Project Documentation

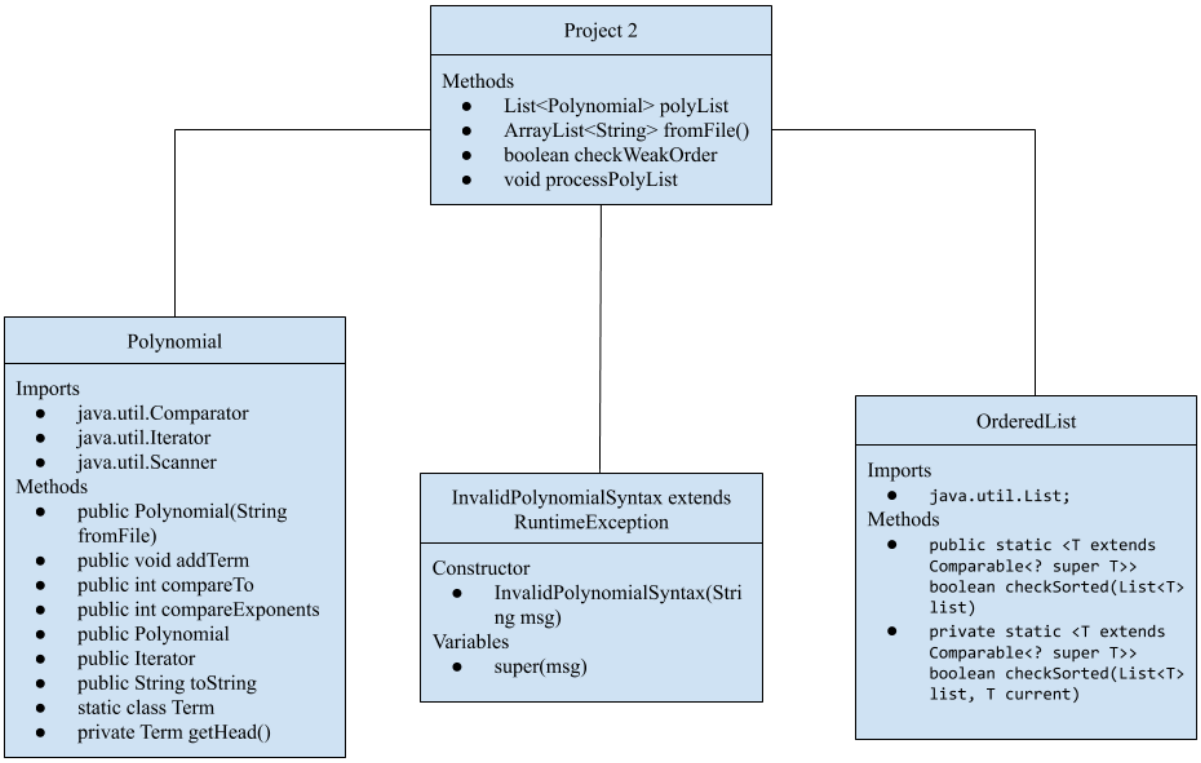
Name: Attiqa Sheikh

Assignment: Project 2

Date: June 15, 2020

Problem Statement: Write a program that examines a file of polynomials and determines whether the polynomials in that file are in strictly ascending order using two different methods of comparison.

Design:



Code:

```
/*
 * Dev: Attiqa Sheikh
 * File: Polynomial.java
 * Date: June 15, 2020
 * Description: Defines an individual polynomial.
 */
import java.util.Comparator;
import java.util.Iterator;
import java.util.Scanner;
public class Polynomial implements Iterable<Polynomial.Term>, Comparable<Polynomial>
{

	Comparator<Polynomial> compare;
	private Term head;

	//Polynomial constructor
	public Polynomial(String fromFile) {
		head = null; // explicitly stating for clarity
		Scanner termScanner = new Scanner(fromFile);
		try{
			while(termScanner.hasNext()){
				addTerm(termScanner.nextDouble(), termScanner.nextInt());
			}
		} catch (Exception ex){
			System.out.println(ex.getLocalizedMessage());
			throw new InvalidPolynomialSyntax("Incorrect Syntax. Check
inputs.");
		}
	}
	//addTerm method
	public void addTerm(double coefficient, int exponent ){
		if (exponent < 0){
			throw new InvalidPolynomialSyntax("No negative exponents. Check
inputs.");
		}
		Term current = head;
		if(current == null){ // then Polynomial is empty
			head = new Term(coefficient, exponent);
			head.next = null;
		} else { //find end by looping to null next link
			while(current.next != null){
				current = current.next;
			}
			current.next = new Term(coefficient, exponent);
		}
	}
	//Overridden compareTo method
	@Override
	public int compareTo(Polynomial otherPoly) {
		Term thisCurrent = this.head;
		Term otherCurrent = otherPoly.head;

		while (thisCurrent != null && otherCurrent != null){
```

```
                    //positive if this is larger, negative otherwise
                    if (thisCurrent.getExponent() != otherCurrent.getExponent()){
                          return thisCurrent.getExponent() -
otherCurrent.getExponent();
                          //casting to an int truncates, so simple checking for
larger
                    }else if(thisCurrent.getCoefficient() !=
otherCurrent.getCoefficient()) {
                          if(otherCurrent.getCoefficient()>
thisCurrent.getCoefficient()){
                                return -1;
                          }else if(otherCurrent.getCoefficient()<
thisCurrent.getCoefficient()){
                                return +1;
                          }
                    }// resetting the values outside of the loop
                    thisCurrent = thisCurrent.getNext();
                    otherCurrent = otherCurrent.getNext();
              }//if both are null, and at this point, they are equal, ret zero
              if (thisCurrent == null && otherCurrent == null){
                    return 0;
              }//this would be the case of one with more terms than other
              if (thisCurrent == null){
                    return -1;
              }else {
                    return +1;
              }
        }
        //compareExponents methods
        public int compareExponents(Polynomial poly2) {
              Term thisPolyTerm = this.head;
              Term otherPolyTerm = poly2.head;
              while(thisPolyTerm != null && otherPolyTerm != null) {
                    if (thisPolyTerm.getExponent() != otherPolyTerm.getExponent()) {
                          return thisPolyTerm.getExponent() -
otherPolyTerm.getExponent();
                    }else {
                          thisPolyTerm = thisPolyTerm.getNext();
                          otherPolyTerm = otherPolyTerm.getNext();
                    }
              }if(thisPolyTerm == null && otherPolyTerm == null){
                    return 0;
              }
              if (otherPolyTerm == null){
                    return +1;
              }else {
                    return -1;
              }
        }
        public Polynomial() {
              compare = (Polynomial poly1, Polynomial poly2) ->
poly1.compareExponents(poly2); }
        public Polynomial(Comparator<Polynomial> compare){
              this.compare = compare;
              }
```

```java
	//Iterator method
	@Override
	public Iterator<Term> iterator() {
		return new Iterator() {
			private Term current = getHead();
			@Override
			public boolean hasNext() {
				return current != null && current.getNext() != null;
			}
			@Override
			public Term next() {
				Term data = current;
				current = current.next;
				return data;
			}
		};
	}
	//toString Method
	@Override
	public String toString() {
		StringBuilder expressionBuilder = new StringBuilder();
		//first check head to avoid +1x^3 +3x^2
		if (head.coefficient > 0){
			expressionBuilder.append(head.toString());
		}else{
			expressionBuilder.append(" - ").append(head.toString());
		}// then check the other nodes if they are not null
		for(Term tmp = head.next; tmp != null; tmp = tmp.next) {
			if (tmp.coefficient < 0) {
				expressionBuilder.append(" - ").append(tmp.toString());
			} else {
				expressionBuilder.append(" + ").append(tmp.toString());
			}
		}
		return expressionBuilder.toString();
	}
	//nested Term class
	static class Term{
		private double coefficient;
		private int exponent;
		private Term next;
		private Term(double c, int e) {
			coefficient = c;
			exponent = e;
			next = null; // explicitly setting to null
		}
		private int getExponent(){
			return this.exponent;
		}
		private double getCoefficient(){
			return this.coefficient;
		}
		private Term getNext(){
			return next;
		}
```

```java
	@Override
	public String toString(){
		String termString = String.format("%.1f", Math.abs(coefficient));
		if (exponent == 0) { //no variable
			return termString;
		}else if(exponent == 1){ // do not display exponent
			return termString + "x";
		} else{ // display exponent after variable
			return termString + "x^" + exponent;
		}
	}
	private Term getHead() {
		return head;
	}

/*
 * Dev: Attiqa Sheikh
 * File: InvalidPolynomialSyntax.java
 * Date: June 15, 2020
 * Description: Defines an unchecked exception
*/
public class InvalidPolynomialSyntax extends RuntimeException {
	InvalidPolynomialSyntax(String msg){super(msg);
	}
}

/*
 * Dev: Attiqa Sheikh
 * File: OrderedList.java
 * Date: June 15, 2020
 * Description: Determines whether a List object in is ascending order
*/
import java.util.List;
public class OrderedList {
	public static <T extends Comparable<? super T>> boolean checkSorted(List<T>
list){
		boolean isSorted = true;
		for(int i = list.size()-1; i > 0 ; i--){
			T current = list.get(i);
			if(!checkSorted(list, current)){
				isSorted = false;
			}
		}
		return isSorted;
	}
	private static <T extends Comparable<? super T>> boolean checkSorted(List<T>
list, T current) {
		T currentValue = list.get(list.indexOf(current));
		T nextValue = list.get(list.indexOf(current) - 1);
		if (nextValue != null) {
			return currentValue.compareTo(nextValue) >= 0; //if next index is
larger return false
		}
		return true;
	}
```

```java
}
/*
 * Dev: Attiqa Sheikh
 * File: Project2.java
 * Date: June 15, 2020
 * Description: Main method
*/
import javax.swing.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class Project2 {
	private static List<Polynomial> polyList = new ArrayList<>();
	public static void main(String[] args) {
		processPolyList();
	}
	//fromFile method
	public static ArrayList<String> fromFile() {
		//Create ArrayList and JFileChooser
		ArrayList<String> expressionList = new ArrayList<>();
		JFileChooser chooser = new JFileChooser();
		//Show both directories and files
		chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
		//use current directory for ease
		chooser.setCurrentDirectory(new File(System.getProperty("user.dir")));
		int response = chooser.showOpenDialog(null);
		if (response == JFileChooser.APPROVE_OPTION){
			File file = chooser.getSelectedFile();
			try {
				Scanner fileIn = new Scanner(file);
				if (file.isFile()){
					while (fileIn.hasNextLine()){
						String expression = fileIn.nextLine();
						expressionList.add(expression);
					}
				}
			}catch (NoSuchElementException nse){

	JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),"File is Empty");
			}catch(FileNotFoundException fne){

	JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),"File Not Found");
			}
		}
		return expressionList;
	}
	//checkWeakOrder method
	public static boolean checkWeakOrder( List<Polynomial> polyList){
		boolean isWeakOrder = true;
		Polynomial previous = polyList.get(polyList.size()-1);
		for(int i = polyList.size()-2; i > 0; i--){
			if (previous.compareExponents(polyList.get(i)) < 0){
				isWeakOrder = false;
			}
```

```
            }
            return isWeakOrder;
        }
        //processPolyList method
        public static void processPolyList(){
            try {
                ArrayList<String> a = fromFile();
                for (String element : a) {
                    Polynomial p = new Polynomial(element);
                    System.out.println(p);
                    polyList.add(p);
                }
            }catch (InvalidPolynomialSyntax ex){

        JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),ex.getMessage());
            }
            /* Call to check sorted for the Strong order check */
            System.out.println("Strong Ordered: " +
OrderedList.checkSorted(polyList));
            /* Check for Weak order (exponents only) */
            System.out.println("Weak Ordered: " + checkWeakOrder(polyList));
        }
}
```

Testing:

Test #1

Data: File TestCase.txt includes the following three sets of data:

4.0 3 2.5 1 8.0 0

5.0 4 5.0 0

4.5 4 5.7 2 8.6 0

Test Cases: TestCase.txt is input to program to return results of polynomial.

Expected Results:

```
4.0x^3 + 2.5x + 8.0
5.0x^4 + 5.0
4.5x^4 + 5.7x^2 + 8.6
Strong Ordered: false
Weak Ordered: true
```

Remarks: Testing to see if the program works appropriately

Test #2

Data: File TestCase2.txt includes the following three sets of data:

1 4 2.5 3 5.6 2 8.6 1 9 0

A 1 F 5 6 3

Test Cases: TestCase2.txt is input into program to show results of polynomial and to test the function of InvalidPolynomialSyntax class.

Expected Results:

```
1.0x^4 + 2.5x^3 + 5.6x^2 + 8.6x + 9.0
null
JOptionPane window with message “Invalid Syntax. Check Inputs.”
```

Remarks: Program does not accept anything other than numbers, if invalid syntax is entered, the user is told that their entry is invalid and to try again.

Test #3

Data: File TestCase3.txt includes the following three sets of data:

1 3 2 2 6 1 8 0

Test Cases: TestCase3.txt is input to program to test whether the program correctly recognizes strong and weak ordered polynomial.

Expected Results:

```
1.0x^3 + 2.0x^2 + 6.0x + 8.0
Strong Ordered: true
Weak Ordered: true
```

Outputs:

	Input	Output
Test #1	TestCase - Notepad File Edit Format View Help 4.0 3 2.5 1 8.0 0 5.0 4 5.0 0 4.5 4 5.7 2 8.6 0	Console <terminated> Project2 [Java Applica 4.0x^3 + 2.5x + 8.0 5.0x^4 + 5.0 4.5x^4 + 5.7x^2 + 8.6 Strong Ordered: false Weak Ordered: true
Test #2	TestCase2 - Notepad File Edit Format View Help 1 4 2.5 3 5.6 2 8.6 1 9 0 A 1 F 5 6 3	Message Incorrect Syntax. Check inputs. OK Console Project2 [Java Application] C:\Program Files\Java\jre1. 1.0x^4 + 2.5x^3 + 5.6x^2 + 8.6x + 9.0 null
Test #3	TestCase3 - Notepad File Edit Format View Help 1 3 2 2 6 1 8 0	Console <terminated> Project2 [Java Application] C 1.0x^3 + 2.0x^2 + 6.0x + 8.0 Strong Ordered: true Weak Ordered: true

Reflection: The most significant points of the assignment to me was designing it in a way that it was user friendly and in a way that if an incorrect input were to be put into the program, then it would throw an exception that would notify the user. For some reason exceptions have been a challenge for me, but after this project I've done better and learned more on exceptions. I had a

few problems where my code wasn’t running appropriately, and I had to go through each line to make sure I didn’t misspell something or misplace a line of code. Another challenge for me was figuring out how to put the numbers in ascending order.