Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий  и анализа данных |

наименование института

**Отчёт**

к лабораторной работе №5

по дисциплине «ООП»:

|  |
| --- |
| «Разработка приложения с графическим пользовательским интерфейсом» |

наименование темы

Вариант №9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент | ИСТб 19-2 |  |  |  | Козлов М.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил | | |  |  | Маланова Т.В |
|  | | | подпись |  | Фамилия ИО |
| Работа защищена с оценкой | | |  | | |

Иркутск 2021

Содержание

[1 Постановка задачи 3](#_Toc67787635)

[2 Проектирование графического интерфейса 4](#_Toc67787636)

[3 Проектирование классов 5](#_Toc67787637)

[4 Проектирование тестов 6](#_Toc67787638)

[5 Листинг программы на Java 15](#_Toc67787639)

[5.1 Главный класс Main 15](#_Toc67787640)

[5.2 Класс Controller 16](#_Toc67787641)

[5.3 Класс View 22](#_Toc67787642)

[5.4 Класс Product 29](#_Toc67787643)

[5.5 Класс Lists 31](#_Toc67787644)

# 1 Постановка задачи

**Цель работы:** Применение объектно-ориентированного подхода в разработке приложений с графическим интерфейсом.

**Задачи:**

* Получение общего представления о шаблоне программирования Модель/Вид/Контроллер и его практического применения;
* Получение практического опыта работы с базовыми графическими библиотеками java.

**Содержание задания**

Для выполнения задания необходимо разработать графический пользовательский интерфейс средствами java для работы с классами реализованными при выполнении либо 3-ей, либо 4-ой лабораторной работы.

Графический пользовательский интерфейс должен предоставлять возможности работы со всеми возможными функциями программы (ввод и вывод данных в соответствующей форме, изменение условий выполнения задания, работа с файлами и т.д.). Состав используемых компонентов определяется самостоятельно, но для них должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах.

Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий. Для этого может понадобиться модифицировать программный код из предыдущих лабораторных работ.

# 2 Проектирование графического интерфейса

Описание

Окно добавления продуктов, представляющая собой форму для внесения продукта в список. Форма состоит из полей “Название”, “Состав продукта”, “Форма для хранения”, “Цена продукта”. При нажатии на кнопку “Хлеб”, произойдёт изменение формы: станут доступны поля для записи “Название”, “Состав продукта”, “Форма”, “Цена продукта”.

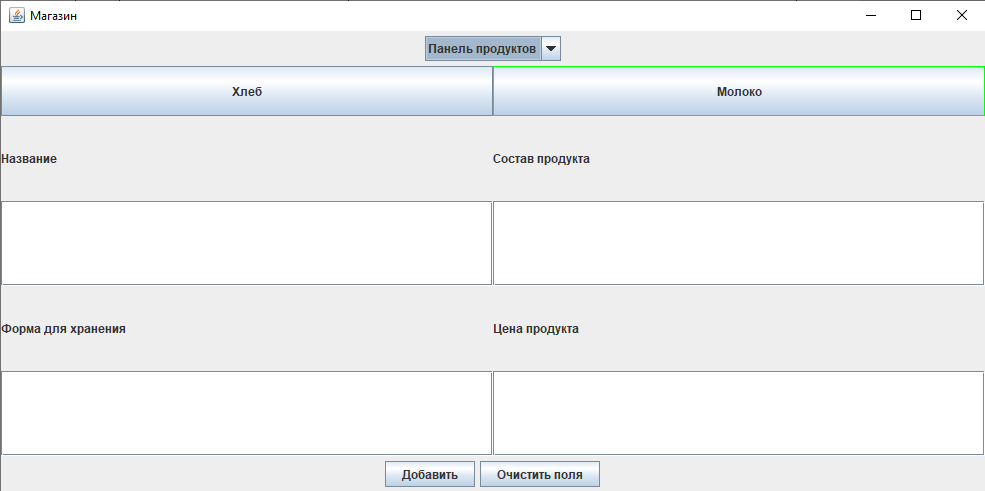


Рисунок 1 – Окно добавления продуктов

Описание

Окно отображения продуктов представляет собой таблицу, отображаемую при нажатии кнопки “Хлеб” или “Молоко”. При нажатии на кнопку “Хлеб” отобразится таблица хлебобулочных изделий, а при нажатии на кнопку “Молоко” отобразится панель молочных продуктов.

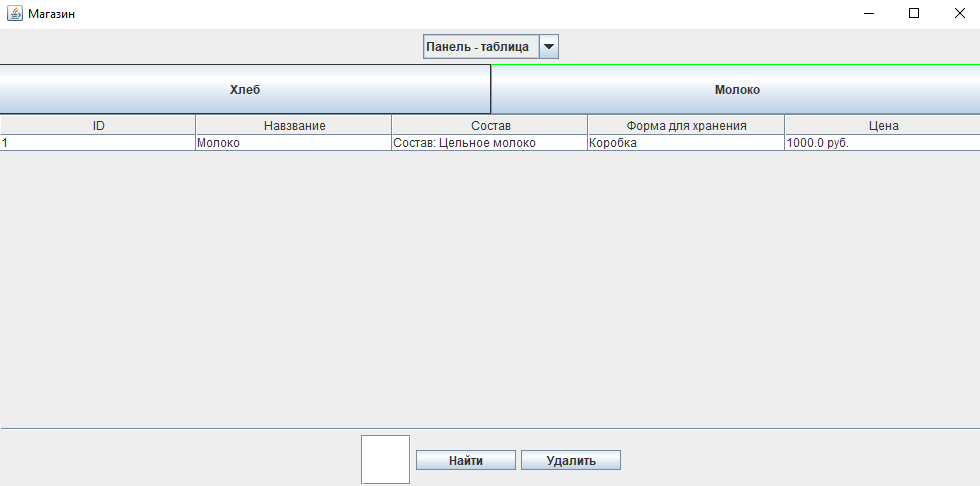


Рисунок 2 – Окно отображения продуктов

# 3 Проектирование классов

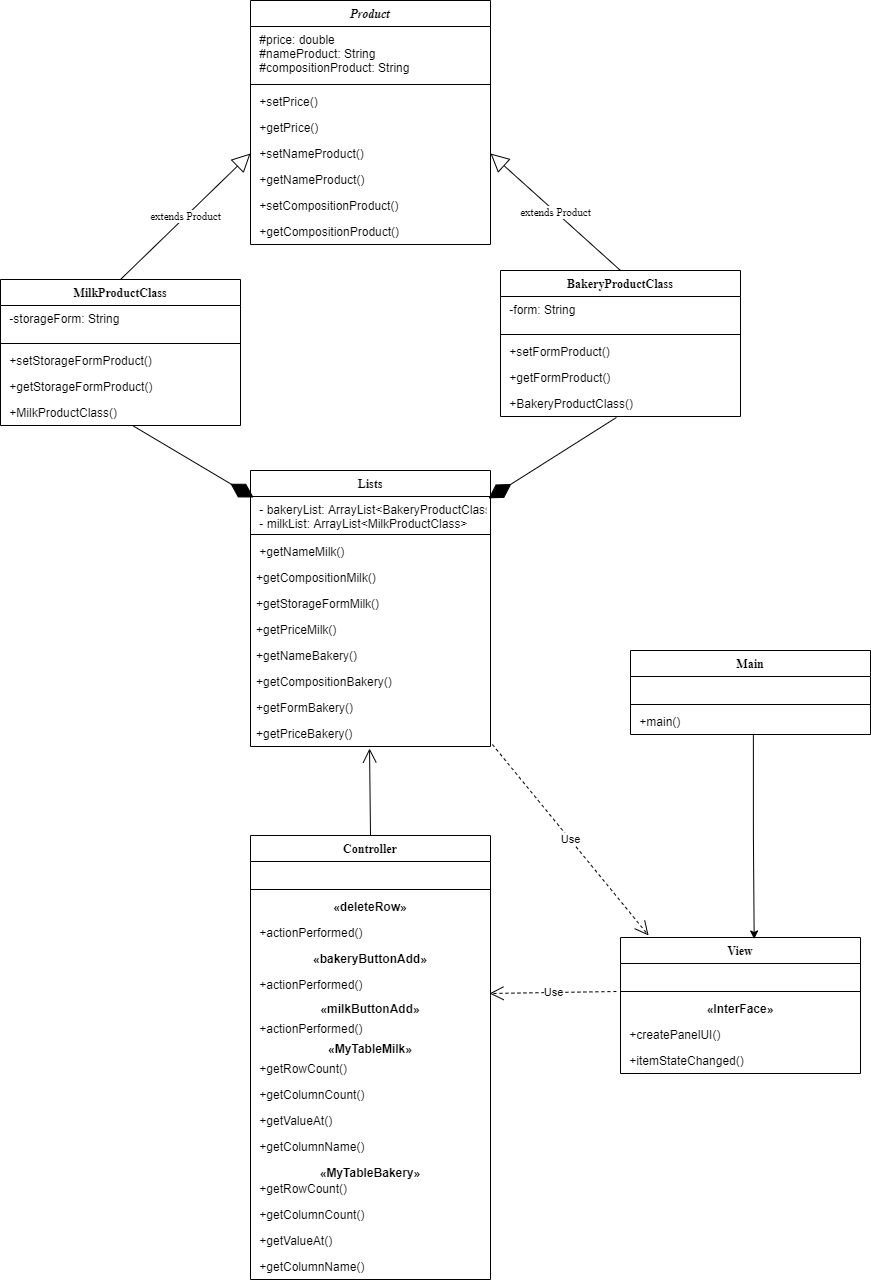


Рисунок 5 – UML-диаграмма проекта

# 4 Проектирование тестов

В ходе разработки было проведено тестирование программы, результаты которого представлены в табл. 1.

Таблица 1 – Тестирование программы

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 1 | Первоначальный запуск программы. | При запуске программы появляется окно добавления продукта – форма добавления молочного продукта. |
| 2 | Нажатие на кнопку “Хлеб” на окне добавления продукта. | Форма добавления молочного продукта сменяется на форму добавления хлебобулочного изделия |
| 3 | Нажатие на кнопку “Молоко” на окне добавления продукта. | Форма добавления хлебобулочного изделия сменяется на форму добавления молочного продукта |
| 4 | Нажатие на кнопку “Добавить” на окне добавления продукта. | Происходит добавления продукта в зависимости от выбранной формы добавления.  Сообщение: “Продукт добавлен” |
| 5 | Нажатие на кнопку “Очистить поля” на окне добавления продукта. | Происходит удаление введённых данных в текстовых полях. |
| 6 | Выбор в CheckBox “Панель - таблица”. | Появляется окно отображения продуктов. |
| 7 | Нажатие на кнопку “Хлеб” на окне отображения продуктов. | Появляется таблица, отображающая хлебобулочные изделия. |
| 8 | Нажатие на кнопку “Молоко” на окне отображения продуктов. | Появляется таблица, отображающая молочные продукты. |
| 9 | Нажатие на кнопку “Найти” или “Удалить” ,при пустой строке поиска, на окне отображения продуктов. | Сообщение: “Пустая строка поиска” |
| 10 | Нажатие на кнопку “Найти” или “Удалить” , на окне отображения продуктов, при введённом индексе большем чем количество строк в списке продукта. | Сообщение: “Некорректный индекс” |
| 11 | Нажатие на кнопку “Найти”, на окне отображения продуктов, при корректно введённом индексе. | Выделение строки под номером, соответствующим номеру введённой строки. |
| 12 | Нажатие на кнопку “Удаление”, на окне отображения продуктов, при корректно введённом индексе. | Удаление строки под номером, соответствующим номеру введённой строки. |
| 13 | Выбор в CheckBox “Панель продуктов”. | Появляется окно добавления продуктов |

Результаты тестирования

Описание:

Проверка запуска программы.

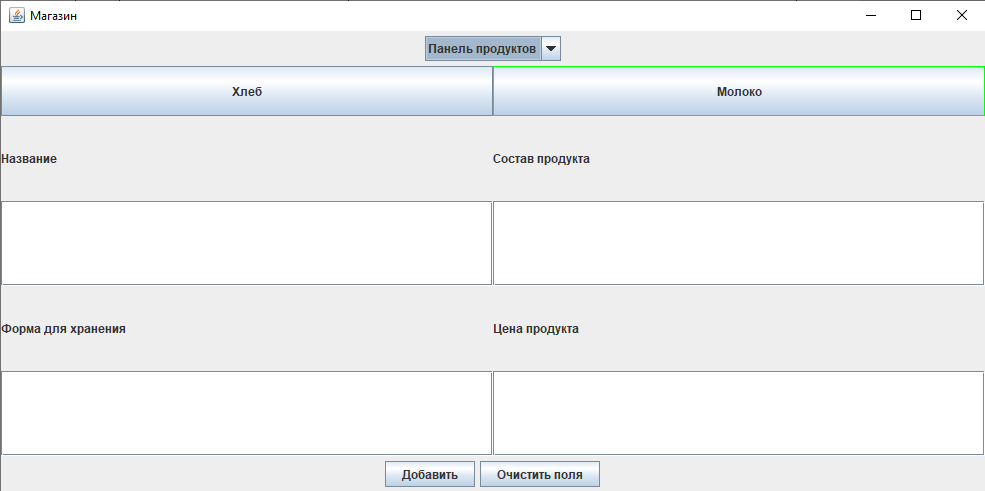


Рисунок – Результат теста №1

Описание:

Проверка работы кнопки “Хлеб”

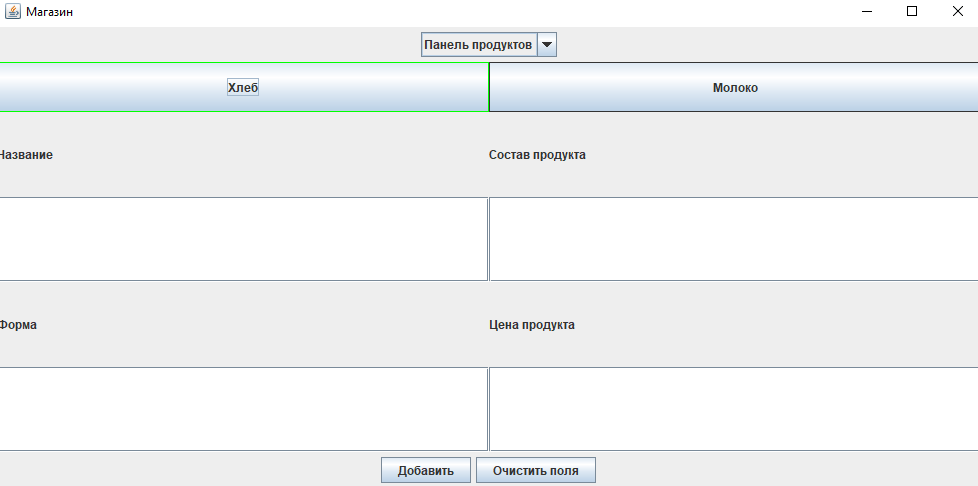


Рисунок – Результат теста №2

Описание:

Проверка работы кнопки “Молоко”

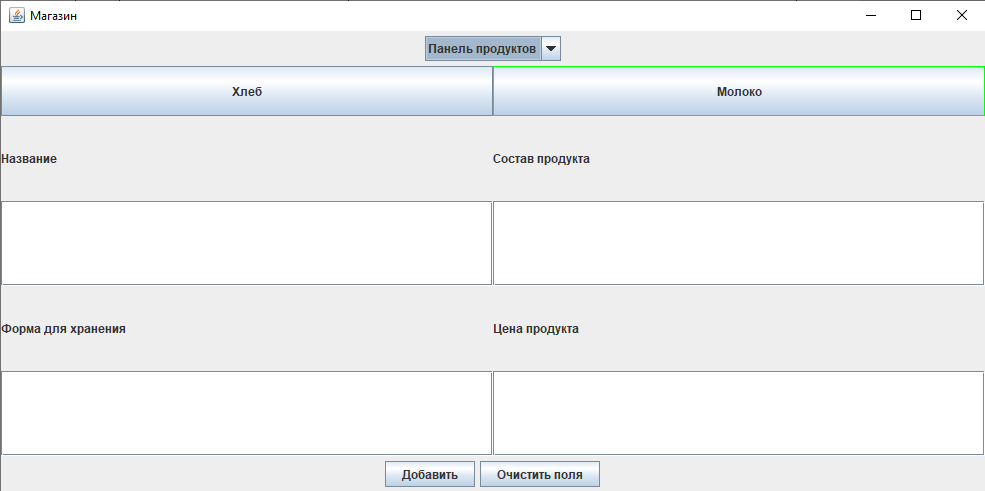


Рисунок – Результат теста №3

Описание:

Проверка работы кнопки “Добавить”

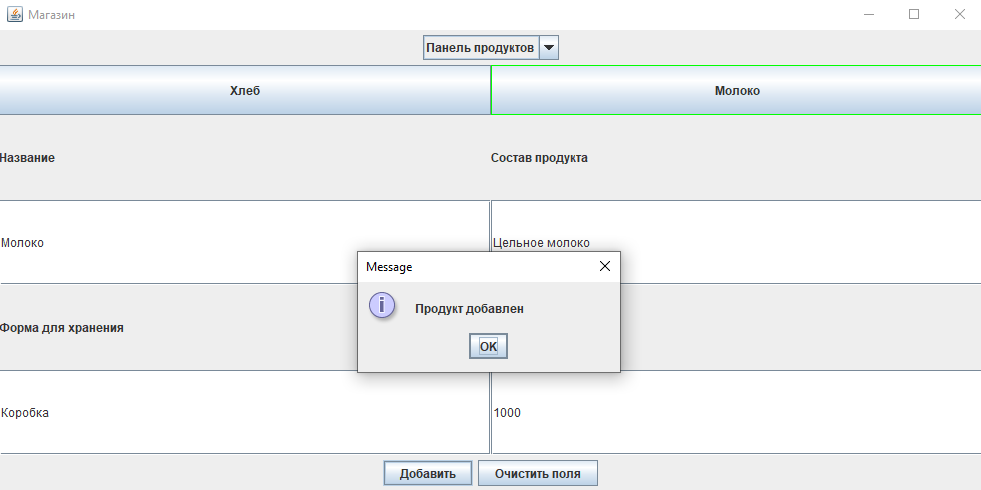


Рисунок – Результат теста №4

Описание:

Проверка работы кнопки “Очистить поля”

Окно до нажатия

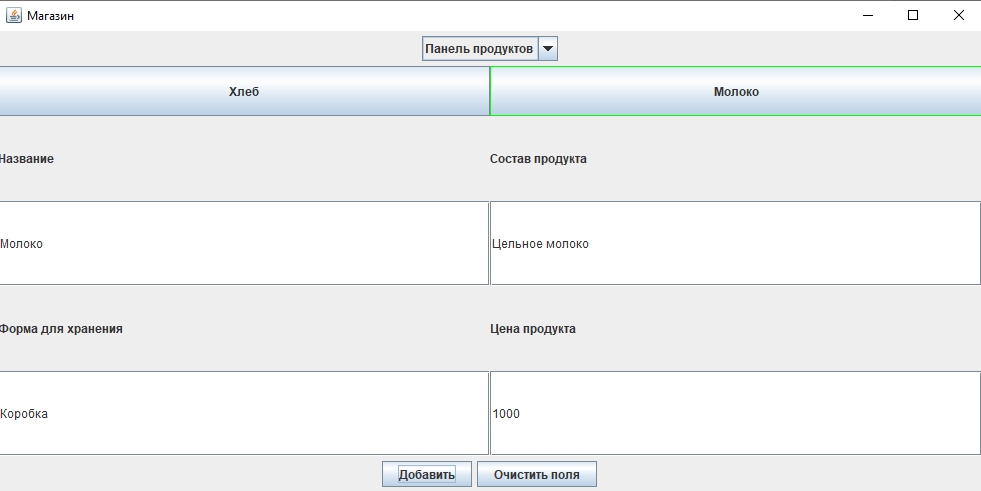


Рисунок *–* Окно до нажатия кнопки “Очистить поля”

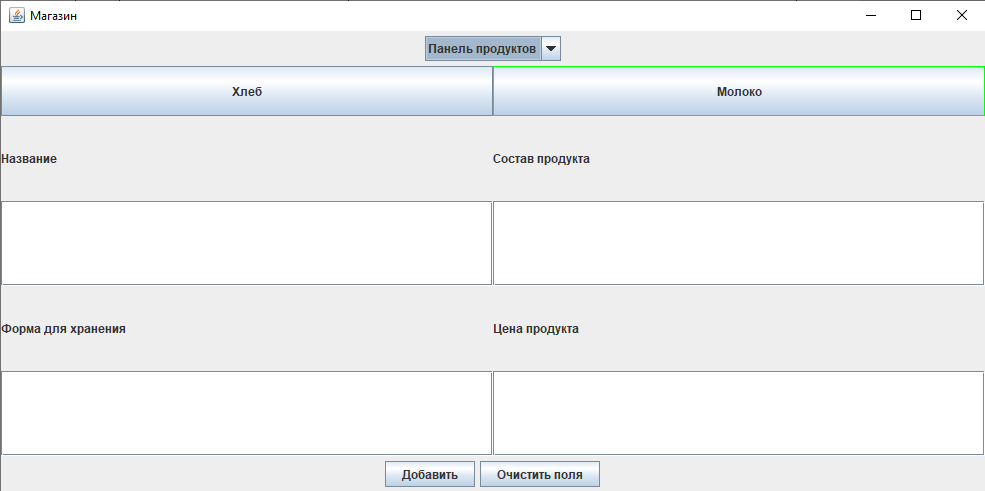


Рисунок – Результат теста №5

Описание:

Проверка работы CheckBox- “Панель - таблица”

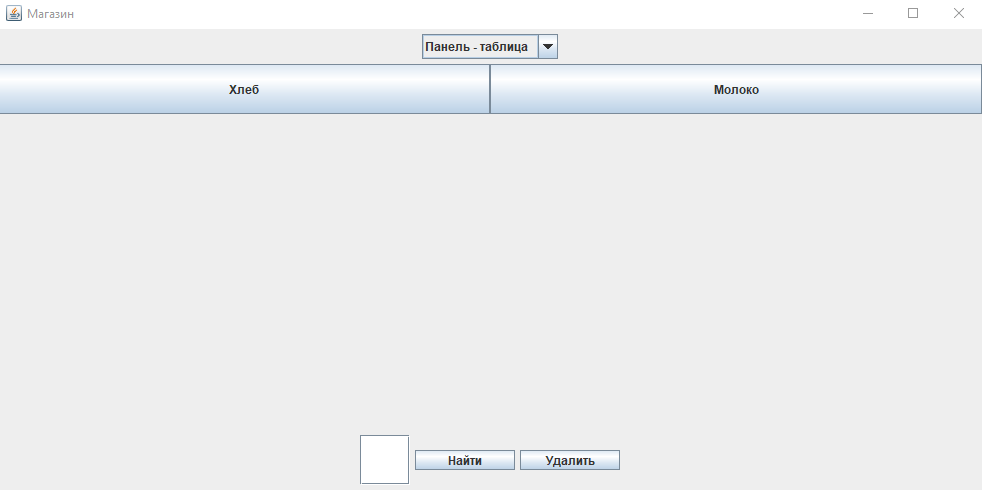


Рисунок – Результат теста №6

Описание:

Проверка работы кнопки “Хлеб”

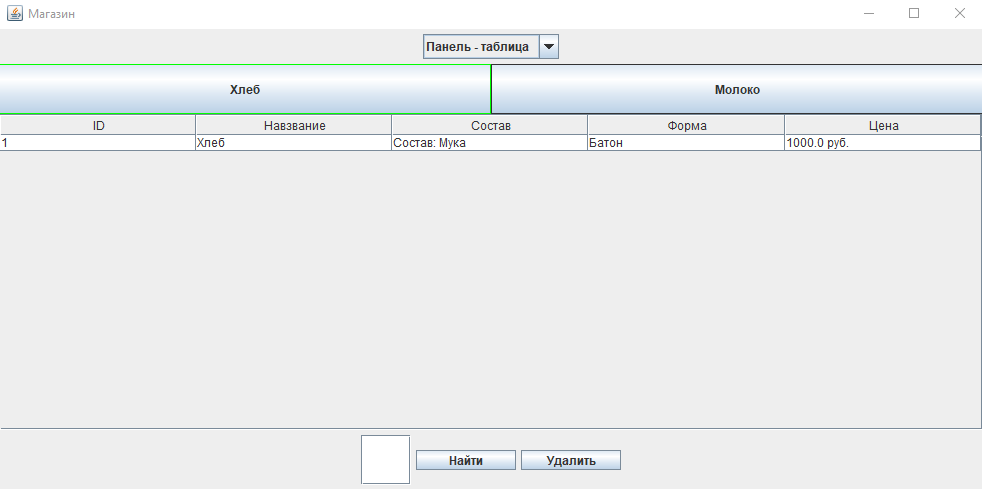


Рисунок – Результат теста №7

Описание:

Проверка работы кнопки “Молоко”

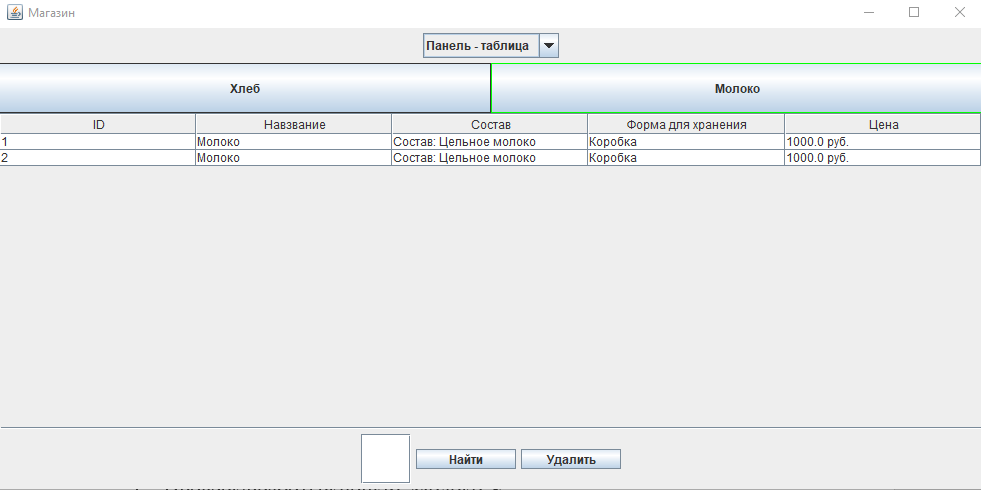


Рисунок – Результат теста №8

Описание:

Проверка теста №9

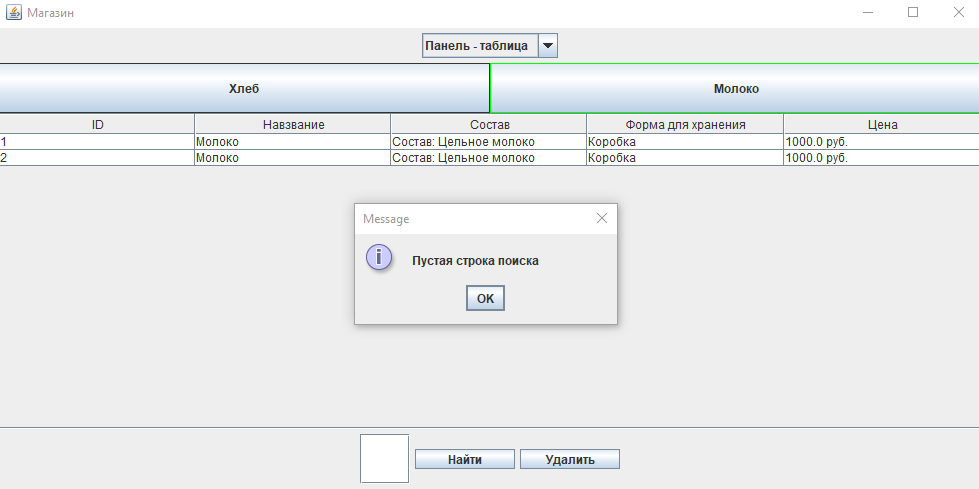


Рисунок – Результат теста №9

Описание:

Проверка теста №10

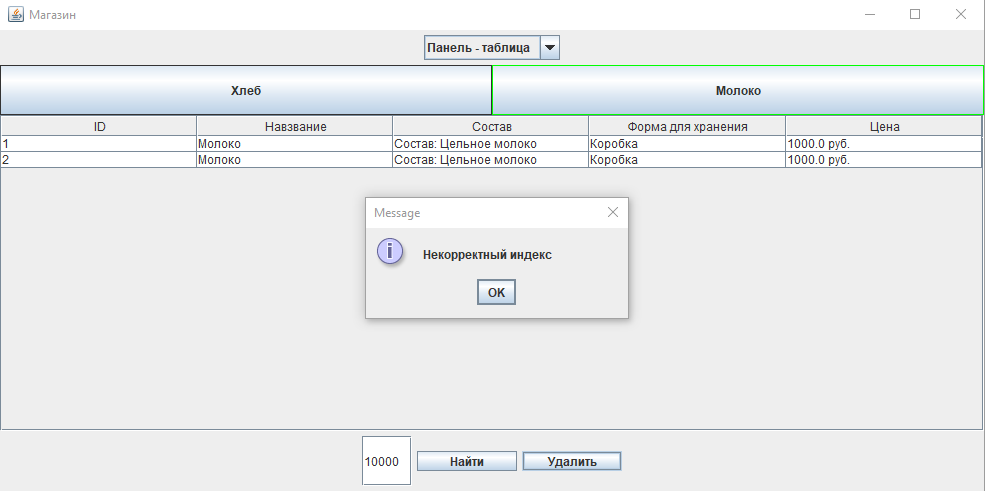


Рисунок – Результат теста №10

Описание:

Проверка теста №11

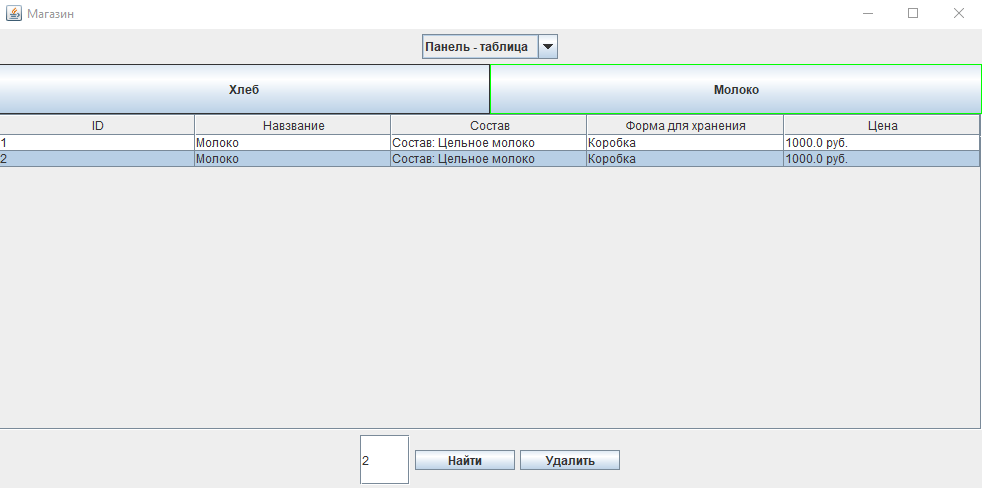


Рисунок – Результат теста №11

Описание:

Проверка теста №12

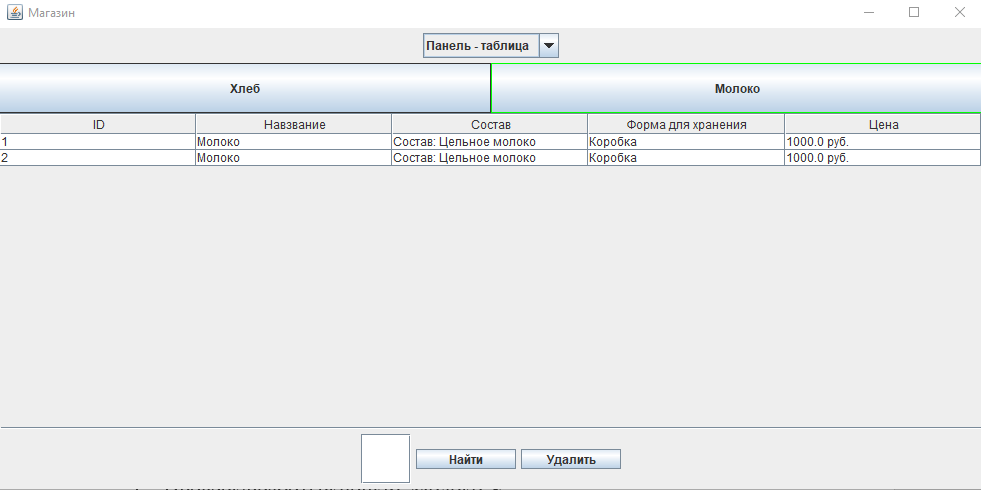


Рисунок – До удаления записи

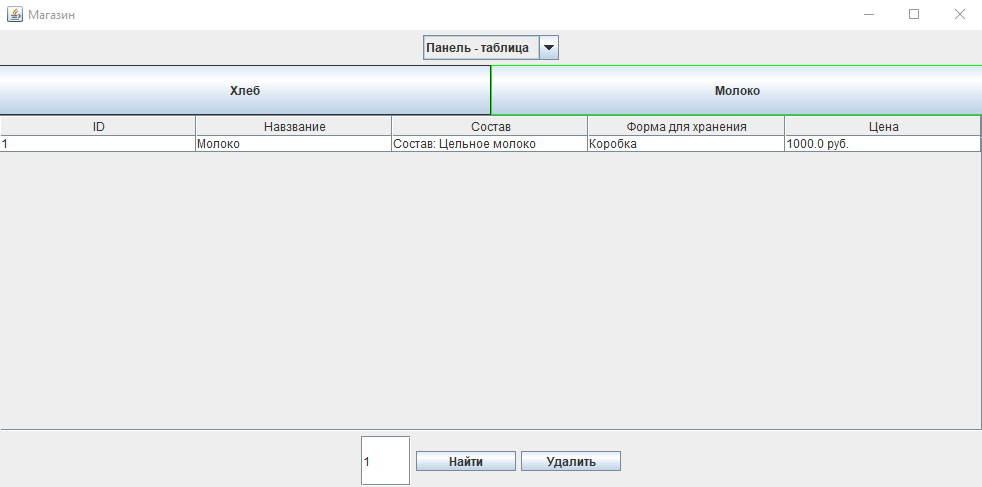


Рисунок – Результат теста №12

Описание:

Проверка теста №13

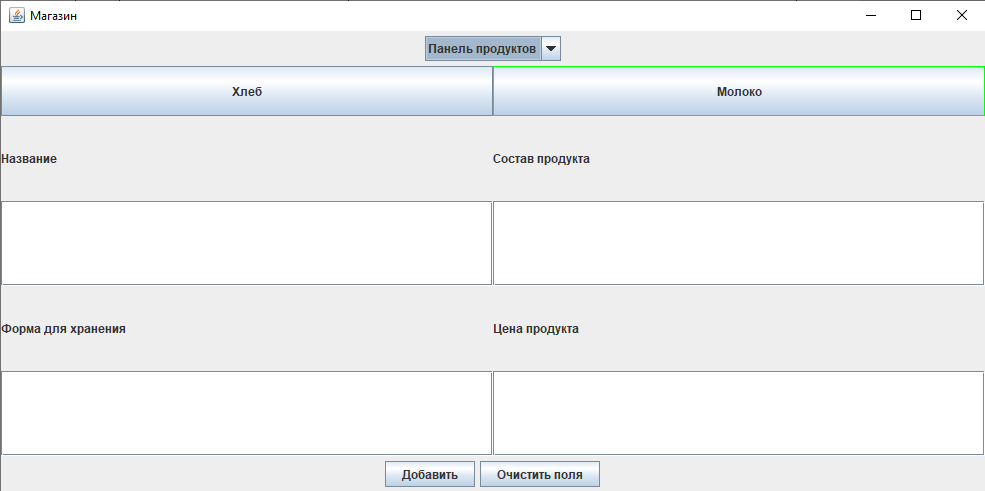


Рисунок – Результат теста №13

# 5 Листинг программы на Java

## 5.1 Главный класс Main

package com.company.test;  
  
import javax.swing.\*;  
  
  
public class Main {  
  
 public static void main(String[] args)  
 {  
 // Создание и настройка окна  
 JFrame frame = new JFrame("Магазин");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 javax.swing.SwingUtilities.*invokeLater*(new Runnable() {  
 public void run() {  
 (new InterFace()).createPanelUI(frame.getContentPane());  
  
 // Открытие окна  
 frame.setSize(1000,500);  
 frame.setVisible(true);  
 }  
 });  
 }  
}

## 5.2 Класс Controller

package com.company.test;  
  
  
import javax.swing.\*;  
import javax.swing.table.AbstractTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
  
public class Controller {  
  
 public static Lists *l* = new Lists();  
  
 public static class bakeryButtonAdd implements ActionListener {  
 public void actionPerformed(ActionEvent e) throws NullPointerException {  
  
 BakeryProductClass bakery = new BakeryProductClass();  
  
 if (InterFace.*bakeryName*.getText().trim().length() > 0 &&  
 InterFace.*bakeryComp*.getText().trim().length() > 0 &&  
 InterFace.*bakeryForm*.getText().trim().length() > 0 &&  
 InterFace.*bakeryPrice*.getText().trim().length() > 0){  
  
 bakery.setNameProduct(InterFace.*bakeryName*.getText());  
 bakery.setCompositionProduct(InterFace.*bakeryComp*.getText());  
 bakery.setFormProduct(InterFace.*bakeryForm*.getText());  
 bakery.setPrice(Double.*parseDouble*(InterFace.*bakeryPrice*.getText()));  
  
 *l*.addBakery(bakery);  
  
 JOptionPane.*showMessageDialog*(null,"Продукт добавлен");  
  
  
 InterFace.*table* = new JTable(new MyTableBakery());  
  
 if(((BorderLayout) InterFace.*card2*.getLayout()).getLayoutComponent(BorderLayout.*CENTER*)!=null) {  
 InterFace.*card2*.remove(((BorderLayout) InterFace.*card2*.getLayout()).getLayoutComponent(BorderLayout.*CENTER*));  
 InterFace.*card2*.add(new JScrollPane(InterFace.*table*), BorderLayout.*CENTER*);  
  
  
 InterFace.*card2bakeryButton*.setBorder(BorderFactory.*createLineBorder*(Color.*green*));  
 InterFace.*card2milkButton*.setBorder(BorderFactory.*createLineBorder*(null));  
  
 InterFace.*card2*.repaint();  
 InterFace.*card2*.validate();  
  
 }  
 }else {JOptionPane.*showMessageDialog*(null,"Заполните все значения продукта");}  
 }  
 }  
  
 public static class milkButtonAdd implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 MilkProductClass milk = new MilkProductClass();  
 if (InterFace.*milkName*.getText().trim().length() > 0 &&  
 InterFace.*milkComp*.getText().trim().length() > 0 &&  
 InterFace.*storageForm*.getText().trim().length() > 0 &&  
 InterFace.*milkPrice*.getText().trim().length() > 0){  
 milk.setNameProduct(InterFace.*milkName*.getText());  
 milk.setCompositionProduct(InterFace.*milkComp*.getText());  
 milk.setStorageFormProduct(InterFace.*storageForm*.getText());  
 milk.setPrice(Double.*parseDouble*(InterFace.*milkPrice*.getText()));  
 *l*.addMilk(milk);  
 JOptionPane.*showMessageDialog*(null,"Продукт добавлен");  
  
 InterFace.*table* = new JTable(new MyTableMilk());  
 if(((BorderLayout) InterFace.*card2*.getLayout()).getLayoutComponent(BorderLayout.*CENTER*)!=null){  
 InterFace.*card2*.remove(((BorderLayout) InterFace.*card2*.getLayout()).getLayoutComponent(BorderLayout.*CENTER*));  
  
 InterFace.*card2*.add(new JScrollPane(InterFace.*table*), BorderLayout.*CENTER*);  
  
 InterFace.*card2milkButton*.setBorder(BorderFactory.*createLineBorder*(Color.*green*));  
 InterFace.*card2bakeryButton*.setBorder(BorderFactory.*createLineBorder*(null));  
  
 InterFace.*card2*.repaint();  
 InterFace.*card2*.validate();  
  
 }  
  
 }else {JOptionPane.*showMessageDialog*(null,"Заполните все значения продукта");}  
 }  
 }  
  
 public static class deleteRow implements ActionListener {  
  
 public void actionPerformed(ActionEvent e) {  
  
 if(InterFace.*choice*!=0){  
 if(InterFace.*choice*==1) {  
 if (InterFace.*filter*.getText().trim().length() > 0) {  
 if (Integer.*parseInt*(InterFace.*filter*.getText()) <= InterFace.*table*.getRowCount()) {  
 InterFace.*table*.setRowSelectionInterval(Integer.*parseInt*(InterFace.*filter*.getText()) - 1, Integer.*parseInt*(InterFace.*filter*.getText()) - 1);  
 *l*.bakeryList.remove(Integer.*parseInt*(InterFace.*filter*.getText())-1);  
 }  
 else {JOptionPane.*showMessageDialog*(null,"Некорректный индекс");}  
 InterFace.*table* = new JTable(new MyTableBakery());  
  
 InterFace.*card2*.remove(((BorderLayout) InterFace.*card2*.getLayout()).getLayoutComponent(BorderLayout.*CENTER*));  
  
 InterFace.*card2*.add(new JScrollPane(InterFace.*table*), BorderLayout.*CENTER*);  
 InterFace.*card2*.repaint();  
 InterFace.*card2*.validate();  
 } else {JOptionPane.*showMessageDialog*(null,"Пустая строка поиска");}  
  
 }  
 if (InterFace.*choice* == 2) {  
 if (InterFace.*filter*.getText().trim().length() > 0) {  
 if (Integer.*parseInt*(InterFace.*filter*.getText()) <= InterFace.*table*.getRowCount()) {  
 InterFace.*table*.setRowSelectionInterval(Integer.*parseInt*(InterFace.*filter*.getText()) - 1, Integer.*parseInt*(InterFace.*filter*.getText()) - 1);  
 *l*.milkList.remove(Integer.*parseInt*(InterFace.*filter*.getText()) - 1);  
 }else {JOptionPane.*showMessageDialog*(null,"Некорректный индекс");}  
 InterFace.*table* = new JTable(new MyTableMilk());  
  
 InterFace.*card2*.remove(((BorderLayout) InterFace.*card2*.getLayout()).getLayoutComponent(BorderLayout.*CENTER*));  
  
 InterFace.*card2*.add(new JScrollPane(InterFace.*table*), BorderLayout.*CENTER*);  
 InterFace.*card2*.repaint();  
 InterFace.*card2*.validate();  
  
 } else {JOptionPane.*showMessageDialog*(null,"Пустая строка поиска");}  
 }  
 }  
 }  
 }  
  
 public static class MyTableBakery extends AbstractTableModel {  
  
 @Override  
 public int getRowCount() {  
 return *l*.bakeryList.size();  
 }  
  
 @Override  
 public int getColumnCount() {  
 return 5;  
 }  
  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 return switch (columnIndex) {  
 case 0 -> rowIndex+1;  
 case 1 -> *l*.getNameBakery(rowIndex);  
 case 2 -> "Состав: " + *l*.getCompositionBakery(rowIndex);  
 case 3 -> *l*.getFormBakery(rowIndex);  
 case 4 -> *l*.getPriceBakery(rowIndex) + " руб.";  
 default -> "Не определена";  
 };  
 }  
  
 @Override  
 public String getColumnName(int cell) {  
 return switch (cell) {  
 case 0 -> "ID";  
 case 1 -> "Навзвание";  
 case 2 -> "Состав";  
 case 3 -> "Форма";  
 case 4 -> "Цена";  
 default -> "Other Column";  
 };  
 }  
 }  
 public static class MyTableMilk extends AbstractTableModel {  
  
 @Override  
 public int getRowCount() {  
 return *l*.milkList.size();  
 }  
  
 @Override  
 public int getColumnCount() {  
 return 5;  
 }  
  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 return switch (columnIndex) {  
 case 0 -> rowIndex+1;  
 case 1 -> *l*.getNameMilk(rowIndex);  
 case 2 -> "Состав: " + *l*.getCompositionMilk(rowIndex);  
 case 3 -> *l*.getStorageFormMilk(rowIndex);  
 case 4 -> *l*.getPriceMilk(rowIndex) + " руб.";  
 default -> "Не определена";  
 };  
 }  
  
 @Override  
 public String getColumnName(int cell) {  
 return switch (cell) {  
 case 0 -> "ID";  
 case 1 -> "Навзвание";  
 case 2 -> "Состав";  
 case 3 -> "Форма для хранения";  
 case 4 -> "Цена";  
 default -> "Other Column";  
 };  
 }  
 }  
  
}

## 5.3 Класс View

package com.company.test;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.ItemEvent;  
import java.awt.event.ItemListener;  
  
  
  
 class InterFace implements ItemListener {  
  
  
 public static JPanel card1CenterPanelBakery = new JPanel(new GridLayout(2,2));  
 public static JPanel card1CenterPanelMilk = new JPanel(new GridLayout(2,2));  
  
 public static JButton bakeryButton = new JButton("Хлеб");  
 public static JButton milkButton = new JButton("Молоко");  
  
 public static JButton card2bakeryButton = new JButton("Хлеб");  
 public static JButton card2milkButton = new JButton("Молоко");  
  
 public static JTextField bakeryName = new JTextField("");  
 public static JTextField bakeryComp = new JTextField("");  
 public static JTextField bakeryForm = new JTextField("");  
 public static JTextField bakeryPrice = new JTextField("");  
  
 public static JTextField milkName = new JTextField("");  
 public static JTextField milkComp = new JTextField("");  
 public static JTextField storageForm = new JTextField("");  
 public static JTextField milkPrice = new JTextField("");  
  
 public static JTable table;  
  
 public static JButton search = new JButton("Найти");  
 public static JButton remove = new JButton("Удалить");  
  
  
 public static JPanel card2 = new JPanel(new BorderLayout());  
  
  
 public static int choice = 0;  
  
 private static JPanel cards;  
  
  
 public static JTextField filter= new JTextField("");  
  
 final static String BUTTONPANEL = "Панель продуктов";  
 final static String TEXTPANEL = "Панель - таблица";  
  
 /\*\*  
 \* Метод определения интерфейса панели  
 \*  
 \* @param container - панель содержимого  
 \*/  
  
 public void createPanelUI(Container container) {  
  
  
 milkButton.setBorder(BorderFactory.createLineBorder(Color.green));  
  
 filter.setPreferredSize(new Dimension(50,50));  
 search.setPreferredSize(new Dimension(100,20));  
 remove.setPreferredSize(new Dimension(100,20));  
  
  
 search.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (filter.getText().trim().length() > 0) {  
 if (Integer.parseInt(InterFace.filter.getText()) <= table.getRowCount()) {  
 table.setRowSelectionInterval(Integer.parseInt(filter.getText()) - 1, Integer.parseInt(filter.getText()) - 1);  
 table.changeSelection(Integer.parseInt(filter.getText()) - 1, Integer.parseInt(filter.getText()) - 1, true, true);  
 }  
 else {JOptionPane.showMessageDialog(null,"Некорректный индекс");}  
 }  
 else {JOptionPane.showMessageDialog(null,"Пустая строка поиска");}  
 }  
 });  
  
 remove.addActionListener(new Controller.deleteRow());  
  
 bakeryButton.setPreferredSize(new Dimension(50,50));  
 milkButton.setPreferredSize(new Dimension(50,50));  
  
 card2bakeryButton.setPreferredSize(new Dimension(50,50));  
 card2milkButton.setPreferredSize(new Dimension(50,50));  
  
 JComboBox<String> combobox = new JComboBox<String>(  
 new String[]{BUTTONPANEL, TEXTPANEL});  
 combobox.setEditable(false);  
 combobox.addItemListener(this);  
  
 JPanel mainPanel = new JPanel();  
  
 mainPanel.add(combobox);  
  
 JPanel card1SouthPanel = new JPanel(new FlowLayout());  
  
 JButton add = new JButton("Добавить");  
 JButton clear = new JButton("Очистить поля");  
  
  
 clear.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(card1CenterPanelBakery.isVisible()) {  
 bakeryName.setText("");  
 bakeryComp.setText("");  
 bakeryForm.setText("");  
 bakeryPrice.setText("");  
 }  
 if(card1CenterPanelMilk.isVisible()){  
 milkName.setText("");  
 milkComp.setText("");  
 storageForm.setText("");  
 milkPrice.setText("");  
 }  
 }  
 });  
  
 Controller.bakeryButtonAdd b = new Controller.bakeryButtonAdd();  
 Controller.milkButtonAdd m = new Controller.milkButtonAdd();  
  
 // Создание "cards".  
 JPanel card1 = new JPanel(new BorderLayout());  
  
 add.addActionListener(m);  
  
 bakeryButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 card1.add(card1CenterPanelBakery, BorderLayout.CENTER);  
 card1CenterPanelBakery.setVisible(true);  
 card1CenterPanelMilk.setVisible(false);  
 add.removeActionListener(m);  
 add.addActionListener(b);  
 bakeryButton.setBorder(BorderFactory.createLineBorder(Color.green));  
 milkButton.setBorder(BorderFactory.createLineBorder(null));  
  
 }  
 });  
  
 milkButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 card1CenterPanelMilk.setVisible(true);  
 card1CenterPanelBakery.setVisible(false);  
 add.removeActionListener(b);  
 add.addActionListener(m);  
 milkButton.setBorder(BorderFactory.createLineBorder(Color.green));  
 bakeryButton.setBorder(BorderFactory.createLineBorder(null));  
  
 }  
 });  
  
 //-------------------------bakeryPanelCenter-----------------//  
 JPanel addBakeryName = new JPanel(new GridLayout(2,1));  
 addBakeryName.add(new JLabel("Название"));  
 addBakeryName.add(bakeryName);  
  
 JPanel addBakeryСomposition = new JPanel(new GridLayout(2,1));  
 addBakeryСomposition.add(new JLabel("Состав продукта"));  
 addBakeryСomposition.add(bakeryComp);  
  
 JPanel addForm = new JPanel(new GridLayout(2,1));  
 addForm.add(new JLabel("Форма"));  
 addForm.add(bakeryForm);  
  
 JPanel addBakeryPrice = new JPanel(new GridLayout(2,1));  
 addBakeryPrice.add(new JLabel("Цена продукта"));  
 addBakeryPrice.add(bakeryPrice);  
  
 card1CenterPanelBakery.add(addBakeryName);  
 card1CenterPanelBakery.add(addBakeryСomposition);  
 card1CenterPanelBakery.add(addForm);  
 card1CenterPanelBakery.add(addBakeryPrice);  
 //-------------------------bakeryPanelCenter-----------------//  
  
 //-------------------------milkPanelCenter-----------------//  
 JPanel addMilkName = new JPanel(new GridLayout(2,1));  
 addMilkName.add(new JLabel("Название"));  
 addMilkName.add(milkName);  
  
 JPanel addMilkСomposition = new JPanel(new GridLayout(2,1));  
 addMilkСomposition.add(new JLabel("Состав продукта"));  
 addMilkСomposition.add(milkComp);  
  
 JPanel addStorageForm = new JPanel(new GridLayout(2,1));  
 addStorageForm.add(new JLabel("Форма для хранения"));  
 addStorageForm.add(storageForm);  
  
 JPanel addMilkPrice = new JPanel(new GridLayout(2,1));  
 addMilkPrice.add(new JLabel("Цена продукта"));  
 addMilkPrice.add(milkPrice);  
  
 card1CenterPanelMilk.add(addMilkName);  
 card1CenterPanelMilk.add(addMilkСomposition);  
 card1CenterPanelMilk.add(addStorageForm);  
 card1CenterPanelMilk.add(addMilkPrice);  
 //-------------------------milkPanelCenter-----------------//  
  
 card1SouthPanel.add(add);  
 card1SouthPanel.add(clear);  
  
  
 //-------------------------mainButtons-----------------//  
 JPanel MainButtons = new JPanel(new GridLayout(1,2));  
 MainButtons.add(bakeryButton);  
 MainButtons.add(milkButton);  
 //-------------------------mainButtons-----------------//  
  
 card1.add(MainButtons,BorderLayout.NORTH);  
 card1.add(card1CenterPanelBakery,BorderLayout.CENTER);  
 card1.add(card1CenterPanelMilk,BorderLayout.CENTER);  
 card1.add(card1SouthPanel,BorderLayout.SOUTH);  
  
  
 JPanel card2MainButtons = new JPanel(new GridLayout(1,2));  
 card2MainButtons.add(card2bakeryButton);  
 card2MainButtons.add(card2milkButton);  
  
  
 JPanel card2SouthButtons = new JPanel(new FlowLayout());  
 card2SouthButtons.add(filter);  
 card2SouthButtons.add(search);  
 card2SouthButtons.add(remove);  
  
  
 InterFace.card2.add(card2MainButtons,BorderLayout.NORTH);  
 card2.add(card2SouthButtons,BorderLayout.SOUTH);  
  
 //--------------------------------------------------Table------------------//  
 card2bakeryButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 table = new JTable(new Controller.MyTableBakery());  
  
 choice = 1;  
 BorderLayout layout = (BorderLayout)card2.getLayout();  
  
 if(layout.getLayoutComponent(BorderLayout.CENTER)==null){  
 card2.add(table);  
 }  
  
 card2.remove(layout.getLayoutComponent(BorderLayout.CENTER));  
  
 card2.add(new JScrollPane(table), BorderLayout.CENTER);  
 card2bakeryButton.setBorder(BorderFactory.createLineBorder(Color.green));  
 card2milkButton.setBorder(BorderFactory.createLineBorder(null));  
 card2.setVisible(true);  
  
  
 card2.repaint();  
 card2.validate();  
  
 }  
 });  
  
 card2milkButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 choice = 2;  
  
 table = new JTable(new Controller.MyTableMilk());  
  
 BorderLayout layout = (BorderLayout)card2.getLayout();  
  
  
 if(layout.getLayoutComponent(BorderLayout.CENTER)==null){  
 card2.add(table);  
 }  
  
  
 card2.remove(layout.getLayoutComponent(BorderLayout.CENTER));  
 card2.add(new JScrollPane(table), BorderLayout.CENTER);  
 card2milkButton.setBorder(BorderFactory.createLineBorder(Color.green));  
 card2bakeryButton.setBorder(BorderFactory.createLineBorder(null));  
  
 card2.setVisible(true);  
  
 card2.repaint();  
 card2.validate();  
 }  
 });  
 //--------------------------------------------------Table------------------//  
 // Создаем панель с менеджером расположения CardLayout  
 cards = new JPanel(new CardLayout());  
 // Добавляем вкладки  
 cards.add(card1, BUTTONPANEL);  
 cards.add(card2, TEXTPANEL);  
 container.add(mainPanel, BorderLayout.PAGE\_START);  
 container.add(cards, BorderLayout.CENTER);  
 }  
 // Обработчик события  
 public void itemStateChanged(ItemEvent event) {  
 CardLayout layout = (CardLayout) (cards.getLayout());  
 layout.show(cards, (String) event.getItem());  
 }  
 }

## 5.4 Класс Product

package com.company.test;  
  
public abstract class Product {  
 protected double price; //цена продукта  
 protected String nameProduct; //название продукта  
 protected String compositionProduct; //состав продукта  
  
 public Product(){  
 price = 0;  
 nameProduct = "";  
 compositionProduct = "";  
 }  
 public boolean setPrice(double val) {  
 if(val <= 0)  
 return false;  
 this.price = val;  
  
 return false;  
 }  
  
 public double getPrice() {  
 return this.price;  
 }  
  
 public void setNameProduct(String n) {  
 this.nameProduct = n;  
 }  
  
 public String getNameProduct() {  
 return this.nameProduct;  
 }  
  
 public void setCompositionProduct(String c) {  
 this.compositionProduct = c;  
 }  
  
 public String getCompositionProduct() {  
 return compositionProduct;  
 }  
}  
class BakeryProductClass extends Product{  
 private String form;  
  
 public BakeryProductClass(){  
 super();  
 form = "";  
 }  
 public void setFormProduct(String f){  
 this.form = f;  
 }  
  
 public String getFormProduct(){  
 return this.form;  
 }  
  
}  
class MilkProductClass extends Product{  
  
 private String storageForm;  
  
 public MilkProductClass(){  
 super();  
 storageForm = "";  
 }  
  
  
 public void setStorageFormProduct(String f){  
 this.storageForm = f;  
 }  
  
 public String getStorageFormProduct(){  
 return this.storageForm;  
 }  
  
}

## 5.5 Класс Lists

package com.company.test;  
  
import java.util.ArrayList;  
  
public class Lists {  
 public ArrayList<BakeryProductClass> bakeryList; //хлебо-булочные изделия  
 public ArrayList<MilkProductClass> milkList; // молочные изделия  
 public Lists(){  
 bakeryList = new ArrayList<>();  
 milkList = new ArrayList<>();  
 }  
 public void addBakery(BakeryProductClass b){  
 bakeryList.add(b);  
 }  
 public void addMilk(MilkProductClass m){  
 milkList.add(m);  
 }  
   
 public String getNameMilk(int rowIndex) {  
 return this.milkList.get(rowIndex).getNameProduct();  
 }  
 public String getCompositionMilk(int rowIndex) {  
 return this.milkList.get(rowIndex).getCompositionProduct();  
 }  
 public String getStorageFormMilk(int rowIndex){  
 return this.milkList.get(rowIndex).getStorageFormProduct();  
 }  
 public String getPriceMilk(int rowIndex){  
 return Double.toString(this.milkList.get(rowIndex).getPrice());  
 }  
 public String getNameBakery(int rowIndex){  
 return this.bakeryList.get(rowIndex).getNameProduct();  
 }  
 public String getCompositionBakery(int rowIndex){  
 return this.bakeryList.get(rowIndex).getCompositionProduct();  
 }  
 public String getFormBakery(int rowIndex){  
 return this.bakeryList.get(rowIndex).getFormProduct();  
 }  
 public String getPriceBakery(int rowIndex){  
 return Double.toString(this.bakeryList.get(rowIndex).getPrice());  
 }  
}