Homework 2

Code

```
// Homework2.java
package com;

import containers.SinglyLinkedList;

public class Homework2 {

	// Implement method to insert value coefficient to polynomial at the end
	static void appendTerm(SinglyLinkedList<Double> polynomial, Double coefficient) {
		polynomial.insertTail(coefficient);
	}


	// Implement method to print the polynomial in proper format
	static void display(SinglyLinkedList<Double> polynomial) {
		SinglyLinkedList<Double>.Element elem = polynomial.getHead();
		// print the polynomial term by term
		for (int index = polynomial.getSize()-1; index >= 0; index--) {
			// check if it is the last term
			if (index == 0) {
				// check pre-conditions
				if (elem.getData()!=0) {
					System.out.print(elem.getData());
				}
			}
			// if it is not the last term
			else {
				// check pre-conditions
				if (elem.getData()!=0) {
					System.out.print(elem.getData()+"x^"+index);
					// add plus sign
					if(elem.getNext().getData()>0) {
						System.out.print("+");
					}
				}
```

```
                }

                elem = elem.getNext();
            }
            System.out.print("\n");
    }

    // Implement method to evaluate the polynomial for given value and return result
    static Double evaluate(SinglyLinkedList<Double> polynomial, Double x) {
            SinglyLinkedList<Double>.Element elem = polynomial.getHead();
            double sum = 0;
            // calculate the answer term by term
            for(int index = (polynomial.getSize()-1); index>=0; index--) {
                sum += Math.pow(x, index)*elem.getData();
                elem = elem.getNext();
            }
            return sum;
    }

    // Test for creating the first polynomial
    static SinglyLinkedList<Double> testCreate1() {
            SinglyLinkedList<Double> polynomial = new SinglyLinkedList<Double>();
            appendTerm(polynomial, 1.0);
            appendTerm(polynomial, 1.0);
            return polynomial;
    }
    // Test for creating the second polynomial
    static SinglyLinkedList<Double> testCreate2() {
            SinglyLinkedList<Double> polynomial = new SinglyLinkedList<Double>();
            appendTerm(polynomial, 1.0);
            appendTerm(polynomial, 0.0);
            appendTerm(polynomial, -1.0);
            return polynomial;
    }

    // Test for creating the third polynomial
    static SinglyLinkedList<Double> testCreate3() {
            SinglyLinkedList<Double> polynomial = new SinglyLinkedList<Double>();
            appendTerm(polynomial, -3.0);
            appendTerm(polynomial, 0.5);
```

```
            appendTerm(polynomial, -2.0);
            appendTerm(polynomial, 0.0);
            return polynomial;
        }

        // Test for creating the fourth polynomial
        static SinglyLinkedList<Double> testCreate4() {
            SinglyLinkedList<Double> polynomial = new SinglyLinkedList<Double>();
            appendTerm(polynomial, -0.3125);
            appendTerm(polynomial, 0.0);
            appendTerm(polynomial, -9.915);
            appendTerm(polynomial, -7.75);
            appendTerm(polynomial, -40.0);
            return polynomial;
        }

        // Test for displaying polynomials
        static void testDisplay(SinglyLinkedList<Double> polynomial) {
            display(polynomial);
        }

        // Test for evaluating polynomials
        static void testEvaluate(SinglyLinkedList<Double> polynomial, Double x) {
            Double answer = evaluate(polynomial, x);
            System.out.printf("The output when x = %f is %.6f. \n", x, answer);
        }

        // Perform testing demonstration
        public static void main(String[] args) {
            testDisplay(testCreate1());
            testEvaluate(testCreate1(), 1.0);
            testDisplay(testCreate2());
            testEvaluate(testCreate2(), 2.03);
            testDisplay(testCreate3());
            testEvaluate(testCreate3(), 05.0);
            testDisplay(testCreate4());
            testEvaluate(testCreate4(), 123.45);
        }
}
```

Output

1.0x^1+1.0

The output when x = 1.000000 is 2.000000.

1.0x^2-1.0

The output when x = 2.030000 is 3.120900.

-3.0x^3+0.5x^2-2.0x^1

The output when x = 5.000000 is -372.500000.

-0.3125x^4-9.915x^2-7.75x^1-40.0

The output when x = 123.450000 is -72731671.686258.