

```
package gui;

import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.util.ArrayList;

import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import ast.Method;

import skeleton.SkeletonMaker;

/**
 * Class that shows and manage the GUI of creating a new method and a new constructor
 * @author Sergio Alcocer
 */
public class CreateMethodConstructorUI extends JFrame{
    private static final long serialVersionUID = 1L;

    private ArrayList<String> params;
    private JTextField txtParamType, txtMethodName, txtAccess, txtReturnedType;
    private JButton jbMore, jbLess, jbSave;
    private int currentParam;
    private boolean isConstructor;
    private String className;

    /**
     * Constructor of the class to use when creating a method
     */
    public CreateMethodConstructorUI(){
        txtParamType = new JTextField();
```

```
txtMethodName = new JTextField();
txtReturnedType = new JTextField();
txtAccess = new JTextField();
jbMore = new JButton(">>");
jbLess = new JButton("<<");
jbSave = new JButton("Save");
params = new ArrayList<String>();
isConstructor = false;
init();
}
/**
 * Constructor of the class to use when creating a Constructor
 * @param _className name of the class
 */
public CreateMethodConstructorUI(String _className){
    txtParamType = new JTextField();
    txtMethodName = new JTextField();
    txtAccess = new JTextField();
    jbMore = new JButton(">>");
    jbLess = new JButton("<<");
    jbSave = new JButton("Save");
    params = new ArrayList<String>();
    isConstructor = true;
    className = _className;
    init();
}

private void init(){
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);

    this.addWindowListener(new WindowListener(){
        public void windowActivated(WindowEvent arg0) {}
        public void windowClosed(WindowEvent arg0) {
            MainFrame.getInstance().unlock();
            MainFrame.getInstance().ToFront();
        }
        public void windowClosing(WindowEvent arg0) {}
        public void windowDeactivated(WindowEvent arg0) {}
        public void windowDeiconified(WindowEvent arg0) {}
    });
}
```

```
    public void windowIconified(WindowEvent arg0) {}
    public void windowOpened(WindowEvent arg0) {}
});

////////////////////////////////////
//  START LAYOUT  //
////////////////////////////////////
Box mainBox = new Box(BoxLayout.Y_AXIS);

{
    Box miniBox = new Box(BoxLayout.X_AXIS);
    if(isConstructor){
        miniBox.add(new JLabel("Access    {ConstructorName}    (    ParamType1,    ParamType2, ...)"));
    }else{
        miniBox.add(new JLabel("Access    ReturnedType    MethodName    (    ParamType1, ParamType2, ...)"));
    }
    mainBox.add(miniBox);
}

Box constructorInfoBox = new Box(BoxLayout.X_AXIS);

txtAccess.setMinimumSize(new java.awt.Dimension(50,27));
txtAccess.setMaximumSize(new java.awt.Dimension(50,27));
txtAccess.setPreferredSize(new java.awt.Dimension(50,27));
txtAccess.setToolTipText("Either public or private");
constructorInfoBox.add(txtAccess);

txtMethodName.setMinimumSize(new java.awt.Dimension(80,27));
txtMethodName.setMaximumSize(new java.awt.Dimension(80,27));
txtMethodName.setPreferredSize(new java.awt.Dimension(80,27));

if(!isConstructor){
    txtReturnedType.setMinimumSize(new java.awt.Dimension(80,27));
    txtReturnedType.setMaximumSize(new java.awt.Dimension(80,27));
    txtReturnedType.setPreferredSize(new java.awt.Dimension(80,27));

    constructorInfoBox.add(txtReturnedType);
}
```

```
        constructorInfoBox.add(txtMethodName);
    }else{
        constructorInfoBox.add(txtMethodName);
        txtMethodName.setEnabled(false);
        txtMethodName.setText(className);
    }
    constructorInfoBox.add(new JLabel("("));

    jbLess.setEnabled(false);
    jbLess.addActionListener(new ActionListener () {
        public void actionPerformed(ActionEvent arg0) {
            lessClick();
        }
    });

    jbMore.addActionListener(new ActionListener () {
        public void actionPerformed(ActionEvent arg0) {
            moreClick();
        }
    });

    jbSave.addActionListener(new ActionListener () {
        public void actionPerformed(ActionEvent arg0) {
            saveObject();
        }
    });

    txtParamType.setMinimumSize(new java.awt.Dimension(80,27));
    txtParamType.setMaximumSize(new java.awt.Dimension(2000,27));
    txtParamType.setPreferredSize(new java.awt.Dimension(200,27));
    constructorInfoBox.add(jbLess);
    constructorInfoBox.add(txtParamType);
    constructorInfoBox.add(jbMore);
    constructorInfoBox.add(new JLabel(" ) "));
    constructorInfoBox.add(jbSave);

    mainBox.add(constructorInfoBox);
    mainBox.add(Box.createGlue());

    setContentPane(mainBox);
```

```
setVisible(true);
currentParam = 0;

pack();

}

private void lessClick(){
    if (txtParamType.getText().equals("")){
        currentParam--;
        if(currentParam == 0) jblLess.setEnabled(false);
        txtParamType.setText(params.get(currentParam));
    }else{
        if(txtParamType.getText().indexOf(' ') == -1 && !txtParamType.getText().equals("")){
            if(params.size() > currentParam){
                params.set(currentParam, txtParamType.getText());
            }else{
                params.add(txtParamType.getText());
            }
            currentParam--;
            if(currentParam == 0) jblLess.setEnabled(false);

            txtParamType.setText(params.get(currentParam));

        }else{
            msgBox("You cannot use types with a <space> on it or be empty.");
        }
    }
}

private void moreClick(){
    if(txtParamType.getText().indexOf(' ') == -1 && !txtParamType.getText().equals("")){
        if(params.size() > currentParam){
            params.set(currentParam, txtParamType.getText());
        }else{
            params.add(txtParamType.getText());
        }
        jblLess.setEnabled(true);
        currentParam++;
        //updateConstructorLabel();
        try{
```

```
        txtParamType.setText(params.get(currentParam));
    }catch(Exception e){
        txtParamType.setText("");
    }
}
}else{
    msgBox("You cannot use types with a <space> on it or be empty.");
}
}

private void saveObject(){
    if (txtAccess.getText().equalsIgnoreCase("private") || txtAccess.getText().equalsIgnoreCase("public")){
        if(isConstructor){
            Method newMethod = new Method();
            newMethod.setReturnedType(className);
            newMethod.setAccess(txtAccess.getText().equalsIgnoreCase("private")?Method.PRIVATE:Method.PUBLIC);
            newMethod.setAsConstructor();
            for (int i = 0; i < params.size(); i++){
                newMethod.addParam(params.get(i));
            }
            SkeletonMaker.getInstance().addMethod(newMethod);
            dispose();
        }else{
            if (!txtMethodName.getText().equals("")){
                if(!txtReturnedType.getText().equals("")){
                    Method newMethod = new Method();
                    newMethod.setName(txtMethodName.getText());
                    newMethod.setReturnedType(txtReturnedType.getText());
                    newMethod.setAccess(txtAccess.getText().equalsIgnoreCase("private")?Method.PRIVATE:Method.PUBLIC);

                    for (int i = 0; i < params.size(); i++){
                        newMethod.addParam(params.get(i));
                    }
                    SkeletonMaker.getInstance().addMethod(newMethod);
                    dispose();
                }else{
                    msgBox("Not empty Returned Type is allowed");
                }
            }else{
                msgBox("Not empty Method's name is allowed.");
            }
        }
    }
}
```

```
        }  
    }else{  
        msgBox("Only public or private are allowed in Access' Field");  
    }  
  
}  
  
private void msgBox(String msg){  
    JOptionPane.showMessageDialog(((Component) this), msg);  
}  
}
```