МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

Кафедра "Обчислювальна техніка та програмування"

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТП

\_\_\_\_\_\_\_\_\_\_ проф.Семенов С. Г.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2019 р.

РОЗРОБКА ГРАФІЧНОГО РЕДАКТОРУ

Текст програми

Лист затвердження

КІТЗ7.15325-01 12 01-1-ЛЗ

Розробники:

Керівник

\_\_\_\_\_\_\_\_\_\_ Молчанов Г.І.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2019 р.

Виконавець

\_\_\_\_\_\_\_\_\_\_ Дерюга М.В.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2019 р.

Харків 2019

ЗАТВЕРДЖЕНО

КІТЗ7.15325-01 12 01-1-ЛЗ

РОЗРОБКА ГРАФІЧНОГО РЕДАКТОРУ

Текст програми

КІТЗ7.15325-01 12 01-1-ЛЗ

Аркушів 25

Харків 2019

АНОТАЦІЯ

Даний документ входить до комплекту документів курсового проекту на тему "Розробка програмного забезпечення з інформаційно-довідковою системою".

Програма розроблена як проект для середовища ECLIPCE.

У процесі розробки використовувалися мова програмування JAVA.

ЗМІСТ

[1   Файл ImageEdit.java…………………………………………………………...4](#_Toc406601793)

[2   Файл ColorDialog.java……………………………………………………….](#_Toc406601794)20

[3   Файл NewFrame.java…………………………………………………………21](#_Toc406601795)

[4   Файл NewPanel.java………………………………………………………….2](#_Toc406601796)2

[5   Файл TextFileFilter.java………………………………………...……………2](#_Toc406601793)3

ImageEdit.java

import java.awt.\*;

import java.awt.event.\*;

import java.awt.image.\*;

import java.io.\*;

import javax.swing.\*;

import javax.swing.event.\*;

import javax.imageio.\*;

public class ImageEdit {

private int mode = 0;

/\*\* РљРѕРѕСЂРґРёРЅР°С‚С‹ РЅР°С‡Р°Р»Р° Рё РєРѕРЅС†Р° РѕС‚СЂРµР·РєРѕРІ. \*/

private int xOne, xTwo, yTwo, yOne;

/\*\* Р¤Р»Р°Рі СЃРѕР±С‹С‚РёСЏ РЅР°Р¶Р°С‚РёСЏ РєР»Р°РІРёС€Рё. \*/

private boolean pressed = false;

/\*\* Р¤Р»Р°Рі СЃРѕР±С‹С‚РёСЏ Р·Р°РіСЂСѓР·РєРё. \*/

private boolean loading = false;

/\*\* Р­РєР·РµРјРїР»СЏСЂ РєР»Р°СЃСЃР° Color. \*/

private Color color;

/\*\* Р­РєР·РµРјРїР»СЏСЂ РєР»Р°СЃСЃР° NewFrame. \*/

private NewFrame newFrame;

/\*\* Р­РєР·РµРјРїР»СЏСЂ РєР»Р°СЃСЃР° NewPanel. \*/

private NewPanel newPanel;

/\*\* Р­РєР·РµРјРїР»СЏСЂ РєР»Р°СЃСЃР° JButton. \*/

private JButton jButton;

/\*\* Р­РєР·РµРјРїР»СЏСЂ РєР»Р°СЃСЃР° JColorChooser. \*/

private static JColorChooser jColorChooser;

/\*\* Р­РєР·РµРјРїР»СЏСЂ РєР»Р°СЃСЃР° BufferedImage. \*/

private static BufferedImage bufferedImage;

/\*\* РҐСЂР°РЅРёС‚ РёРјСЏ С„Р°Р№Р»Р°. \*/

private String fileName;

/\*\* Р�РЅРёС†РёР°Р»РёР·Р°С†РёСЏ РЅРѕРІРѕР№ РІРєР»Р°РґРєРё "File". \*/

private JMenu fileMenu = new JMenu("File");

/\*\* Р�РЅРёС†РёР°Р»РёР·Р°С†РёСЏ РЅРѕРІРѕР№ РІРєР»Р°РґРєРё "About". \*/

private JMenu aboutMenu = new JMenu("About");

/\*\* Р�РЅРёС†РёР°Р»РёР·Р°С†РёСЏ РЅРѕРІРѕР№ РїР°РЅРµР»Рё РёРЅСЃС‚СЂСѓРјРµРЅС‚РѕРІ. \*/

private JToolBar toolBar = new JToolBar("Toolbar", JToolBar.VERTICAL);

/\*\* Р�РЅРёС†РёР°Р»РёР·Р°С†РёСЏ РЅРѕРІРѕР№ РїР°РЅРµР»Рё РІС‹Р±РѕСЂР° С†РІРµС‚Р°. \*/

private JToolBar colorBar = new JToolBar("Colorbar", JToolBar.HORIZONTAL);

/\*\* Р�РЅС„РѕСЂРјР°С†РёСЏ Рѕ СЂР°Р·СЂР°Р±РѕС‚С‡РёРєРµ РїСЂРѕРіСЂР°РјРјС‹. \*/

private static final String AUTHOR\_INFO = "Developed by student of group CIT-37, Deriuha Maxim";

/\*\*

\* РЎРѕР·РґР°РЅРёРµ, СѓСЃС‚Р°РЅРѕРІРєР° СЂР°Р·РјРµСЂР°, СЂР°Р·РјРµС‰РµРЅРёСЏ Рё С†РІРµС‚Р° РґР»СЏ РіР»Р°РІРЅРѕРіРѕ РѕРєРЅР°, РїР°РЅРµР»Рё

\* РёРЅСЃС‚СЂСѓРјРµРЅС‚РѕРІ, РїР°РЅРµР»Рё С†РІРµС‚Р° Рё РІРєР»Р°РґРѕРє.

\*/

public ImageEdit() {

color = Color.BLACK;

newFrame = new NewFrame("Graphical Redactor");

newFrame.setSize(800, 500);

newFrame.setLocation(200, 100);

newFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JMenuBar jMenuBar = new JMenuBar();

newFrame.setJMenuBar(jMenuBar);

jMenuBar.setBounds(200, 0, 100, 500);

jMenuBar.add(fileMenu);

jMenuBar.add(aboutMenu);

newPanel = new NewPanel();

newPanel.setBounds(30, 30, 260, 260);

newPanel.setBackground(Color.white);

newPanel.setOpaque(true);

newFrame.add(newPanel);

toolBar.setBounds(0, 0, 30, 300);

colorBar.setBounds(30, 0, 260, 40);

jButton = new JButton();

jButton.setBackground(color);

jButton.setBounds(15, 5, 20, 20);

}

/\*\*

\* Р’РѕР·РІСЂР°С‰Р°РµС‚ Р·РЅР°С‡РµРЅРёРµ РїРѕР»СЏ.

\*

\* @return С‚РµРєСѓС‰РµРµ Р·РЅР°С‡РµРЅРёРµ.

\*/

public static JColorChooser getJColorChooser() {

return jColorChooser;

}

/\*\*

\* РЈСЃС‚Р°РЅРѕРІРєР° Р·РЅР°С‡РµРЅРёСЏ РґР»СЏ РїРѕР»СЏ.

\*

\* @param jColorChooser

\* - Р·РЅР°С‡РµРЅРёРµ РґР»СЏ РёРЅРёС†РёР°Р»РёР·Р°С†РёРё.

\*/

public static void setJColorChooser(final JColorChooser jColorChooser) {

ImageEdit.jColorChooser = jColorChooser;

}

/\*\*

\* Р’РѕР·РІСЂР°С‰Р°РµС‚ Р·РЅР°С‡РµРЅРёРµ РїРѕР»СЏ.

\*

\* @return С‚РµРєСѓС‰РµРµ Р·РЅР°С‡РµРЅРёРµ.

\*/

public static BufferedImage getBufferedImage() {

return bufferedImage;

}

/\*\*

\* РЈСЃС‚Р°РЅРѕРІРєР° Р·РЅР°С‡РµРЅРёСЏ РґР»СЏ РїРѕР»СЏ.

\*

\* @param bufferedImage

\* - Р·РЅР°С‡РµРЅРёРµ РґР»СЏ РёРЅРёС†РёР°Р»РёР·Р°С†РёРё.

\*/

public static void setBufferedImage(final BufferedImage bufferedImage) {

ImageEdit.bufferedImage = bufferedImage;

}

/\*\*

\* Р’РѕР·РІСЂР°С‰Р°РµС‚ Р·РЅР°С‡РµРЅРёРµ РїРѕР»СЏ.

\*

\* @return С‚РµРєСѓС‰РµРµ Р·РЅР°С‡РµРЅРёРµ.

\*/

public int getMode() {

return mode;

}

/\*\*

\* РЈСЃС‚Р°РЅРѕРІРєР° Р·РЅР°С‡РµРЅРёСЏ РґР»СЏ РїРѕР»СЏ.

\*

\* @param mode

\* - СЂРµР¶РёРј СЂРёСЃРѕРІР°РЅРёСЋ.

\*/

public final void setMode(final int mode) {

this.mode = mode;

}

/\*\*

\* РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёРµ Р·Р°РіСЂСѓР·РєРё РёР·РѕР±СЂР°Р¶РµРЅРёР№.

\*/

public void loadAction() {

Action loadAction = new AbstractAction("Load...") {

/\*\* РђРІС‚РѕРјР°С‚РёС‡РµСЃРєРё СЃРіРµРЅРµСЂРёСЂРѕРІР°РЅРЅР°СЏ РєРѕРЅСЃС‚Р°РЅС‚Р°. \*/

private static final long serialVersionUID = 1L;

@Override

public void actionPerformed(final ActionEvent event) {

JFileChooser jFileChooser = new JFileChooser();

int result = jFileChooser.showOpenDialog(null);

if (result == JFileChooser.APPROVE\_OPTION)

try {

fileName = jFileChooser.getSelectedFile().getAbsolutePath();

File file = new File(fileName);

jFileChooser.addChoosableFileFilter(new TextFileFilter(".png"));

jFileChooser.addChoosableFileFilter(new TextFileFilter(".jpg"));

setBufferedImage(ImageIO.read(file));

loading = true;

newFrame.setSize(getBufferedImage().getWidth() + 40, getBufferedImage().getWidth() + 80);

newPanel.setSize(getBufferedImage().getWidth(), getBufferedImage().getWidth());

newPanel.repaint();

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(newFrame, "Error! File not found");

} catch (IOException ex) {

JOptionPane.showMessageDialog(newFrame, "Error! I/O Exception");

} catch (Exception ex) {

JOptionPane.showMessageDialog(newFrame, "Error! " + ex.getMessage());

}

}

};

JMenuItem loadMenu = new JMenuItem(loadAction);

fileMenu.add(loadMenu);

}

/\*\*

\* РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёРµ СЃРѕС…СЂР°РЅРµРЅРёСЏ РёР·РѕР±СЂР°Р¶РµРЅРёР№.

\*/

public void saveAction() {

Action saveAction = new AbstractAction("Save") {

/\*\* РђРІС‚РѕРјР°С‚РёС‡РµСЃРєРё СЃРіРµРЅРµСЂРёСЂРѕРІР°РЅРЅР°СЏ РєРѕРЅСЃС‚Р°РЅС‚Р°. \*/

private static final long serialVersionUID = 1L;

@Override

public void actionPerformed(final ActionEvent event) {

try {

JFileChooser jFileChooser = new JFileChooser();

TextFileFilter pngFilter = new TextFileFilter(".png");

TextFileFilter jpgFilter = new TextFileFilter(".jpg");

if (fileName == null) {

jFileChooser.addChoosableFileFilter(pngFilter);

jFileChooser.addChoosableFileFilter(jpgFilter);

int result = jFileChooser.showSaveDialog(null);

if (result == JFileChooser.APPROVE\_OPTION)

fileName = jFileChooser.getSelectedFile().getAbsolutePath();

}

if (jFileChooser.getFileFilter() == pngFilter)

ImageIO.write(getBufferedImage(), "png", new File(fileName + ".png"));

else

ImageIO.write(getBufferedImage(), "jpeg", new File(fileName + ".jpg"));

} catch (IOException ex) {

JOptionPane.showMessageDialog(newFrame, "Error! I/O Exception");

}

}

};

JMenuItem saveMenu = new JMenuItem(saveAction);

fileMenu.add(saveMenu);

}

/\*\*

\* РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёРµ СЃРѕС…СЂР°РЅРµРЅРёСЏ РёР·РѕР±СЂР°Р¶РµРЅРёР№ СЃ РїР°СЂР°РјРµС‚СЂР°РјРё.

\*/

public void saveAsAction() {

Action saveAsAction = new AbstractAction("Save as...") {

/\*\* РђРІС‚РѕРјР°С‚РёС‡РµСЃРєРё СЃРіРµРЅРµСЂРёСЂРѕРІР°РЅРЅР°СЏ РєРѕРЅСЃС‚Р°РЅС‚Р°. \*/

private static final long serialVersionUID = 1L;

@Override

public void actionPerformed(final ActionEvent event) {

try {

JFileChooser jFileChooser = new JFileChooser();

TextFileFilter pngFilter = new TextFileFilter(".png");

TextFileFilter jpgFilter = new TextFileFilter(".jpg");

jFileChooser.addChoosableFileFilter(pngFilter);

jFileChooser.addChoosableFileFilter(jpgFilter);

int result = jFileChooser.showSaveDialog(null);

if (result == JFileChooser.APPROVE\_OPTION)

fileName = jFileChooser.getSelectedFile().getAbsolutePath();

if (jFileChooser.getFileFilter() == pngFilter)

ImageIO.write(getBufferedImage(), "png", new File(fileName + ".png"));

else

ImageIO.write(getBufferedImage(), "jpeg", new File(fileName + ".jpg"));

} catch (IOException ex) {

JOptionPane.showMessageDialog(newFrame, "Error! I/O Exception");

}

}

};

JMenuItem saveasMenu = new JMenuItem(saveAsAction);

fileMenu.add(saveasMenu);

}

/\*\*

\* РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёРµ РІС‹РІРѕРґР° РёРЅС„РѕСЂРјР°С†РёРё Рѕ Р°РІС‚РѕСЂРµ.

\*/

public void aboutAuthorAction() {

Action aboutAuthorAction = new AbstractAction("About author") {

/\*\* РђРІС‚РѕРјР°С‚РёС‡РµСЃРєРё СЃРіРµРЅРµСЂРёСЂРѕРІР°РЅРЅР°СЏ РєРѕРЅСЃС‚Р°РЅС‚Р°. \*/

private static final long serialVersionUID = 1L;

@Override

public void actionPerformed(final ActionEvent event) {

JOptionPane.showMessageDialog(null, AUTHOR\_INFO);

}

};

JMenuItem aboutAuthorMenu = new JMenuItem(aboutAuthorAction);

aboutMenu.add(aboutAuthorMenu);

}

/\*\*

\* Р”РѕР±Р°РІР»СЏРµС‚ С”Р»РµРјРµРЅС‚С‹ РЅР° РїР°РЅРµР»СЊ РёРЅСЃС‚СЂСѓРјРµРЅС‚РѕРІ.

\*/

public void toolBar() {

// Р”РѕР±Р°РІР»РµРЅРёРµ СЂСѓС‡РєРё.

JButton penButton = new JButton(new ImageIcon("pen.png"));

penButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

setMode(0);

}

});

toolBar.add(penButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ РєРёСЃС‚Рё.

JButton brushButton = new JButton(new ImageIcon("brush.png"));

brushButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

setMode(1);

}

});

toolBar.add(brushButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ Р»Р°СЃС‚РёРєР°.

JButton eraserButton = new JButton(new ImageIcon("eraser.png"));

eraserButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

setMode(2);

}

});

toolBar.add(eraserButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ С‚РµРєСЃС‚Р°.

JButton textButton = new JButton(new ImageIcon("text.png"));

textButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

setMode(3);

}

});

toolBar.add(textButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ Р»РёРЅРёРё.

JButton lineButton = new JButton(new ImageIcon("line.png"));

lineButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

setMode(4);

}

});

toolBar.add(lineButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ СЌР»РёРїСЃР°.

JButton ellipseButton = new JButton(new ImageIcon("ellipse.png"));

ellipseButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

setMode(5);

}

});

toolBar.add(ellipseButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ РїСЂСЏРјРѕСѓРіРѕР»СЊРЅРёРєР°.

JButton rectangleButton = new JButton(new ImageIcon("rectangle.png"));

rectangleButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

setMode(6);

}

});

toolBar.add(rectangleButton);

newFrame.add(toolBar);

}

/\*\*

\* Р”РѕР±Р°РІР»СЏРµС‚ СЌР»РµРјРµРЅС‚С‹ РЅР° РїР°РЅРµР»СЊ С†РІРµС‚Р°.

\*/

public void colorBar() {

// Р”РѕР±Р°РІР»РµРЅРёРµ РґРёР°Р»РѕРіР° РІС‹Р±РѕСЂР° С†РІРµС‚Р°.

jButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

ColorDialog colorDialog = new ColorDialog(newFrame, "Color selection");

colorDialog.setVisible(true);

}

});

colorBar.add(jButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ РєСЂР°СЃРЅРѕРіРѕ С†РІРµС‚Р°.

JButton redButton = new JButton();

redButton.setBackground(Color.RED);

redButton.setBounds(40, 5, 15, 15);

redButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.RED;

jButton.setBackground(color);

}

});

colorBar.add(redButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ РѕСЂР°РЅР¶РµРІРѕРіРѕ С†РІРµС‚Р°.

JButton orangeButton = new JButton();

orangeButton.setBackground(Color.ORANGE);

orangeButton.setBounds(60, 5, 15, 15);

orangeButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.ORANGE;

jButton.setBackground(color);

}

});

colorBar.add(orangeButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ Р¶С‘Р»С‚РѕРіРѕ С†РІРµС‚Р°.

JButton yellowButton = new JButton();

yellowButton.setBackground(Color.YELLOW);

yellowButton.setBounds(80, 5, 15, 15);

yellowButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.YELLOW;

jButton.setBackground(color);

}

});

colorBar.add(yellowButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ Р·РµР»С‘РЅРѕРіРѕ С†РІРµС‚Р°.

JButton greenButton = new JButton();

greenButton.setBackground(Color.GREEN);

greenButton.setBounds(100, 5, 15, 15);

greenButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.GREEN;

jButton.setBackground(color);

}

});

colorBar.add(greenButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ СЃРёРЅРµРіРѕ С†РІРµС‚Р°.

JButton blueButton = new JButton();

blueButton.setBackground(Color.BLUE);

blueButton.setBounds(120, 5, 15, 15);

blueButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.BLUE;

jButton.setBackground(color);

}

});

colorBar.add(blueButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ С†РІРµС‚Р° СЃРёР°РЅ.

JButton cyanButton = new JButton();

cyanButton.setBackground(Color.CYAN);

cyanButton.setBounds(140, 5, 15, 15);

cyanButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.CYAN;

jButton.setBackground(color);

}

});

colorBar.add(cyanButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ С†РІРµС‚Р° РјР°РґР¶РµРЅС‚Р°.

JButton magentaButton = new JButton();

magentaButton.setBackground(Color.MAGENTA);

magentaButton.setBounds(160, 5, 15, 15);

magentaButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.MAGENTA;

jButton.setBackground(color);

}

});

colorBar.add(magentaButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ Р±РµР»РѕРіРѕ С†РІРµС‚Р°.

JButton whiteButton = new JButton();

whiteButton.setBackground(Color.WHITE);

whiteButton.setBounds(180, 5, 15, 15);

whiteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.WHITE;

jButton.setBackground(color);

}

});

colorBar.add(whiteButton);

// Р”РѕР±Р°РІР»РµРЅРёРµ С‡С‘СЂРЅРѕРіРѕ С†РІРµС‚Р°.

JButton blackButton = new JButton();

blackButton.setBackground(Color.BLACK);

blackButton.setBounds(200, 5, 15, 15);

blackButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(final ActionEvent event) {

color = Color.BLACK;

jButton.setBackground(color);

}

});

colorBar.add(blackButton);

colorBar.setLayout(null);

newFrame.add(colorBar);

}

/\*\*

\* РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёСЏ РјС‹С€Рё.

\*/

public void drawing() {

setJColorChooser(new JColorChooser(color));

getJColorChooser().getSelectionModel().addChangeListener(new ChangeListener() {

@Override

public void stateChanged(final ChangeEvent e) {

color = getJColorChooser().getColor();

jButton.setBackground(color);

}

});

newPanel.addMouseMotionListener(new MouseMotionAdapter() {

// РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёРµ РїРµСЂРµС‚Р°СЃРєРёРІР°РЅРёСЏ РјС‹С€Рё.

@Override

public void mouseDragged(final MouseEvent e) {

if (pressed) {

Graphics graphics = getBufferedImage().getGraphics();

Graphics2D graphicsTwoD = (Graphics2D) graphics;

graphicsTwoD.setColor(color);

switch (getMode()) {

case 0:

graphicsTwoD.drawLine(xOne, yOne, e.getX(), e.getY());

break;

case 1:

graphicsTwoD.setStroke(new BasicStroke(3.0f));

graphicsTwoD.drawLine(xOne, yOne, e.getX(), e.getY());

break;

case 2:

graphicsTwoD.setStroke(new BasicStroke(3.0f));

graphicsTwoD.setColor(Color.WHITE);

graphicsTwoD.drawLine(xOne, yOne, e.getX(), e.getY());

break;

default:

break;

}

xOne = e.getX();

yOne = e.getY();

}

newPanel.repaint();

}

});

// РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёСЏ РЅР°Р¶Р°С‚РёСЏ Рё РѕС‚Р¶Р°С‚РёСЏ РјС‹С€Рё.

newPanel.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(final MouseEvent e) {

Graphics graphics = getBufferedImage().getGraphics();

Graphics2D graphicsTwoD = (Graphics2D) graphics;

graphicsTwoD.setColor(color);

switch (getMode()) {

case 0:

graphicsTwoD.drawLine(xOne, yOne, xOne + 1, yOne + 1);

break;

case 1:

graphicsTwoD.setStroke(new BasicStroke(3.0f));

graphicsTwoD.drawLine(xOne, yOne, xOne + 1, yOne + 1);

break;

case 2:

graphicsTwoD.setStroke(new BasicStroke(3.0f));

graphicsTwoD.setColor(Color.WHITE);

graphicsTwoD.drawLine(xOne, yOne, xOne + 1, yOne + 1);

break;

case 3:

newPanel.requestFocus();

break;

default:

break;

}

xOne = e.getX();

yOne = e.getY();

pressed = true;

newPanel.repaint();

}

@Override

public void mousePressed(final MouseEvent e) {

xOne = e.getX();

yOne = e.getY();

xTwo = e.getX();

yTwo = e.getY();

pressed = true;

}

@Override

public void mouseReleased(final MouseEvent e) {

Graphics graphics = getBufferedImage().getGraphics();

Graphics2D graphicsTwoD = (Graphics2D) graphics;

graphicsTwoD.setColor(color);

int x1 = xTwo, x2 = xOne, y1 = yTwo, y2 = yOne;

if (xTwo > xOne) {

x2 = xTwo;

x1 = xOne;

}

if (yTwo > yOne) {

y2 = yTwo;

y1 = yOne;

}

switch (getMode()) {

case 4:

graphics.drawLine(xTwo, yTwo, e.getX(), e.getY());

break;

case 5:

graphics.drawOval(x1, y1, (x2 - x1), (y2 - y1));

break;

case 6:

graphics.drawRect(x1, y1, (x2 - x1), (y2 - y1));

break;

default:

break;

}

xTwo = 0;

yTwo = 0;

pressed = false;

newPanel.repaint();

}

});

// РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёСЏ РЅР°Р¶Р°С‚РёСЏ Рё РѕС‚Р¶Р°С‚РёСЏ РєР»Р°РІРёС€ РїСЂРё РІРІРѕРґРµ СЃРёРјРІРѕР»РѕРІ.

newPanel.addKeyListener(new KeyAdapter() {

@Override

public void keyReleased(final KeyEvent e) {

newPanel.requestFocus();

}

@Override

public void keyTyped(final KeyEvent e) {

if (getMode() == 3) {

Graphics graphics = getBufferedImage().getGraphics();

Graphics2D graphicsTwoD = (Graphics2D) graphics;

graphicsTwoD.setColor(color);

graphicsTwoD.setStroke(new BasicStroke(2.0f));

String str = new String("");

str += e.getKeyChar();

graphicsTwoD.setFont(new Font("Verdana", 0, 16));

graphicsTwoD.drawString(str, xOne, yOne);

xOne += 10;

newPanel.requestFocus();

newPanel.repaint();

}

}

});

// РћР±СЂР°Р±Р°С‚С‹РІР°РµС‚ СЃРѕР±С‹С‚РёРµ РёР·РјРµРЅРµРЅРёСЏ СЂР°Р·РјРµСЂРѕРІ РєРѕРјРїРѕРЅРµРЅС‚РѕРІ РѕРєРЅР°.

newFrame.addComponentListener(new ComponentAdapter() {

@Override

public void componentResized(final ComponentEvent evt) {

if (!loading) {

newPanel.setSize(newFrame.getWidth() - 40, newFrame.getHeight() - 80);

BufferedImage tempImage = new BufferedImage(newPanel.getWidth(), newPanel.getHeight(),

BufferedImage.TYPE\_INT\_RGB);

Graphics2D graphicsTwoD = (Graphics2D) tempImage.createGraphics();

graphicsTwoD.setColor(Color.WHITE);

graphicsTwoD.fillRect(0, 0, newPanel.getWidth(), newPanel.getHeight());

tempImage.setData(getBufferedImage().getRaster());

setBufferedImage(tempImage);

newPanel.repaint();

}

loading = false;

}

});

newFrame.setLayout(null);

newFrame.setVisible(true);

}

/\*\*

\* РўРѕС‡РєР° РІС…РѕРґР° РІ РїСЂРѕРіСЂР°РјРјСѓ. Р’С‹Р·С‹РІР°РµС‚ С„СѓРЅРєС†РёСЋ run().

\*

\* @param args

\* - РїР°СЂР°РјРµС‚СЂС‹ Р·Р°РїСѓСЃРєР° РїСЂРѕРіСЂР°РјРјС‹.

\*/

public static void main(final String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

ImageEdit imageEdit = new ImageEdit();

imageEdit.loadAction();

imageEdit.saveAction();

imageEdit.saveAsAction();

imageEdit.aboutAuthorAction();

imageEdit.toolBar();

imageEdit.colorBar();

imageEdit.drawing();

}

});

}

}

ColorDialog.java

import javax.swing.JDialog;

import javax.swing.JFrame;

/\*\*

\* РЎРѕР·РґР°С‘С‚ РґРёР°Р»РѕРіРѕРІРѕРµ РѕРєРЅРѕ РІС‹Р±РѕСЂР° С†РІРµС‚Р°.

\*

\*

\* @version 1.0.0

\*/

public class ColorDialog extends JDialog {

/\*\* РђРІС‚РѕРјР°С‚РёС‡РµСЃРєРё СЃРіРµРЅРµСЂРёСЂРѕРІР°РЅРЅР°СЏ РєРѕРЅСЃС‚Р°РЅС‚Р°. \*/

private static final long serialVersionUID = 1L;

/\*\*

\* Р”РѕР±Р°РІР»СЏРµС‚ РЅРѕРІРѕРµ РѕРєРЅРѕ РІ РјР°СЃСЃРёРІ. РЈСЃС‚Р°РЅР°РІР»РёРІР°РµС‚ РµРіРѕ СЂР°Р·РјРµСЂ Рё РїРѕР»РѕР¶РµРЅРёРµ.

\*

\* @param jFrame

\* - РґРµСЃРєСЂРёРїС‚РѕСЂ РѕРєРЅР°.

\* @param title

\* - Р·Р°РіРѕР»РѕРІРѕРє РѕРєРЅР°.

\*/

public ColorDialog(final JFrame jFrame, final String title) {

super(jFrame, title, true);

add(ImageEdit.getJColorChooser());

setSize(600, 400);

setLocation(300, 150);

}

}

NewFrame.java

import java.awt.Graphics;

import javax.swing.JFrame;

/\*\*

\* Р РёСЃСѓРµС‚ РѕРєРЅР°.

\*

\*

\* @version 1.0.0

\*/

public class NewFrame extends JFrame {

/\*\* РђРІС‚РѕРјР°С‚РёС‡РµСЃРєРё СЃРіРµРЅРµСЂРёСЂРѕРІР°РЅРЅР°СЏ РєРѕРЅСЃС‚Р°РЅС‚Р°. \*/

private static final long serialVersionUID = 1L;

/\*\*

\* РќР°СЃР»РµРґСѓРµС‚ РєРѕРЅСЃС‚СЂСѓРєС‚РѕСЂ СЃСѓРїРµСЂРєР»Р°СЃСЃР°.

\*

\* @param title

\* - Р·Р°РіРѕР»РѕРІРѕРє РѕРєРЅР°.

\*/

public NewFrame(final String title) {

super(title);

}

@Override

public void paint(final Graphics graphics) {

super.paint(graphics);

}

}

NewPanel.java

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.image.BufferedImage;

import javax.swing.JPanel;

/\*\*

\* РЎРѕР·РґР°С‘С‚ РїР°РЅРµР»Рё.

\*

\*

\* @version 1.0.0

\*/

public class NewPanel extends JPanel {

/\*\* РђРІС‚РѕРјР°С‚РёС‡РµСЃРєРё СЃРіРµРЅРµСЂРёСЂРѕРІР°РЅРЅР°СЏ РєРѕРЅСЃС‚Р°РЅС‚Р°. \*/

private static final long serialVersionUID = 1L;

/\*\*

\* Empty constructor.

\*/

public NewPanel() {

// Nothing to do.

}

@Override

public void paintComponent(final Graphics g) {

if (ImageEdit.getBufferedImage() == null) {

ImageEdit

.setBufferedImage(new BufferedImage(this.getWidth(), this.getHeight(), BufferedImage.TYPE\_INT\_RGB));

Graphics2D d2 = (Graphics2D) ImageEdit.getBufferedImage().createGraphics();

d2.setColor(Color.white);

d2.fillRect(0, 0, this.getWidth(), this.getHeight());

}

super.paintComponent(g);

g.drawImage(ImageEdit.getBufferedImage(), 0, 0, this);

}

}

TextFileFilter.java

import java.io.File;

import javax.swing.filechooser.FileFilter;

/\*\*

\* РћС‚Р±РѕСЂ С„Р°Р№Р»РѕРІ РїРѕ СЂР°СЃС€РёСЂРµРЅРёСЋ.

\*

\*

\* @version 1.0.0

\*/

public class TextFileFilter extends FileFilter {

/\*\* Р Р°СЃС€РёСЂРµРЅРёРµ С„Р°Р№Р»Р°. \*/

private String ext;

/\*\*

\* Р�РЅРёС†РёР°Р»РёР·Р°С†РёСЏ РїРѕР»СЏ.

\*

\* @param ext

\* - Р·РЅР°С‡РµРЅРёРµ РґР»СЏ РёРЅРёС†РёР°Р»РёР·Р°С†РёРё.

\*/

public TextFileFilter(final String ext) {

this.ext = ext;

}

@Override

public boolean accept(final File file) {

if (file.isDirectory())

return true;

return (file.getName().endsWith(ext));

}

@Override

public String getDescription() {

return "\*" + ext;

}

}