**Tutorial No. 6**

**Problem statement:**

**Design Assumptions:**

**Design Diagrams:**

**Code:**

**Composite\_tester.java**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package tutorial.pkg6;

import java.util.Stack;

/\*\*

\*

\* @author Jatin

\*/

public class Composite\_tester {

private Stack<Expr\_Component> values = null;

private Stack<Character> ops = null;

private char[] tokens = null;

private String s="";

public Composite\_tester(String s){

this.s=s;

}

public double calculate(){

tokens = s.toCharArray();

values = new Stack<>();

ops = new Stack<>();

for (int i = 0; i < tokens.length; i++){

if (tokens[i] == ' ')

continue;

if ((tokens[i] >= '0' && tokens[i] <= '9') || (tokens[i]=='.')){

StringBuffer sbuf = new StringBuffer();

while (i < tokens.length &&((tokens[i] >= '0' && tokens[i] <= '9') || (tokens[i]=='.')))

sbuf.append(tokens[i++]);

i--;

values.push(new Expr\_leaf(Double.parseDouble(sbuf.toString())));

}

else if (tokens[i] == '+' || tokens[i] == '-' || tokens[i] == '\*' || tokens[i] == '/'){

while (!ops.empty() && hasPrecedence(tokens[i], ops.peek()))

values.push(new Expr\_leaf((new Expr\_Composite(ops.pop(), values.pop(), values.pop())).calculate()));

ops.push(tokens[i]);

}

}

while (!ops.empty())

values.push(new Expr\_leaf((new Expr\_Composite(ops.pop(), values.pop(), values.pop())).calculate()));

return (values.pop()).calculate();

}

public static boolean hasPrecedence(char op1, char op2)

{

return !((op1 == '\*' || op1 == '/') && (op2 == '+' || op2 == '-'));

}

}

**Expr\_Component.java**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package tutorial.pkg6;

public interface Expr\_Component {

public double calculate();

}

**Expr\_Composite.java**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package tutorial.pkg6;

import java.util.Stack;

public class Expr\_Composite implements Expr\_Component{

private char op;

private Expr\_Component a;

private Expr\_Component b;

public Expr\_Composite( char op, Expr\_Component b, Expr\_Component a){

this.a=a;

this.b=b;

this.op=op;

}

@Override

public double calculate(){

switch (op)

{

case '+':

return Add(a,b);

case '-':

return Sub(a,b);

case '\*':

return Mul(a,b);

case '/':

return Div(a,b);

}

return 0;

}

public double Add(Expr\_Component left, Expr\_Component right){

return left.calculate() + right.calculate();

}

public double Sub(Expr\_Component left, Expr\_Component right){

return left.calculate() - right.calculate();

}

public double Mul(Expr\_Component left, Expr\_Component right){

return left.calculate() \* right.calculate();

}

public double Div(Expr\_Component left, Expr\_Component right){

if (right.calculate() == 0)

throw new

UnsupportedOperationException("Cannot divide by zero");

return left.calculate() / right.calculate();

}

}

**Expr\_leaf.java**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package tutorial.pkg6;

public class Expr\_leaf implements Expr\_Component{

double value;

public Expr\_leaf(double value) {

this.value = value;

}

@Override

public double calculate() {

return value;

}

}

**Main.java**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package tutorial.pkg6;

/\*\*

\*

\* @author Jatin

\*/

public class Main {

public static void main(String[] args) {

System.out.println( String.format("%.2f",(new Composite\_tester("5+6\*9")).calculate()));

}

}

**Observation:**