**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ БЕЛОРУССКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА»**

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ УПРАВЛЕНИЯ

**ОТЧЕТ**

ПО УЧЕБНОЙ ПРАКТИКЕ

Петрова Андрея Александрович

студента 2 курса, группа 14

специальность «Прикладная информатика»

**Руководитель**:

старший преподаватель Орешко И.Г.

Минск, 2021

**ОГЛАВЛЕНИЕ**

[ЗАДАНИЕ №1. 3](#_Toc72778305)

[ЗАДАНИЕ №2. 5](#_Toc72778306)

[ЗАДАНИЕ №3. 7](#_Toc72778307)

[ЗАДАНИЕ №4. 9](#_Toc72778308)

[ЗАДАНИЕ №5. 11](#_Toc72778309)

[ЗАДАНИЕ №6. 14](#_Toc72778310)

[ЗАДАНИЕ №7. 16](#_Toc72778311)

[ЗАДАНИЕ №8. 18](#_Toc72778312)

[ЗАДАНИЕ №9. 20](#_Toc72778313)

[ЗАДАНИЕ №10. 22](#_Toc72778314)

[ЗАДАНИЕ №11. 25](#_Toc72778315)

[ЗАДАНИЕ №12. 27](#_Toc72778316)

## **ЗАДАНИЕ №1.**

**Цель работы:** Изобразить сектор круга, вращающийся в плоскости экрана вокруг своего центра по часовой стрелке. Для изображения указанной фигуры создать класс, реализующий интерфейс Shape (можно взять базовым библиотечный класс, реализующий Shape):

- выполнить указанные в задании перемещения указанной фигуры с помощью аффинного преобразования координат;

- выполнить рисунок в окне фрейма с выбранной толщиной границы фигуры, цветом границы и цветом внутренней области (вводить толщину и цвет в качестве аргументов ваших программ).

Ход работы:

Для изображения вращения отрезка используется аффинное преобразование.

Код:

public ShiftF6(double centerX, double centerY, double radius, double angle) {  
 this.centerX = centerX;  
 this.centerY = centerY;  
 this.radius = radius;  
 this.angle = angle;  
  
 transform = AffineTransform.*getRotateInstance*(Math.*toRadians*(-angle),getCenterX(),getCenterY());  
 arc2D = new Arc2D.Double(getCenterX()-getRadius(), getCenterY()- getRadius(), getRadius() \* 2, getRadius() \* 2, 60, -120, Arc2D.*PIE*);  
  
 Update();  
}

AffineTransform transform;

public void Rotate(){  
 arc2D = transform.createTransformedShape(arc2D);  
 Update();  
}

Фрейму задается цвет фона, цвет отрезка и его радиус (размер).

Код:

class MyComponent extends JPanel{  
 ShiftF6 shiftF6;  
 Color arc2DColor;  
 Color strokeColor;  
 int strokeWidth;  
  
 public MyComponent(Color arc2DColor, Color strokeColor, int strokeWidth) {  
 shiftF6 = new ShiftF6(frSize.getHeight() /2 , frSize.getWidth() / 2, radius,angle);  
 this.arc2DColor = arc2DColor;  
 this.strokeColor = strokeColor;  
 this.strokeWidth = strokeWidth;  
 }  
 public void paintComponent(Graphics g) {  
 Graphics2D g2d = (Graphics2D) g;  
  
 g2d.setColor(arc2DColor);  
 g2d.fill(shiftF6.getArc2D());  
  
 g2d.setColor(strokeColor);  
 g2d.setStroke(new BasicStroke(strokeWidth));  
  
 g2d.draw(shiftF6);  
 shiftF6.Rotate();  
 }  
}

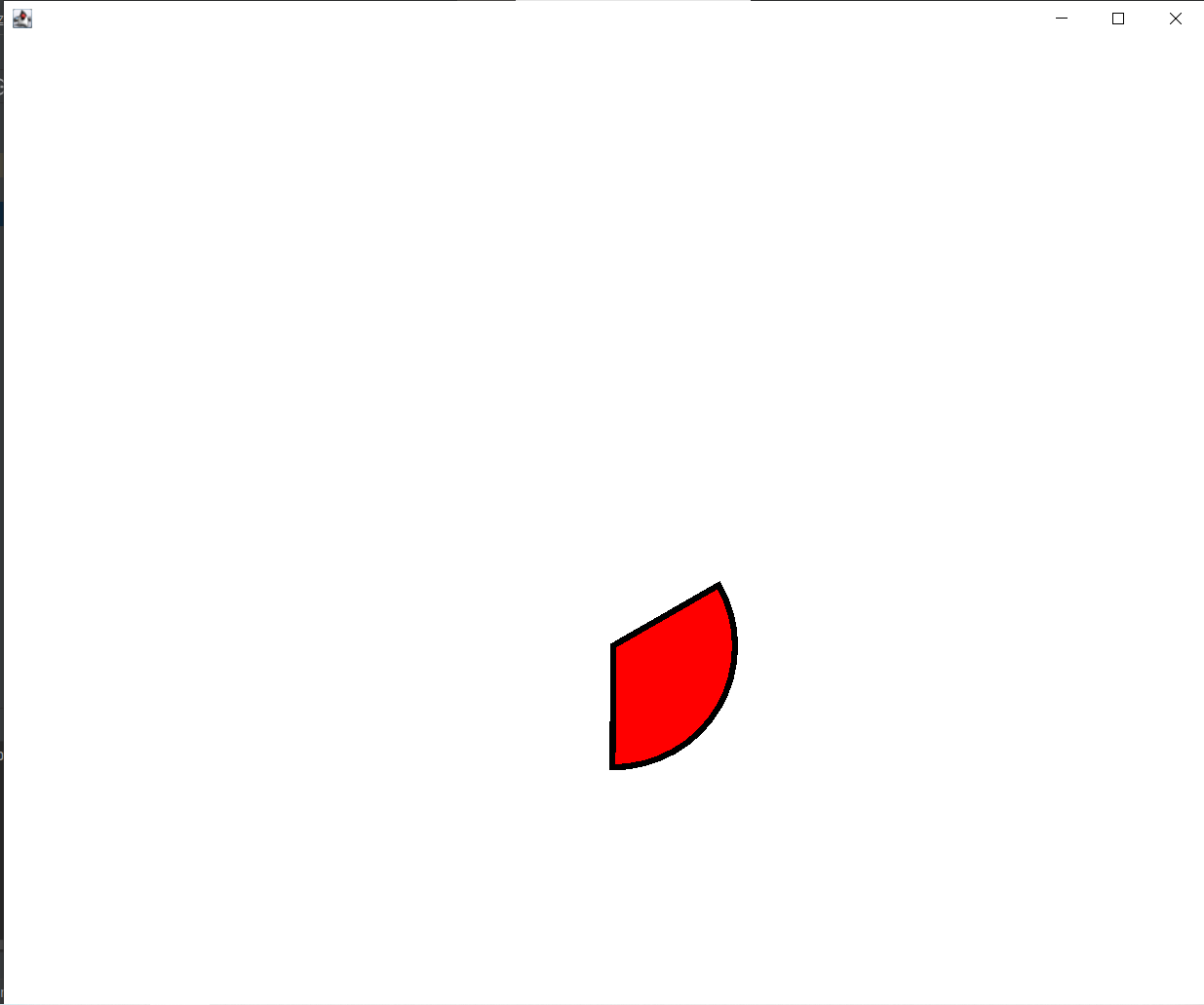


Рисунок 1. Результат выполнения задания №1

## **ЗАДАНИЕ №2.**

**Цель работы:** Изобразить фигуру (дорожный знак): надпись WAIT в прямоугольнике, цвет прямоугольника и надписи – жёлтый, цвет фона – серый с градиентной заливкой снизу-вверх. Фильтр: Rotate CW 45 degrees (поворот по часовой на 45)

Создать тестовое приложение (Frame/JFrame) для демонстрации решения, при этом:

- для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape;

- создать указанный фильтр изображения; при тестировании вывести фигуру без фильтра и с фильтром (аналогично фильтрам из примеров);

- моделировать освещение и тень от объекта при помощи альфа-канала и/или механизма обработки изображения;

- при рисовании использовать сглаживание, внеэкранный буфер и преобразования координат.

**Ход работы:**

Рисуем основную фигуру.

Код:

public Sign(int xP,int yP,int sid) {  
 this.xPos=xP;  
 this.yPos=yP;  
 this.side=sid;  
 this.rect = new Rectangle2D.Double(xPos,yPos,side\*1.5,side);  
}  
  
@Override  
public void paint(Graphics graphics) {  
 Graphics2D graphics2D = (Graphics2D) graphics;  
   
 graphics2D.setColor(Color.*BLACK*);  
 graphics.fillPolygon( new Polygon(  
 new int[]{  
 (int) (xPos+side\*1.5),  
 (int) ((xPos+side\*1.5)+side/2),  
 (int) (xPos+side\*1.5)  
 },  
 new int[] {  
 (int) (yPos + side),  
 (int) ((yPos + side + yPos)/2),  
 (int) ((yPos + side + yPos) / 2 )  
 },  
 3));  
  
 graphics2D.setColor(Color.*WHITE*);  
 graphics2D.fillRect((int)rect.getX(),(int)rect.getY(),(int)rect.getWidth(),(int)rect.getHeight());  
  
 graphics2D.setColor(Color.*YELLOW*);  
 graphics2D.setStroke(new BasicStroke(strokeWidth));  
  
 Font font = new Font("Serif", Font.*PLAIN*, textSize);  
 graphics2D.setFont(font);  
 graphics2D.drawString("WAIT", xPos+side/2-textSize/3, yPos+side/2+textSize/3);  
 graphics2D.draw(this.rect);  
}

Применяем Rotate CW 45 degrees.

Код:

AffineTransform at = AffineTransform.*getRotateInstance*(Math.*PI*/4,700,100);  
gr2.transform(at);

Результат работы программы (Рис 1.)

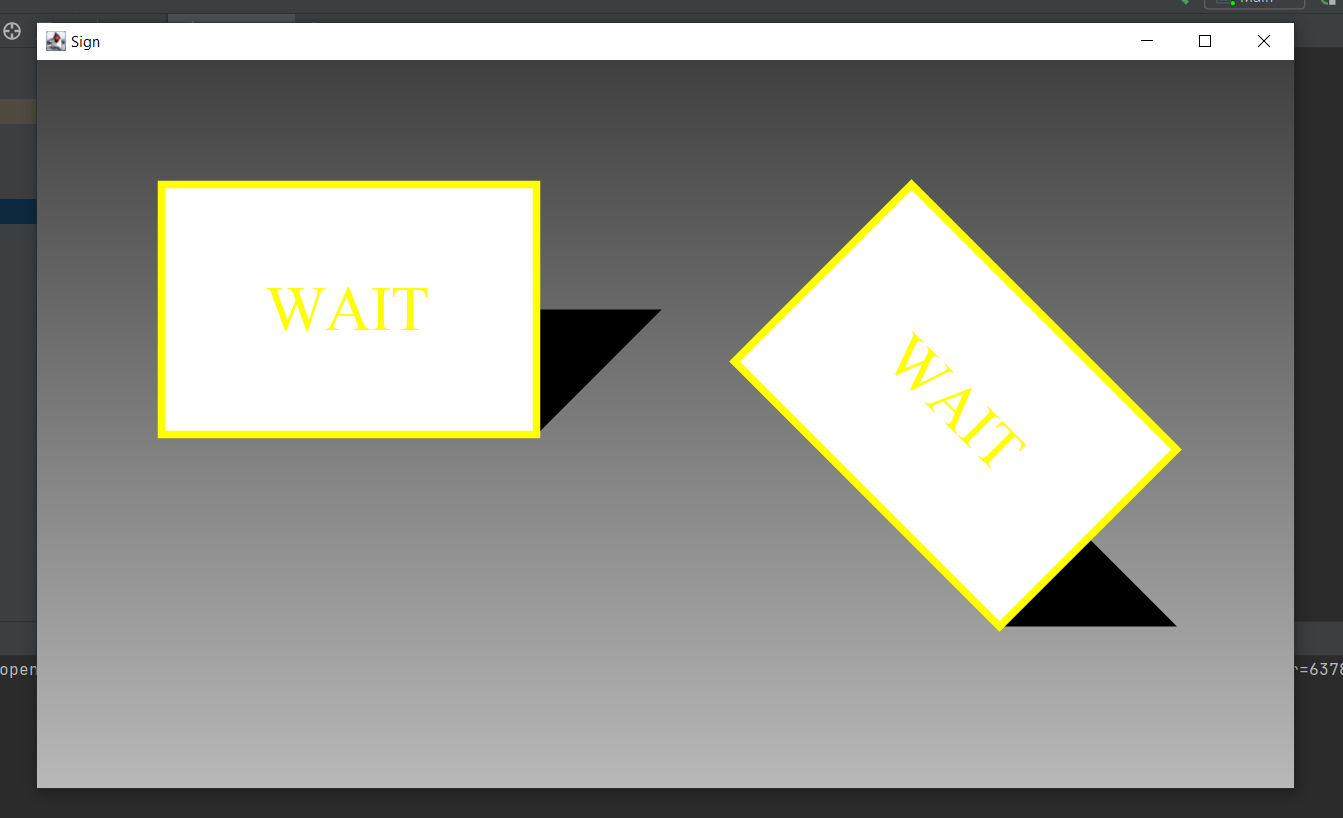
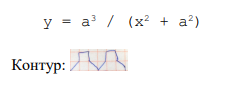


Рисунок 2. Результат выполнения задания №2

## **ЗАДАНИЕ №3.**

**Цель работы:** Разработайте пользовательский класс Shape реализующий рисование указанной алгебраической линии. Разработайте пользовательский класс Stroke для отображения указанного контура, используя в качестве исходных точек результаты класса Shape, созданного на шаге 1). Создайте приложение (Frame/JFrame) для тестирования и демонстрации разработанных классов.

Линия Версьера:



**Ход работы:**

Был разработан класс Witch\_of\_Agnesi, который реализует Shape. Отрисовка линии происходит в методе currentSegment()

Код:

public int currentSegment(float[] xy) {  
 if (start) {  
 xy[0] = (float) (a \* Math.*tan*(t))+centerX;  
 xy[1] = (float) (a \* Math.*pow*(Math.*cos*(t), 2))+centerY;  
  
 start = false;  
 if (aff != null)  
 aff.transform(xy, 0, xy, 0, 1);  
  
 return *SEG\_MOVETO*;  
  
 }  
 if (t >= Math.*PI*/2-h) {  
 done = true;  
 return *SEG\_CLOSE*;  
 }  
  
 xy[0] = (float) (a \* Math.*tan*(t))+centerX;  
 xy[1] = (float) (a \* Math.*pow*(Math.*cos*(t), 2))+centerY;  
 return *SEG\_LINETO*;  
}

где:

xy[0] = (float) (a \* Math.*tan*(t))+centerX;  
xy[1] = (float) (a \* Math.*pow*(Math.*cos*(t), 2))+centerY;

– уравнение Версьера в параметрическом виде.

Контур реализует класс HouseStroke с помощью метода createStrokedShape().

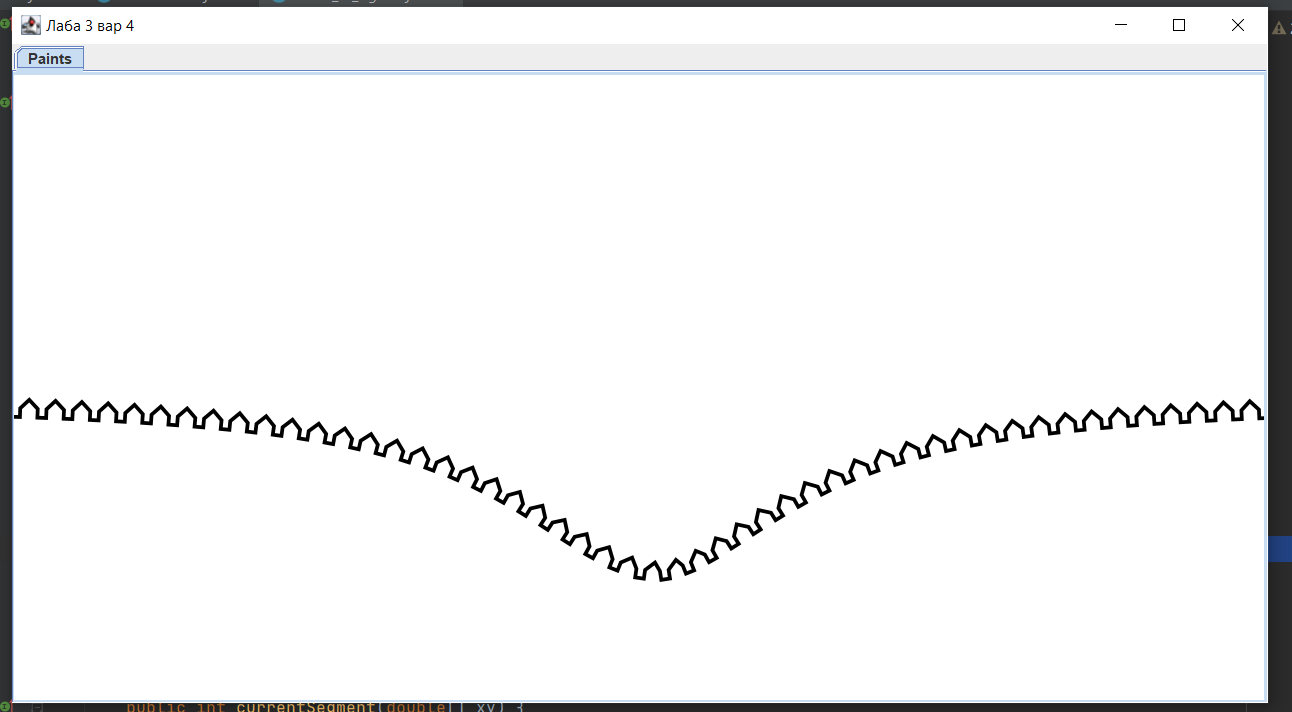


Рисунок 3. Результат выполнения задания №3

## **ЗАДАНИЕ №4.**

**Цель работы:** Модифицируйте вашу программу следующим образом. В демонстрационное приложение добавьте возможность печати небольшого отчёта о решении задания No3. Отчёт должен содержать следующее: - рисунок с подписью (по стандарту!) алгебраической линии вашего задания - исходный текст класса Shape, реализующий рисование указанной алгебраической линии; для длинных строк, выходящих за границы области печати, организуйте перенос текста на новую строку с разрывом по пробельным символам.

При печати используйте режим альбомной ориентации страницы и двустороннюю печать. Рисунок должен занимать не более половины страницы, при печати выровнять его по горизонтали.

**Ход работы:**

Был разработан класс PrintableExample, который реализует интерфейс Printable с методом

Код:

public int print(Graphics g, PageFormat pf, int pageIndex) {  
 if (pageIndex != 0) return *NO\_SUCH\_PAGE*;  
  
 Graphics2D g2 = (Graphics2D)g;  
 g2.translate(pf.getImageableX(), pf.getImageableY());  
  
 double pageWidth = pf.getImageableWidth();  
 double pageHeight = pf.getImageableHeight();  
 double exampleWidth = example.getWidth();  
 double exampleHeight = example.getHeight();  
  
 // Scale the example if needed  
 double scalex = 1.0, scaley = 1.0;  
 if (exampleWidth > pageWidth) scalex = pageWidth/exampleWidth;  
 if (exampleHeight > pageHeight) scaley = pageHeight/exampleHeight;  
 double scalefactor = Math.*min*(scalex, scaley);  
 if (scalefactor != 1) g2.scale(scalefactor, scalefactor);  
  
 example.draw(g2, GraphSampleFrame.this);  
  
 return *PAGE\_EXISTS*;  
}

для печати объекта.

Также для смены ориентации реализован метод Print класса GraphSampleFrame.

public void print(final GraphSample example) {   
 PrinterJob job = PrinterJob.*getPrinterJob*();  
 PrintRequestAttributeSet pf = new HashPrintRequestAttributeSet();  
 pf.add(OrientationRequested.*LANDSCAPE*); // Меняем формат на АЛЬБОМНЫЙ  
 job.setPrintable(new PrintableExample(example));  
  
 if (job.printDialog(pf)) {  
 try {  
 job.print(pf);  
 }  
 catch(PrinterException e) {  
 System.*out*.println("Couldn't print: " + e.getMessage());  
 }  
 }  
}

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 4. Результат выполнения задания №4

## **ЗАДАНИЕ №5.**

**Цель работы:** Разработайте систему классов/интерфейсов для предметной области Вашего варианта задания. Данные необходимо упорядочить по атрибутам/свойствам товаров, предметов и т.п. в виде дерева. Разработайте графическое приложение для ввода/отображения данных Вашего варианта задания. При отображении структуры данных в виде дерева реализуйте интерфейс javax.swing.Tree.TreeModel. Листья дерева отображайте в виде таблицы, для этого реализуйте интерфейс javax.swing.table.TableModel. (пример похожего приложения – Проводник Windows)

**Ход работы:**

Был разработан класс CookNode, реализующий сборник кулинарных рецептов.

Код:

class CookNode {  
 String name, worldCuisine, type, level;  
  
 CookNode(String worldCuisine, String type, String level, String name) {  
 this.worldCuisine = worldCuisine;  
 this.type = type;  
 this.level = level;  
 this.name = name;  
 }  
  
 public String getName() { return name; }  
  
 public String getWorldCuisine() {  
 return worldCuisine;  
 }  
  
 public String getLevel() {  
 return level;  
 }  
  
 public String getType() {  
 return type;  
 }  
}

Для графического взаимодействия со сборником кулинарных рецептов были разработаны классы, реализующие интерфейс javax.swing.table.TableModel

Код:

class myTableModel implements TableModel {

static final String[] *columnNames* = new String[]{"Название",   
 "Сложность", "Тип", "Категория кухни"};

static final Class[] *columnTypes* = new Class[]{String.class,

String.class, String.class, String.class, String.class, Integer.class,

Integer.class};

private Set<TableModelListener> listeners = new

HashSet<TableModelListener>();  
 private ArrayList<CookNode> infoNodes;  
 public myTableModel() {  
 infoNodes = new ArrayList<CookNode>(); }  
  
 public myTableModel(ArrayList<CookNode> al){this.infoNodes = al;}  
  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 CookNode nb = infoNodes.get(rowIndex);  
 switch (columnIndex) {  
 case 0:  
 return nb.getName();  
 case 1:  
 return nb.getWorldCuisine();  
 case 2:  
 return nb.getType();  
 case 3:  
 return nb.getLevel();  
 }  
 return "";  
 }  
  
 public int getColumnCount() { return *columnNames*.length; }  
 public int getRowCount() { return infoNodes.size(); }  
 public Class getColumnClass(int columnIndex) {  
 return *columnTypes*[columnIndex];  
 }  
  
 public String getColumnName(int columnIndex) {  
 return *columnNames*[columnIndex];  
 }  
 public void setInfoArray(ArrayList<CookNode> al) { infoNodes = al; }  
 @Override  
 public void addTableModelListener(TableModelListener listener) {  
 listeners.add(listener); }  
 @Override  
 public void removeTableModelListener(TableModelListener listener) {  
 listeners.remove(listener);}  
 @Override  
 public boolean isCellEditable(int rowIndex,int columnIndex){return

false;}  
 @Override  
 public void setValueAt(Object value, int rowIndex, int columnIndex){}  
}

и интерфейс javax.swing.Tree.TreeModel

Код:

tableModel = new myTableModel();  
infoPanel = new JTable(tableModel);  
treeModel = new myTreeModel(new treeNode("Книга"));  
cookTree = new JTree(treeModel);  
  
cookTree.addTreeSelectionListener(new TreeSelectionListener() {  
 @Override  
 public void valueChanged(TreeSelectionEvent e) {  
 treeNode node = (treeNode) cookTree.getLastSelectedPathComponent();  
 if (node == null) {  
 return;  
 }  
 ArrayList<CookNode> array = node.getAllNodes();  
 tableModel = new myTableModel(array);  
 infoPanel.setModel(tableModel);  
 }  
});  
JSplitPane splitPane = new JSplitPane(JSplitPane.*HORIZONTAL\_SPLIT*, true, new JScrollPane(cookTree), new JScrollPane(infoPanel));  
splitPane.setDividerLocation(200);

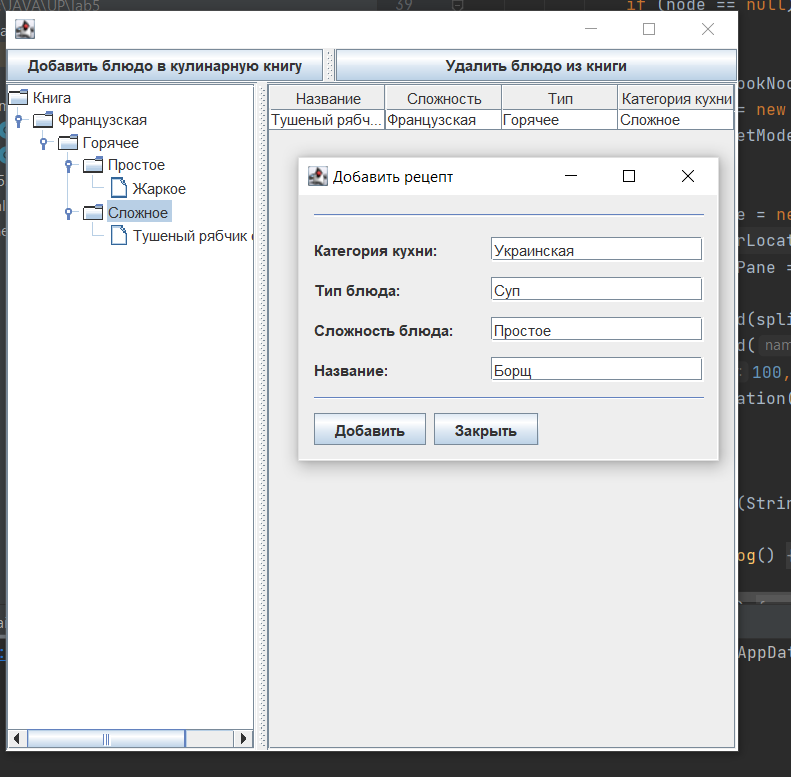


Рисунок 5. Результат выполнения задания №5

## **ЗАДАНИЕ №6.**

**Цель работы:** для выполнения задания используется ваш вариант решения задания №3. Модифицируйте вашу программу следующим образом.

Создайте тестовое приложение, добавьте в ваш класс рисования алгебраической линии возможность «перетаскивание» (drag-and-drop). Реализуйте необходимые интерфейсы в классе и в приложении для демонстрации «перетаскивания» алгебраической линии между несколькими копиями тестового приложения. При реализации интерфейса тестового приложения следуйте рекомендациям стандарта CUI (Common User Interface).

**Ход работы:**

Было создано 2 одинаковых jframe для работы с классом MyDragAndDrop. Класс в свою очередь в зависимости от выбранного режима вызывает методы рисования или перемещения (клонирования, удаления, рисования) моей кривой. Пример метода рисования по нажатию мышки.

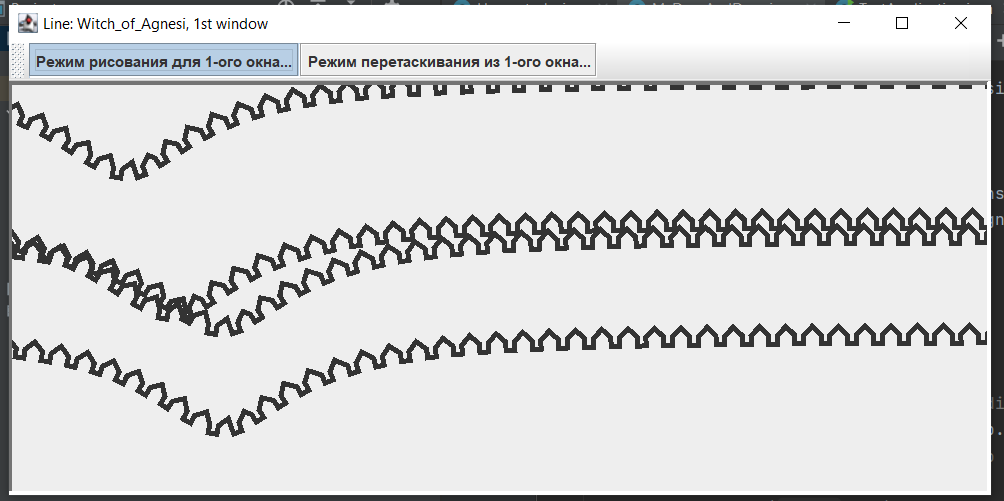
Код:

public void mousePressed(MouseEvent e)  
{  
 if (dragMode)  
 return;  
 currentScribble = new Witch\_of\_Agnesi(e.getX() - 150, e.getY() - 200, 300, 300, 75);  
 leafs.add(currentScribble);  
  
 repaint();  
}

В класс кривой добавлена реализация Cloneable, Transferable, Serializable. Пример переопределения метода clone.

Код:

public Object clone() {  
 try {  
 return super.clone();  
 } catch (CloneNotSupportedException e) { // This should never happen  
 return this;  
 }  
}



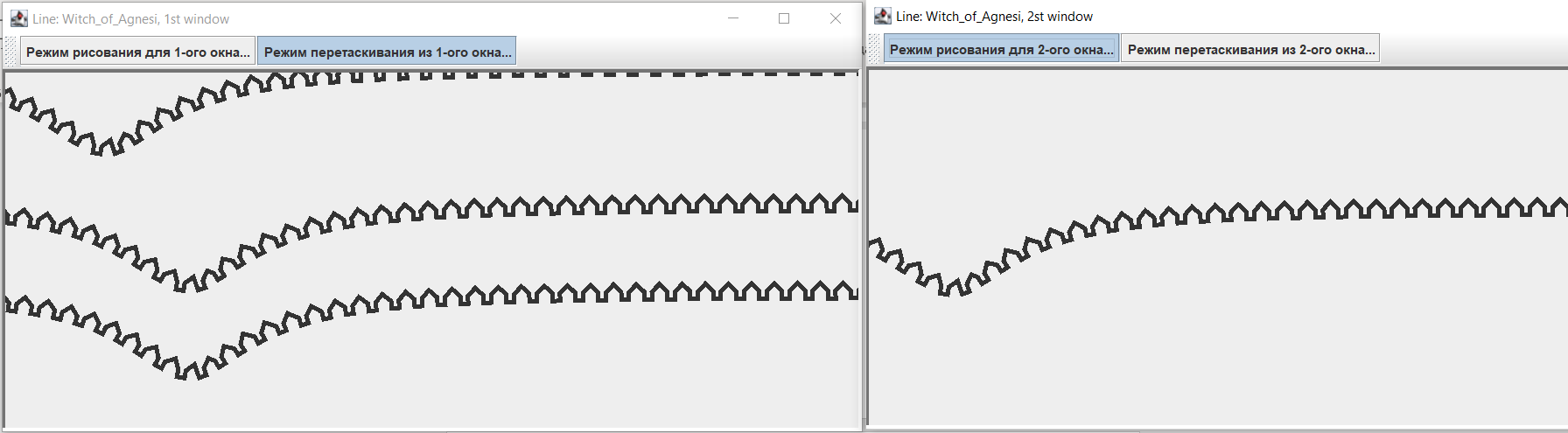


Рисунок 6. Результат выполнения задания №6

## **ЗАДАНИЕ №7.**

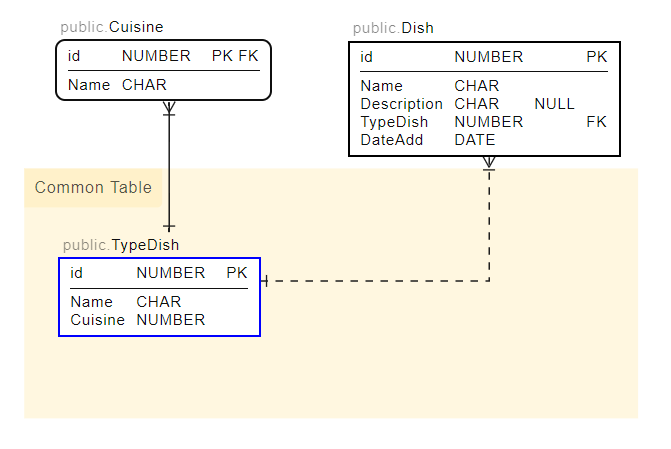
**Цель работы:** Исследовать предложенную предметную область, спроектировать структуру базы данных объектов выбранной предметной области (из не менее чем 2-х таблиц объектов). Согласуйте проект БД с преподавателем. Обязательно работаем с Derby — решение в другой СУБД не принимается!

Разработайте графическое приложение для создания/ввода/отображения БД Вашего варианта задания. Содержимое БД отображайте в виде таблиц.

При реализации интерфейса следуйте рекомендациям стандарта CUI (Common User Interface).

**Ход работы:**

Был разработана архитектура базы данных, состоящая из 3 таблиц Cuisine, TypeDish, Dish.



Для графического взаимодействия со сборником кулинарных рецептов были разработаны классы, реализующие интерфейс javafx.scene.control.TableView и javafx.scene.control.TreeView

Код:

public static void createCommonTable(ResultSet rec) {  
 *tableView*.getColumns().clear();  
 ObservableList<ObservableList> data = FXCollections.*observableArrayList*();  
 try {  
 for(int i=0 ; i<rec.getMetaData().getColumnCount(); i++){  
 final int j = i;  
 var col = new TableColumn(rec.getMetaData().getColumnName(i+1));  
 col.setCellValueFactory(  
 new Callback<TableColumn.CellDataFeatures<ObservableList,String>,  
 ObservableValue<String>>(){  
 public ObservableValue<String> call(TableColumn.CellDataFeatures<ObservableList, String> param) {  
 return new SimpleStringProperty(param.getValue().get(j).toString());  
 }  
 });  
 *tableView*.getColumns().addAll(col);  
 }  
  
 while(rec.next()){  
 ObservableList<String> row = FXCollections.*observableArrayList*();  
 for(int i = 1 ; i <= rec.getMetaData().getColumnCount(); i++){  
 row.add(rec.getString(i));  
 }  
 data.add(row);  
 }  
 *tableView*.setItems(data);  
 *cellContextMenu*("");  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
}

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 7. Результат выполнения задания №7

## **ЗАДАНИЕ №8.**

**Цель работы:** Изучите материал примера по быстрому введению в среду разработки NetBeans и компоненты JavaBeans по адресу: http://docs.oracle.com/javase/tutorial/javabeans/quick/index.html 2) Разработайте простой компонент вашего варианта задания на базе класса Canvas. Создайте файл манифеста и упакуйте компонент вместе с исходным кодом разработанных классов. При разработке поместите все ваши классы в пакет: bsu.fpmi.educational\_practice 3) Создайте тестовое приложение в NetBeans с использованием вашего компонента.

Компонент: круглая заливка. Свойства: диаметр и цвет.

**Ход работы:**

Компонент Java Beans

public class MyCanvas extends Canvas {

private int diam;

private Color color;

public void setDiametr(int d) {

this.diam = d;}

public void setColor(Color c) {

this.color = c;}

public MyCanvas() {

this(Color.BLUE, 250);

this.setSize(new Dimension(250, 250)); }

public MyCanvas(Color c, int d) {

this.color = c;

this.diam = d; }

@Override

public void paint(Graphics g) {

Graphics2D g2d = (Graphics2D)g;

Shape circle = new Ellipse2D.Float(0, 0 ,diam, diam );

g2d.draw(circle);

g2d.setColor(color);

g2d.fill(circle);

}

}

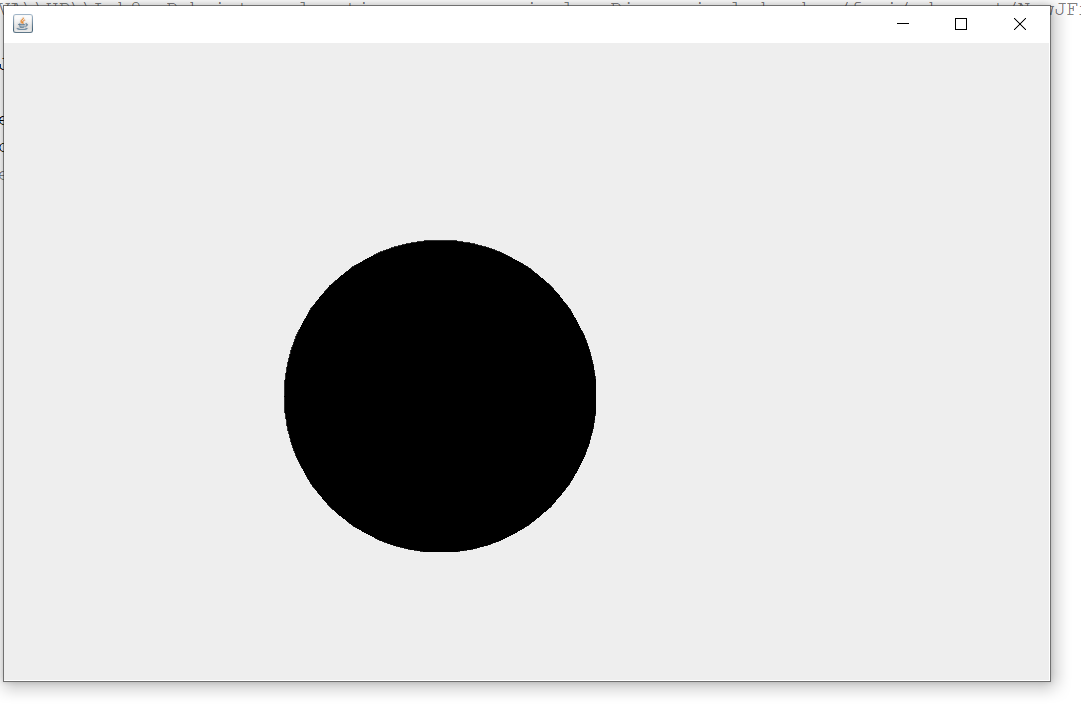


Рисунок 8. Результат выполнения задания №8

## **ЗАДАНИЕ №9.**

**Цель работы:** Разработайте компонент вашего варианта задания. Создайте файл манифеста и упакуйте компонент вместе с исходным кодом разработанных классов. При разработке поместите все ваши классы в пакет: bsu.fpmi.edupract. Компонент должен реализовать класс BeanInfo с информацией о компоненте. Создайте тестовое приложение в NetBeans с использованием вашего компонента.

Однострочный статический текст, список и обычная кнопка. Свойства: текст, текст кнопки, массив строк для заполнения списка. Событие генерируется при нажатии на обычную кнопку. Событие передаёт ещё и индекс выбранного элемента списка.

**Ход работы:**

Компонент Java Beans. Манифест:

Name: myNewPanel.class

Java-Bean: true

Обработка событий:

public class myNewPanel extends javax.swing.JPanel {  
 private javax.swing.JButton jButton;  
 private javax.swing.JComboBox<String> jComboBox;  
 private javax.swing.JLabel jLabel;  
 private char commandKey = 'z';  
 ArrayList<String> comboBoxes=new ArrayList<String>();  
   
 public myNewPanel() {  
 initComponents();  
 }  
   
 @SuppressWarnings("unchecked")  
 private void initComponents() {...}  
  
 private void jButtonActionPerformed(java.awt.event.ActionEvent evt){ }  
  
 private void jComboBoxActionPerformed(java.awt.event.ActionEvent evt) { }  
  
 private void jButtonMousePressed(java.awt.event.MouseEvent evt) {  
 showInfo("Button");  
 }  
  
 private void jButtonKeyPressed(java.awt.event.KeyEvent evt) {  
 keyPressed(evt.getKeyChar());  
 }  
  
 private void jComboBoxKeyPressed(java.awt.event.KeyEvent evt) {  
 keyPressed(evt.getKeyChar());  
 }  
  
 private void showInfo(String pressed) {  
 JOptionPane.showMessageDialog(this, "Selected items index = " +   
 jComboBox.getSelectedIndex() + "\n" + pressed + " pressed");  
 jButton.setVisible(true);  
 }  
  
 private void keyPressed(char ev) {  
 if (ev == commandKey) {  
 showInfo("Command key");  
 }  
 }  
  
 public void setComboBoxText(String[] lines) {  
 jComboBox.removeAllItems();  
 for (String line :lines) {  
 jComboBox.addItem(line);  
 }  
 }  
  
 public void setjLabel(String labels) {  
 jLabel.setText(labels);  
 }  
  
 public String getjLabel() {  
 return jLabel.getText();  
 }  
  
  
 public void setjButton(String buttons) {  
 jButton.setText(buttons);  
 }  
  
 public String getjButton() {  
 return jButton.getText();  
 }  
}

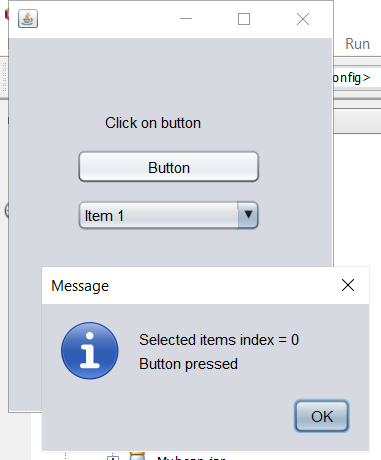


Рисунок 9. Результат выполнения задания №9

## **ЗАДАНИЕ №10.**

**Цель работы:** Создаём собственный редактор для каждого свойства компонента. Каждый редактор ограничивает возможные значения свойства, предоставляя выбор из списка трёх – пяти допустимых значений (т.е. определяем методы getTags( ))

Регистрируем редакторы в классе BeanInfo компонента.

Дополнительно:

Попытайтесь создать настройщик компонента, который позволит изменять списки допустимых значений для свойств вашего компонента.

**Ход работы:**

Компонент Java Beans. Манифест:

Name: TextWithButtonPanelBean.class

Java-Bean: true

Код основного компонента:

public class TextWithButtonPanel extends Panel{  
 protected String messageText; // The message to display  
 protected Alignment alignment; // The alignment of the message  
 protected String simpleButtonLabel; // Text for the simpleButton button  
 protected String[] listText;  
  
 protected SingleLineLabel message;  
 protected Button simpleButton;  
 protected javax.swing.JList<String> list;  
public TextWithButtonPanel() {  
 this("Your Message Here");  
 }  
  
 public TextWithButtonPanel(String messageText) {  
 this(messageText, new String[] {"item1", "item2", "item3", "item4"});  
 }  
  
 public TextWithButtonPanel(String messageText, String[] listText) {  
 this(messageText, listText, "OK");  
 }  
  
 */\*\* A constructor for programmers using this class "by hand" \*/* public TextWithButtonPanel(String messageText, String[] listText, String simpleButtonLabel)  
 {  
 setLayout(new BorderLayout(15, 15));  
 message = new SingleLineLabel(messageText);  
 add(message, BorderLayout.NORTH);  
 Panel buttonbox = new Panel();  
 buttonbox.setLayout(new FlowLayout(FlowLayout.CENTER, 25, 15));  
 add(buttonbox, BorderLayout.SOUTH);  
 simpleButton = new Button(); // Create buttons  
 buttonbox.add(simpleButton);  
 simpleButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 fireEvent(new AcceptEvent(TextWithButtonPanel.this));  
 }  
 });  
  
 list = new javax.swing.JList<>();  
 add(list, BorderLayout.CENTER);  
 setMessageText(messageText);  
 setSimpleButtonLabel(simpleButtonLabel);  
 setListText(listText);  
 }  
 public String getMessageText() { return messageText; }  
 public String getSimpleButtonLabel() { return simpleButtonLabel; }  
 public Alignment getAlignment() { return alignment; }  
 public String[] getListText() { return listText; }  
  
 public void setMessageText(String messageText) {  
 this.messageText = messageText;  
 message.setLabel(messageText);  
 validate();  
 }  
  
 public void setAlignment(Alignment alignment) {  
 this.alignment = alignment;  
 message.setAlignment(alignment);  
 }  
  
 public void setListText(String[] listText) {  
 this.listText = listText;  
 list.setListData(listText);  
 validate();  
 }  
  
 public void setSimpleButtonLabel(String l) {  
 simpleButtonLabel = l;  
 simpleButton.setLabel(l);  
 simpleButton.setVisible((l != null) && (l.length() > 0));  
 validate();  
 }  
  
 public void setFont(Font f) {  
 super.setFont(f); // Invoke the superclass method  
 message.setFont(f);  
 simpleButton.setFont(f);  
 list.setFont(f);  
 validate();  
 }protected Vector<AcceptListener> listeners = new Vector<>();public void addAcceptListener(AcceptListener l) {  
 listeners.addElement(l);  
 }public void removeAcceptListener(AcceptListener l) {  
 listeners.removeElement(l);  
 }public void fireEvent(AcceptEvent e) {  
 // Make a copy of the list and fire the events using that copy.  
 // This means that listeners can be added or removed from the original  
 // list in response to this event. We ought to be able to just use an  
 // enumeration for the vector, but that doesn't actually copy the list.  
 Vector list = (Vector) listeners.clone();  
 for(int i = 0; i < list.size(); i++) {  
 AcceptListener listener = (AcceptListener)list.elementAt(i);  
 AcceptEvent: listener.simpleButton(e); break;  
 }  
 }public static void main(String[] args) throws IOException {  
 // Create an instance of InfoPanel, with title and message specified:  
 TextWithButtonPanel p = new TextWithButtonPanel("Do you really want to quit?");  
  
 p.addAcceptListener(new AcceptListener() {  
 public void simpleButton(AcceptEvent e) {  
 System.out.printf("Selected row index: %d",

p.list.getLeadSelectionIndex());  
 JOptionPane.showMessageDialog(p, "Selected items index = " +   
 p.list.getLeadSelectionIndex());  
 p.setMessageText("Selected row index: " +

p.list.getLeadSelectionIndex());  
 System.exit(0);  
 }  
 });  
  
 Frame f = new Frame();  
 f.add(p);  
 f.pack();  
 f.setVisible(true);  
 }  
}

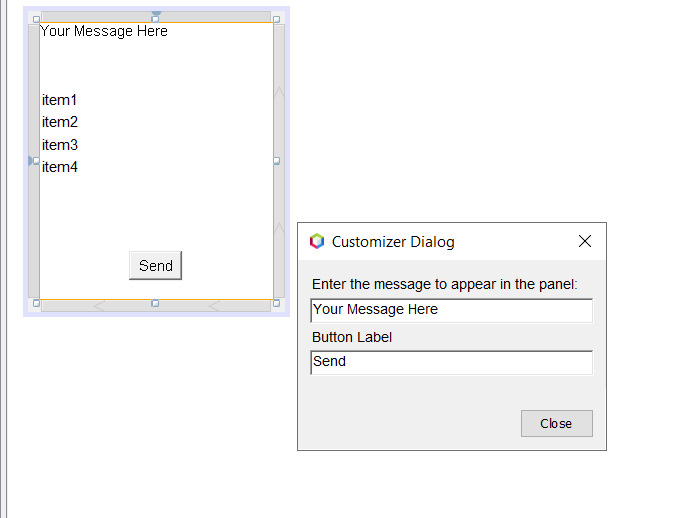
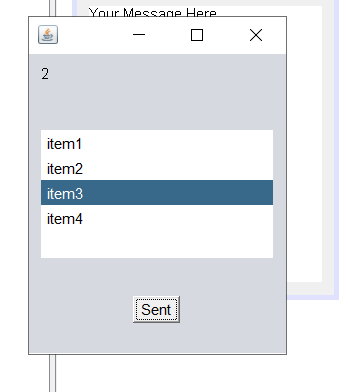


Рисунок 10. Результат выполнения задания №10

## **ЗАДАНИЕ №11.**

**Цель работы:** Создать сервлет и взаимодействующие с ним пакеты Java-классов и HTML-документов, выполняющие действия для решения вашего варианта задания. Представить решение в виде web-приложения (как в примере).

Реализовать игру с сервером в 21 очко.

**Ход работы:**

Код сервлета:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Рисунки 11, 12. Результат выполнения задания №11

## **ЗАДАНИЕ №12.**

**Цель работы:** Проанализируйте ваш вариант задания. Можно ли его реализовать как часть MUD системы (например, в одной из комнат MudPlace), требуется ли для этого внести изменения в парадигму MUD? Какие изменения потребует реализация клиента MUD, другие классы примера? Оформите эти размышления в вашем отчёте в качестве анализа предметной области. При реализации, по возможности, используйте парадигму MUD и классы примера 2 при реализации вашего варианта задания.

Создайте на основе технологии RMI клиент/серверное приложение:

Игра по сети в “Морской бой”. Игра между двумя клиентами через сервер. Клиент предлагает поиграть и ждёт, пока другой клиент согласится на игру. Сервер организует связь между играющими клиентами.

**Ход работы:**

На стороне сервера реализован класс *Worker,* который выполняет всю логику приложения. В частности, он умеет отслеживаться количество подключённых игроков, отслеживать игровой процесс, выводить его логи и отправлять данные, необходимые для игры клиенту.

Непосредственное создание сервера:

Код:

BattleshipInterface obj = new Worker();  
Registry rgsty = LocateRegistry.*createRegistry*(5000);  
rgsty.rebind("BattleshipImplementationobj", obj);

Клиент может запросить у сервера все новости, либо новости за указанный период. Подключение клиента:

Код:

try{  
 obj = (BattleshipInterface)Naming.*lookup*("//localhost:5000/BattleshipImplementationobj");  
}  
catch(Exception e){  
 System.*out*.println("Exception in lookup"+e.getMessage());  
 e.printStackTrace();  
}

Сначала запускается Server, лишь после этого можно запустить Client(ограничение стоит на 2 человека, ибо игроков может быть только 2) для трансляции игрового процесса, а также запросов клиента.

Также в этом задании предлагалось провести рассуждение, подходит ли концепция MUD для реализации данного приложения.

Если говорить коротко, парадигма MUD не подходит для реализации данного задания. К этому есть следующие причины:

MUD само по себе расшифровывается как Multi user domain. Ключевое здесь то, что MUD ориентировано на наличие большего числа взаимодействующих между собой пользователей. В данном задании взаимодействие осуществляется только между сервером и клиентом.

Парадигма MUD хоть и представляет большую гибкость при создании виртуального мира, однако эта гибкость несёт с собой дополнительные расходы. Кажется неразумным поддерживать интерфейс класс MudPlace, если пользоваться будут только методами получения новостей

Единственный вариант внедрения приложения в парадигму MUD – создание отдельной комнат, где можно будет осуществлять несколько игр между разными игроками одновременно. Однако для этого необходимо немного менять интерфейс MUD-парадигмы, а именно теперь описание комнаты должно быть динамически изменяемым (иметь разные id комнаты).

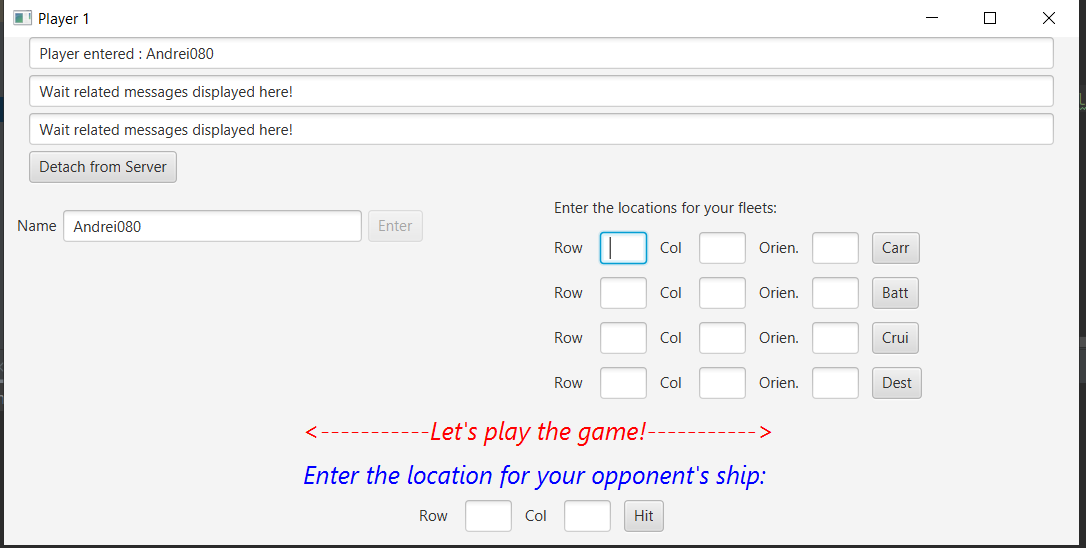


Рисунок 13. Результат выполнения задания №12