KLASA SUDOKUWINDOW

```
public class SudokuWindow extends JFrame implements FieldValueChangeListener {
    public List<String> puzzle;
    // private JButton SudokuWindow[][] = new JButton[9][9];
    SudokuField[][] tablicaGui = new SudokuField[9][9];
    String[] linia = new String[9];


    public SudokuWindow(){ // konstuktor
        readPuzzle();
        createGui();
       // showList();

    }
```

```
    public void showList(){ // do testowania
        for(int i = 0; i < 9; i++){
                System.out.print(puzzle.get(i));
                System.out.print("\n");
        }
    }
```

```
public void readPuzzle(){
    try{
        File file = new File("puzzle2.txt");
        URI fileUri = file.toURI();
        Path puzzlePath = Paths.get(fileUri);
        puzzle = Files.readAllLines(puzzlePath);
    } catch (IOException e){
        e.printStackTrace();
    }
}
```

```java
public void createGui(){
    setTitle("Sudoku");
    setVisible(true);
    setSize(500,500);
    setLayout(new GridLayout(9,9));
    for(int i = 0; i < 9; i++){ // wpisywanie linii do tablicy linii
        linia[i] = puzzle.get(i);
        System.out.print(linia[i]);
    }

    for(int wiersz = 0; wiersz < 9; wiersz++){
        for(int kolumna = 0; kolumna < 9; kolumna++){
            if( linia[wiersz].charAt(kolumna) == '0'){
                VariableSudokuField fieldV = new VariableSudokuField(this);
                tablicaGui[wiersz][kolumna] = fieldV;
                    add(fieldV.label);
            }else if(linia[wiersz].charAt(kolumna) == '1' ||
                linia[wiersz].charAt(kolumna) == '2' ||
                linia[wiersz].charAt(kolumna) == '3' ||
                linia[wiersz].charAt(kolumna) == '4' ||
                linia[wiersz].charAt(kolumna) == '5' ||
                linia[wiersz].charAt(kolumna) == '6' ||
                linia[wiersz].charAt(kolumna) == '7' ||
                linia[wiersz].charAt(kolumna) == '8' ||
                linia[wiersz].charAt(kolumna) == '9'){
                    int puzzleInt = linia[wiersz].charAt(kolumna) - 48;
                    FixedSudokuField fieldF = new FixedSudokuField(puzzleInt,this);
                    tablicaGui[wiersz][kolumna] = fieldF;
                        add(fieldF.label);
            }
            System.out.print(linia[wiersz].charAt(kolumna));
        }
    }
```

```java
@Override
public void fieldsChanged(){ // metoda z interfejsu FieldValueChangeListener
    for(int i = 0; i < 9; i++){
        for(int j = 0; j < 9; j++){
            tablicaGui[i][j].setError(false);
        }
    }

    // sprawdzanie wierszy
    for(int i = 0; i < 9; i++){ ···

    // sprawdzanie kolumn
    for(int kolumna = 0; kolumna < 9; kolumna++){ ···
```

```
// zaznaczanie kwadratu 3x3
    for(int kwadrat = 0; kwadrat < 7; kwadrat+=3){ // kwadraty po lewo
        for(int innerRow = 0+kwadrat; innerRow < 3+kwadrat; innerRow++){
            if(tablicaGui[innerRow][2] == tablicaGui[innerRow][1] || tablicaGui[innerRow][2] == tablicaGui[innerRow][0] ||
               tablicaGui[innerRow][1] == tablicaGui[innerRow][0]){ // sprawdzanie wiersza w kwadracie
                    for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                        for(int kolumna = 0; kolumna < 3; kolumna++){
                            tablicaGui[wiersz][kolumna].setError(true);
                        }
                    }
            }else if(tablicaGui[0+kwadrat][2] == tablicaGui[1+kwadrat][2] ||
            tablicaGui[0+kwadrat][2] == tablicaGui[2+kwadrat][2] ||
            tablicaGui[2+kwadrat][2] == tablicaGui[1+kwadrat][2] ||
            tablicaGui[0+kwadrat][1] == tablicaGui[1+kwadrat][1] ||
            tablicaGui[0+kwadrat][1] == tablicaGui[2+kwadrat][1] ||
            tablicaGui[2+kwadrat][1] == tablicaGui[1+kwadrat][1] ||
            tablicaGui[0+kwadrat][0] == tablicaGui[1+kwadrat][0] ||
            tablicaGui[0+kwadrat][0] == tablicaGui[2+kwadrat][0] ||
            tablicaGui[2+kwadrat][0] == tablicaGui[1+kwadrat][0]){ // sprawdzanie kolumn w kwadracie
               for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                    for(int kolumna = 0; kolumna < 3; kolumna++){
                        tablicaGui[wiersz][kolumna].setError(true);
                    }
               }
    }
```

```
    }
    if(tablicaGui[0+kwadrat][0] == tablicaGui[1+kwadrat][1] ||
                tablicaGui[0+kwadrat][0] == tablicaGui[2+kwadrat][2] ||
                tablicaGui[0+kwadrat][1] == tablicaGui[1+kwadrat][2] ||
                tablicaGui[1+kwadrat][0] == tablicaGui[2+kwadrat][1] ||
                tablicaGui[0+kwadrat][2] == tablicaGui[1+kwadrat][1] ||
                tablicaGui[0+kwadrat][2] == tablicaGui[0+kwadrat][0] ||
                tablicaGui[0+kwadrat][1] == tablicaGui[1+kwadrat][0] ||
                tablicaGui[2+kwadrat][2] == tablicaGui[2+kwadrat][1]){ // przekatna
                   for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                       for(int kolumna = 0; kolumna < 3; kolumna++){
                           tablicaGui[wiersz][kolumna].setError(true);
                       }
                   }
        }
}

for(int kwadrat = 0; kwadrat < 7; kwadrat+=3){ // kwadraty po srodku
    for(int innerRow = 0+kwadrat; innerRow < 3+kwadrat; innerRow++){
        if(tablicaGui[innerRow][5] == tablicaGui[innerRow][4] || tablicaGui[innerRow][5] == tablicaGui[innerRow][3] ||
           tablicaGui[innerRow][4] == tablicaGui[innerRow][3]){ // sprawdzanie wiersza w kwadracie
            for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                for(int kolumna = 3; kolumna < 6; kolumna++){
                    tablicaGui[wiersz][kolumna].setError(true);
                }
            }
```

```
            }else if(tablicaGui[0+kwadrat][6] == tablicaGui[1+kwadrat][6] ||
            tablicaGui[0+kwadrat][6] == tablicaGui[2+kwadrat][6] ||
            tablicaGui[2+kwadrat][6] == tablicaGui[1+kwadrat][6] ||
            tablicaGui[0+kwadrat][5] == tablicaGui[1+kwadrat][5] ||
            tablicaGui[0+kwadrat][5] == tablicaGui[2+kwadrat][5] ||
            tablicaGui[2+kwadrat][5] == tablicaGui[1+kwadrat][5] ||
            tablicaGui[0+kwadrat][4] == tablicaGui[1+kwadrat][4] ||
            tablicaGui[0+kwadrat][4] == tablicaGui[2+kwadrat][4] ||
            tablicaGui[2+kwadrat][4] == tablicaGui[1+kwadrat][4]){ // sprawdzanie kolumn w kwadracie
               for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                  for(int kolumna = 3; kolumna < 6; kolumna++){
                     tablicaGui[wiersz][kolumna].setError(true);
                  }
               }
   }
        }
        if(tablicaGui[0+kwadrat][3] == tablicaGui[1+kwadrat][4] ||
                  tablicaGui[0+kwadrat][3] == tablicaGui[2+kwadrat][5] ||
                  tablicaGui[0+kwadrat][4] == tablicaGui[1+kwadrat][5] ||
                  tablicaGui[1+kwadrat][3] == tablicaGui[2+kwadrat][4] ||
                  tablicaGui[0+kwadrat][5] == tablicaGui[1+kwadrat][4] ||
                  tablicaGui[0+kwadrat][5] == tablicaGui[0+kwadrat][3] ||
                  tablicaGui[0+kwadrat][4] == tablicaGui[1+kwadrat][3] ||
                  tablicaGui[2+kwadrat][5] == tablicaGui[2+kwadrat][4]){ // przekatna
                     for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                        for(int kolumna = 3; kolumna < 6; kolumna++){
                           tablicaGui[wiersz][kolumna].setError(true);
                        }
                     }
            }
    }
```

```
for(int kwadrat = 0; kwadrat < 7; kwadrat+=3){ // kwadraty po prawo
    for(int innerRow = 0+kwadrat; innerRow < 3+kwadrat; innerRow++){
        if(tablicaGui[innerRow][8] == tablicaGui[innerRow][7] || tablicaGui[innerRow][8] == tablicaGui[innerRow][6] ||
           tablicaGui[innerRow][7] == tablicaGui[innerRow][6]){ // sprawdzanie wiersza w kwadracie
            for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                for(int kolumna = 6; kolumna < 9; kolumna++){
                    tablicaGui[wiersz][kolumna].setError(true);
                }
            }
        }else if(tablicaGui[0+kwadrat][8] == tablicaGui[1+kwadrat][8] ||
                 tablicaGui[0+kwadrat][8] == tablicaGui[2+kwadrat][8] ||
                 tablicaGui[2+kwadrat][8] == tablicaGui[1+kwadrat][8] ||
                 tablicaGui[0+kwadrat][7] == tablicaGui[1+kwadrat][7] ||
                 tablicaGui[0+kwadrat][7] == tablicaGui[2+kwadrat][7] ||
                 tablicaGui[2+kwadrat][7] == tablicaGui[1+kwadrat][7] ||
                 tablicaGui[0+kwadrat][6] == tablicaGui[1+kwadrat][6] ||
                 tablicaGui[0+kwadrat][6] == tablicaGui[2+kwadrat][6] ||
                 tablicaGui[2+kwadrat][6] == tablicaGui[1+kwadrat][6]){ // sprawdzanie kolumn w kwadracie
                    for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                        for(int kolumna = 6; kolumna < 9; kolumna++){
                            tablicaGui[wiersz][kolumna].setError(true);
                        }
                    }
        }
    }
```

```
                if(tablicaGui[0+kwadrat][3] == tablicaGui[1+kwadrat][4] ||
                        tablicaGui[0+kwadrat][6] == tablicaGui[2+kwadrat][8] ||
                        tablicaGui[0+kwadrat][7] == tablicaGui[1+kwadrat][8] ||
                        tablicaGui[1+kwadrat][6] == tablicaGui[2+kwadrat][7] ||
                        tablicaGui[0+kwadrat][8] == tablicaGui[1+kwadrat][7] ||
                        tablicaGui[0+kwadrat][8] == tablicaGui[0+kwadrat][6] ||
                        tablicaGui[0+kwadrat][7] == tablicaGui[1+kwadrat][6] ||
                        tablicaGui[2+kwadrat][8] == tablicaGui[2+kwadrat][7]){ // przekatna
                    for(int wiersz = 0+kwadrat; wiersz < 3+kwadrat; wiersz++){
                        for(int kolumna = 6; kolumna < 9; kolumna++){
                            tablicaGui[wiersz][kolumna].setError(true);
                        }
                    }
                }
            }
        }

    }
```

KLASA SUDOKUFIELD

```
abstract class SudokuField extends JLabel{
    private int value;
    JLabel label = new JLabel();
    public SudokuField(int valueInjected){
        Integer intTmp = valueInjected;
        String temp = intTmp.toString();
        this.label = new JLabel(temp);
        this.setDisplayedValue(valueInjected); // nie dziala nie wiem czemu, wiec musialem dodac to wyzej
        this.label.setHorizontalAlignment(CENTER);
        this.label.setVerticalAlignment(CENTER);
    }

    protected SudokuField getSudokuField(){
        return this;
    }
    protected void setSudokuField(){
      //
    }

    protected void setDisplayedValue(int valueInjected){
        if(value != 0){
            this.value = valueInjected;
            Integer integerValue = valueInjected;
            String valueString = integerValue.toString();
            this.label = new JLabel(valueString);
        }
    }
```

```
    public void setError(boolean bboolean){
    if(bboolean == true){
        this.setBackground(Color.red);
    }else if(bboolean == false){
        UIManager.getColor("Panel.background");
    }


    }
}
```

KLASA VARIABLESUDOKUFIELD

```
public class VariableSudokuField extends SudokuField{
    public VariableSudokuField(final FieldValueChangeListener listener){
        super(0);
        JPopupMenu popupMenu = new JPopupMenu();
        JMenuItem mi;

        label.setBorder(BorderFactory.createEtchedBorder());

        popupMenu.add(mi = new JMenuItem("Wyczysc"));
        popupMenu.add(mi = new JMenuItem("1"));
        popupMenu.add(mi = new JMenuItem("2"));
        popupMenu.add(mi = new JMenuItem("3"));
        popupMenu.add(mi = new JMenuItem("4"));
        popupMenu.add(mi = new JMenuItem("5"));
        popupMenu.add(mi = new JMenuItem("6"));
        popupMenu.add(mi = new JMenuItem("7"));
        popupMenu.add(mi = new JMenuItem("8"));
        popupMenu.add(mi = new JMenuItem("9"));

        mi.addActionListener(e->{
            listener.fieldsChanged();
        });
        label.add(mi);
```

```
        label.addMouseListener(new MouseAdapter() { // menu sie nie pokazuje, jednakze nic dalej nie mozna zrobic
            @Override
             public void mouseClicked(MouseEvent e){
                 VariableSudokuField field = VariableSudokuField.this;
                 // popupMenu.show(field,field.getWidth()/2,field.getHeight()/2);
                 popupMenu.setVisible(true);
                 // tutaj funkcja setValue() dla odpowiedniej labelki
                 //popupMenu.setVisible(false);
             }
         });
    }

    //  @Override
    //  public void setValue(int value){
    //      super.setDisplayedValue(value);
    //  }
}
```

KLASA FIXEDSUDOKUFIELD

```
public class FixedSudokuField extends SudokuField {
    public FixedSudokuField(int value, final FieldValueChangeListener listener) {
        super(value);
        super.setDisplayedValue(value);
        //super.setBackground(Color.green);
    }
}
```

KLASA RUN

```java
import javax.swing.SwingUtilities;

public class Run {
    Run | Debug
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            (new SudokuWindow()).setVisible(true);
            //(new SudokuWindow()).readPuzzle();
            //(new SudokuWindow()).createGui();
        });
    }
}
```

INTERFEJS FIELDFVALUECHANGELISTENER

```java
public interface FieldValueChangeListener {
    public void fieldsChanged();
}
```

Podsumowanie:

Brakuje ustawiania elementu JLabel, kiedy jest równy zero poprzez MouseListener i PopUPMenu.

Wyświetla się: