|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»  Факультет Бизнес-информатики, отделение Программной инженерии  **Кафедра Управление разработкой программного обеспечения** | | | |
| СОГЛАСОВАНО  Преподаватель Национального исследовательского университета «Высшая школа экономики»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Каган Э.М.  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 г. |  | | УТВЕРЖДАЮ  Заведующий отделением Программной инженерии Национального исследовательского университета «Высшая школа экономики»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Авдошин С.М.  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 г. |
| **Интерактивная доска с возможностью многопользовательского сетевого редактирования**  Текст программы | | | |
| Подп. и дата |  | **ЛИСТ УТВЕРЖДЕНИЯ**  RU.17701729.502900 – 01 12 | | | |
| Инв. № дубл. |  |  | | | |
| Взам. инв. № |  |  | | . | |
| Подп. и дата |  |  | |
| Инв. № подл. | RU.17701729.502900 – 01 12 | 2014 | | | |

|  |  |  |
| --- | --- | --- |
|  |  | УТВЕРЖДЕНО  **RU.17701729.502900 – 01 12**  **интерактивная доска с возможностью многопользовательского сетевого редактирования**  Текст программы  RU.17701729.502900 – 01 12    Листов \_\_\_  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Каган Э. М./  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 г.  Исполнитель: студент группы 271 ПИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Осипов Л. И. /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 г. |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл. | RU.17701729.502900 – 01 12 |

2014

**СОДЕРЖАНИЕ**

**Текст файла MyJFrame.java 3**

**Текст файла ControlPanel.java 12**

**Текст файла BoardPanel.java 15**

**Текст файла BoardContainer.java 21**

**Текст файла TextContainer.java 27**

**Текст файла ImageContainer.java 29**

**Текст файла ImageSlider.java 31**

**Текст файла AddingListener.java 33**

**Текст файла ImageTranserHandler.java 34**

**Текст файла JFontChooser.java 35**

**Текст файла MouseScrollListener.java 39**

**Текст файла MoveListener.java 40**

**Текст файла PointerListener.java 42**

**Текст файла ResizeListener.java 43**

**Текст файла SliderListener.java 47**

**Текст файла StandardPopUpMenu.java 47**

**Текст файла TextTransferHandler.java 50**

**Текст файла BoardClient.java 51**

**Текст файла BoardServer.java 56**

**Текст файла Command.java 60**

**Текст файла CommandConstants.java 62**

**Текст файла ConnectionData.java 62**

**Текст файла MySocketChannel.java 64**

**Текст файла SerBoardContainer.java 65**

**Текст файла SerBoardPanel.java 67**

**Лист регистрации изменений 69**

**Текст файла** **MyJFrame.java**

package MainPanels;

import BoardCompFeatures.PointerListener;

import BoardCompFeatures.JFontChooser;

import BoardCompFeatures.MouseScrollListener;

import BoardComponents.BoardContainer;

import ClientServerPart.\*;

import javax.swing.\*;

import javax.swing.event.MenuEvent;

import javax.swing.event.MenuListener;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

public class MyJFrame extends JFrame {

/\*\* File chooser \*/

private JFileChooser fc;

/\*\* Scroll \*/

private JScrollPane mainScroll;

/\*\* Control panel \*/

private ControlPanel controlPanel;

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* Client thread \*/

private Thread clientThread;

/\*\* Server thread \*/

private Thread serverThread;

/\*\* Server \*/

private BoardServer boardServer;

final private String TITLE = "Interactive whiteboard";

final public static ImageIcon TEXT\_ICON =

new ImageIcon(MyJFrame.class.getResource("/images/textIcon.png"));

final public static ImageIcon TEXT\_BUTTON\_ICON =

new ImageIcon(MyJFrame.class.getResource("/images/textButton.png"));

final public static ImageIcon IMAGE\_ICON =

new ImageIcon(MyJFrame.class.getResource("/images/imageIcon.png"));

final public static ImageIcon IMAGE\_BUTTON\_ICON =

new ImageIcon(MyJFrame.class.getResource("/images/imageButton.png"));

final public static ImageIcon CONTROL\_EXTEND =

new ImageIcon(MyJFrame.class.getResource("/images/controlExtend.png"));

final public static ImageIcon CONTROL\_POINTER =

new ImageIcon(MyJFrame.class.getResource("/images/controlPointer.png"));

final public static ImageIcon CONTROL\_TEXT\_CONTAINER =

new ImageIcon(MyJFrame.class.getResource("/images/controlText.png"));

final public static ImageIcon CONTROL\_IMAGE\_CONTAINER =

new ImageIcon(MyJFrame.class.getResource("/images/controlImage.png"));

final public static ImageIcon CONTROL\_CLEAR =

new ImageIcon(MyJFrame.class.getResource("/images/controlClear.png"));

/\*\*

\* Constructor.

\*/

public MyJFrame() {

fc = new JFileChooser();

mainScroll = new JScrollPane();

controlPanel = new ControlPanel();

this.setIconImage(new ImageIcon(this.getClass().getResource("/images/logo.png")).getImage());

JFontChooser.getInstance();

createGUI();

}

/\*\*

\* Creates GUI components.

\*/

private void createGUI() {

// Creating frame settings

Toolkit tk = Toolkit.getDefaultToolkit();

Dimension d = tk.getScreenSize();

double part = 3 / 4.0;

double w = d.width \* part;

double h = d.height \* part;

int width = (int)Math.round(w);

int height = (int)Math.round(h);

this.setLocation((int)Math.round((1 - part) \* w / 2), (int)Math.round(((1 - part) \* h / 2)));

this.setDefaultCloseOperation(WindowConstants.DO\_NOTHING\_ON\_CLOSE);

this.setTitle(TITLE);

this.setLayout(new BorderLayout());

// Control panel

this.add(controlPanel, BorderLayout.NORTH);

// Modifying file chooser

fc.setFileFilter(new javax.swing.filechooser.FileFilter() {

@Override

public String getDescription() {

return "Interactive boards (\*.brd)";

}

@Override

public boolean accept(File f) {

if (f.isDirectory()) {

return true;

} else {

return f.getName().toLowerCase().endsWith(".brd");

}

}

});

// Constructing menu bar

this.setJMenuBar(createMenu());

// Board panel

boardPanel = new BoardPanel(mainScroll, width \* 2, (height - 150) \* 2);

boardPanel.setFrame(this);

controlPanel.setBoardPanel(boardPanel);

// Setting scrolls

JViewport viewport = mainScroll.getViewport();

viewport.add(boardPanel);

mainScroll.setViewport(viewport);

MouseAdapter msl = new MouseScrollListener(boardPanel, viewport,

Cursor.getPredefinedCursor(Cursor.MOVE\_CURSOR), boardPanel);

viewport.addMouseListener(msl);

viewport.addMouseMotionListener(msl);

mainScroll.setPreferredSize(new Dimension(width, height - 150));

this.add(mainScroll, BorderLayout.CENTER);

this.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

super.windowClosing(e);

if (! checkSaving()) {

return;

}

disconnect(serverThread, boardPanel);

disconnect(clientThread, boardPanel);

System.exit(0);

}

});

this.pack();

this.setVisible(true);

}

/\*\*

\* Checks and asks if the document needed to be saved

\*/

private boolean checkSaving() {

if (boardPanel.isSaved()) {

return true;

}

else {

int result = JOptionPane.showConfirmDialog(null, "The changes were not saved. Would you like to save them?",

"Saving", JOptionPane.YES\_NO\_CANCEL\_OPTION);

if (result == JOptionPane.CANCEL\_OPTION) {

return false;

}

if (result == JOptionPane.NO\_OPTION) {

return true;

}

if (fc.getSelectedFile() != null) {

saveBoard(fc.getSelectedFile());

return true;

}

else {

return saveAs();

}

}

}

/\*\*

\* Sets the new board to screen.

\* @param newBoard the new board

\*/

public void setNewBoard(BoardPanel newBoard) {

JViewport viewport = mainScroll.getViewport();

BoardClient client = boardPanel.getClient();

viewport.remove(boardPanel);

controlPanel.getButtonPointer().setSelected(false);

boardPanel = newBoard;

boardPanel.setScroll(mainScroll);

boardPanel.setFrame(this);

boardPanel.setPl(new PointerListener(boardPanel, boardPanel));

for (Component component : boardPanel.getComponents()) {

BoardContainer bc = ((BoardContainer)component);

bc.setPl(new PointerListener(bc, boardPanel));

}

controlPanel.setBoardPanel(boardPanel);

viewport.add(boardPanel);

if (client != null) {

client.setBoardPanel(boardPanel);

boardPanel.setClient(client);

}

if (boardServer != null) {

boardServer.setBoardPanel(boardPanel);

}

MouseAdapter msl = new MouseScrollListener(boardPanel, viewport,

Cursor.getPredefinedCursor(Cursor.MOVE\_CURSOR), boardPanel);

viewport.addMouseListener(msl);

viewport.addMouseMotionListener(msl);

boardPanel.repaint();

}

/\*\*

\* Opens FileChooser and opens the board if the directory is selected.

\*/

private void open() {

fc.setSelectedFile(new File(""));

int result = fc.showOpenDialog(null);

if (result == JFileChooser.APPROVE\_OPTION) {

File file = fc.getSelectedFile();

if (! file.getName().toLowerCase().endsWith(".brd")) {

file = new File(file.getAbsolutePath() + ".brd");

}

openBoard(file);

}

fc.setSelectedFile(null);

}

/\*\*

\* Opens FileChooser and saves the board if the directory is selected.

\*/

private boolean saveAs() {

fc.setSelectedFile(new File(""));

int result = fc.showSaveDialog(null);

if (result == JFileChooser.APPROVE\_OPTION) {

File file = fc.getSelectedFile();

if (! file.getName().toLowerCase().endsWith(".brd")) {

file = new File(file.getAbsolutePath() + ".brd");

}

saveBoard(file);

return true;

}

fc.setSelectedFile(null);

return false;

}

/\*\*

\* Loads th board from file.

\* @param file the file

\*/

private void openBoard(File file) {

try {

FileInputStream fin = new FileInputStream(file);

ObjectInputStream lol = new ObjectInputStream(fin);

BoardPanel newBoard = new BoardPanel(mainScroll, (SerBoardPanel)lol.readObject());

setNewBoard(newBoard);

lol.close();

fin.close();

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_BOARD);

command.setNewBoard(newBoard.toSerializable());

boardPanel.getClient().addCommand(command);

}

}

catch (IOException e1) {

//e1.printStackTrace();

JOptionPane.showMessageDialog(null, "A error occurred in opening the board!",

"Error", JOptionPane.ERROR\_MESSAGE);

}

catch (ClassNotFoundException e2) {

//e2.printStackTrace();

JOptionPane.showMessageDialog(null, "A error occurred in opening the board!",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* Save the board to file.

\* @param file the file

\*/

private void saveBoard(File file) {

try {

FileOutputStream out = new FileOutputStream(file);

ObjectOutputStream stream = new ObjectOutputStream(out);

stream.writeObject(boardPanel.toSerializable());

stream.close();

out.close();

JOptionPane.showMessageDialog(null, "The board was successfully saved!");

boardPanel.setSaved(true);

}

catch (IOException e1) {

e1.printStackTrace();

JOptionPane.showMessageDialog(null, "A error occurred in saving the board!",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* Constructs menu Bar;

\*/

private JMenuBar createMenu() {

final JMenuBar menuBar = new JMenuBar();

// File options

final JMenu menuFile = new JMenu("File");

// Creates a new board

final JMenuItem menuItemNew = new JMenuItem("New");

menuItemNew.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (! checkSaving()) {

return;

}

int width = boardPanel.getActualWidth();

int height = boardPanel.getActualHeight();

BoardPanel bp = new BoardPanel(mainScroll, width, height);

setNewBoard(bp);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_BOARD);

command.setNewBoard(bp.toSerializable());

boardPanel.getClient().addCommand(command);

}

}

});

menuFile.add(menuItemNew);

menuFile.addSeparator();

// Opens a board from file

final JMenuItem menuItemOpen = new JMenuItem("Open");

menuItemOpen.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (! checkSaving()) {

return;

}

open();

}

});

menuFile.add(menuItemOpen);

menuFile.addSeparator();

// Saves to file

final JMenuItem menuItemSave = new JMenuItem("Save");

menuItemSave.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (fc.getSelectedFile() != null) {

saveBoard(fc.getSelectedFile());

}

else {

saveAs();

}

}

});

menuFile.add(menuItemSave);

// Saves as

final JMenuItem menuItemSaveAs = new JMenuItem("Save As");

menuItemSaveAs.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

saveAs();

}

});

menuFile.add(menuItemSaveAs);

menuBar.add(menuFile);

// Board settings

final JMenu menuSettings = new JMenu("Settings");

final JMenuItem menuItemSetBG = new JMenuItem("Set board background color");

menuItemSetBG.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Color old = boardPanel.getCurrentColor();

Color color = JColorChooser.showDialog(null,

"Choose board background color", boardPanel.getBackground());

if (color != null && ! old.equals(color)) {

boardPanel.setBackground(color);

//System.out.println(color);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_BOARD\_BACKGROUND);

command.setNewColor(color);

boardPanel.getClient().addCommand(command);

}

}

}

});

menuSettings.add(menuItemSetBG);

menuSettings.addSeparator();

final JMenuItem menuItemSetGeneralFont = new JMenuItem("Set general text font");

menuItemSetGeneralFont.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFontChooser fc = JFontChooser.getInstance();

int result = fc.showDialog(boardPanel.getCurrentFont());

if (result == JFontChooser.OK\_OPTION) {

boardPanel.setGeneralTextFont(fc.getFont());

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_GENERAL\_FONT);

command.setNewFont(fc.getFont());

boardPanel.getClient().addCommand(command);

}

}

}

});

menuSettings.add(menuItemSetGeneralFont);

final JMenuItem menuItemSetGeneralColor = new JMenuItem("Set general text color");

menuItemSetGeneralColor.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Color old = boardPanel.getCurrentColor();

Color color = JColorChooser.showDialog(null,

"Choose general text color", boardPanel.getBackground());

if (color != null && ! old.equals(color)) {

boardPanel.setGeneralTextColor(color);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_GENERAL\_COLOR);

command.setNewColor(color);

boardPanel.getClient().addCommand(command);

}

}

}

});

menuSettings.add(menuItemSetGeneralColor);

menuBar.add(menuSettings);

// Connections

JMenu menuConnections = new JMenu("Connections");

final JMenuItem menuItemDisconnect = new JMenuItem("Disconnect");

menuItemDisconnect.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

disconnect(serverThread, boardPanel);

disconnect(clientThread, boardPanel);

}

});

final JMenuItem menuItemCreateServer = new JMenuItem("Create server");

menuItemCreateServer.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

disconnect(serverThread, boardPanel);

disconnect(clientThread, boardPanel);

createServer();

}

});

menuConnections.add(menuItemCreateServer);

menuConnections.addSeparator();

final JMenuItem menuItemConnect = new JMenuItem("Connect to board");

menuItemConnect.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

disconnect(serverThread, boardPanel);

disconnect(clientThread, boardPanel);

createClient();

}

});

menuConnections.add(menuItemConnect);

menuConnections.addSeparator();

menuConnections.add(menuItemDisconnect);

menuBar.add(menuConnections);

JLabel buttonAbout = new JLabel(" About");

buttonAbout.setOpaque(false);

buttonAbout.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

JOptionPane.showMessageDialog(null, "Interactive whiteboard with multi-user editing via " +

"network connection.\nLev Osipov 2014");

}

});

menuBar.add(buttonAbout);

return menuBar;

}

private void createClient() {

ConnectionData cd = new ConnectionData(true);

if (cd.askForData()) {

final BoardClient boardClient = new BoardClient(boardPanel, cd.getIpAddress(), cd.getPort(), cd.getPassword());

if (boardClient.setClient()) {

clientThread = new Thread() {

@Override

public void run() {

boardClient.start();

MyJFrame.this.setTitle(TITLE);

}

};

clientThread.start();

this.setTitle(TITLE + " - Client, port: " + cd.getPort());

}

}

}

private void createServer() {

ConnectionData cd = new ConnectionData(false);

if (cd.askForData()) {

boardServer = new BoardServer(boardPanel, cd.getPort(), cd.getPassword());

if (boardServer.createServer()) {

serverThread = new Thread() {

@Override

public void run() {

boardServer.start();

}

};

serverThread.start();

this.setTitle(TITLE + " - Server, ip: " + BoardServer.getIpAddress() + ", port: " + cd.getPort());

final BoardClient boardClient = new BoardClient(boardPanel, cd.getIpAddress(), cd.getPort(), cd.getPassword());

if (boardClient.setClient()) {

clientThread = new Thread() {

@Override

public void run() {

boardClient.start();

MyJFrame.this.setTitle(TITLE);

}

};

clientThread.start();

}

}

}

}

private void disconnect(Thread thread, BoardPanel boardPanel) {

if (thread != null && thread.isAlive()) {

thread.interrupt();

try {

thread.join();

}

catch (InterruptedException e1) {

e1.printStackTrace();

}

boardPanel.setClient(null);

}

}

public ControlPanel getControlPanel() {

return controlPanel;

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

new MyJFrame();

}

});

}

}

**Текст файла ControlPanel.java**

package MainPanels;

import BoardCompFeatures.AddingListener;

import BoardComponents.BoardContainer;

import BoardComponents.ImageContainer;

import BoardComponents.TextContainer;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

/\*\*

\* Created by Lev on 04.04.14.

\*/

/\*\*

\* Panel with common commands.

\*/

public class ControlPanel extends JPanel {

/\*\* Link for board panel \*/

private BoardPanel boardPanel;

/\*\* General dimension \*/

final public static Dimension BUTTON\_DIM = new Dimension(50, 50);

/\*\* Button attention \*/

private JToggleButton buttonPointer;

public ControlPanel() {

createButtons();

}

/\*\*

\* Create control buttons.

\*/

private void createButtons() {

this.setLayout(new FlowLayout(FlowLayout.LEFT));

ButtonGroupNSS bg = new ButtonGroupNSS();

// Extend button

final JButton buttonExtend = new JButton();

buttonExtend.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

boardPanel.extendBoard();

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.EXTEND\_BOARD);

boardPanel.getClient().addCommand(command);

}

}

});

buttonExtend.setPreferredSize(BUTTON\_DIM);

buttonExtend.setIcon(MyJFrame.CONTROL\_EXTEND);

buttonExtend.setToolTipText("Extend board x2");

this.add(buttonExtend);

// Attention button

buttonPointer = new JToggleButton();

buttonPointer.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (buttonPointer.isSelected()) {

setPL();

} else {

removePL();

}

}

});

this.add(buttonPointer);

buttonPointer.setPreferredSize(BUTTON\_DIM);

buttonPointer.setIcon(MyJFrame.CONTROL\_POINTER);

buttonPointer.setToolTipText("Point to something, just click");

bg.add(buttonPointer);

// Text container button

final JToggleButton buttonTC = new JToggleButton();

buttonTC.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setButtonAction(buttonTC,

new AddingListener(boardPanel, BoardPanel.TEXT\_CONTAINER, buttonTC));

}

});

this.add(buttonTC);

buttonTC.setPreferredSize(BUTTON\_DIM);

buttonTC.setIcon(MyJFrame.CONTROL\_TEXT\_CONTAINER);

buttonTC.setToolTipText("Paint a text container");

bg.add(buttonTC);

// Image container button

final JToggleButton buttonIC = new JToggleButton();

buttonIC.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setButtonAction(buttonIC,

new AddingListener(boardPanel, BoardPanel.IMAGE\_CONTAINER, buttonIC));

}

});

this.add(buttonIC);

buttonIC.setPreferredSize(BUTTON\_DIM);

buttonIC.setIcon(MyJFrame.CONTROL\_IMAGE\_CONTAINER);

buttonIC.setToolTipText("Paint an image container");

bg.add(buttonIC);

// Clear button

JButton buttonClear = new JButton();

buttonClear.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int result = JOptionPane.showConfirmDialog(null, "Are you sure?", "Clearing the board",

JOptionPane.OK\_CANCEL\_OPTION);

if (result == JOptionPane.OK\_OPTION) {

boardPanel.clearBoard();

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.CLEAR\_BOARD);

boardPanel.getClient().addCommand(command);

}

}

}

});

buttonClear.setPreferredSize(BUTTON\_DIM);

buttonClear.setIcon(MyJFrame.CONTROL\_CLEAR);

buttonClear.setToolTipText("Clear the board");

this.add(buttonClear);

}

/\*\*

\* Sets buttons behaviour

\*/

private void setButtonAction(JToggleButton button, MouseAdapter ma) {

removeMMListeners(boardPanel);

if (button.isSelected()) {

boardPanel.addMouseListener(ma);

boardPanel.addMouseMotionListener(ma);

}

else {

boardPanel.setCursor(Cursor.getDefaultCursor());

}

if (buttonPointer.isSelected()) {

removePL();

}

}

/\*\*

\* Removes all mouse listeners from component

\*/

public static void removeMMListeners(JComponent component) {

for (MouseListener ml : component.getMouseListeners()) {

component.removeMouseListener(ml);

}

for (MouseMotionListener mml : component.getMouseMotionListeners()) {

component.removeMouseMotionListener(mml);

}

}

public void setPL() {

boardPanel.addMouseListener(boardPanel.getPl());

for (Component component : boardPanel.getComponents()) {

BoardContainer bc = (BoardContainer)component;

bc.addMouseListener(bc.getPl());

if (bc.getType() == BoardPanel.TEXT\_CONTAINER) {

bc.getComponent().addMouseListener(((TextContainer) bc.getComponent()).getPl());

}

else {

bc.getComponent().addMouseListener(((ImageContainer)bc.getComponent()).getPl());

}

}

}

public void removePL() {

boardPanel.removeMouseListener(boardPanel.getPl());

for (Component component : boardPanel.getComponents()) {

BoardContainer bc = (BoardContainer)component;

bc.removeMouseListener(bc.getPl());

if (bc.getType() == BoardPanel.TEXT\_CONTAINER) {

bc.getComponent().removeMouseListener(((TextContainer) bc.getComponent()).getPl());

}

else {

bc.getComponent().removeMouseListener(((ImageContainer) bc.getComponent()).getPl());

}

}

}

public void setBoardPanel(BoardPanel boardPanel) {

this.boardPanel = boardPanel;

}

public JToggleButton getButtonPointer() {

return buttonPointer;

}

/\*\*

\* Supported no selection

\*/

class ButtonGroupNSS extends ButtonGroup {

@Override

public void setSelected(ButtonModel m, boolean b) {

if (b) {

super.setSelected(m, b);

}

else {

clearSelection();

}

}

}

}

**Текст файла BoardPanel.java**

package MainPanels;

import BoardCompFeatures.PointerListener;

import BoardComponents.\*;

import ClientServerPart.\*;

import javax.swing.\*;

import java.awt.\*;

import java.util.ArrayList;

import java.util.Random;

/\*\*

\* Created by Lev on 04.04.14.

\*/

public class BoardPanel extends JLayeredPane {

private int actualWidth;

private int actualHeight;

/\*\* Frame \*/

private MyJFrame frame;

/\*\* Upper left corner of the new component \*/

private Point startCompPoint;

/\*\* Down right corner of the new component \*/

private Point endCompPoint;

/\*\* Scroll of this panel \*/

private JScrollPane scroll;

/\*\* Constants \*/

final public static int TEXT\_CONTAINER = 0;

final public static int IMAGE\_CONTAINER = 1;

/\*\* Font of the board components \*/

private Font currentFont;

/\*\* Color of the board components \*/

private Color currentColor;

/\*\* If the current state is saved \*/

private boolean saved;

/\*\* This board as a client (in order to send/receive commands) \*/

private BoardClient client;

/\*\* Attention listener \*/

private PointerListener pl;

public BoardPanel(JScrollPane scroll, int actualWidth, int actualHeight) {

this.scroll = scroll;

this.actualWidth = actualWidth;

this.actualHeight = actualHeight;

currentFont = new Font("Serif", Font.ITALIC, 20);

currentColor = Color.BLACK;

pl = new PointerListener(this, this);

this.setOpaque(true);

this.setBackground(new Color(0, 255, 153));

this.setPreferredSize(new Dimension(actualWidth, actualHeight));

saved = true;

this.updateUI();

}

public BoardPanel(JScrollPane scroll, SerBoardPanel sbp) {

this(scroll, sbp.getActualWidth(), sbp.getActualHeight());

currentColor = sbp.getCurrentColor();

currentFont = sbp.getCurrentFont();

for (SerBoardContainer sbc : sbp.getContainers()) {

this.addComponent(sbc.getLocation(), new BoardContainer(sbc, this));

}

this.setBackground(sbp.getBgColor());

saved = sbp.isSaved();

}

/\*\* Clears the board \*/

public void clearBoard() {

this.removeAll();

scroll.revalidate();

this.repaint();

}

/\*\* Extends the board \*/

public void extendBoard() {

actualWidth \*= 2;

actualHeight \*= 2;

this.setPreferredSize(new Dimension(actualWidth, actualHeight));

this.revalidate();

}

/\*\* Adds a component to the board \*/

public void addComponent(Point location, BoardContainer bc) {

if (frame != null) {

frame.getControlPanel().removePL();

frame.getControlPanel().getButtonPointer().setSelected(false);

}

bc.setLocation(location);

this.add(bc);

this.setLayer(bc, 1);

for (Component component1 : this.getComponents()) {

this.setLayer(component1, this.getLayer(component1) + 1);

}

saved = false;

this.updateUI();

this.repaint();

}

/\*\* Sets general text font \*/

public void setGeneralTextFont(Font font) {

currentFont = font;

for (Component component : this.getComponents()) {

((BoardContainer) component).setContainerTextFont(font);

}

}

/\*\* Sets general text color \*/

public void setGeneralTextColor(Color color) {

currentColor = color;

for (Component component : this.getComponents()) {

((BoardContainer) component).setContainerTextColor(color);

}

}

/\*\* Sets component to front \*/

public void componentToFront(BoardContainer container) {

int oldLayer = this.getLayer(container);

this.setLayer(container, this.getComponents().length);

for (Component component : this.getComponents()) {

int layer = this.getLayer(component);

if (component != container && layer > oldLayer) {

this.setLayer(component, --layer);

}

}

this.repaint();

}

/\*\* Sets component to background \*/

public void componentToBackground(BoardContainer container) {

int oldLayer = this.getLayer(container);

this.setLayer(container, 1);

for (Component component : this.getComponents()) {

int layer = this.getLayer(component);

if (component != container && layer < oldLayer) {

this.setLayer(component, ++layer);

}

}

this.repaint();

}

/\*\* Deletes the component from board \*/

public void deleteComponent(BoardContainer container) {

int id = container.getId();

int oldLayer = this.getLayer(container);

this.remove(container);

for (Component component : this.getComponents()) {

int layer = this.getLayer(component);

if (layer > oldLayer) {

this.setLayer(component, -- layer);

}

BoardContainer bcc = (BoardContainer)component;

if (bcc.getId() > id) {

bcc.setId(bcc.getId() - 1);

}

}

this.repaint();

this.getScroll().revalidate();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

if (startCompPoint != null) {

g.fillOval((int)startCompPoint.getX() - 4, (int)startCompPoint.getY() - 4, 8, 8);

}

if (endCompPoint != null && startCompPoint != null && endCompPoint.getX() > startCompPoint.getX() &&

endCompPoint.getY() > startCompPoint.getY()) {

g.drawRect((int)startCompPoint.getX(), (int)startCompPoint.getY(), (int)endCompPoint.getX() -

(int)startCompPoint.getX(), (int)endCompPoint.getY() - (int)startCompPoint.getY());

}

}

/\*\* Generate the ID \*/

public int generateUID() {

Random random = new Random();

int i = random.nextInt(1000000) + 1;

for (Component component : this.getComponents()) {

if (((BoardContainer)component).getId() == i) {

return generateUID();

}

}

return i;

}

public static void pointerCommand(JComponent component, Point point) {

Graphics g = component.getGraphics();

g.setColor(Color.RED);

try {

for (int i = 0; i <= 30; i += 2) {

g.fillOval(point.x - i / 2, point.y - i / 2, i, i);

Thread.sleep(11);

}

}

catch (InterruptedException e1) {

e1.printStackTrace();

}

component.repaint();

}

public SerBoardPanel toSerializable() {

ArrayList<SerBoardContainer> containers = new ArrayList<SerBoardContainer>();

for (Component component : this.getComponents()) {

containers.add(((BoardContainer) component).toSerializable());

}

return new SerBoardPanel(actualWidth, actualHeight,

this.getBackground(), currentFont, currentColor, saved, containers);

}

/\*\* Receiving a command \*/

public void processCommand(Command command) {

saved = false;

// Need to have a reference to the container if the command is connected to it

BoardContainer target = null;

int commandId = command.getUid();

if (command.getCommandType() >= CommandConstants.GENERAL\_COMMANDS && commandId != 0) {

for (Component component : this.getComponents()) {

if (((BoardContainer)component).getId() == command.getUid()) {

target = (BoardContainer)component;

break;

}

}

if (target == null) {

throw new NullPointerException("No container id!");

}

}

switch (command.getCommandType()) {

case CommandConstants.CLEAR\_BOARD :

this.clearBoard();

break;

case CommandConstants.EXTEND\_BOARD :

this.extendBoard();

break;

case CommandConstants.POINTER:

if (commandId == 0) {

pointerCommand(this, command.getNewPoint());

}

else {

if (command.getNewValue1() == -1) {

BoardPanel.pointerCommand(target, command.getNewPoint());

}

else {

BoardPanel.pointerCommand(target.getComponent(), command.getNewPoint());

}

}

break;

case CommandConstants.NEW\_TEXT\_CONTAINER :

this.addComponent(command.getNewPoint(), new BoardContainer(TEXT\_CONTAINER, commandId,

command.getNewValue1(), command.getNewValue2(), this));

break;

case CommandConstants.NEW\_IMAGE\_CONTAINER :

this.addComponent(command.getNewPoint(), new BoardContainer(IMAGE\_CONTAINER, commandId,

command.getNewValue1(), command.getNewValue2(), this));

break;

case CommandConstants.DELETE\_CONTAINER :

this.deleteComponent(target);

break;

case CommandConstants.MOVE\_CONTAINER :

target.setLocation(command.getNewPoint());

break;

case CommandConstants.RESIZE\_CONTAINER :

target.setBounds(command.getNewPoint().x, command.getNewPoint().y,

command.getNewValue1(), command.getNewValue2());

target.setActualWidth(command.getNewValue1());

target.setActualHeight(command.getNewValue2());

break;

case CommandConstants.NEW\_NAME :

target.setContainerName(command.getNewText());

break;

case CommandConstants.NEW\_FONT :

target.setContainerTextFont(command.getNewFont());

break;

case CommandConstants.NEW\_COLOR :

target.setContainerTextColor(command.getNewColor());

break;

case CommandConstants.NEW\_GENERAL\_FONT :

this.setGeneralTextFont(command.getNewFont());

break;

case CommandConstants.NEW\_GENERAL\_COLOR :

this.setGeneralTextColor(command.getNewColor());

break;

case CommandConstants.NEW\_TEXT :

JTextArea textArea = (JTextArea)target.getComponent();

//int position = textArea.getCaretPosition();

textArea.setText(command.getNewText());

//textArea.setCaretPosition(position);

break;

case CommandConstants.NEW\_IMAGE :

ImageContainer ic = (ImageContainer)target.getComponent();

ic.clear();

ic.setImage(command.getNewImage());

break;

case CommandConstants.TO\_FRONT :

this.componentToFront(target);

break;

case CommandConstants.TO\_BACKGROUND :

this.componentToBackground(target);

break;

case CommandConstants.CLEAR\_CONTAINER :

target.clearContainer();

break;

case CommandConstants.NEW\_BOARD :

frame.setNewBoard(new BoardPanel(scroll, command.getNewBoard()));

break;

case CommandConstants.NEW\_BOARD\_BACKGROUND:

this.setBackground(command.getNewColor());

break;

case CommandConstants.IMAGE\_RESIZE:

ImageSlider slider = ((ImageContainer)target.getComponent()).getSlider();

slider.setChanged(false);

slider.setValue(command.getNewValue1());

slider.setChanged(true);

slider.resizeImage(command.getNewValue1());

break;

case CommandConstants.IMAGE\_SCROLL:

Point point = command.getNewPoint();

Point old = new Point(command.getNewValue1(), command.getNewValue2());

JViewport viewport = target.getScroll().getViewport();

Point vp = viewport.getViewPosition();

vp.translate(old.x - point.x, old.y - point.y);

target.getComponent().scrollRectToVisible(new Rectangle(vp, viewport.getSize()));

break;

}

}

public void setStartCompPoint(Point startCompPoint) {

this.startCompPoint = startCompPoint;

}

public void setEndCompPoint(Point endCompPoint) {

this.endCompPoint = endCompPoint;

}

public void setScroll(JScrollPane scroll) {

this.scroll = scroll;

}

public Font getCurrentFont() {

return currentFont;

}

public boolean isSaved() {

return saved;

}

public void setSaved(boolean saved) {

this.saved = saved;

}

public void setPl(PointerListener pl) {

this.pl = pl;

}

public void setFrame(MyJFrame frame) {

this.frame = frame;

}

public BoardClient getClient() {

return client;

}

public Color getCurrentColor() {

return currentColor;

}

public void setClient(BoardClient client) {

this.client = client;

}

public Point getStartCompPoint() {

return startCompPoint;

}

public Point getEndCompPoint() {

return endCompPoint;

}

public JScrollPane getScroll() {

return scroll;

}

public int getActualWidth() {

return actualWidth;

}

public int getActualHeight() {

return actualHeight;

}

public PointerListener getPl() {

return pl;

}

}

**Текст файла BoardContainer.java**

package BoardComponents;

import BoardCompFeatures.\*;

import ClientServerPart.SerBoardContainer;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import MainPanels.MyJFrame;

import javax.swing.\*;

import java.awt.\*;

import java.awt.Image;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.event.MouseAdapter;

/\*\*

\* Created by Lev on 08.04.14.

\*/

public class BoardContainer extends JPanel {

/\*\* The board \*/

private BoardPanel boardPanel;

/\*\* Component type \*/

private int type;

/\*\* ID number \*/

private int id;

/\*\* The width of the container \*/

private int actualWidth;

/\*\* The height of the container \*/

private int actualHeight;

/\*\* The name of the container \*/

private JLabel containerName;

/\*\* The background \*/

private ImageIcon image;

/\*\* The scroll \*/

private JScrollPane scroll;

/\*\* Button label \*/

private JLabel labelButton;

/\*\* Component itself \*/

private JComponent component;

/\*\* Component params \*/

private int compWidth;

private int compHeight;

/\*\* Text color \*/

private Color color;

/\*\* Text font \*/

private Font font;

/\*\* Move listener (in order to turn it on and off while resizing) \*/

private MoveListener moveListener;

/\*\* If ml is on\*/

private boolean ml;

/\*\* Attention listener \*/

private PointerListener pl;

public BoardContainer(int type, int id, int actualWidth, int actualHeight, BoardPanel boardPanel) {

int minimumSize = type == BoardPanel.TEXT\_CONTAINER? 90 : 200;

if (actualWidth < minimumSize) {

actualWidth = minimumSize;

}

if (actualHeight < minimumSize) {

actualHeight = minimumSize;

}

this.type = type;

this.id = id;

this.actualWidth = actualWidth;

this.actualHeight = actualHeight;

pl = new PointerListener(this, boardPanel);

this.boardPanel = boardPanel;

this.setOpaque(false);

this.setSize(new Dimension(actualWidth, actualHeight));

this.setLayout(new BoxLayout(this, BoxLayout.Y\_AXIS));

// Adding main elements

// Label button

labelButton = new JLabel();

labelButton.setAlignmentX(CENTER\_ALIGNMENT);

this.add(labelButton);

// Label name

containerName = new JLabel();

containerName.setAlignmentX(CENTER\_ALIGNMENT);

containerName.setFont(boardPanel.getCurrentFont());

this.add(containerName);

// Component itself

if (type == BoardPanel.TEXT\_CONTAINER) {

component = new TextContainer("Drag a file here", this, boardPanel);

}

else {

component = new ImageContainer(this, boardPanel);

}

// Scroll to it

scroll = new JScrollPane();

final JViewport viewport = scroll.getViewport();

viewport.setOpaque(false);

viewport.setBorder(null);

viewport.add(component);

scroll.setViewport(viewport);

if (type == BoardPanel.TEXT\_CONTAINER) {

scroll.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER);

}

else {

scroll.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

}

scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

scroll.setAlignmentX(CENTER\_ALIGNMENT);

scroll.setOpaque(false);

scroll.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

this.add(scroll);

// If IC, add slider

if (type == BoardPanel.IMAGE\_CONTAINER) {

ImageContainer ic = (ImageContainer)component;

ImageSlider slider = new ImageSlider(viewport, ic);

this.add(slider);

ic.setSlider(slider);

}

this.setContainerTextFont(boardPanel.getCurrentFont());

this.setContainerTextColor(boardPanel.getCurrentColor());

// Set features

this.setFeatures();

this.updateUI();

this.repaint();

}

public BoardContainer(SerBoardContainer sbc, BoardPanel boardPanel) {

this(sbc.getType(), sbc.getUid(), sbc.getActualWidth(), sbc.getActualHeight(), boardPanel);

this.setContainerTextFont(sbc.getFont());

this.setContainerTextColor(sbc.getColor());

this.setContainerName(sbc.getName());

if (type == BoardPanel.TEXT\_CONTAINER) {

((TextContainer)component).setText(sbc.getText());

}

else {

((ImageContainer)component).setImage(sbc.getImage());

}

this.updateUI();

this.repaint();

}

public int getActualWidth() {

return actualWidth;

}

public void setActualWidth(int actualWidth) {

this.actualWidth = actualWidth;

}

public int getActualHeight() {

return actualHeight;

}

public void setActualHeight(int actualHeight) {

this.actualHeight = actualHeight;

}

public JLabel getContainerName() {

return containerName;

}

public void setContainerName(String name) {

this.containerName.setText(name);

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public JScrollPane getScroll() {

return scroll;

}

public JLabel getLabelButton() {

return labelButton;

}

public int getType() {

return type;

}

public int getCompWidth() {

return compWidth;

}

public int getCompHeight() {

return compHeight;

}

public JComponent getComponent() {

return component;

}

public MoveListener getMoveListener() {

return moveListener;

}

public boolean isMl() {

return ml;

}

public void setPl(PointerListener pl) {

this.pl = pl;

if (type == BoardPanel.TEXT\_CONTAINER) {

((TextContainer)component).setPl(new PointerListener(component, boardPanel));

}

else {

((ImageContainer)component).setPl(new PointerListener(component, boardPanel));

}

}

public PointerListener getPl() {

return pl;

}

public void setMl(boolean ml) {

this.ml = ml;

}

/\*\* Repaints the image if needed \*/

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

compWidth = 6 \* actualWidth / 7;

compHeight = 6 \* (actualHeight - labelButton.getHeight() - containerName.getHeight()) / 7;

if (type == BoardPanel.IMAGE\_CONTAINER) {

compHeight -= 30;

}

scroll.setMaximumSize(new Dimension(compWidth, compHeight));

scroll.setPreferredSize(new Dimension(compWidth, compHeight));

Image image1 = image.getImage();

g.drawImage(image1, 0, 0, actualWidth, actualHeight, null);

this.revalidate();

}

/\*\*

\* Sets all features to board container

\*/

public void setFeatures() {

// Moving

moveListener = new MoveListener(this, boardPanel);

this.addMouseListener(moveListener);

this.addMouseMotionListener(moveListener);

ml = true;

// Resizing

ResizeListener rl = new ResizeListener(this, boardPanel);

this.addMouseListener(rl);

this.addMouseMotionListener(rl);

// PopUp menu

StandardPopUpMenu popUpMenu = new StandardPopUpMenu(boardPanel, this);

this.addMouseListener(popUpMenu.setPopUpMenu());

// If image container

if (type == BoardPanel.IMAGE\_CONTAINER) {

image = MyJFrame.IMAGE\_ICON;

labelButton.setIcon(MyJFrame.IMAGE\_BUTTON\_ICON);

MouseAdapter msl = new MouseScrollListener(component, scroll.getViewport(),

Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR), boardPanel);

component.addMouseListener(msl);

component.addMouseMotionListener(msl);

ImageContainer ic = (ImageContainer)component;

ic.getSlider().addChangeListener(new SliderListener(this, boardPanel));

ic.setTransferHandler(new ImageTransferHandler(ic));

}

else {

image = MyJFrame.TEXT\_ICON;

labelButton.setIcon(MyJFrame.TEXT\_BUTTON\_ICON);

final TextContainer tc = (TextContainer)component;

tc.setTransferHandler(new TextTransferHandler(tc));

tc.addKeyListener(new KeyAdapter() {

@Override

public void keyReleased(KeyEvent e) {

super.keyReleased(e);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_TEXT);

command.setUid(id);

command.setNewText(((TextContainer)component).getText());

boardPanel.getClient().addCommand(command);

}

}

});

}

}

/\*\* Sets text font \*/

public void setContainerTextFont(Font font) {

this.font = font;

if (type == BoardPanel.TEXT\_CONTAINER) {

component.setFont(font);

}

containerName.setFont(font);

}

/\*\* Sets text color \*/

public void setContainerTextColor(Color color) {

this.color = color;

if (type == BoardPanel.TEXT\_CONTAINER) {

component.setForeground(color);

}

containerName.setForeground(color);

}

/\*\* Clears the container \*/

public void clearContainer() {

this.setContainerName("");

containerName.setFont(boardPanel.getCurrentFont());

if (type == BoardPanel.TEXT\_CONTAINER) {

((JTextArea)component).setText("");

component.setFont(boardPanel.getCurrentFont());

}

else {

((ImageContainer)component).clear();

}

}

public SerBoardContainer toSerializable() {

String text = null;

ImageIcon image = null;

if (type == BoardPanel.TEXT\_CONTAINER) {

text = ((TextContainer)component).getText();

}

else {

image = ((ImageContainer)component).getII();

}

return new SerBoardContainer(type, id, this.getLocation(), actualWidth,

actualHeight, font, color, containerName.getText(), text, image);

}

}

**Текст файла TextContainer.java**

package BoardComponents;

import BoardCompFeatures.PointerListener;

import BoardCompFeatures.TextTransferHandler;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import javax.swing.\*;

import java.io.\*;

import java.util.concurrent.TimeoutException;

/\*\*

\* Created by Lev on 05.04.14.

\*/

public class TextContainer extends JTextArea {

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* Board container \*/

private BoardContainer boardContainer;

/\*\* Attention listener \*/

private PointerListener pl;

public TextContainer(String s, BoardContainer boardContainer, BoardPanel boardPanel) {

super(s);

this.boardPanel = boardPanel;

this.boardContainer = boardContainer;

pl = new PointerListener(this, boardPanel);

this.setOpaque(false);

this.setBorder(null);

this.setLineWrap(true);

this.setFont(boardPanel.getCurrentFont());

// Drag settings

this.setDragEnabled(true);

this.setDropMode(DropMode.INSERT);

this.setTransferHandler(new TextTransferHandler(this));

}

/\*\*

\* Reads data from text file

\*/

public void readFile(File file) {

BufferedReader reader = null;

try {

int position = this.getCaretPosition();

reader = new BufferedReader(new InputStreamReader(new FileInputStream(file), "windows-1251"));

String line;

long time = System.currentTimeMillis();

while ((line = reader.readLine()) != null) {

line += "\n";

this.insert(line, position);

position += line.length();

if (System.currentTimeMillis() - time > 10000) {

throw new TimeoutException();

}

}

boardPanel.setSaved(false);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_TEXT);

command.setUid(boardContainer.getId());

command.setNewText(this.getText());

boardPanel.getClient().addCommand(command);

}

} catch (IOException e1) {

e1.printStackTrace();

} catch (TimeoutException e2) {

JOptionPane.showMessageDialog(null, "A error occurred when reading the file :" +

" timeout (10 sec) has been reached!", "Error", JOptionPane.ERROR\_MESSAGE);

} finally {

try {

reader.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

public BoardContainer getBoardContainer() {

return boardContainer;

}

public PointerListener getPl() {

return pl;

}

public void setPl(PointerListener pl) {

this.pl = pl;

}

}

**Текст файла ImageContainer.java**

package BoardComponents;

import BoardCompFeatures.PointerListener;

import BoardCompFeatures.ImageTransferHandler;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import javax.swing.\*;

import java.awt.\*;

/\*\*

\* Created by Lev on 07.04.14.

\*/

public class ImageContainer extends JPanel {

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* Container \*/

private BoardContainer boardContainer;

/\*\* Image itself \*/

private ImageIcon image;

/\*\* Attention listener \*/

private PointerListener pl;

/\*\* Image params \*/

private int originalWidth;

private int originalHeight;

private int imageWidth;

private int imageHeight;

/\*\* Image slider \*/

private ImageSlider slider;

public ImageContainer(BoardContainer boardContainer, BoardPanel boardPanel) {

this.boardContainer = boardContainer;

this.boardPanel = boardPanel;

pl = new PointerListener(this, boardPanel);

this.setOpaque(false);

this.setTransferHandler(new ImageTransferHandler(this));

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

if (image != null) {

int x = 0;

int y = 0;

int width = boardContainer.getCompWidth() - boardContainer.getScroll().getVerticalScrollBar().getWidth() - 2;

int height = boardContainer.getCompHeight() - boardContainer.getScroll().getHorizontalScrollBar().getHeight() - 2;

if (imageWidth < width) {

x = (width - imageWidth) / 2;

}

else {

width = imageWidth;

}

if (imageHeight < height) {

y = (height - imageHeight) / 2;

}

else {

height = imageHeight;

}

this.setPreferredSize(new Dimension(width, height));

this.revalidate();

Image image1 = image.getImage();

g.drawImage(image1, x, y, imageWidth, imageHeight, null);

}

}

public void setImage(ImageIcon image) {

this.image = image;

if (image != null) {

imageWidth = image.getImage().getWidth(null);

imageHeight = image.getImage().getHeight(null);

originalWidth = imageWidth;

originalHeight = imageHeight;

this.setPreferredSize(new Dimension(imageWidth, imageHeight));

}

this.revalidate();

this.repaint();

}

public ImageIcon getII() {

return image;

}

public Image getImage() {

if (image == null) {

return null;

}

return image.getImage();

}

public int getImageWidth() {

return imageWidth;

}

public void setImageWidth(int imageWidth) {

this.imageWidth = imageWidth;

}

public int getImageHeight() {

return imageHeight;

}

public void setImageHeight(int imageHeight) {

this.imageHeight = imageHeight;

}

public int getOriginalWidth() {

return originalWidth;

}

public int getOriginalHeight() {

return originalHeight;

}

public BoardContainer getBoardContainer() {

return boardContainer;

}

public ImageSlider getSlider() {

return slider;

}

public void setSlider(ImageSlider slider) {

this.slider = slider;

}

public PointerListener getPl() {

return pl;

}

public void setPl(PointerListener pl) {

this.pl = pl;

}

/\*\*

\* Clears the image container.

\*/

public void clear() {

setImage(null);

int width = 100;//boardContainer.getCompWidth();

int height = 100;//boardContainer.getCompHeight();

imageWidth = width;

imageHeight = height;

imageWidth = width;

originalHeight = height;

this.setPreferredSize(new Dimension(width, height));

slider.setValue(100);

}

public void setNewImage(ImageIcon image) {

this.clear();

this.setImage(image);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_IMAGE);

command.setUid(boardContainer.getId());

command.setNewImage(image);

boardPanel.getClient().addCommand(command);

}

}

}

**Текст файла ImageSlider.java**

package BoardComponents;

import javax.swing.\*;

import java.awt.\*;

/\*\*

\* Created by Lev on 20.04.14.

\*/

public class ImageSlider extends JSlider {

/\*\* Scroll \*/

private JViewport viewport;

/\*\* Image \*/

private ImageContainer image;

/\*\* If the change was called by the current user \*/

private boolean changed;

public ImageSlider(JViewport viewport, ImageContainer image) {

super(0, 500, 100);

this.viewport = viewport;

this.image = image;

changed = true;

this.setMajorTickSpacing(100);

this.setPaintTicks(true);

this.setPaintLabels(true);

this.setOpaque(false);

}

public void resizeImage(int percent) {

if (image.getImage() != null) {

//Point vp = viewport.getViewPosition();

double delta = percent / 100.0;

int newWidth = (int)Math.round(delta \* image.getOriginalWidth());

int newHeight = (int)Math.round(delta \* image.getOriginalHeight());

image.setImageWidth(newWidth);

image.setImageHeight(newHeight);

//BoardContainer bc = image.getBoardContainer();

int x = image.getImageWidth() / 2 - viewport.getSize().width / 2;

int y = image.getImageHeight() / 2 - viewport.getSize().height / 2;

//image.scrollRectToVisible(new Rectangle(new Point(x, y), viewport.getSize()));

//System.out.println(viewport.getSize());

//System.out.println(bc.getCompWidth());

//System.out.println(bc.getCompHeight());

image.repaint();

}

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

BoardContainer parent = (BoardContainer)this.getParent();

Dimension dimension = new Dimension(5 \* parent.getActualWidth() / 6,

5 \* (parent.getActualHeight() - parent.getLabelButton().getHeight()

- parent.getScroll().getHeight() - parent.getContainerName().getHeight()) / 6);

this.setPreferredSize(dimension);

this.setMaximumSize(dimension);

}

public boolean isChanged() {

return changed;

}

public void setChanged(boolean changed) {

this.changed = changed;

}

}

**Текст файла AddingListener.java**

package BoardCompFeatures;

/\*\*

\* Created by Lev on 04.04.14.

\*/

import BoardComponents.BoardContainer;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* Adds elements to board panel.

\*/

public class AddingListener extends MouseAdapter {

/\*\* Parent \*/

private BoardPanel boardPanel;

/\*\* Component type \*/

private int type;

/\*\* Corresponding button \*/

private JToggleButton button;

public AddingListener(BoardPanel boardPanel, int type, JToggleButton button) {

this.boardPanel = boardPanel;

this.type = type;

this.button = button;

boardPanel.setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR\_CURSOR));

}

@Override

public void mouseReleased(MouseEvent e) {

super.mouseReleased(e);

if (boardPanel.getStartCompPoint() != null && boardPanel.getEndCompPoint() != null) {

boardPanel.removeMouseListener(this);

boardPanel.removeMouseMotionListener(this);

boardPanel.setCursor(Cursor.getDefaultCursor());

button.setSelected(false);

int id = boardPanel.generateUID();

Point location = new Point(boardPanel.getStartCompPoint());

int width = boardPanel.getEndCompPoint().x - location.x;

int height = boardPanel.getEndCompPoint().y - location.y;

boardPanel.addComponent(location, new BoardContainer(type, id, width, height, boardPanel));

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(type == BoardPanel.TEXT\_CONTAINER? CommandConstants.NEW\_TEXT\_CONTAINER :

CommandConstants.NEW\_IMAGE\_CONTAINER);

command.setUid(id);

command.setNewPoint(location);

command.setNewValue1(width);

command.setNewValue2(height);

boardPanel.getClient().addCommand(command);

}

}

boardPanel.setStartCompPoint(null);

boardPanel.setEndCompPoint(null);

boardPanel.repaint();

}

@Override

public void mousePressed(MouseEvent e) {

super.mousePressed(e);

boardPanel.setStartCompPoint(e.getPoint());

boardPanel.repaint();

/\*Graphics g = boardPanel.getGraphics();

g.fillOval(e.getX() - 4, e.getY() - 4, 8, 8);

p = e.getPoint();\*/

}

@Override

public void mouseDragged(MouseEvent e) {

super.mouseDragged(e);

boardPanel.setEndCompPoint(e.getPoint());

boardPanel.repaint();

Graphics g = boardPanel.getGraphics();

/\*g.fillOval((int)p.getX() - 4, (int)p.getY() - 4, 8, 8);

g.drawRect((int)p.getX(), (int)p.getY(), e.getX() -

(int)p.getX(), e.getY() - (int)p.getY());\*/

}

}

**Текст файла ImageTransferHandler.java**

package BoardCompFeatures;

import BoardComponents.ImageContainer;

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.datatransfer.DataFlavor;

import java.awt.datatransfer.UnsupportedFlavorException;

import java.io.File;

import java.io.IOException;

/\*\*

\* Created by Lev on 06.05.14.

\*/

public class ImageTransferHandler extends TransferHandler {

/\*\* Image container \*/

private ImageContainer container;

public ImageTransferHandler(ImageContainer container) {

this.container = container;

}

@Override

public boolean canImport(TransferSupport support) {

if (support.getUserDropAction() == LINK) {

return false;

}

for (DataFlavor flavor : support.getDataFlavors()) {

if (flavor.equals(DataFlavor.imageFlavor) ||

flavor.equals(DataFlavor.javaFileListFlavor)) {

return true;

}

}

return false;

}

@Override

public boolean importData(TransferSupport support) {

if (! canImport(support)) {

return false;

}

if (support.isDataFlavorSupported(DataFlavor.imageFlavor)) {

try {

ImageIcon image = new ImageIcon((Image)support.getTransferable().getTransferData(

DataFlavor.imageFlavor));

container.setNewImage(image);

return true;

} catch (UnsupportedFlavorException e) {

e.printStackTrace();

}

catch (IOException e) {

e.printStackTrace();

}

}

if (support.isDataFlavorSupported(DataFlavor.javaFileListFlavor)) {

java.util.List files;

try {

files = (java.util.List)support.getTransferable().getTransferData(DataFlavor.javaFileListFlavor);

if (files.size() != 1) {

return false;

}

ImageIcon image = new ImageIcon(ImageIO.read((File) files.get(0)));

container.setNewImage(image);

return true;

}

catch (UnsupportedFlavorException e) {

e.printStackTrace();

}

catch (IOException e) {

e.printStackTrace();

}

}

return false;

}

}

**Текст файла JFontChooser.java**

package BoardCompFeatures;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.Dimension;

import java.awt.GraphicsEnvironment;

import java.awt.Font;

import java.awt.Frame;

import javax.swing.JButton;

import javax.swing.JCheckBox;

import javax.swing.JDialog;

import javax.swing.JList;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.ListSelectionModel;

public class JFontChooser extends JDialog

{

private static JFontChooser instance;

public static JFontChooser getInstance() {

if (instance == null) {

instance = new JFontChooser(null);

}

return instance;

}

public static int OK\_OPTION = 0;

public static int CANCEL\_OPTION = 1;

private JList fontList, sizeList;

private JCheckBox cbBold, cbItalic;

private JTextArea txtSample;

private int option;

public int showDialog(Font font)

{

setFont(font);

return showDialog();

}

public int showDialog()

{

setVisible(true);

return option;

}

private JFontChooser(Frame frame)

{

super(frame, true);

setLocationRelativeTo(frame);

setTitle("Choose font");

option = JFontChooser.CANCEL\_OPTION;

// create all components

JButton btnOK = new JButton("OK");

btnOK.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

JFontChooser.this.option = JFontChooser.OK\_OPTION;

JFontChooser.this.setVisible(false);

}

});

JButton btnCancel = new JButton("Cancel");

btnCancel.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

JFontChooser.this.option = JFontChooser.CANCEL\_OPTION;

JFontChooser.this.setVisible(false);

}

});

fontList = new JList(GraphicsEnvironment.getLocalGraphicsEnvironment().

getAvailableFontFamilyNames())

{

public Dimension getPreferredScrollableViewportSize()

{ return new Dimension(150, 144); }

};

fontList.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

String[] sizes = new String[]

{ "2","4","6","8","10","12","14","16","18","20","22","24","30","36","48","72" };

sizeList = new JList(sizes)

{

public Dimension getPreferredScrollableViewportSize()

{ return new Dimension(25, 144); }

};

sizeList.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

cbBold = new JCheckBox("Bold");

cbItalic = new JCheckBox("Italic");

txtSample = new JTextArea()

{

public Dimension getPreferredScrollableViewportSize()

{ return new Dimension(385, 80); }

};

txtSample.setText("The quick brown fox jumped over the fence");

// set the default font

setFont(null);

// add the listeners

ListSelectionListener listListener = new ListSelectionListener()

{

public void valueChanged(ListSelectionEvent e)

{ txtSample.setFont(getCurrentFont()); }

};

fontList.addListSelectionListener(listListener);

sizeList.addListSelectionListener(listListener);

ActionListener cbListener = new ActionListener()

{

public void actionPerformed(ActionEvent e)

{ txtSample.setFont(getCurrentFont()); }

};

cbBold.addActionListener(cbListener);

cbItalic.addActionListener(cbListener);

// build the container

getContentPane().setLayout(new java.awt.BorderLayout());

JPanel leftPanel = new JPanel();

leftPanel.setLayout(new java.awt.BorderLayout());

leftPanel.add(new JScrollPane(fontList), java.awt.BorderLayout.CENTER);

leftPanel.add(new JScrollPane(sizeList), java.awt.BorderLayout.EAST);

getContentPane().add(leftPanel, java.awt.BorderLayout.CENTER);

JPanel rightPanel = new JPanel();

rightPanel.setLayout(new java.awt.BorderLayout());

JPanel rightPanelSub1 = new JPanel();

rightPanelSub1.setLayout(new java.awt.FlowLayout());

rightPanelSub1.add(cbBold);

rightPanelSub1.add(cbItalic);

rightPanel.add(rightPanelSub1, java.awt.BorderLayout.NORTH);

JPanel rightPanelSub2 = new JPanel();

rightPanelSub2.setLayout(new java.awt.GridLayout(2, 1));

rightPanelSub2.add(btnOK);

rightPanelSub2.add(btnCancel);

rightPanel.add(rightPanelSub2, java.awt.BorderLayout.SOUTH);

getContentPane().add(rightPanel, java.awt.BorderLayout.EAST);

getContentPane().add(new JScrollPane(txtSample), java.awt.BorderLayout.SOUTH);

setSize(200, 200);

setResizable(false);

pack();

}

public void setFont(Font font)

{

if (font == null) font = txtSample.getFont();

fontList.setSelectedValue(font.getName(), true);

fontList.ensureIndexIsVisible(fontList.getSelectedIndex());

sizeList.setSelectedValue("" + font.getSize(), true);

sizeList.ensureIndexIsVisible(sizeList.getSelectedIndex());

cbBold.setSelected(font.isBold());

cbItalic.setSelected(font.isItalic());

}

public Font getFont()

{

if (option == OK\_OPTION)

{

return getCurrentFont();

}

else return null;

}

private Font getCurrentFont()

{

String fontFamily = (String)fontList.getSelectedValue();

int fontSize = Integer.parseInt((String)sizeList.getSelectedValue());

int fontType = Font.PLAIN;

if (cbBold.isSelected()) fontType += Font.BOLD;

if (cbItalic.isSelected()) fontType += Font.ITALIC;

return new Font(fontFamily, fontType, fontSize);

}

}

**Текст файла MouseScrollListener.java**

package BoardCompFeatures;

/\*\*

\* Created by Lev on 26.04.14.

\*/

import BoardComponents.ImageContainer;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import MainPanels.BoardPanel;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\* Processing mouse drag scrolling \*/

public class MouseScrollListener extends MouseAdapter {

/\*\* Cursors \*/

private Cursor defCursor;

private Cursor moveCursor;

/\*\* Component params \*/

private JComponent component;

private JViewport viewport;

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* Old point \*/

private Point old;

public MouseScrollListener(JComponent component, JViewport viewport, Cursor moveCursor,

BoardPanel boardPanel) {

this.component = component;

this.viewport = viewport;

defCursor = Cursor.getPredefinedCursor(Cursor.DEFAULT\_CURSOR);

this.moveCursor = moveCursor;

this.boardPanel = boardPanel;

}

@Override public void mouseDragged(MouseEvent e) {

Point point = e.getPoint();

Point vp = viewport.getViewPosition();

vp.translate(old.x - point.x, old.y - point.y);

component.scrollRectToVisible(new Rectangle(vp, viewport.getSize()));

if (component instanceof BoardPanel) {

old = e.getPoint();

}

else {

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.IMAGE\_SCROLL);

command.setUid(((ImageContainer) component).getBoardContainer().getId());

command.setNewPoint(point);

command.setNewValue1(old.x);

command.setNewValue2(old.y);

boardPanel.getClient().addCommand(command);

}

}

}

@Override public void mousePressed(MouseEvent e) {

component.setCursor(moveCursor);

old = e.getPoint();

}

@Override public void mouseReleased(MouseEvent e) {

component.setCursor(defCursor);

}

}

**Текст файла MoveListener.java**

package BoardCompFeatures;

import BoardComponents.BoardContainer;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* Created by Lev on 10.04.14.

\*/

public class MoveListener extends MouseAdapter {

/\*\* The container to move \*/

final private BoardContainer container;

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* Old point \*/

private Point old;

/\*\* Source cursor \*/

private Cursor sourceCursor;

/\*\* If moving \*/

private boolean moving;

/\*\* Delta \*/

final public static int DELTA = 50;

public MoveListener(BoardContainer container, BoardPanel boardPanel) {

this.container = container;

this.boardPanel = boardPanel;

}

public boolean isMoving() {

return moving;

}

/\*\*

\* For the cursor.

\*/

@Override

public void mouseEntered(MouseEvent e) {

if (! moving) {

sourceCursor = container.getCursor();

}

}

/\*\*

\* For the cursor.

\*/

@Override

public void mouseExited(MouseEvent e) {

if (! moving) {

container.setCursor(sourceCursor);

}

}

@Override

public void mousePressed(MouseEvent e) {

container.setCursor(Cursor.getPredefinedCursor(Cursor.MOVE\_CURSOR));

old = e.getPoint();

moving = true;

}

@Override

public void mouseReleased(MouseEvent e) {

container.setCursor(sourceCursor);

moving = false;

if (! old.equals(e.getPoint())) {

boardPanel.setSaved(false);

}

}

@Override

public void mouseDragged(MouseEvent e) {

Point current = e.getPoint();

int x = container.getX() + current.x - old.x;

int y = container.getY() + current.y - old.y;

if (x > boardPanel.getActualWidth() - DELTA) {

x = boardPanel.getActualWidth() - DELTA;

}

if (x + container.getActualWidth() < DELTA) {

x = DELTA - container.getActualWidth();

}

if (y > boardPanel.getActualHeight() - DELTA) {

y = boardPanel.getActualHeight() - DELTA;

}

if (y + container.getActualHeight() < DELTA) {

y = DELTA - container.getActualHeight();

}

container.setLocation(x, y);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.MOVE\_CONTAINER);

command.setUid(container.getId());

command.setNewPoint(container.getLocation());

boardPanel.getClient().addCommand(command);

}

}

}

**Текст файла PointerListener.java**

package BoardCompFeatures;

import BoardComponents.BoardContainer;

import BoardComponents.ImageContainer;

import BoardComponents.TextContainer;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import javax.swing.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* Created by Lev on 10.05.14.

\*/

public class PointerListener extends MouseAdapter {

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* Component \*/

private JComponent component;

public PointerListener(JComponent component, BoardPanel boardPanel) {

this.component = component;

this.boardPanel = boardPanel;

}

@Override

public void mouseReleased(MouseEvent e) {

super.mouseClicked(e);

BoardPanel.pointerCommand(component, e.getPoint());

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.POINTER);

command.setNewPoint(e.getPoint());

if (component instanceof BoardContainer) {

command.setUid(((BoardContainer) component).getId());

command.setNewValue1(-1);

}

if (component instanceof TextContainer) {

command.setUid(((TextContainer) component).getBoardContainer().getId());

command.setNewValue1(0);

}

if (component instanceof ImageContainer) {

command.setUid(((ImageContainer) component).getBoardContainer().getId());

command.setNewValue1(0);

}

boardPanel.getClient().addCommand(command);

}

}

}

**Текст файла ResizeListener.java**

package BoardCompFeatures;

import BoardComponents.BoardContainer;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

import javax.swing.SwingUtilities;

public class ResizeListener extends MouseAdapter {

/\*\* The container to resize \*/

private BoardContainer container;

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* For defining resizing direction \*/

private static Map<Integer, Integer> cursors = new HashMap<Integer, Integer>();

{

cursors.put(1, Cursor.N\_RESIZE\_CURSOR);

cursors.put(2, Cursor.W\_RESIZE\_CURSOR);

cursors.put(4, Cursor.S\_RESIZE\_CURSOR);

cursors.put(8, Cursor.E\_RESIZE\_CURSOR);

cursors.put(3, Cursor.NW\_RESIZE\_CURSOR);

cursors.put(9, Cursor.NE\_RESIZE\_CURSOR);

cursors.put(6, Cursor.SW\_RESIZE\_CURSOR);

cursors.put(12, Cursor.SE\_RESIZE\_CURSOR);

}

private int direction;

private static final int NORTH = 1;

private static final int WEST = 2;

private static final int SOUTH = 4;

private static final int EAST = 8;

/\*\* Source cursor \*/

private Cursor sourceCursor;

/\*\* If resizing \*/

private boolean resizing;

/\*\* Old values \*/

private Rectangle bounds;

private Point old;

/\*\* Temp values \*/

private int tempX;

private int tempY;

public ResizeListener(BoardContainer container, BoardPanel boardPanel) {

this.container = container;

this.boardPanel = boardPanel;

}

/\*\*

\* For the cursor.

\*/

@Override

public void mouseEntered(MouseEvent e) {

if (! (resizing || container.getMoveListener().isMoving())) {

sourceCursor = container.getCursor();

}

}

/\*\*

\* For the cursor.

\*/

@Override

public void mouseExited(MouseEvent e) {

if (! (resizing || container.getMoveListener().isMoving())) {

container.setCursor(sourceCursor);

}

}

@Override

public void mouseMoved(MouseEvent e) {

Component source = e.getComponent();

Point location = e.getPoint();

direction = 0;

int error = 10;

if (location.x < error)

direction += WEST;

if (location.x > source.getWidth() - error - 1)

direction += EAST;

if (location.y < error)

direction += NORTH;

if (location.y > source.getHeight() - error - 1)

direction += SOUTH;

if (direction == 0) {

container.setCursor( sourceCursor );

if (! container.isMl()) {

container.addMouseListener(container.getMoveListener());

container.addMouseMotionListener(container.getMoveListener());

container.setMl(true);

}

}

else {

Cursor cursor = Cursor.getPredefinedCursor(cursors.get(direction));

container.setCursor(cursor);

if (container.isMl()) {

container.removeMouseListener(container.getMoveListener());

container.removeMouseMotionListener(container.getMoveListener());

container.setMl(false);

}

}

}

/\*\*

\* Remembering params.

\*/

@Override

public void mousePressed(MouseEvent e) {

if (direction == 0) {

return;

}

resizing = true;

old = e.getPoint();

SwingUtilities.convertPointToScreen(old, container);

bounds = container.getBounds();

tempX = bounds.x;

tempY = bounds.y;

}

/\*\*

\* For the cursor.

\*/

@Override

public void mouseReleased(MouseEvent e) {

resizing = false;

container.setCursor(sourceCursor);

if (! container.isMl()) {

container.addMouseListener(container.getMoveListener());

container.addMouseMotionListener(container.getMoveListener());

container.setMl(true);

}

if (old != null) {

if (! old.equals(e.getPoint())) {

//((BoardPanel)(container.getScroll().getParent())).setSaved(false);

}

}

}

/\*\*

\* Checking and resizing.

\*/

@Override

public void mouseDragged(MouseEvent e) {

if (! resizing) {

return;

}

Point dragged = e.getPoint();

SwingUtilities.convertPointToScreen(dragged, container);

resize(direction, bounds, old, dragged);

}

/\*\*

\* Resizing itself.

\* @param direction the direction of resizing

\* @param bounds old bounds

\* @param pressed the old point

\* @param current the dragged point

\*/

private void resize(int direction, Rectangle bounds, Point pressed, Point current) {

int x = bounds.x;

int y = bounds.y;

int width = bounds.width;

int height = bounds.height;

int dragX = current.x - pressed.x;

int dragY = current.y - pressed.y;

if (WEST == (direction & WEST)) {

x += dragX;

width -= dragX;

}

if (NORTH == (direction & NORTH)) {

y += dragY;

height -= dragY;

}

if (EAST == (direction & EAST)) {

width += dragX;

}

if (SOUTH == (direction & SOUTH)) {

height += dragY;

}

int minimumSize = container.getType() == BoardPanel.TEXT\_CONTAINER ? 90 : 200;

width = Math.max(width, minimumSize);

height = Math.max(height, minimumSize);

if (width == minimumSize) {

x = tempX;

}

if (height == minimumSize) {

y = tempY;

}

if (x > boardPanel.getActualWidth() - MoveListener.DELTA) {

x = boardPanel.getActualWidth() - MoveListener.DELTA;

}

if (x + container.getActualWidth() < MoveListener.DELTA) {

x = MoveListener.DELTA - container.getActualWidth();

}

if (y > boardPanel.getActualHeight() - MoveListener.DELTA) {

y = boardPanel.getActualHeight() - MoveListener.DELTA;

}

if (y + container.getActualHeight() < MoveListener.DELTA) {

y = MoveListener.DELTA - container.getActualHeight();

}

container.setActualWidth(width);

container.setActualHeight(height);

container.setBounds(x, y, width, height);

tempX = x;

tempY = y;

container.validate();

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.RESIZE\_CONTAINER);

command.setUid(container.getId());

command.setNewPoint(container.getLocation());

command.setNewValue1(container.getActualWidth());

command.setNewValue2(container.getActualHeight());

boardPanel.getClient().addCommand(command);

}

}

}

**Текст файла SliderListener.java**

package BoardCompFeatures;

import BoardComponents.BoardContainer;

import BoardComponents.ImageSlider;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import MainPanels.BoardPanel;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

/\*\*

\* Created by Lev on 29.04.14.

\*/

public class SliderListener implements ChangeListener {

/\*\* Board container \*/

private BoardContainer container;

/\*\* Board panel \*/

private BoardPanel boardPanel;

public SliderListener(BoardContainer container, BoardPanel boardPanel) {

this.container = container;

this.boardPanel = boardPanel;

}

@Override

public void stateChanged(ChangeEvent e) {

ImageSlider slider = (ImageSlider)e.getSource();

if (slider.isChanged()) {

slider.resizeImage(slider.getValue());

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.IMAGE\_RESIZE);

command.setUid(container.getId());

command.setNewValue1(slider.getValue());

boardPanel.getClient().addCommand(command);

}

}

}

}

**Текст файла StandardPopUpMenu.java**

package BoardCompFeatures;

import BoardComponents.BoardContainer;

import MainPanels.BoardPanel;

import ClientServerPart.Command;

import ClientServerPart.CommandConstants;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* Created by Lev on 06.04.14.

\*/

public class StandardPopUpMenu extends JPopupMenu {

/\*\* The board \*/

private BoardPanel boardPanel;

/\*\* The container \*/

private BoardContainer container;

public StandardPopUpMenu(final BoardPanel boardPanel, final BoardContainer container) {

this.boardPanel = boardPanel;

this.container = container;

createButtons();

}

private void createButtons() {

// Setting name

JMenuItem menuItemSetName = new JMenuItem("Set name");

menuItemSetName.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String newName = JOptionPane.showInputDialog(container, "Enter name:");

container.setContainerName(newName);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_NAME);

command.setUid(container.getId());

command.setNewText(newName);

boardPanel.getClient().addCommand(command);

}

}

});

this.add(menuItemSetName);

// Set font

JMenuItem menuItemSetFont = new JMenuItem("Set text font");

menuItemSetFont.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFontChooser fc = JFontChooser.getInstance();

int result = fc.showDialog(container.getFont());

if (result == JFontChooser.OK\_OPTION) {

container.setContainerTextFont(fc.getFont());

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_FONT);

command.setUid(container.getId());

command.setNewFont(fc.getFont());

boardPanel.getClient().addCommand(command);

}

}

}

});

this.add(menuItemSetFont);

// Set color

JMenuItem menuItemSetColor = new JMenuItem("Set text color");

menuItemSetColor.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Color old = container.getContainerName().getForeground();

Color result = JColorChooser.showDialog(null,

"Choose color", container.getContainerName().getForeground());

if (result != null && ! result.equals(old)) {

container.setContainerTextColor(result);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.NEW\_COLOR);

command.setUid(container.getId());

command.setNewColor(result);

boardPanel.getClient().addCommand(command);

}

}

}

});

this.add(menuItemSetColor);

// Front button

JMenuItem menuItemFront = new JMenuItem("To front");

menuItemFront.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

boardPanel.componentToFront(container);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.TO\_FRONT);

command.setUid(container.getId());

boardPanel.getClient().addCommand(command);

}

}

});

this.add(menuItemFront);

// Background button

JMenuItem menuItemBackground = new JMenuItem("To background");

menuItemBackground.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

boardPanel.componentToBackground(container);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.TO\_BACKGROUND);

command.setUid(container.getId());

boardPanel.getClient().addCommand(command);

}

}

});

this.add(menuItemBackground);

// Clear button

JMenuItem menuItemClear = new JMenuItem("Clear");

menuItemClear.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

container.clearContainer();

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.CLEAR\_CONTAINER);

command.setUid(container.getId());

boardPanel.getClient().addCommand(command);

}

}

});

this.add(menuItemClear);

// Delete button

JMenuItem menuItemDelete = new JMenuItem("Delete");

menuItemDelete.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

boardPanel.deleteComponent(container);

if (boardPanel.getClient() != null) {

// Send to others

Command command = new Command(CommandConstants.DELETE\_CONTAINER);

command.setUid(container.getId());

boardPanel.getClient().addCommand(command);

}

}

});

this.add(menuItemDelete);

}

public MouseAdapter setPopUpMenu() {

final StandardPopUpMenu menu = this;

return new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

super.mousePressed(e);

pop(e);

}

@Override

public void mouseReleased(MouseEvent e) {

super.mouseReleased(e);

pop(e);

}

private void pop(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON3 && e.isPopupTrigger()) {

menu.show(e.getComponent(), e.getX(), e.getY());

}

}

};

}

}

**Текст файла TexTransferHandler.java**

package BoardCompFeatures;

import BoardComponents.TextContainer;

import javax.swing.\*;

import java.awt.datatransfer.DataFlavor;

import java.awt.datatransfer.Transferable;

import java.awt.datatransfer.UnsupportedFlavorException;

import java.io.File;

import java.io.IOException;

/\*\*

\* Created by Lev on 06.05.14.

\*/

public class TextTransferHandler extends TransferHandler {

/\*\* Text container \*/

private TextContainer container;

public TextTransferHandler(TextContainer container) {

this.container = container;

}

@Override

public boolean canImport(TransferSupport support) {

// Checking

return support.isDataFlavorSupported(DataFlavor.javaFileListFlavor);

}

@Override

public boolean importData(TransferSupport support) {

if (!support.isDrop()) {

return false;

}

// Get the data

Transferable t = support.getTransferable();

java.util.List files;

// Checking

try {

files = (java.util.List)t.getTransferData(DataFlavor.javaFileListFlavor);

}

catch (UnsupportedFlavorException e) {

e.printStackTrace();

return false;

}

catch (IOException e) {

e.printStackTrace();

return false;

}

if (files.size() != 1) {

System.out.println(files.size());

return false;

}

container.readFile((File) files.get(0));

return true;

}

}

**Текст файла BoardClient.java**

package ClientServerPart;

import MainPanels.BoardPanel;

import javax.swing.\*;

import java.io.\*;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.ClosedByInterruptException;

import java.nio.channels.NotYetConnectedException;

import java.nio.channels.SocketChannel;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Vector;

/\*\*

\* Created by Lev on 04.05.14.

\*/

public class BoardClient {

/\*\* Commands to be delivered to the server \*/

private Vector<Command> commands;

/\*\* Board panel \*/

private BoardPanel boardPanel;

/\*\* IP address \*/

private String ipAddress;

/\*\* Port \*/

private int port;

/\*\* Password \*/

private String password;

/\*\* Channel \*/

private MySocketChannel msc;

public BoardClient(BoardPanel boardPanel, String ipAddress, int port, String password) {

this.boardPanel = boardPanel;

this.port = port;

this.password = password;

if (ipAddress == null) {

this.ipAddress = "localhost";

}

else {

this.ipAddress = ipAddress;

}

boardPanel.setClient(this);

commands = new Vector<Command>();

}

public void start() {

// Working

while (true) {

try {

SocketChannel socketChannel = msc.getSocketChannel();

// Checking if it is the end of connection

if (! socketChannel.isConnected() || Thread.interrupted()) {

socketChannel.close();

return;

}

// Checking input

readCommandFromChannel(msc);

if (msc.isRead()) {

// Checking output

while (! commands.isEmpty()) {

Command command = commands.lastElement();

commands.remove(command);

writeCommandToChannel(msc, command);

}

}

}

catch (ClosedByInterruptException e1) {

//e1.printStackTrace();

// all is normal

return;

}

catch (IOException e2) {

//e2.printStackTrace();

// dc

return;

}

catch (ClassNotFoundException e3) {

e3.printStackTrace();

}

}

}

public void addCommand(Command command) {

commands.add(command);

}

public void setBoardPanel(BoardPanel boardPanel) {

this.boardPanel = boardPanel;

}

public static void writeCommandToChannel(MySocketChannel msc, Command command)

throws IOException {

SocketChannel channel = msc.getSocketChannel();

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ObjectOutputStream oos = new ObjectOutputStream(baos);

oos.writeObject(command);

byte[] objData = baos.toByteArray();

ByteBuffer buffer = ByteBuffer.allocate(objData.length + 4);

// The length of the command

buffer.putInt(objData.length);

buffer.put(objData);

buffer.flip();

baos.close();

oos.close();

while (buffer.hasRemaining()) {

channel.write(buffer);

}

// this message is unread

msc.setRead(false);

//System.out.println(objData.length + " written");

}

private boolean readCommandFromChannel(MySocketChannel msc)

throws IOException, ClassNotFoundException {

boolean flag = true;

SocketChannel channel = msc.getSocketChannel();

if (msc.getCommandLength() == -1) {

// Reading size;

flag = false;

ByteBuffer buffer = ByteBuffer.allocate(4);

msc.setReadBuffer(buffer);

int bytes = channel.read(buffer);

if (bytes != 0) {

flag = true;

buffer.position(0);

int length = buffer.getInt();

// if the command that the message from here was read (or end of connection)

if (length == -1 || length == 0) {

msc.setRead(true);

return true;

}

msc.setCommandLength(length);

}

}

if (flag) {

if (msc.getBytesRead() == 0) {

ByteBuffer buffer = ByteBuffer.allocate(msc.getCommandLength());

msc.setReadBuffer(buffer);

}

ByteBuffer buffer = msc.getReadBuffer();

int bytesRead = channel.read(buffer);

msc.setBytesRead(msc.getBytesRead() + bytesRead);

//System.out.println(msc.getBytesRead() + " received, buffer size: " + buffer.capacity());

// If all the message is read

if (msc.getBytesRead() == msc.getCommandLength()) {

//System.out.println(msc.getBytesRead() + " received at last");

msc.setCommandLength(-1);

msc.setBytesRead(0);

// process it

buffer.flip();

//System.out.println(buffer.capacity() + " read");

ByteArrayInputStream bais = new ByteArrayInputStream(buffer.array());

ObjectInputStream ois = new ObjectInputStream(bais);

Command command = (Command)ois.readObject();

bais.close();

ois.close();

boardPanel.processCommand(command);

return true;

}

}

return false;

}

public static ByteBuffer transferCommand(MySocketChannel msc)

throws IOException {

SocketChannel channel = msc.getSocketChannel();

boolean flag = true;

if (msc.getCommandLength() == -1) {

// Reading size;

flag = false;

ByteBuffer buffer = ByteBuffer.allocate(4);

msc.setReadBuffer(buffer);

int bytes = channel.read(buffer);

if (bytes != 0) {

flag = true;

buffer.position(0);

int length = buffer.getInt();

//System.out.println(length + " length got");

msc.setCommandLength(length);

}

}

if (flag) {

if (msc.getBytesRead() == 0) {

ByteBuffer buffer = ByteBuffer.allocate(msc.getCommandLength() + 4);

msc.setReadBuffer(buffer);

buffer.putInt(msc.getCommandLength());

}

ByteBuffer buffer = msc.getReadBuffer();

int bytesRead = channel.read(buffer);

msc.setBytesRead(msc.getBytesRead() + bytesRead);

//System.out.println(bytesRead + "; total: " + msc.getBytesRead());

//System.out.println(msc.getBytesRead() + " bytes got, buffer size: " + buffer.capacity());

// If all the message is read

if (msc.getBytesRead() == msc.getCommandLength()) {

//System.out.println(msc.getBytesRead() + " bytes got at last");

msc.setCommandLength(-1);

msc.setBytesRead(0);

buffer.flip();

return buffer;

}

}

return null;

}

public boolean setClient() {

// Connecting

try {

msc = new MySocketChannel(SocketChannel.open());

SocketChannel socketChannel = msc.getSocketChannel();

socketChannel.configureBlocking(false);

socketChannel.connect(new InetSocketAddress(ipAddress, port));

long time = System.currentTimeMillis();

while (! socketChannel.finishConnect()) {

if (System.currentTimeMillis() - time > 1000) {

throw new NotYetConnectedException();

}

}

/\*if (socketChannel.finishConnect()) {

System.out.println("connected");

}\*/

// sending password

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] array = md.digest(password.getBytes("UTF-8"));

ByteBuffer buffer = ByteBuffer.wrap(array);

while (buffer.hasRemaining()) {

socketChannel.write(buffer);

}

// Waiting for server answer

ByteBuffer readBuffer = ByteBuffer.allocate(1);

while (true) {

int bytes = socketChannel.read(readBuffer);

if (bytes != 0) {

if (bytes != 1) {

throw new IllegalArgumentException();

}

readBuffer.flip();

byte answer = readBuffer.get();

switch (answer) {

case BoardServer.ACCEPT:

try {

while (! readCommandFromChannel(msc)) {

// wait to receive new board

}

}

catch (ClassNotFoundException e) {

e.printStackTrace();

}

return true;

case BoardServer.DENY:

socketChannel.socket().close();

JOptionPane.showMessageDialog(null, "Access denied: wrong password!",

"Error", JOptionPane.ERROR\_MESSAGE);

return false;

default:

throw new IllegalArgumentException();

}

}

}

}

catch (NotYetConnectedException e1) {

JOptionPane.showMessageDialog(null, "Could not connect to server with IP Address \"" + ipAddress +

"\"and port \"" + port + "\"!", "Error", JOptionPane.ERROR\_MESSAGE);

}

catch (NoSuchAlgorithmException e2) {

e2.printStackTrace();

}

catch (IllegalArgumentException e3) {

JOptionPane.showMessageDialog(null, "Could not connect to server with IP Address \"" + ipAddress +

"\"and port \"" + port + "\"!", "Error", JOptionPane.ERROR\_MESSAGE);

}

catch (IOException e4) {

JOptionPane.showMessageDialog(null, "Could not connect to server with IP Address \"" + ipAddress +

"\"and port \"" + port + "\"!", "Error", JOptionPane.ERROR\_MESSAGE);

}

return false;

}

}

**Текст файла BoardServer.java**

package ClientServerPart;

import MainPanels.BoardPanel;

import javax.swing.\*;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.net.BindException;

import java.net.InetAddress;

import java.net.InetSocketAddress;

import java.net.UnknownHostException;

import java.nio.ByteBuffer;

import java.nio.channels.\*;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Iterator;

/\*\*

\* Created by Lev on 04.05.14.

\*/

public class BoardServer {

/\*\* Constants \*/

final public static byte ACCEPT = 33;

final public static byte DENY = 66;

/\*\* Server port \*/

private int port;

/\*\* Server password \*/

private byte[] password;

/\*\* All socket channels \*/

private ArrayList<MySocketChannel> channels;

/\*\* Waiting channels (they are being written) \*/

private ArrayList<MySocketChannel> awaitingChannels;

/\*\* Selector \*/

private Selector selector;

/\*\* Server socket channel \*/

private ServerSocketChannel serverSocketChannel;

/\*\* Board panel \*/

private BoardPanel boardPanel;

public BoardServer(BoardPanel boardPanel, int port, String password) {

this.boardPanel = boardPanel;

this.port = port;

try {

MessageDigest md = MessageDigest.getInstance("MD5");

this.password = md.digest(password.getBytes("UTF-8"));

}

catch (NoSuchAlgorithmException e1) {

e1.printStackTrace();

}

catch (UnsupportedEncodingException e2) {

e2.printStackTrace();

}

channels = new ArrayList<MySocketChannel>();

awaitingChannels = new ArrayList<MySocketChannel>();

}

public void start() {

while (selector.isOpen()) {

try {

if (Thread.interrupted()) {

serverSocketChannel.close();

selector.close();

return;

}

// If no updates

if (selector.select() == 0) {

continue;

}

Iterator it = selector.selectedKeys().iterator();

while (it.hasNext()) {

SelectionKey key = (SelectionKey)it.next();

it.remove();

if (key.isAcceptable()) {

// Accept a new client

SocketChannel channel = serverSocketChannel.accept();

if (channel != null) {

channel.configureBlocking(false);

channel.register(selector, SelectionKey.OP\_READ | SelectionKey.OP\_WRITE);

channels.add(new MySocketChannel(channel));

}

}

if (key.isReadable()) {

SocketChannel channel = (SocketChannel)key.channel();

// finding the socket

MySocketChannel msc = null;

for (MySocketChannel msc1 : channels) {

if (msc1.getSocketChannel() == channel) {

msc = msc1;

break;

}

}

if (msc == null) {

throw new NullPointerException("The channel not found!");

}

try {

// if the channel is not authorized, try to get the password

if (! msc.isAuthorized()) {

// getting password

ByteBuffer buffer = ByteBuffer.allocate(16);

channel.read(buffer);

ByteBuffer writeBuffer = ByteBuffer.allocate(1);

// checking

if (Arrays.equals(buffer.array(), password)) {

msc.setAuthorized(true);

writeBuffer.put(ACCEPT);

writeBuffer.flip();

channel.write(writeBuffer);

// All is blocked in order to sent the board to the new client

Command command = new Command(CommandConstants.NEW\_BOARD);

command.setNewBoard(boardPanel.toSerializable());

BoardClient.writeCommandToChannel(msc, command);

}

else {

key.cancel();

channels.remove(msc);

writeBuffer.put(DENY);

writeBuffer.flip();

channel.write(writeBuffer);

}

}

// else read the command

else {

// if no channels to write the previous command

if (awaitingChannels.size() == 0) {

// Reading the command

ByteBuffer buffer = BoardClient.transferCommand(msc);

// If the message is read

if (buffer != null) {

// tell the client about it

ByteBuffer buffer1 = ByteBuffer.allocate(4);

buffer1.putInt(-1);

buffer1.flip();

channel.write(buffer1);

msc.setReadBuffer(null);

// Sending to other

for (MySocketChannel channel1 : channels) {

if (msc == channel1) {

continue;

}

awaitingChannels.add(channel1);

buffer.position(msc.getBytesWritten());

channel1.setWriteBuffer(buffer);

int bytes = channel1.getSocketChannel().write(buffer);

channel1.setBytesWritten(channel1.getBytesWritten() + bytes);

//System.out.println(channel1.getBytesWritten() + "written to client");

// if all is written

if (channel1.getBytesWritten() == buffer.capacity()) {

//System.out.println(channel1.getBytesWritten() + "written to client");

channel1.setBytesWritten(0);

msc.setWriteBuffer(null);

awaitingChannels.remove(channel1);

}

}

}

}

}

}

catch(IOException e) {

// it disconnected

key.cancel();

channels.remove(msc);

}

}

if (key.isWritable()) {

// If there is smth to write

if (awaitingChannels.size() != 0) {

for (int i = 0; i < awaitingChannels.size();) {

MySocketChannel msc1 = awaitingChannels.get(i);

ByteBuffer buffer = msc1.getWriteBuffer();

buffer.position(msc1.getBytesWritten());

int bytes = msc1.getSocketChannel().write(buffer);

msc1.setBytesWritten(msc1.getBytesWritten() + bytes);

//System.out.println(msc1.getBytesWritten() + "written to client");

if (msc1.getBytesWritten() == buffer.capacity()) {

msc1.setBytesWritten(0);

msc1.setWriteBuffer(null);

awaitingChannels.remove(msc1);

}

else {

++ i;

}

}

}

}

}

}

catch (CancelledKeyException e1) {

// all is normal

}

catch (ClosedByInterruptException e2) {

// all is normal

}

catch (IOException e3) {

e3.printStackTrace();

}

}

}

public boolean createServer() {

try {

serverSocketChannel = ServerSocketChannel.open();

serverSocketChannel.socket().bind(new InetSocketAddress(port));

serverSocketChannel.configureBlocking(false);

selector = Selector.open();

serverSocketChannel.register(selector, SelectionKey.OP\_ACCEPT);

return true;

}

catch (BindException e1) {

JOptionPane.showMessageDialog(null, "Server on this port already exists!",

"Error", JOptionPane.ERROR\_MESSAGE);

}

catch (IOException e2) {

JOptionPane.showMessageDialog(null, "A error occurred when creating the server!",

"Error", JOptionPane.ERROR\_MESSAGE);

}

return false;

}

public static String getIpAddress() {

String result = null;

try {

result = InetAddress.getLocalHost().getHostAddress();

}

catch (UnknownHostException e) {

e.printStackTrace();

}

return result;

}

public void setBoardPanel(BoardPanel boardPanel) {

this.boardPanel = boardPanel;

}

}

**Текст файла Command.java**

package ClientServerPart;

import javax.swing.\*;

import java.awt.\*;

import java.io.Serializable;

/\*\*

\* Created by Lev on 04.05.14.

\*/

public class Command implements Serializable {

/\*\* Command type (see CommandConstants) \*/

private byte commandType;

/\*\* Component uid \*/

private int uid;

/\*\* If the new board occurred \*/

private SerBoardPanel newBoard;

/\*\* If the new text or name occurred \*/

private String newText;

/\*\* If the new image occurred \*/

private ImageIcon newImage;

/\*\* If the new font occurred \*/

private Font newFont;

/\*\* If the new color occurred \*/

private Color newColor;

/\*\* If moved, new point \*/

private Point newPoint;

/\*\* New width (or caret position) \*/

private int newValue1;

/\*\* New height (or char) \*/

private int newValue2;

public Command(byte commandType) {

this.commandType = commandType;

}

public byte getCommandType() {

return commandType;

}

public int getUid() {

return uid;

}

public void setUid(int uid) {

this.uid = uid;

}

public String getNewText() {

return newText;

}

public void setNewText(String newText) {

this.newText = newText;

}

public ImageIcon getNewImage() {

return newImage;

}

public void setNewImage(ImageIcon newImage) {

this.newImage = newImage;

}

public Color getNewColor() {

return newColor;

}

public void setNewColor(Color newColor) {

this.newColor = newColor;

}

public int getNewValue2() {

return newValue2;

}

public void setNewValue2(int newValue2) {

this.newValue2 = newValue2;

}

public int getNewValue1() {

return newValue1;

}

public void setNewValue1(int newValue1) {

this.newValue1 = newValue1;

}

public Point getNewPoint() {

return newPoint;

}

public void setNewPoint(Point newPoint) {

this.newPoint = newPoint;

}

public Font getNewFont() {

return newFont;

}

public void setNewFont(Font newFont) {

this.newFont = newFont;

}

public SerBoardPanel getNewBoard() {

return newBoard;

}

public void setNewBoard(SerBoardPanel newBoard) {

this.newBoard = newBoard;

}

}

**Текст файла CommandConstants.java**

package ClientServerPart;

/\*\*

\* Created by Lev on 04.05.14.

\*/

public class CommandConstants {

final public static byte CLEAR\_BOARD = 0;

final public static byte EXTEND\_BOARD = 1;

final public static byte NEW\_BOARD = 2;

final public static byte NEW\_TEXT\_CONTAINER = 3;

final public static byte NEW\_IMAGE\_CONTAINER = 4;

final public static byte NEW\_GENERAL\_FONT = 5;

final public static byte NEW\_GENERAL\_COLOR = 6;

final public static byte NEW\_BOARD\_BACKGROUND = 7;

final public static byte POINTER = 8;

final public static byte DELETE\_CONTAINER = 9;

final public static byte MOVE\_CONTAINER = 10;

final public static byte RESIZE\_CONTAINER = 11;

final public static byte NEW\_NAME = 12;

final public static byte NEW\_FONT = 13;

final public static byte NEW\_COLOR = 14;

final public static byte NEW\_TEXT = 15;

final public static byte NEW\_IMAGE = 16;

final public static byte TO\_FRONT = 17;

final public static byte TO\_BACKGROUND = 18;

final public static byte CLEAR\_CONTAINER = 19;

final public static byte IMAGE\_SCROLL = 20;

final public static byte IMAGE\_RESIZE = 21;

final public static byte GENERAL\_COMMANDS = 8;

}

**Текст файла ConnectionData.java**

package ClientServerPart;

import javax.swing.\*;

/\*\*

\* Created by Lev on 07.05.14.

\*/

public class ConnectionData {

/\*\* IP Address\*/

private String ipAddress;

/\*\* Port \*/

private int port;

/\*\* Password \*/

private String password;

/\*\* Ask for ipAddress \*/

private boolean askIP;

public ConnectionData(boolean askIP) {

this.askIP = askIP;

}

public boolean askForData() {

if (askIP) {

ipAddress = JOptionPane.showInputDialog(null, "Enter IP Address:");

if (ipAddress == null) {

return false;

}

}

while (true) {

try {

String message = JOptionPane.showInputDialog(null, "Enter port (from 1 to 65535):");

if (message == null) {

return false;

}

port = (Integer.parseInt(message));

}

catch (NumberFormatException e ) {

JOptionPane.showMessageDialog(null, "Incorrect input! See the conditions!",

"Error", JOptionPane.ERROR\_MESSAGE);

continue;

}

if (port < 1 || port > 65535) {

JOptionPane.showMessageDialog(null, "Incorrect input! See the conditions!",

"Error", JOptionPane.ERROR\_MESSAGE);

continue;

}

break;

}

while (true) {

password = JOptionPane.showInputDialog(null, "Enter password (no more than 12 characters): ");

if (password == null) {

return false;

}

if (password.length() > 12) {

JOptionPane.showMessageDialog(null, "Incorrect input! See the conditions!",

"Error", JOptionPane.ERROR\_MESSAGE);

continue;

}

return true;

}

}

public int getPort() {

return port;

}

public String getPassword() {

return password;

}

public String getIpAddress() {

return ipAddress;

}

}

**Текст файла MySocketChannel.java**

package ClientServerPart;

import java.nio.ByteBuffer;

import java.nio.channels.SocketChannel;

/\*\*

\* Created by Lev on 05.05.14.

\*/

public class MySocketChannel {

/\*\* Socket channel itself\*/

private SocketChannel socketChannel;

/\*\* If this channel was authorized \*/

private boolean authorized;

/\*\* The command length \*/

private int commandLength;

/\*\* Counter for already read bytes \*/

private int bytesRead;

/\*\* Counter for already read bytes \*/

private int bytesWritten;

/\*\* Buffer to read the data \*/

private ByteBuffer readBuffer;

/\*\* Buffer to write the data \*/

private ByteBuffer writeBuffer;

/\*\* If the current command was read by the server \*/

private volatile boolean read;

public MySocketChannel(SocketChannel socketChannel) {

this.socketChannel = socketChannel;

commandLength = -1;

bytesRead = 0;

bytesWritten = 0;

read = true;

authorized = false;

}

public SocketChannel getSocketChannel() {

return socketChannel;

}

public int getCommandLength() {

return commandLength;

}

public void setCommandLength(int commandLength) {

this.commandLength = commandLength;

}

public int getBytesRead() {

return bytesRead;

}

public void setBytesRead(int bytesRead) {

this.bytesRead = bytesRead;

}

public ByteBuffer getReadBuffer() {

return readBuffer;

}

public void setReadBuffer(ByteBuffer readBuffer) {

this.readBuffer = readBuffer;

}

public ByteBuffer getWriteBuffer() {

return writeBuffer;

}

public void setWriteBuffer(ByteBuffer writeBuffer) {

this.writeBuffer = writeBuffer;

}

public int getBytesWritten() {

return bytesWritten;

}

public void setBytesWritten(int bytesWritten) {

this.bytesWritten = bytesWritten;

}

public boolean isRead() {

return read;

}

public void setRead(boolean read) {

this.read = read;

}

public boolean isAuthorized() {

return authorized;

}

public void setAuthorized(boolean authorized) {

this.authorized = authorized;

}

}

**Текст файла SerBoardContainer.java**

package ClientServerPart;

import javax.swing.\*;

import java.awt.\*;

import java.io.Serializable;

/\*\*

\* Created by Lev on 10.05.14.

\*/

public class SerBoardContainer implements Serializable {

/\*\* Component type \*/

private int type;

/\*\* UID number \*/

private int uid;

/\*\* Component location \*/

private Point location;

/\*\* The width of the container \*/

private int actualWidth;

/\*\* The height of the container \*/

private int actualHeight;

/\*\* Text font of the container \*/

private Font font;

/\*\* Text color of the container \*/

private Color color;

/\*\* Name \*/

private String name;

/\*\* Text \*/

private String text;

/\*\* Image \*/

private ImageIcon image;

public SerBoardContainer(int type, int uid, Point location, int actualWidth, int actualHeight, Font font,

Color color, String name, String text, ImageIcon image) {

this.type = type;

this.uid = uid;

this.location = location;

this.actualWidth = actualWidth;

this.actualHeight = actualHeight;

this.font = font;

this.color = color;

this.name = name;

this.text = text;

this.image = image;

}

public int getType() {

return type;

}

public int getUid() {

return uid;

}

public Point getLocation() {

return location;

}

public int getActualWidth() {

return actualWidth;

}

public int getActualHeight() {

return actualHeight;

}

public Font getFont() {

return font;

}

public Color getColor() {

return color;

}

public String getText() {

return text;

}

public ImageIcon getImage() {

return image;

}

public String getName() {

return name;

}

}

**Текст файла SerBoardPanel.cs**

package ClientServerPart;

import java.awt.\*;

import java.io.Serializable;

import java.util.ArrayList;

/\*\*

\* Created by Lev on 10.05.14.

\*/

public class SerBoardPanel implements Serializable {

private int actualWidth;

private int actualHeight;

/\*\* Background color \*/

private Color bgColor;

/\*\* Font of the board components \*/

private Font currentFont;

/\*\* Color of the board components \*/

private Color currentColor;

/\*\* If the current state is saved \*/

private boolean saved;

/\*\* Containers \*/

private ArrayList<SerBoardContainer> containers;

public SerBoardPanel(int actualWidth, int actualHeight, Color bgColor, Font currentFont,

Color currentColor, boolean saved, ArrayList<SerBoardContainer> containers) {

this.actualWidth = actualWidth;

this.actualHeight = actualHeight;

this.bgColor = bgColor;

this.currentFont = currentFont;

this.currentColor = currentColor;

this.saved = saved;

this.containers = containers;

}

public int getActualWidth() {

return actualWidth;

}

public ArrayList<SerBoardContainer> getContainers() {

return containers;

}

public boolean isSaved() {

return saved;

}

public Color getCurrentColor() {

return currentColor;

}

public Font getCurrentFont() {

return currentFont;

}

public int getActualHeight() {

return actualHeight;

}

public Color getBgColor() {

return bgColor;

}

}

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |