

```
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 generator.Assert;
import generator.Instruction;
import generator.ObjectCreated;
import generator.TestInfo;

import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextField;

/**
 * Class that shows and manage the GUI of creating/modifying an instruction
 * @author Sergio Alcocer
 */
public class CreateInstructionUI extends JFrame {

    private static final long serialVersionUID = 1L;

    private boolean editing;
    private ast.Method candidate;
    public static final int ASSERT_TRUE = 0;
    public static final int ASSERT_FALSE = 1;
    public static final int ASSERT_EQUALS = 2;
    public static final int CUSTOM = 3;
```

```
private int optionSelected, index;
private JTextField jtCustomCode;
private JTextField jtAssertEquals1, jtAssertEquals2;
private ArrayList<JComboBox> comboBoxes;
private ArrayList<JTextField> textFields;

/**
 * Constructor of the class to use when creating an instruction
 * @param candidate method that would be associated with the instruction
 */
public CreateInstructionUI(ast.Method candidate) {
    editing = false;
    this.candidate = candidate;
    optionSelected = -1;
    jtCustomCode = new JTextField();
    comboBoxes = new ArrayList<JComboBox>();
    textFields = new ArrayList<JTextField>();
    init();
}

/**
 * Constructor of the class to use when editing an instruction
 * @param candidate method that is associated with the instructino
 * @param position position of the instruction
 */
public CreateInstructionUI(ast.Method candidate, int position) {
    editing = true;
    this.candidate = candidate;
    index = position;
    optionSelected = -1;
    jtCustomCode = new JTextField();
    comboBoxes = new ArrayList<JComboBox>();
    textFields = new ArrayList<JTextField>();
    init();
}

private void init() {
    setTitle("Instruction Creation for " + candidate.toString());
}
```

```
this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);

////////////////////
//          WINDOW LISTENER          //
////////////////////
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) {}
});

ButtonGroup bgChoice = new ButtonGroup();
Box mainBox = new Box(BoxLayout.Y_AXIS);

int instType = -1;
Instruction currentInst = null;
if (editing) {
    currentInst = TestInfo.getInstance().getThingToDo(candidate.toString(), index);
    instType = currentInst.getType();
}

if (!candidate.getReturnedType().equals("void")) {
    setSize(400, 160);

    //////////////////
    // FIRST ROW, ASSERT TRUE AND FALSE //
    //////////////////
    Box longBox = new Box(BoxLayout.X_AXIS);
    JRadioButton rbAssertTrue;
    JRadioButton rbAssertFalse;
    {
        Box temp = new Box(BoxLayout.Y_AXIS);
        rbAssertTrue = new JRadioButton("Assert True");
```

```

        rbAssertTrue.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                optionSelected = ASSERT_TRUE;
            }
        });
        bgChoice.add(rbAssertTrue);
        rbAssertFalse = new JRadioButton("Assert False");
        rbAssertFalse.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                optionSelected = ASSERT_FALSE;
            }
        });
        bgChoice.add(rbAssertFalse);
        temp.add(rbAssertTrue);
        temp.add(rbAssertFalse);

        longBox.add(temp);
    }

    // for through parameters to add a field for each parameter. (if is
    // the object, show a list of current available objects)
    //////////////////////////////////////
    //      BOX WITH PARAMETERS      //
    //////////////////////////////////////
    Box inLongBox = new Box(BoxLayout.X_AXIS);
    {
        JComboBox newcmb = new JComboBox(TestInfo.getInstance().getObjectsString().toArray());
        newcmb.setPreferredSize(new java.awt.Dimension(100, 25));
        newcmb.setMaximumSize(new java.awt.Dimension(100, 25));
        newcmb.setMinimumSize(new java.awt.Dimension(100, 25));
        inLongBox.add(newcmb);
        inLongBox.add(new JLabel("." + candidate.getName() + "("));
        if(editing && instType == Instruction.T_ASSERT && ((Assert)currentInst).getAssertType() != Assert.
ASSERT_EQUALS){
            newcmb.setSelectedIndex(TestInfo.getInstance().getObjectIndex(((Assert)currentInst).getObjectInvolved()));
        }
        comboBoxes.add(newcmb);
    }

    String myClass = TestInfo.getInstance().getClassName();

```

```
for (int i = 0; i < candidate.getParamNumber(); i++) {
    inLongBox.add(Box.createHorizontalStrut(10));
    if (candidate.getParam(i).equals(myClass)) {
        JComboBox newcmb = new JComboBox(TestInfo.getInstance().getObjectsString().toArray());
        newcmb.setPreferredSize(new java.awt.Dimension(100, 25));
        newcmb.setMaximumSize(new java.awt.Dimension(100, 25));
        newcmb.setMinimumSize(new java.awt.Dimension(100, 25));
        if (editing && instType == Instruction.T_ASSERT && ((Assert)currentInst).getAssertType() != Assert.
ASSERT_EQUALS) {
            newcmb.setSelectedIndex(TestInfo.getInstance().getObjectIndex(((Assert) currentInst).getParams().get(i)
));
        }

        inLongBox.add(newcmb);

        comboBoxes.add(newcmb);
    } else {
        JLabel newlbl = new JLabel(candidate.getParam(i));
        inLongBox.add(newlbl);
        JTextField newtxt = new JTextField();
        newtxt.setPreferredSize(new java.awt.Dimension(100, 25));
        newtxt.setMaximumSize(new java.awt.Dimension(100, 25));
        newtxt.setMinimumSize(new java.awt.Dimension(100, 25));
        if (editing && instType == Instruction.T_ASSERT && ((Assert)currentInst).getAssertType() != Assert.
ASSERT_EQUALS) {
            newtxt.setText(((Assert) currentInst).getParams().get(i));
        }
        inLongBox.add(newtxt);
        textFields.add(newtxt);
    }

}

inLongBox.add(Box.createHorizontalStrut(10));
inLongBox.add(new JLabel(""));
inLongBox.add(Box.createHorizontalStrut(10));

JLabel jlextra = new JLabel(" { Rest of the code: * } ");
jlextra.setToolTipText("Add here the rest of the code (i.e. \" == 50\")");
```

```
inLongBox.add(jlextra);
JTextField jtextra = new JTextField();
jtextra.setPreferredSize(new java.awt.Dimension(100, 25));
jtextra.setMaximumSize(new java.awt.Dimension(1000, 25));
jtextra.setMinimumSize(new java.awt.Dimension(100, 25));
if(editing && instType == Instruction.T_ASSERT && ((Assert)currentInst).getAssertType() != Assert.ASSERT_EQUALS) {
    jtextra.setText(((Assert)currentInst).getRestCode());
}
inLongBox.add(jtextra);
textFieldds.add(jtextra);

inLongBox.add(Box.createHorizontalStrut(10));
// end for

JButton btSave = new JButton("Save");
btSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        save();
    }
});
inLongBox.add(btSave);

JScrollPane jsp = new JScrollPane(inLongBox);
jsp.setPreferredSize(new java.awt.Dimension(20, 60));
jsp.setMaximumSize(new java.awt.Dimension(2000, 60));
jsp.setMinimumSize(new java.awt.Dimension(20, 60));
longBox.add(jsp);
mainBox.add(longBox);

mainBox.add(Box.createGlue());

if(editing){ // if editing and asserts true and false are visible
    if(instType == Instruction.T_ASSERT){
        if(((Assert)currentInst).getAssertType() == Assert.ASSERT_TRUE){
            rbAssertTrue.setSelected(true);
            optionSelected = ASSERT_TRUE;
        }else if(((Assert)currentInst).getAssertType() == Assert.ASSERT_FALSE){
            rbAssertFalse.setSelected(true);
            optionSelected = ASSERT_FALSE;
        }
    }
}
```

```
        }
    }
} else {
    setSize(400, 95);
}

////////////////////////////////////
//          CUSTOM CODE BOX          //
////////////////////////////////////
Box shortBox = new Box(BoxLayout.X_AXIS);
JRadioButton optCustom = new JRadioButton("Custom Code");
optCustom.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        optionSelected = CUSTOM;
    }
});
bgChoice.add(optCustom);
shortBox.add(optCustom);

jtCustomCode = new JTextField();

jtCustomCode.setPreferredSize(new java.awt.Dimension(100, 27));
jtCustomCode.setMaximumSize(new java.awt.Dimension(1500, 27));
// jtCustomCode.setMinimumSize(new java.awt.Dimension(100,30));
if (instType == Instruction.T_CUSTOM) {
    jtCustomCode.setText(TestInfo.getInstance().getThingToDo(candidate.toString(), this.index).getCode());
}
shortBox.add(jtCustomCode);

JButton btSave2 = new JButton("Save");
btSave2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        save();
    }
});
shortBox.add(btSave2);

mainBox.add(shortBox);
```

```
////////////////////////////////////////
//          ASSERT_EQUALS          //
////////////////////////////////////////

shortBox = new Box(BoxLayout.X_AXIS);
JRadioButton optEquals = new JRadioButton("Assert Equals");
optEquals.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        optionSelected = ASSERT_EQUALS;
    }
});
bgChoice.add(optEquals);
shortBox.add(optEquals);

jtAssertEquals1 = new JTextField();
jtAssertEquals2 = new JTextField();

jtAssertEquals1.setPreferredSize(new java.awt.Dimension(100, 27));
jtAssertEquals1.setMaximumSize(new java.awt.Dimension(900, 27));
jtAssertEquals2.setPreferredSize(new java.awt.Dimension(100, 27));
jtAssertEquals2.setMaximumSize(new java.awt.Dimension(900, 27));

if (instType == Instruction.T_ASSERT && ((Assert)currentInst).getAssertType() == Assert.ASSERT_EQUALS) {
    Assert ass = (Assert)TestInfo.getInstance().getThingToDo(candidate.toString(), this.index);
    ArrayList<String> paramsRetrieved = ass.getParams();
    jtAssertEquals1.setText(paramsRetrieved.get(0));
    jtAssertEquals2.setText(paramsRetrieved.get(1));
}

shortBox.add(new JLabel("( "));
shortBox.add(jtAssertEquals1);
shortBox.add(new JLabel(" , "));
shortBox.add(jtAssertEquals2);
shortBox.add(new JLabel(" )"));

mainBox.add(shortBox);

setContentPane(mainBox);
```

```
    if (editing){ // set the type of instruction
        if(instType == Instruction.T_CUSTOM){
            optionSelected = CUSTOM;
            optCustom.setSelected(true);
        }else{
            if(((Assert)currentInst).getAssertType() == Assert.ASSERT_EQUALS){
                optEquals.setSelected(true);
                optionSelected = ASSERT_EQUALS;
            }
        }
    }
    setVisible(true);
    // pack();

}

private void save() {
    switch (optionSelected) {
        case ASSERT_TRUE:
        case ASSERT_FALSE:
            Assert newAssert = new Assert((optionSelected == 0), ((ObjectCreated) TestInfo.getInstance().getObject(comboBoxes.
get(0).getSelectedIndex())).getObjectName(), candidate.getName());
            String myClass = TestInfo.getInstance().getClassName();
            int comboIndex = 1,
            textFieldIndex = 0;
            for (int i = 0; i < candidate.getParamNumber(); i++) {
                if (candidate.getParam(i).equals(myClass)) {
                    newAssert.addParam(((ObjectCreated) TestInfo.getInstance().getObject(comboBoxes.get(comboIndex++) .
getSelectedIndex())).getObjectName());
                } else {
                    String textfield = textFields.get(textFieldIndex++).getText();
                    if (textfield.equals("")){
                        msgBox("You cannot leave fields of the Assert " + (optionSelected==0) + " empty.");
                        return;
                    }else{
                        newAssert.addParam(textfield);
                    }
                }
            }
    }
}
```

```
newAssert.setRestCode(textFields.get(textFields.size()-1).getText());

if (editing){
    TestInfo.getInstance().addThingsToDo(candidate.toString(),newAssert,index);
}else{
    TestInfo.getInstance().addThingsToDo(candidate.toString(),newAssert);
}

dispose();
break;
case ASSERT_EQUALS:
    String obj1 = jtAssertEquals1.getText();
    String obj2 = jtAssertEquals2.getText();
    if(obj1.equals("") || obj2.equals("")){
        msgBox("Please fill both fields for Assert Equals");
    }else{
        Assert myAssert = new Assert(candidate.getName(), obj1, obj2);
        if(editing){
            TestInfo.getInstance().addThingsToDo(candidate.toString(), myAssert, index);
        }else{
            TestInfo.getInstance().addThingsToDo(candidate.toString(), myAssert);
        }
        dispose();
    }

    break;
case CUSTOM:
    generator.Custom c = new generator.Custom();
    String code = jtCustomCode.getText();
    if (code.equals("")){
        msgBox("The code field cannot be left empty");
        return;
    }
    c.setCode(code);
    if (editing){
        TestInfo.getInstance().addThingsToDo(candidate.toString(), c, index);
    }else{
        TestInfo.getInstance().addThingsToDo(candidate.toString(), c);
    }
    dispose();
```

```
        break;
    }
}

private void msgBox(String msg) {
    JOptionPane.showMessageDialog(((Component) this), msg);
}
}
```