Subject code

CSE228

MINI PROJECT FINAL REPORT

Polynomial Evaluator using LinkedList

Submitted by

Shahana T

Registration No:

12219504 Roll

No

RK22UNA05

Submitted to

Shubham Sharma (64339)

Acknowledgement:

I would like to extend my sincere thanks to the Lovely School of Computer Science and Engineering for providing me with the opportunity to fulfil my wish and achieve my goal. I am grateful to **Shubham Sharma** sir for providing me with the opportunity to undertake this project and for providing me with all the necessary facilities. I am highly thankful to sir for his active support, valuable time and advice, whole-hearted guidance, sincere cooperation, and pain-taking involvement during the study and in completing the assignment of preparing the said project within the time stipulated.

Lastly, I am thankful to all those, particularly the various friends, who have been instrumental in creating a proper, healthy, and conducive environment and including new and fresh innovative ideas for me during the project. Without their help, it would have been extremely difficult for me to prepare the project in a timebound framework.

Shahana T

12219504

Table of Contents:

Introduction:

Polynomials are mathematical expressions that consist of variables and coefficients. They can be used to represent a wide range of relationships between variables, including linear, quadratic, cubic, and higher-order relationships. Polynomials are used in many different fields of science and engineering, including mathematics, physics, engineering, and computer science.

Polynomial evaluation is the process of calculating the value of a polynomial for a given value of **x**. This is a fundamental task in many scientific and engineering applications. For example, polynomial evaluation is used in physics to calculate the trajectory of a projectile, in engineering to design control systems, and in computer science to develop algorithms for image processing and machine learning.

LinkedList are a type of data structure that can be used to store and manipulate lists of data. They are particularly well-suited for storing and manipulating polynomials, as they can be used to represent the terms of a polynomial in a linked list.

This project implements a **Polynomial Evaluator using a LinkedList** data structure in Java. This project provides a tool that can efficiently evaluate polynomials for a given value of **x**. The significance of this project lies in its potential applications in mathematics, physics, engineering, and computer science.

Objectives and Scope:

The primary objectives of this project are as follows:

1. Create a **Polynomial** class to represent polynomials using a LinkedList.
2. Implement a method **evaluate(double xValue)** that evaluates the polynomial for a given **xValue**.
3. Calculate the degree and number of non-zero terms in the polynomial.
4. Utilize built-in Java methods, such as **Math.pow()**, for efficient polynomial evaluation.

The scope of this project is limited to the creation of the **Polynomial** class and the implementation of polynomial evaluation. The class will be designed to handle polynomials with real coefficients.

Application Tools:

1. Programming Language

- Java: The primary programming language used for implementing the polynomial evaluator. Java provides the core functionality for creating the Polynomial class and performing polynomial evaluation.

2. Integrated Development Environment (IDE)

- IntelliJ IDEA: An Integrated Development Environment used for Java programming. IntelliJ IDEA provides a user-friendly interface for writing, testing, and running Java code. It greatly facilitates the development and debugging of the Polynomial class and the polynomial evaluator.

3. Data Structure

- LinkedList: A data structure available in Java for storing the coefficients of the polynomial. A LinkedList is used to represent the polynomial terms as nodes in a linked list, providing efficient manipulation and traversal capabilities.

Methodology:

1. **Creating the Polynomial Class**: We define the **Polynomial** class, which includes the LinkedList to store coefficients, and implement methods to calculate the degree and number of terms.
2. **Evaluating the Polynomial**: The **evaluate (double xValue)** method uses the LinkedList to calculate the value of the polynomial for a given **xValue**. It iterates through the list, computes each term, and sums them to obtain the final result.
3. **Testing and Validation**: We thoroughly test the **Polynomial** class using various polynomials and **x** values to ensure accurate evaluation.

Flowchart:

Start

1. Ask the user for the degree of the polynomial

Prompt the user for the degree of the polynomial, which determines the number of terms.

2. Initialize the polynomial object

Create a Polynomial object, setting its degree and number of terms to 0.

3. Create a linked list to store the terms of the polynomial

Establish a linked list to store polynomial terms, holding coefficients and exponents.

4. Loop through the degree of the polynomial - Iterate through the polynomial degree to add terms.

~ Ask the user for the coefficient and exponent of the current term

Collect user input for the coefficient and exponent of each term.

~Add the term to the linked list

Create a new node in the linked list to store coefficient and exponent for the term.

4. Check if the degree of the polynomial and the exponent of the first term match

Verify if the degree of the polynomial matches the exponent of the first term; display an error and exit if they don't match.

5. Evaluate the polynomial

Calculate the polynomial's value for a specified variable value.

6. Print the result

Display the computed polynomial value to the console.

End]

Screenshots of Execution:

CODE OUTPUT:

```
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
Enter the degree of the polynomial:
3
Enter the number of terms in the polynomial:
3
Enter the coefficient of term 1:
4
Enter the exponent of term 1:
3
Enter the coefficient of term 2:
2
Enter the exponent of term 2:
2
Enter the coefficient of term 3:
1
Enter the exponent of term 3:
1
The polynomial equation is:
+4.0x^3+2.0x^2+1.0x^1
Enter the value of x for which you want to evaluate the polynomial:
2
The result of evaluating the polynomial at x = 2.0 is: 42.0

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk-1.8\bin\java.exe" ...
Enter the degree of the polynomial:
4
Enter the number of terms in the polynomial:
3
Enter the coefficient of term 1:
2
Enter the exponent of term 1:
3
Error: The degree of the polynomial and the exponent of the current term do not match.

Process finished with exit code 0
```

CODE:

Polynomial.java

```java
import java.util.Scanner;

public class Polynomial {

    6 usages
    private Node head;
    4 usages
    private int degree;
    3 usages
    private int numTerms;

    1 usage
    public Polynomial() {
        this.head = null;
        this.degree = 0;
        this.numTerms = 0;
    }

    1 usage
    public void addTerm(double coefficient, int exponent) {
        if (coefficient == 0) {
            return;
        }

        Node newNode = new Node(coefficient, exponent);

        if (head == null) {
            head = newNode;
        }

```

Polynomial.java

```
26          else {
27              Node current = head;
28              while (current.next != null) {
29                  current = current.next;
30              }
31              current.next = newNode;
32          }
33
34          if (exponent > degree) {
35              degree = exponent;
36          }
37
38          numTerms++;
39      }
40
        1 usage
41      public double evaluate(double xValue) {
42          double result = 0;
43          Node current = head;
44
45          while (current != null) {
46              result += current.coefficient * Math.pow(xValue, current.exponent);
47              current = current.next;
48          }
49
50          return result;
51      }
52
        no usages
53      public int getDegree() {
54          return degree;
55      }
56
```

Polynomial.java

```
57      public int getNumTerms() {
58          return numTerms;
59      }
        7 usages
60      private class Node {
            4 usages
61          double coefficient;
            3 usages
62          int exponent;
            6 usages
63          Node next;
            1 usage
64          public Node(double coefficient, int exponent) {
65              this.coefficient = coefficient;
66              this.exponent = exponent;
67              this.next = null;
68          }
69      }
70      public static void main(String[] args) {
71          Scanner scanner = new Scanner(System.in);
72
73          System.out.println("Enter the degree of the polynomial: ");
74          int degree = scanner.nextInt();
75
76          System.out.println("Enter the number of terms in the polynomial: ");
77          int numTerms = scanner.nextInt();
78
79          Polynomial polynomial = new Polynomial();
80
81          for (int i = 0; i < numTerms; i++) {
82              System.out.println("Enter the coefficient of term " + (i + 1) + ": ");
83              double coefficient = scanner.nextDouble();
84
```

Polynomial.java

```
            System.out.println("Enter the exponent of term " + (i + 1) + ": ");
            int exponent = scanner.nextInt();

            if (degree != exponent) {
                System.out.println("Error: The degree of the polynomial and the exponent of the current term do not match.");
                return;
            }

            polynomial.addTerm(coefficient, exponent);
        }

        System.out.println("The polynomial equation is: ");
        Node current = polynomial.head;
        while (current != null) {
            if (current.coefficient > 0) {
                System.out.print("+");
            }
            System.out.print(current.coefficient + "x^" + current.exponent);
            current = current.next;
        }

        System.out.println("\nEnter the value of x for which you want to evaluate the polynomial: ");
        double xValue = scanner.nextDouble();

        double result = polynomial.evaluate(xValue);

        System.out.println("The result of evaluating the polynomial at x = " + xValue + " is: " + result);
    }
}
```

Summary:

In conclusion, this project aims to create a Polynomial Evaluator using a LinkedList in Java. The **Polynomial** class provides an efficient way to represent and evaluate polynomials for different values of **x**. The project's objectives are centered around creating a robust polynomial evaluation tool. The implementation relies on fundamental Java programming concepts and built-in math functions, making it accessible and useful across various fields of study and application.

This project report provides an overview of the purpose, objectives, and scope of the project, along with an outline of the tools and classes used for polynomial evaluation. The successful completion of this project will result in a versatile polynomial evaluator that can be employed in diverse mathematical and scientific scenarios.

Bibliography:

https://github.com/LiskaDimitri/PolynomialCalculator

https://rajeshpedia.wordpress.com/2011/12/20/evaluation-of-polynomial-usingsingly-linked-list/

Annexure:

GitHub Link: https://github.com/ShahanaT12219504/Cse228-DSA