Kpi-best

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра обчислювальної техніки

Лабораторна робота №1,2

з Теорії Проектування Комп’ютерних Систем-1

Виконав студент групи ІО-11

Ротенберг О.В

Номер з.к - 1128

Київ — 2014

***Мета роботи:***

Здобуття навичок з побудови редактора блок-схем алгоритмів. Розробка інтерфейсу користувача та функціонального наповнення. Розробка засобів перетворення форматів зберігання даних.

***Варіант***

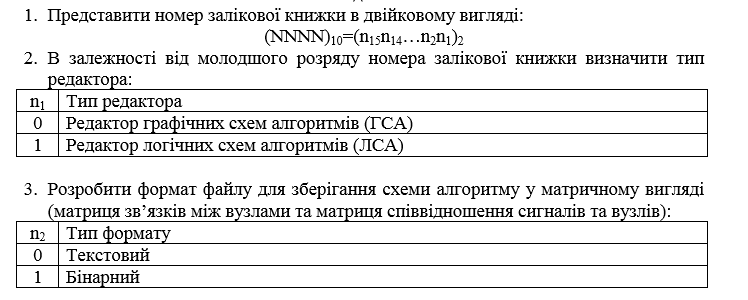
112810 = 100011010002

n1=0

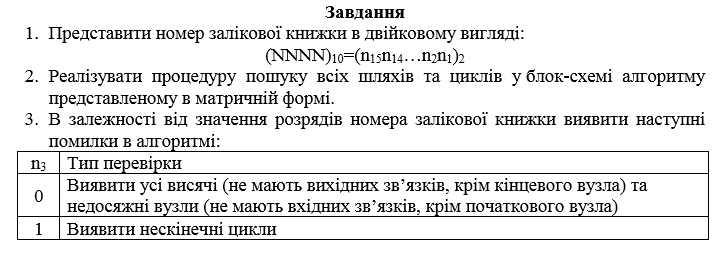
n2=0

n3=0

***Лабораторна Робота №1***



***Лабораторна Робота №2***



Лістинг:

package redactor.gui;

import redactor.Mediator;

import redactor.Message;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

/\*\*

\* Class MainFrame defines and describes the behavior of the main window.

\* Includes a description of the menu bar and toolbar.

\*/

public class MainFrame extends JFrame{

private Mediator mediator;

/\*\*

\* Buttons - the list of toolbar buttons (can be disabled).

\*/

private ArrayList<JButton> buttons;

private JMenuItem saveAlgo, saveGraph, setCondition, setOperator, controlEntry, findEndlessCycles, showMSA,allWays,allCycles,showGraph, saveMsa;

/\*\*

\* ShowGrid - indicator of display grid, information about it should be available

\* Class Canvas, for the correct drawing (can be disabled).

\*/

private JCheckBoxMenuItem showGrid;

/\*\*

\* This method sets setMenuEnabled availability of menu buttons and toolbars

\* that change their availability depending of the operating mode PROGRAM IS.

\* Called when need lock or unlock the key part of the main window

\* @Param b category access

\*/

JToolBar toolBar;

JMenuBar menuBar;

public static final int toolBarStateChangedId = 1;

public static final int newFileId = 2;

public static final int openAlgoFileId = 3;

public static final int openGraphFileId = 4;

public static final int saveAlgoFileId = 5;

public static final int saveGraphFileId = 6;

public static final int exitId = 7;

public static final int setConditionId = 8;

public static final int setOperatorId = 9;

public static final int controlEntryId = 10;

// public static final int findEndlessCyclesId = 11;

public static final int showMSAId = 12;

public static final int findAllWaysId = 13;

public static final int findAllCyclesId = 14;

// public static final int showGraphId = 15;

public static final int aboutId = 17;

public static final int newOperationDialog = 19;

public static final int newConditionDialog = 22;

public static final int saveMsaId = 23;

public void setMenuEnabled(final boolean b) {

saveAlgo.setEnabled(b);

// saveGraph.setEnabled(b);

setCondition.setEnabled(b);

setOperator.setEnabled(b);

controlEntry.setEnabled(b);

// findEndlessCycles.setEnabled(b);

// showGrid.setEnabled(b);

showMSA.setEnabled(b);

// allWays.setEnabled(b);

// allCycles.setEnabled(b);

// showGraph.setEnabled(b);

saveMsa.setEnabled(b);

for (JButton button : buttons) {

button.setEnabled(b);

}

}

/\*\*

\* isGridSelected

\* check whether the menu item is selected the program display grid.

\* @return checkbox state

\*/

public boolean isGridSelected() {

return showGrid.isSelected();

}

public MainFrame(){

// set size and title of main windows

Toolkit kit = Toolkit.getDefaultToolkit();

Dimension screenSize = kit.getScreenSize();

setBounds(screenSize.width / 8, screenSize.height / 8, 6 \* screenSize.width / 8, 6 \* screenSize.height / 8);

setTitle("GSA Editor");

// link to application mediator

mediator = new Mediator(this);

// create panel, define reaction on button pressed

buttons = new ArrayList<JButton>();

toolBar = new JToolBar(SwingConstants.VERTICAL);

toolBar.setEnabled(false);

addDrawButton("cursor.gif",1,"Cursor");

addDrawButton("operator.gif",2,"OperationalVertex Y");

addDrawButton("condition.gif",3,"Condition X");

addDrawButton("line.gif",4,"Line");

addDrawButton("directedline.gif",5,"Directed Line");

buttons.get(3).setVisible(false);

buttons.get(4).setVisible(false);

menuBar = new JMenuBar();

JMenu menu = new JMenu("File");

menuBar.add(menu);

addMenuItem(menu,"New",newFileId);

addMenuItem(menu,"Open",openAlgoFileId);

saveAlgo = new JMenuItem(createActionForItem("Save",saveAlgoFileId));

menu.add(saveAlgo);

saveMsa = new JMenuItem(createActionForItem("Save as MSA",saveMsaId));

menu.add(saveMsa);

// addMenuItem(menu,"Open algorithm",openAlgoFileId);

// addMenuItem(menu,"Open graph",openGraphFileId);

// saveAlgo = new JMenuItem(createActionForItem("Save algorithm",saveAlgoFileId));

// menu.add(saveAlgo);

// saveGraph = new JMenuItem(createActionForItem("Save graph",saveGraphFileId));

// menu.add(saveGraph);

addMenuItem(menu,"Exit",exitId);

menu = new JMenu("Edit");

menuBar.add(menu);

setCondition = new JMenuItem(createActionForItem("Set conditions",newConditionDialog));

menu.add(setCondition);

setOperator = new JMenuItem(createActionForItem("Set operators",newOperationDialog));

menu.add(setOperator);

controlEntry = new JMenuItem(createActionForItem("Mistake control",controlEntryId));

menu.add(controlEntry);

// findEndlessCycles = new JMenuItem(createActionForItem("Find endless cycles",findEndlessCyclesId));

// menu.add(findEndlessCycles);

menu = new JMenu("View");

menuBar.add(menu);

showMSA = new JMenuItem(createActionForItem("Show MSA",showMSAId));

menu.add(showMSA);

/\*allWays = new JMenuItem(createActionForItem("Show all ways",findAllWaysId));

menu.add(allWays);

allCycles = new JMenuItem(createActionForItem("Show all cycles",findAllCyclesId));

menu.add(allCycles);

showGraph = new JMenuItem(createActionForItem("Show graph",showGraphId));

menu.add(showGraph);\*/

addShowGridItem(menu);

menu = new JMenu("Help");

menuBar.add(menu);

Action action = new AbstractAction("About") {

public void actionPerformed(final ActionEvent event) {

JOptionPane.showMessageDialog(MainFrame.this, "GSA builder v1.0", "Product", JOptionPane.INFORMATION\_MESSAGE);

}

};

JMenuItem item = new JMenuItem(action);

menu.add(item);

// addMenuItem(menu,"About",aboutId);

// add created structures to frame

setMenuEnabled(false);

add(mediator.getCanvas());

setJMenuBar(menuBar);

add(toolBar, BorderLayout.WEST);

}

public Action createActionForItem(String menuName, final int typeOfMessage){

return new AbstractAction(menuName) {

public void actionPerformed(final ActionEvent event) {

Message message = new Message(typeOfMessage);

mediator.update(message);

}

};

}

public void addShowGridItem(JMenu m){

showGrid = new JCheckBoxMenuItem("Show Grid");

showGrid.setSelected(true);

showGrid.setEnabled(false);

showGrid.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent event) {

Message message = new Message(Message.showGrid);

mediator.update(message);

}

});

// m.add(showGrid);

}

public void addMenuItem(JMenu m, String menuName, final int typeOfMessage){

Action action = new AbstractAction(menuName) {

public void actionPerformed(final ActionEvent event) {

Message message = new Message(typeOfMessage);

mediator.update(message);

}

};

JMenuItem item = new JMenuItem(action);

m.add(item);

}

public void addDrawButton(String nameOfGifImage, final int numberOfButton, String shortDiscrioption){

Action action = new AbstractAction(null, new ImageIcon(nameOfGifImage)) {

public void actionPerformed(final ActionEvent event) {

for (JButton button : buttons) {

button.setBackground(Color.WHITE);

}

Message message = new Message(Message.toolBarStateChanged);

message.setToolBarState(numberOfButton);

mediator.update(message);

buttons.get(numberOfButton-1).setBackground(Color.GRAY);

}

};

action.putValue(Action.SHORT\_DESCRIPTION, shortDiscrioption);

JButton button = new JButton(action);

button.setBackground(Color.WHITE);

buttons.add(button);

toolBar.add(button);

}

}

**public** String getInitialValue() {

**return** initialValue;

}

@Override

**public** **boolean** hasErrors() {

**return** initialValue == **null** || initialValue.isEmpty()

|| inputLinks.isEmpty() || outputLink == **null**; //To change body of implemented methods use File | Settings | File Templates.

}

@Override

**public** String getErrors() {

StringBuilder sb = **new** StringBuilder();

**if** (initialValue == **null** || initialValue.isEmpty()) {

sb.append("Value not set");

}

**if** (inputLinks.isEmpty()) {

**if** (sb.length() > 0)

sb.append("; ");

sb.append("Vertex is unreachable");

}

**if** (outputLink == **null**) {

**if** (sb.length() > 0)

sb.append("; ");

sb.append("Output not used");

}

**return** sb.toString();

}

}