**Faculty of Computing**

**SE-314: Software Construction**

**Class: BESE 13AB**

# Lab 10: Representing Expression

**CLO-03:** Design and develop solutions based on Software Construction principles.  
**CLO-04:** Use modern tools such as Eclipse, NetBeans etc. for software construction.

**Date: 30th Nov 2024**

**Time: 10:00 AM** **- 12:50 PM   
 02:30 PM – 04:50 PM**

**MUHAMMAD MUSFIR BAIG**

**409968**

**BESE 13A**

**Lab Tasks**

## Expression.Java:

/\* Copyright (c) 2015-2016 MIT 6.005 course staff, all rights reserved.

 \* Redistribution of original or derived work requires permission of course staff.

 \*/

package expressivo;

import java.util.Map;

import org.antlr.v4.runtime.ANTLRInputStream;

import org.antlr.v4.runtime.CharStream;

import org.antlr.v4.runtime.CommonTokenStream;

import org.antlr.v4.runtime.TokenStream;

import org.antlr.v4.runtime.tree.ParseTree;

import org.antlr.v4.runtime.tree.ParseTreeWalker;

import expressivo.parser.ExpressionLexer;

import expressivo.parser.ExpressionParser;

/\*\*

 \* An immutable data type representing a polynomial expression of:

 \*   + and \*

 \*   nonnegative integers and floating-point numbers

 \*   variables (case-sensitive nonempty strings of letters)

 \*/

public interface Expression {

    /\*\*

     \* Parse an expression.

     \* @param input expression to parse, as defined in the PS3 handout.

     \* @return expression AST for the input

     \* @throws IllegalArgumentException if the expression is invalid

     \*/

    public static Expression parse(String input) {

        CharStream stream = new ANTLRInputStream(input);

        ExpressionLexer lexer = new ExpressionLexer(stream);

        lexer.reportErrorsAsExceptions();

        TokenStream tokens = new CommonTokenStream(lexer);

        ExpressionParser parser = new ExpressionParser(tokens);

        parser.reportErrorsAsExceptions();

        ParseTree tree = parser.root();

        ExpressionMaker maker = new ExpressionMaker();

        new ParseTreeWalker().walk(maker, tree);

        return maker.getExpression();

    }

    /\*\*

     \* @return a parsable representation of this expression, such that

     \* for all e:Expression, e.equals(Expression.parse(e.toString())).

     \*/

    @Override

    public String toString();

    /\*\*

     \* @param thatObject any object

     \* @return true if and only if this and thatObject are structurally-equal

     \*/

    @Override

    public boolean equals(Object thatObject);

    /\*\*

     \* @return hash code value consistent with the equals() definition of structural

     \* equality, such that for all e1,e2:Expression,

     \*/

    @Override

    public int hashCode();

    /\*\*

     \* Differentiate an expression with respect to a variable.

     \* @param variable the variable to differentiate by, a case-sensitive nonempty string of letters.

     \* @return expression's derivative with respect to variable.

     \*/

    public Expression differentiate(String variable);

    /\*\*

     \* Simplify an expression.

     \* @param environment maps variables to values. Variables are required to be case-sensitive nonempty

     \*         strings of letters. The set of variables in environment is allowed to be different than the

     \*         set of variables actually found in expression. Values must be nonnegative numbers.

     \* @return an expression equal to the input, but after substituting every variable v that appears in both

     \*/

    public Expression simplify(Map<String, Double> environment);

}

## ExpressionTest.Java:

## 

## Github Link:

## Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. In case of any problems with submissions on LMS, submit your Lab assignments by

emailing it to [aftab.farooq@seecs.edu.pk.](mailto:aftab.farooq@seecs.edu.pk.)