduced you to these concepts of Java:

1. More ADTs, including: LinkedList<K>
2. More sorting, including: Shell sort
3. More work with GUIs
4. Understanding an existing GUI to convert to a new GUI