```
import javafx.scene.Node;
import javafx.scene.control.Label;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;

public class LiteralExpression implements Expression {
    private CompoundExpression parent = null;
    private String str;
    private Label label;
    public LiteralExpression(String str){
        this.str = str;
    }
    /**
     * Returns the expression's parent.
     *
     * @return the expression's parent
     */
    public CompoundExpression getParent() {
        return parent;
    }
    public String toString(){
        return str;
    }
    /**
     * Sets the parent be the specified expression.
     *
     * @param parent the CompoundExpression that should be the parent of the
target object
     */
    public void setParent(CompoundExpression parent) {
        this.parent = parent;
    }

    /**
     * Creates and returns a deep copy of the expression.
     * The entire tree rooted at the target node is copied, i.e.,
     * the copied Expression is as deep as possible.
     *
     * @return the deep copy
     */
    public Expression deepCopy() {
        final LiteralExpression copy = new LiteralExpression(this.str);
        return copy;
    }

    /**
     it's already flat, does nothing
     */
    public void flatten() {

    }

    /**
     * Creates a String representation by recursively printing out (using
indentation) the
     * tree represented by this expression, starting at the specified
indentation level.
     *
     * @param stringBuilder the StringBuilder to use for building the String
representation
```

```
57      * @param indentLevel   the indentation level (number of tabs from the
    left margin) at which to start
58      */
59     public void convertToString(StringBuilder stringBuilder, int indentLevel)
    {
60         Expression.indent(stringBuilder, indentLevel);
61         stringBuilder.append(this.str);
62         stringBuilder.append("\n");
63     }
64     private Label createNode(){
65         label = new Label();
66         label.setText(this.str);
67         label.setFont(Font.font(Expression.FONT_FAMILY,
    Expression.TEXT_SIZE));
68         return label;
69     }
70     public Node getNode(){
71         if(label == null){
72             label = createNode();
73         }
74         return label;
75     }
76 }
77
```