calculator\JCalculator.java

```java
package calculator;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JScrollPane;
import javax.swing.JTextField;

//import java.awt.event.*;

public class JCalculator extends JFrame
{
  // Tableau representant une pile vide
  private static final String[] empty = { "< empty stack >" };

  // Zone de texte contenant la valeur introduite ou resultat courant
  private final JTextField jNumber = new JTextField("0");

  // Composant liste representant le contenu de la pile
  private final JList<String> jStack = new JList<>(empty);

  // Contraintes pour le placement des composants graphiques
  private final GridBagConstraints constraints = new GridBagConstraints();

  private State state = new State();

  public void setText(String s){
    jNumber.setText(s);
  }

  public String getText(){
    return jNumber.getText();
  }


  // Mise a jour de l'interface apres une operation (jList et jStack)
  private void update()
  {
    jNumber.setText(state.getCurrentInString());

    String[] values = state.getStackInString();
    if(values != null){
      jStack.setListData(values);
    }
    else{
      jStack.setListData(empty);
    }
  }

```

```
57    // Ajout d'un bouton dans l'interface et de l'operation associee,
58    // instance de la classe Operation, possedeant une methode execute()
59    private void addOperatorButton(String name, int x, int y, Color color,
60                    final Operator operator)
61    {
62      JButton b = new JButton(name);
63      b.setForeground(color);
64      constraints.gridx = x;
65      constraints.gridy = y;
66      getContentPane().add(b, constraints);
67      b.addActionListener((e) -> {
68      operator.execute();
69      update();
70        });
71    }
72
73    public JCalculator()
74    {
75      super("JCalculator");
76      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
77      getContentPane().setLayout(new GridBagLayout());
78
79      // Contraintes des composants graphiques
80      constraints.insets = new Insets(3, 3, 3, 3);
81      constraints.fill = GridBagConstraints.HORIZONTAL;
82
83      // Nombre courant
84      jNumber.setEditable(false);
85      jNumber.setBackground(Color.WHITE);
86      jNumber.setHorizontalAlignment(JTextField.RIGHT);
87      constraints.gridx = 0;
88      constraints.gridy = 0;
89      constraints.gridwidth = 5;
90      getContentPane().add(jNumber, constraints);
91      constraints.gridwidth = 1; // reset width
92
93      // Rappel de la valeur en memoire
94      addOperatorButton("MR", 0, 1, Color.RED, new MemoryRecall(state));
95
96      // Stockage d'une valeur en memoire
97      addOperatorButton("MS", 1, 1, Color.RED, new MemoryStore(state));
98
99      // Backspace
100     addOperatorButton("<=", 2, 1, Color.RED, new Backspace(state));
101
102     // Mise a zero de la valeur courante + suppression des erreurs
103     addOperatorButton("CE", 3, 1, Color.RED, new ClearError(state));
104
105     // Comme CE + vide la pile
106     addOperatorButton("C",  4, 1, Color.RED, new Clear(state));
107
108     // Boutons 1-9
109     for (int i = 1; i < 10; i++)
110       addOperatorButton(String.valueOf(i), (i - 1) % 3, 4 - (i - 1) / 3,
111             Color.BLUE, new Digit(state, i));
112     // Bouton 0
113     addOperatorButton("0", 0, 5, Color.BLUE, new Digit(state, 0) );
114
115     // Changement de signe de la valeur courante
```

```
116     addOperatorButton("+/-", 1, 5, Color.BLUE, new Negate(state));
117
118     // Operateur point (chiffres apres la virgule ensuite)
119     addOperatorButton(".", 2, 5, Color.BLUE, new Point(state));
120
121     // Operateurs arithmetiques a deux operandes: /, *, -, +
122     addOperatorButton("/", 3, 2, Color.RED, new Division(state));
123     addOperatorButton("*", 3, 3, Color.RED, new Multiplication(state));
124     addOperatorButton("-", 3, 4, Color.RED, new Subtraction(state));
125     addOperatorButton("+", 3, 5, Color.RED, new Addition(state));
126
127     // Operateurs arithmetiques a un operande: 1/x, x^2, Sqrt
128     addOperatorButton("1/x", 4, 2, Color.RED, new Inverse(state));
129     addOperatorButton("x^2", 4, 3, Color.RED, new Power(state));
130     addOperatorButton("Sqrt", 4, 4, Color.RED, new SquareRoot(state));
131
132     // Entree: met la valeur courante sur le sommet de la pile
133     addOperatorButton("Ent", 4, 5, Color.RED, new Enter(state));
134
135     // Affichage de la pile
136     JLabel jLabel = new JLabel("Stack");
137     jLabel.setFont(new Font("Dialog", 0, 12));
138     jLabel.setHorizontalAlignment(JLabel.CENTER);
139     constraints.gridx = 5;
140     constraints.gridy = 0;
141     getContentPane().add(jLabel, constraints);
142
143     jStack.setFont(new Font("Dialog", 0, 12));
144     jStack.setVisibleRowCount(8);
145     JScrollPane scrollPane = new JScrollPane(jStack);
146     constraints.gridx = 5;
147     constraints.gridy = 1;
148     constraints.gridheight = 5;
149     getContentPane().add(scrollPane, constraints);
150     constraints.gridheight = 1; // reset height
151
152     setResizable(false);
153     pack();
154     setVisible(true);
155   }
156 }
157
```