package eReader;  
  
import org.apache.pdfbox.pdmodel.PDDocument;  
import org.apache.pdfbox.rendering.PDFRenderer;  
  
import javax.imageio.ImageIO;  
import javax.swing.\*;  
import javax.swing.border.LineBorder;  
import javax.swing.filechooser.FileFilter;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
import java.net.URL;  
  
import static java.awt.event.ActionEvent.*CTRL\_MASK*;  
  
  
public class FunctionalPanel extends JPanel {  
  
 private MyListener listener = new MyListener();  
 private JPanel toolBar;  
 private Color color;  
  
 private int currentPage, imgHeight, imgWidth;  
 private float scale;  
  
 private BufferedImage bufferedImage;  
 private PDFRenderer render;  
 private PDDocument document;  
 private JFileChooser fileChooser, fileChooser2;  
 private File file;  
  
 private JLabel label;  
 private JTextField textField;  
 private JButton button, menuHelp;  
 private JMenuBar menuBar;  
 private JMenu menuFile, menuView, submenuNav;  
 private JMenuItem fileOpen, fileClose, fileExit,  
 firstPage, lastPage, prevPage, nextPage, submenuScale;  
  
 // конструкор всей панели  
 protected FunctionalPanel() {  
  
 //CONSTR & KeyList plus focus on our panel  
 super();  
 this.addKeyListener(new FieldKeyListener()); // подключение клавиатуры  
 this.setFocusable(true); // фокус  
  
 //DEFAULT DATA  
 currentPage = -1;  
 imgWidth = imgHeight = 0;  
 scale = (float) 1.0424;  
 color = new Color(40, 40, 40);  
 bufferedImage = new BufferedImage(1, 1, BufferedImage.*TYPE\_BYTE\_BINARY*);  
  
 //FILE CHOOSE AND JFileChoose's SETTINGS  
 fileChooser = new JFileChooser();  
 fileChooser.setMultiSelectionEnabled(false);  
 fileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 fileChooser.setDialogTitle("Открыть");  
 fileChooser2 = new JFileChooser();  
 fileChooser2.setDialogTitle("Сохранить сттраницу как изображение");  
 fileChooser2.setMultiSelectionEnabled(false);  
  
 }  
  
 /\*\*/  
 @Override  
 protected void paintComponent(Graphics graphics) {  
  
 //Draw all the necessary components  
 super.paintComponent(graphics);  
 graphics.setColor(color);  
 graphics.fillRect(0, 0, getWidth(), getHeight());  
 graphics.drawImage(bufferedImage, (getWidth() / 2 - imgWidth / 2),  
 (getHeight() / 2 - imgHeight / 2), null);  
 }  
  
  
 protected JMenuBar createBarMenu() {  
  
 //  
 menuBar = new JMenuBar();  
 menuFile = new JMenu("Файл");  
 menuView = new JMenu("Просмотр");  
 submenuNav = new JMenu("Навигация");  
 menuHelp = new JButton("Справка");  
 submenuScale = new JMenuItem("Масштаб");  
 fileOpen = new JMenuItem("Открыть");  
 fileClose = new JMenuItem("Закрыть файл");  
 fileExit = new JMenuItem("Выйти");  
 firstPage = new JMenuItem("Первая страница");  
 lastPage = new JMenuItem("Последняя страница");  
 nextPage = new JMenuItem("Следующая страница");  
 prevPage = new JMenuItem("Предыдущая страница");  
  
 menuHelp.setActionCommand("Справка");  
 menuHelp.setAlignmentY(*CENTER\_ALIGNMENT*);  
 menuHelp.setAlignmentX(*LEFT\_ALIGNMENT*);  
 menuHelp.setFocusPainted(false);  
 menuHelp.setFont(menuFile.getFont());  
 menuHelp.setContentAreaFilled(false);  
 menuHelp.setMnemonic(KeyEvent.*VK\_F1*);  
  
 submenuNav.setEnabled(false);  
 submenuScale.setEnabled(false);  
 fileClose.setEnabled(false);  
 firstPage.setEnabled(false);  
 lastPage.setEnabled(false);  
 nextPage.setEnabled(false);  
 prevPage.setEnabled(false);  
  
 menuHelp.addActionListener(listener);  
 submenuScale.addActionListener(listener);  
 fileOpen.addActionListener(listener);  
 fileClose.addActionListener(listener);  
 fileExit.addActionListener(listener);  
 firstPage.addActionListener(listener);  
 lastPage.addActionListener(listener);  
 nextPage.addActionListener(listener);  
 prevPage.addActionListener(listener);  
  
 submenuScale.setAccelerator(KeyStroke.*getKeyStroke*(KeyEvent.*VK\_S*, *CTRL\_MASK*));  
 fileOpen.setAccelerator(KeyStroke.*getKeyStroke*(KeyEvent.*VK\_O*, *CTRL\_MASK*));  
 fileClose.setAccelerator(KeyStroke.*getKeyStroke*(KeyEvent.*VK\_W*, *CTRL\_MASK*));  
 fileExit.setAccelerator(KeyStroke.*getKeyStroke*(KeyEvent.*VK\_Q*, *CTRL\_MASK*));  
 firstPage.setAccelerator(KeyStroke.*getKeyStroke*(KeyEvent.*VK\_HOME*, *CTRL\_MASK*));  
 lastPage.setAccelerator(KeyStroke.*getKeyStroke*(KeyEvent.*VK\_END*, *CTRL\_MASK*));  
  
 menuFile.add(fileOpen);  
 menuFile.addSeparator();  
 menuFile.add(fileClose);  
 menuFile.add(fileExit);  
  
 menuView.add(submenuNav);  
 menuView.add(submenuScale);  
  
 submenuNav.add(firstPage);  
 submenuNav.add(lastPage);  
 submenuNav.add(nextPage);  
 submenuNav.add(prevPage);  
  
 menuBar.add(menuFile);  
 menuBar.add(menuView);  
 menuBar.add(Box.*createHorizontalGlue*());  
 menuBar.add(menuHelp);  
  
 return menuBar;  
 }  
  
 protected JPanel createToolBar() {  
  
 //  
 toolBar = new JPanel();  
 toolBar.setBackground(new Color(65, 65, 65));  
 addButtons(toolBar);  
 toolBar.setVisible(false);  
 return toolBar;  
 }  
  
 protected JButton makeButton(String imageName, String com) {  
  
 //  
 URL imageURL = FunctionalPanel.class.getResource("/images/" + imageName + ".png");  
 JButton button = new JButton();  
  
 button.setActionCommand(com);  
 button.setBorderPainted(false);  
 button.setBackground(new Color(63, 63, 63));  
 button.setPreferredSize(new Dimension(30, 30));  
 button.addActionListener(listener);  
 button.setContentAreaFilled(false);  
  
 if (com.equals("Увеличить масштаб")) {  
 button.setMnemonic(KeyEvent.*VK\_PLUS*);  
 } else if (com.equals("Уменьшить масштаб")) {  
 button.setMnemonic(KeyEvent.*VK\_MINUS*);  
 }  
  
 if (imageURL != null) {  
 button.setIcon(new ImageIcon(imageURL, com));  
 } else {  
 button.setText(com);  
 System.*err*.println("Resource not found images/" + imageName + ".png");  
 }  
  
 return button;  
 }  
  
 protected void addButtons(JPanel toolBar) {  
  
 //  
 //toolBar.add(Box.createHorizontalStrut(1));  
 button = makeButton("save", "Сохранить страницу");  
 toolBar.add(button);  
 toolBar.add(Box.*createHorizontalStrut*(20));  
  
 button = makeButton("lastPage", "Предыдущая страница");  
 toolBar.add(button);  
  
 button = makeButton("nextPage", "Следующая страница");  
 toolBar.add(button);  
 toolBar.add(Box.*createHorizontalStrut*(20));  
  
 textField = new JTextField("");  
 textField.setColumns(5);  
 textField.setBorder(new LineBorder(new Color(54, 54, 54)));  
 textField.setFont(new Font("TimesRoman", Font.*PLAIN*, 16));  
 textField.addActionListener(listener);  
 textField.setBackground(new Color(65, 65, 65));  
 textField.setActionCommand("Перейти");  
 textField.setForeground(new Color(255, 255, 255));  
 textField.setHorizontalAlignment(SwingConstants.*CENTER*);  
 toolBar.add(textField);  
  
 label = new JLabel();  
 label.setText("/ ");  
 label.setForeground(new Color(218, 227, 227));  
 label.setFont(new Font("TimesRoman", Font.*PLAIN*, 16));  
 toolBar.add(label);  
 toolBar.add(Box.*createHorizontalStrut*(20));  
  
 button = makeButton("inZoom", "Увеличить масштаб");  
 toolBar.add(button);  
  
 button = makeButton("outZoom", "Уменьшить масштаб");  
 toolBar.add(button);  
 }  
  
  
 protected class MyListener extends Component implements ActionListener {  
  
 private class JpgSaveFilter extends FileFilter {  
  
 //  
 @Override  
 public boolean accept(File f) {  
  
 if (f.isDirectory()) {  
 return false;  
 }  
 String s = f.getName();  
 return s.endsWith(".jpg") || s.endsWith(".JPG") || s.endsWith("jpeg") || s.endsWith("JPEG");  
 }  
  
 //  
 @Override  
 public String getDescription() {  
 return "\*.jpg,\*.JPG,\*.jpeg,\*.JPEG";  
 }  
 }  
  
 private class PdfFileFilter extends FileFilter {  
  
 //  
 @Override  
 public boolean accept(File f) {  
  
 if (f.isDirectory()) {  
 return false;  
 }  
 String s = f.getName();  
 return s.endsWith(".pdf") || s.endsWith(".PDF");  
 }  
  
 //  
 @Override  
 public String getDescription() {  
 return "\*.pdf,\*.PDF";  
 }  
 }  
  
 private void grabFocus() {  
 FunctionalPanel.this.getComponentAt(FunctionalPanel.this.getWidth() / 2,  
 FunctionalPanel.this.getHeight() / 2).requestFocus();  
 }  
  
 private void buttonEnDis(boolean onOff) {  
 fileClose.setEnabled(onOff);  
 submenuScale.setEnabled(onOff);  
 firstPage.setEnabled(onOff);  
 lastPage.setEnabled(onOff);  
 nextPage.setEnabled(onOff);  
 prevPage.setEnabled(onOff);  
 submenuNav.setEnabled(onOff);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 //  
 if (e.getActionCommand().equals("Справка")) {  
 JOptionPane.*showMessageDialog*(null,  
 " Структурное подразделение\n" +  
 " Новосибирского Государственного Университета\n" +  
 " Высший Колледж Информатики университета (ВКИ НГУ)\n" +  
 "\nПриложение 'eReader' предназначено для чтения PDF-файлов\n" +  
 "\nРазработали студенты 2-го курса группы 703са1\n" +  
 "Майер Артём и Нестеренко Мария © eReader 2018",  
 "eReader", JOptionPane.*PLAIN\_MESSAGE*);  
 FunctionalPanel.this.repaint();  
 grabFocus();  
 }  
  
 if (e.getActionCommand().equals("Открыть")) {  
  
 //  
 fileChooser.addChoosableFileFilter(new PdfFileFilter());  
  
 if (currentPage != -1) {  
  
 try {  
 document.close();  
 scale = (float) 1.0424;  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
  
 //  
  
 if (fileChooser.showOpenDialog(this) == JFileChooser.*APPROVE\_OPTION*) {  
  
 //  
 long startTime = System.*currentTimeMillis*();  
 currentPage = 0;  
 String ext = fileChooser.getFileFilter().getDescription();  
 file = fileChooser.getSelectedFile();  
  
 //  
 try {  
 document = PDDocument.*load*(file);  
 render = new PDFRenderer(document);  
 //вызов функции цикла проверки размера страниц в книге под вопросом  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 buttonEnDis(true);  
 toolBar.setVisible(true);  
 textField.setText("" + (currentPage + 1));  
 label.setText("/ " + (document.getNumberOfPages()));  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 grabFocus();  
 System.*out*.println("\n" + file.getAbsolutePath());  
 System.*out*.println("File's type: " + ext);  
 System.*out*.println("Resolution: " + bufferedImage.getHeight() +  
 "x" + bufferedImage.getWidth());  
 System.*out*.println("Open time: " + (System.*currentTimeMillis*() - startTime) + " мс");  
 }  
 }  
  
 if (e.getActionCommand().equals("Сохранить страницу")) {  
  
 //  
 fileChooser2.addChoosableFileFilter(new JpgSaveFilter());  
 if (fileChooser2.showSaveDialog(this) == JFileChooser.*APPROVE\_OPTION*) {  
  
 try {  
 ImageIO.*write*(bufferedImage, "jpg",  
 new File(fileChooser2.getCurrentDirectory().getAbsolutePath() +  
 "\\" + fileChooser2.getSelectedFile().getName() + ".jpg"));  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
 grabFocus();  
 }  
  
 if (e.getActionCommand().equals("Перейти")) {  
  
 //  
 try {  
 int tmp = Integer.*parseInt*(textField.getText());  
 if (tmp < 1 || tmp > document.getNumberOfPages()) {  
  
 JOptionPane.*showMessageDialog*(null, ("Нет страницы с номером " + tmp),  
 "eReader", JOptionPane.*ERROR\_MESSAGE*);  
 } else {  
  
 bufferedImage = render.renderImage(currentPage = tmp - 1, scale);  
 textField.setText("" + (currentPage + 1));  
 FunctionalPanel.this.repaint();  
 }  
 } catch (NumberFormatException nfe) {  
 JOptionPane.*showMessageDialog*(null,  
 ("Нет страницы с номером" + textField.getText()),  
 "eReader", JOptionPane.*ERROR\_MESSAGE*);  
 nfe.getStackTrace();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 grabFocus();  
 }  
  
 if (e.getActionCommand().equals("Первая страница")) {  
  
 //  
 if (currentPage != -1) {  
  
 currentPage = 0;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 textField.setText("" + (currentPage + 1));  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 grabFocus();  
 }  
 }  
  
 if (e.getActionCommand().equals("Последняя страница")) {  
  
 //  
 if (currentPage != -1) {  
  
 currentPage = document.getNumberOfPages() - 1;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 textField.setText("" + (currentPage + 1));  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 grabFocus();  
 }  
 }  
  
 if (e.getActionCommand().equals("Следующая страница")) {  
  
 //  
 if (currentPage != -1 && currentPage < document.getNumberOfPages() - 1) {  
  
 currentPage++;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 textField.setText("" + (currentPage + 1));  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 grabFocus();  
 }  
 }  
  
 if (e.getActionCommand().equals("Предыдущая страница")) {  
  
 //  
 if (currentPage != -1 && currentPage > 0) {  
  
 currentPage--;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 textField.setText("" + (currentPage + 1));  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 grabFocus();  
 }  
 }  
  
 if (e.getActionCommand().equals("Увеличить масштаб")) {  
  
 if (scale <= (float) 1.9798) {  
  
 scale += (float) 0.1042;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 FunctionalPanel.this.repaint();  
 } else {  
 scale = (float) 2.084;  
 }  
 grabFocus();  
 }  
  
 if (e.getActionCommand().equals("Уменьшить масштаб")) {  
  
 if (scale >= (float) 0.2084) {  
  
 scale -= (float) 0.1042;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 FunctionalPanel.this.repaint();  
 } else {  
 scale = (float) 0.01042;  
 }  
 grabFocus();  
 }  
  
 if (e.getActionCommand().equals("Масштаб")) {  
  
 String str = JOptionPane.*showInputDialog*(null, "Масштаб: ",  
 "Масштаб", JOptionPane.*OK\_CANCEL\_OPTION*);  
 try {  
 scale = (float) (0.01042 \* Integer.*parseInt*(str));  
 } catch (NumberFormatException nfe) {  
 nfe.printStackTrace();  
 System.*out*.println("Trouble in parsing string to float" + nfe.getCause() + " " + str);  
 }  
 if (scale >= 0.01042 && scale <= 2.0848) {  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(null,  
 ("Невозможно отобразить масштаб " + str + "Введите масштаб в диапазоне от 1% до 200%"),  
 "eReader", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 FunctionalPanel.this.repaint();  
 grabFocus();  
 }  
  
 if (e.getActionCommand().equals("Закрыть файл")) {  
  
 //  
 toolBar.setVisible(false);  
 buttonEnDis(false);  
 bufferedImage = new BufferedImage(1, 1, BufferedImage.*TYPE\_BYTE\_BINARY*);  
 try {  
 document.close();  
 currentPage = -1;  
 scale = (float) 1.0424;  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 grabFocus();  
 }  
  
 if (e.getActionCommand().equals("Выйти")) {  
  
 //  
 if (currentPage != -1) {  
 try {  
 document.close();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
  
 int yesNo = JOptionPane.*showConfirmDialog*(null,  
 "Вы действительно хотите закрыть приложение?",  
 "eReader", JOptionPane.*YES\_NO\_OPTION*);  
  
 if (yesNo == JOptionPane.*YES\_OPTION*) {  
 System.*exit*(0);  
 } else {  
 grabFocus();  
 }  
 }  
 }  
 }  
  
 protected class FieldKeyListener extends KeyAdapter {  
  
 //  
 @Override  
 public void keyPressed(KeyEvent e) {  
  
 super.keyPressed(e);  
 int key = e.getKeyCode();

//  
 if (key == KeyEvent.*VK\_LEFT* && currentPage > 0) {  
  
 currentPage--;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 textField.setText("" + (currentPage + 1));  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
  
 //  
 if (key == KeyEvent.*VK\_RIGHT* && currentPage != -1 && currentPage < document.getNumberOfPages() - 1) {  
  
 currentPage++;  
 try {  
 bufferedImage = render.renderImage(currentPage, scale);  
 imgWidth = bufferedImage.getWidth();  
 imgHeight = bufferedImage.getHeight();  
 textField.setText("" + (currentPage + 1));  
 FunctionalPanel.this.repaint();  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
  
 //  
 if (key == KeyEvent.*VK\_F1*) {  
  
 JOptionPane.*showMessageDialog*(null,  
 " Структурное подразделение\n" +  
 " Новосибирского Государственного Университета\n" +  
 " Высший Колледж Информатики университета (ВКИ НГУ)\n" +  
 "\nПриложение 'eReader' предназначено для чтения PDF-файлов\n" +  
 "\nРазработали студенты 2-го курса группы 703са1\n" +  
 "Майер Артём и Нестеренко Мария © eReader 2018",  
 "eReader", JOptionPane.*PLAIN\_MESSAGE*);  
 FunctionalPanel.this.repaint();  
 grabFocus();  
 }  
 //  
 if (key == KeyEvent.*VK\_F9*) {  
  
 if (menuBar.isVisible()) {  
 menuBar.setVisible(false);  
 } else {  
 menuBar.setVisible(true);  
 }  
 grabFocus();  
 }  
 }  
 }  
}