Isaac Ray Shoebottom
CS 1083 (FR02B)
Assignment 10
3429069

Source Code:

Part A) Student.java:

```
/**
 * Represents a student.
 * @author Isaac Shoebottom (3429069)
*/
public class Student implements Comparable<Student>{

	/**
	The last name of the student.
	*/
	private final String lastName;

	/**
	The first name of the student.
	*/
	private final String firstName;

	/**
	The student's ID number.
	*/
	private final long id;



	/**
	Constructs a student given their first and last name, and student
ID.
	@param firstNameIn The first name of the student.
	@param lastNameIn The last name of the student.
	@param idIn The student's ID number.
	*/
	public Student(String firstNameIn, String lastNameIn, int idIn){
		firstName = firstNameIn;
		lastName = lastNameIn;
		id = idIn;
	}

	/**
	Prints all the information about the student.
	@return The student's information.
	*/
	public String toString(){
		return lastName + ", " + firstName + " (" + id + ")";
	}



	/**
	 * CompareTo method compares the two students on last name, then
first name, and then on id
	 * if lastname/firstname is lexicographically greater than
o.lastname return positive int
	 * we return a positive int when the student o is less than the
current object
	 * we want id's to be sorted smallest to largest so an object is
greater when it's id is greater
	 * @param o The student being taken in
	 * @return 0 if students are equal, 1 if the student being taken in
is lesser, and -1 if the student being taken in is greater.
	 */
	@Override
	public int compareTo(Student o) {
```

```
            if (lastName.compareToIgnoreCase(o.lastName) > 0)
                  return 1;
            if (lastName.compareToIgnoreCase(o.lastName) < 0)
                  return -1;
            if (firstName.compareToIgnoreCase(o.firstName) > 0)
                  return 1;
            if (firstName.compareToIgnoreCase(o.firstName) < 0)
                  return -1;
            return Long.compare(id, o.id);
      }
}
```

Part B) ClassList.java:

```
import java.util.NoSuchElementException;


/**

 * Note: when implementing this doubly linked list, I though of "end" to
be the end as in it is similar to a stack

 * and front refers to the top element (or the first). The name might
not be the most sensible if you think of the front

 * like like front of the linked list, but the code works.

 * @author Isaac Shoebottom (3429069)

 */


public class ClassList {


    /**

     * The bottom element of the linked list

     */

    private StudentNode end;

    /**

     * The top element of the linked list

     */

    private StudentNode front;

    /**

     * The size of the linked list.

     */

    private int size;


    /**

     * Method to add a student to the linked list in sorted order

     * @param studentIn The student ot be added.

     */
```

```
public void add(Student studentIn) {
    StudentNode newNode = new StudentNode(studentIn);

    if (end == null) {
        end = newNode;
        front = newNode;
    }
    else if(newNode.data.compareTo(end.data) <= 0) {
        newNode.next = end;
        end.prev = newNode;
        end = newNode;
    }
    else if(newNode.data.compareTo(front.data) >= 0) {
        newNode.prev = front;
        front.next = newNode;
        front = newNode;
    }
    else {
        StudentNode current = end;
        while (newNode.data.compareTo(current.data) > 0) {
            if (current != front)
                current = current.next;
            else {
                break;
            }
        }
        newNode.next = current;
        newNode.prev = current.prev;
        current.prev.next = newNode;
        current.prev = newNode;
        }
        size++;
    }

    /**
     * Method to get the size of the linked list, also known as the
number of students
     * @return The size of the linked list
     */
    public int getNumStudents() {
        return size;
    }

    /**
     * Method to get a reversed list in array form.
     * @return The reversed array representation of the linked list
     */
    public Student[] getReversedList() {
        Student[] students = new Student[size];
        StudentNode current = front;
        for(int i = 0; i < size; i++) {
            students[i] = current.data;
            current = current.prev;
        }
        return students;
    }

    /**
     * Method to remove a student from the linked list
     * @param studentOut The student to be removed
     * @throws NoSuchElementException When the student is not found
     */
    public void remove(Student studentOut) throws NoSuchElementException
{
        if(studentOut.compareTo(end.data) == 0) {
            end = end.next;
            size--;
            return;
        }
        else if(studentOut.compareTo(front.data) == 0) {
            front = front.prev;
            size--;
            return;
        }
        else {
            StudentNode current = end;
            for (int i = 0; i < size; i++) {
                if (current.data.compareTo(studentOut) == 0) {
                    current.prev.next = current.next;
                    current.next.prev = current.prev;
                    size--;
                    return;
                }
                current = current.next;
            }
        }
        throw new NoSuchElementException("There was no such student in
the list");
    }
    /**
     * Method to represent the contents of the linked list
     * @return A string representing the linked list
     */
    public String toString() {
        StringBuilder str = new StringBuilder("[");
        StudentNode current = end;
        for (int i = 0; i < size; i++) {
            str.append(current.data.toString()).append(", ");
            current = current.next;
        }
        str = new StringBuilder(str.substring(0, str.length() - 2));
        str.append("]");
        return str.toString();
    }

    /**
     * A node representation of a student, containing the next and
previous student
     * @author Isaac Shoebottom (3429069)
     */
    public static class StudentNode {
        /**
         * The data Student object contained in the node
         */
        Student data;
        /**
         * The next student node in the chain
         */
        StudentNode next;
        /**
         * The previous student node in the chain
         */
        StudentNode prev;



        /**
         * The constructor for the student node
         * @param dataIn Takes in a student object
         */
        StudentNode(Student dataIn) {
            data = dataIn;
            next = null;
            prev = null;
        }

        /**
         * Returns what the student node contains for data
         * @return The data contained within the student node
         */
        @Override
        public String toString() {
            return data.toString();
        }
    }
}
```

Part C) LinkedListDriver.java:

```
import java.util.NoSuchElementException;


/**

 * Driver for the linked list

 * @author Isaac Shoebottom (3429069)

 */


public class LinkedListDriver {

    @SuppressWarnings("SpellCheckingInspection")

    public static void main(String[] args) {

        Student isaac = new Student("Isaac", "Shoebottom", 3429069);

        Student studentOne = new Student("Student", "One", 1);

        Student studentAfterMe = new Student("Student", "After",
9999999);

        Student test1 = new Student("Isaac", "Shoebottom", 3429063);

        Student test2 = new Student("Student", "After", 1);


        Student sequence1 = new Student("a", "a", 100);

        Student sequence2 = new Student("a", "a", 102);

        Student sequence3 = new Student("a", "a", 101);


        Student test3 = new Student("Adam", "Test", 89);

        Student test4 = new Student("Lam", "Gorsk", 8990);

        Student test5 = new Student("Zzzzz", "Zzzzz", 100000000);



        ClassList classList = new ClassList();

        classList.add(isaac);

        classList.add(studentOne);
        classList.add(studentAfterMe);
        classList.add(test1);
        classList.add(test2);
        classList.add(test2);

        classList.add(sequence1);
        classList.add(sequence2);
        classList.add(sequence3);

        System.out.println(classList.toString());
        classList.remove(isaac);
        classList.remove(test2);
        classList.remove(sequence1);
        System.out.println(classList.toString());



        classList.add(test3);
        classList.add(test4);
        classList.add(test5);

        //would this be allowed?

//System.out.println(Arrays.toString(classList.getReversedList()));
        StringBuilder reversedList = new StringBuilder("[");
        Student[] students = classList.getReversedList();
        for(int i = 0; i < classList.getNumStudents(); i++) {
            reversedList.append(students[i].toString()).append(", ");
        }
        reversedList = new StringBuilder(reversedList.substring(0,
reversedList.length() - 2));
```

```
        reversedList.append("]");
        System.out.println(reversedList);

        try {
            classList.remove(isaac);
        }catch (NoSuchElementException exception) {
            System.out.println(exception.getMessage());
        }

        System.out.println(classList.toString());
        System.out.println(classList.getNumStudents());

    }
}
```

Output of Driver:

```
[a, a (100), a, a (101), a, a (102), After, Student (1), After, Student
(1), After, Student (9999999), One, Student (1), Shoebottom, Isaac
(3429063), Shoebottom, Isaac (3429069)]

[a, a (101), a, a (102), After, Student (1), After, Student (9999999),
One, Student (1), Shoebottom, Isaac (3429063)]

[Zzzzz, Zzzzz (100000000), Test, Adam (89), Shoebottom, Isaac (3429063),
One, Student (1), Gorsk, Lam (8990), After, Student (9999999), After,
Student (1), a, a (102), a, a (101)]

There was no such student in the list

[a, a (101), a, a (102), After, Student (1), After, Student (9999999),
Gorsk, Lam (8990), One, Student (1), Shoebottom, Isaac (3429063), Test,
Adam (89), Zzzzz, Zzzzz (100000000)]

9
```