|  |  |  |  |
| --- | --- | --- | --- |
|  | | для прик эмбл |  |
| *МИНОБРНАУКИ РОССИИ* | | | |
| ***Федеральное государственное бюджетное образовательное учреждение***  ***высшего профессионального образования***  *"Московский государственный университет информационных технологий, радиотехники и* *электроники"*  *(МИРЭА МГУПИ)* | | | |
| Институт Информационных Технологий | | | |
| Кафедра МОСИТ | | | |

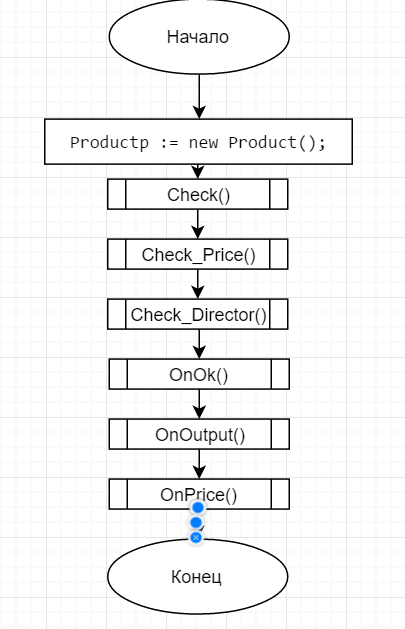
|  |
| --- |
| Лабораторная работа №1  По дисциплине: Объектно ориентированное программирование |
| Выполнила студентка группы: ИКБО-02-13  Силаева А.Э. |
| *2017 год* |
|  |
|  |
|  |

Задание 1

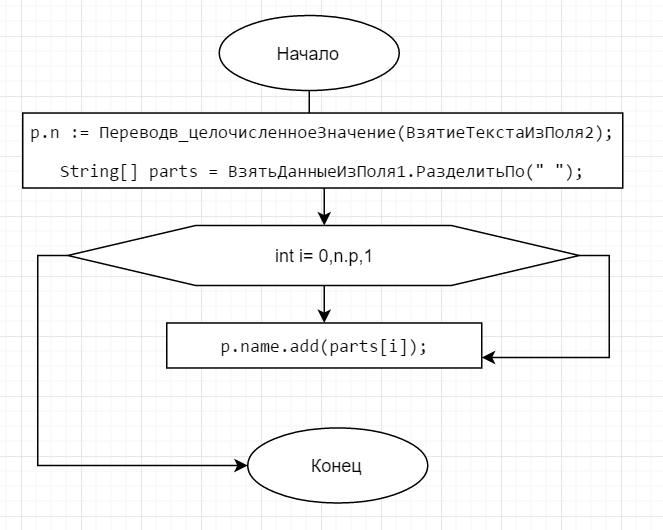
Цель: Ввести n строк с консоли. Вывести на консоль те строки, длина которых меньше средней, также их длины.

Блок-схема задачи

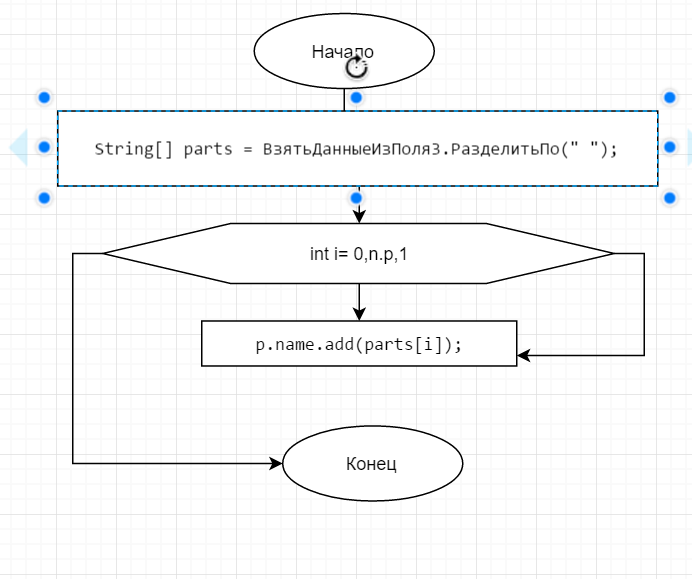
ProductMain() – метод по выполнению всех функций в классе связанные с вызовов методом кнопок, чекбоксов.



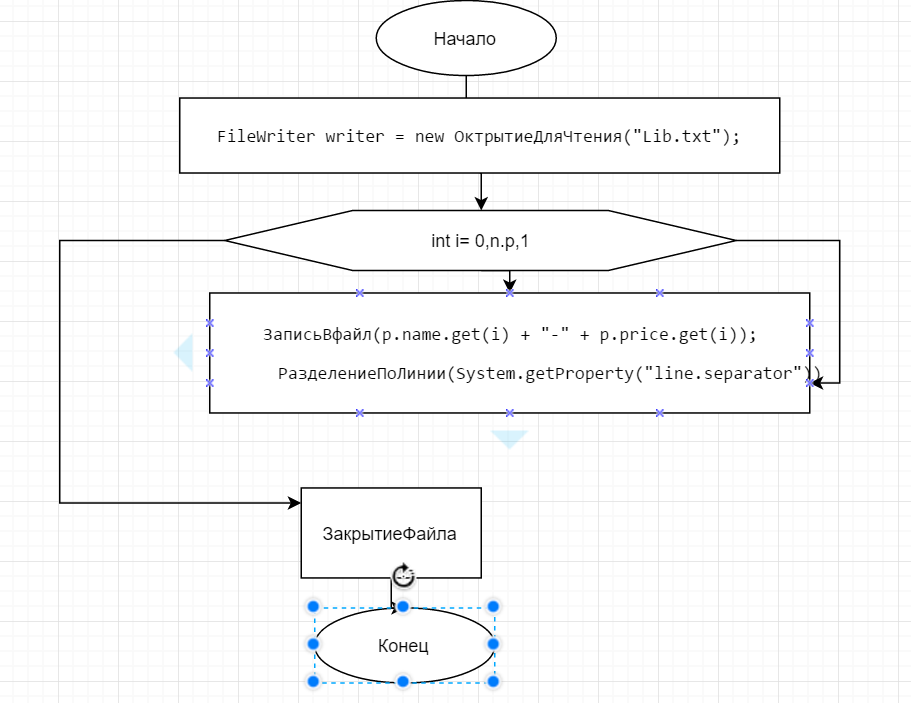
OnOk() – метод для кнопки Input, ввод количества товаров и ввод названия товаров.



OnPrice() – метод по вводу цены товаров



OnOutput() – вывод названия товара и цены товара в файл



Check() Check\_Price() Check\_Director() – методы по управлению полномочиями каждого сотрудника

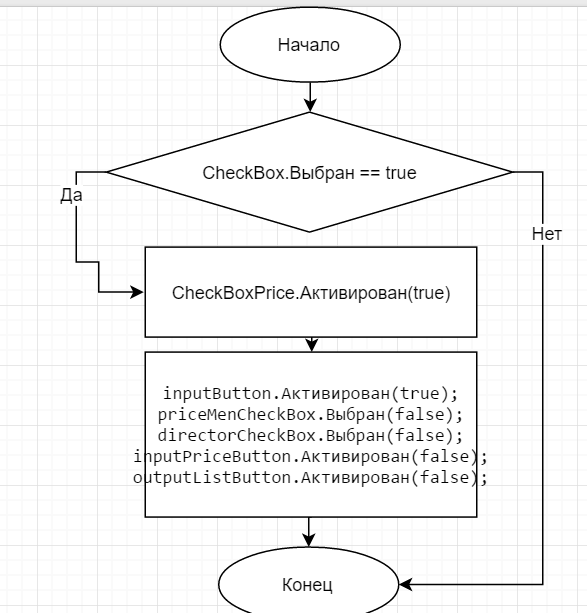
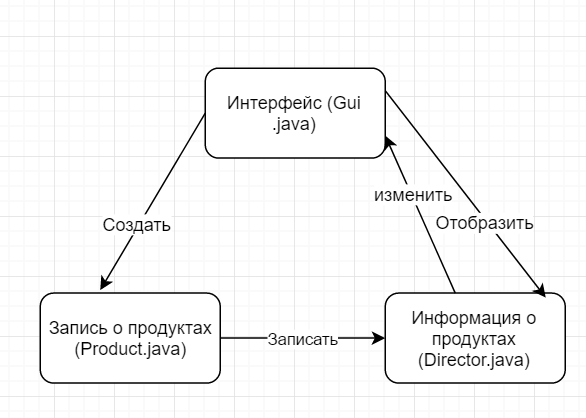


Диаграмма Объектной декомпозиции



Концептуальная диаграмма классов

D:\Загрузки\Untitled Diagram (2).png

Диаграмма «вариантов использования»

D:\Загрузки\Untitled Diagram (3).png

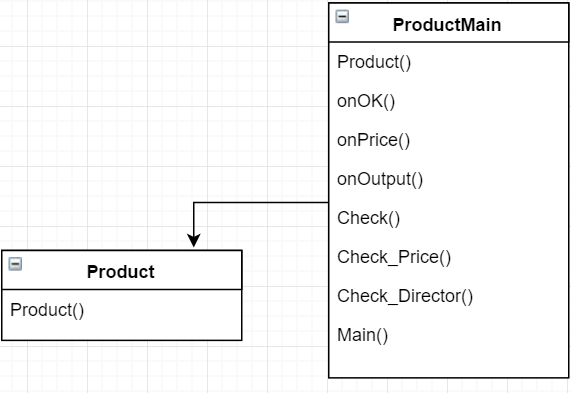
Диаграмма последовательности системы

D:\Загрузки\Untitled Diagram (6).png

Диаграмма деятельностей

D:\Загрузки\Untitled Diagram (5).png

Проект-схемы иерархии классов задачи. Описание классов и методов.



Product – класс по созданию количества товара, цены и названия.

ProductMain() – основной класс по выполнению программы

onOK() – метод заполнения названия и количества товара

onPrice() – метод заполнения цены

onOutput() – метод вывода данных в файл

Check() – метод для активирования полномочий Работника

Check\_Price() – метод для активирования полномочий Бухгалтера

Check\_Director() – метод для активирования полномочий Директора

Main() – метод, вызывает форму.

Программный код реализации задачи.

*//package com.company;***import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
  
*/\*\*  
 \* Created by Grotter on 25.03.2017.  
 \*/***class** Product { *// Класс товар* **int n**;  
 ArrayList<String> **name**;  
 ArrayList<String> **price**;  
  
 Product() {  
 **n** = 0;  
 **name** = **new** ArrayList<String>(); *// Пустой список для названия товара* **price** = **new** ArrayList<String>();*// Пустой список для цены товара* }  
  
}  
  
**public class** ProductMain **extends** JFrame {  
 **private** JPanel **root**;  
 **private** JButton **inputButton**;  
 **private** JButton **inputPriceButton**;  
 **private** JButton **outputListButton**;  
 **private** JCheckBox **directorCheckBox**;  
 **private** JCheckBox **priceMenCheckBox**;  
 **private** JFormattedTextField **formattedTextField2**;  
 **private** JCheckBox **workerCheckBox**;  
 **private** JList **list1**;  
 **private** JFormattedTextField **Text1**;  
 **private** JFormattedTextField **Text2**;  
 **private** JTextArea **textArea1**;  
 **private** JButton **button1**;  
 Product **p** = **new** Product();  
  
 **public** ProductMain() {  
 setContentPane(**root**);  
  
 setVisible(**true**);  
 setSize(800, 600);  
 getRootPane().setDefaultButton(**inputButton**);  
 JOptionPane.*showMessageDialog*(**null**, **"If want to see the grogram, you should rotate the window"**);  
  
  
 **workerCheckBox**.addActionListener(**new** ActionListener() { *// Активация кнопок* **public void** actionPerformed(ActionEvent e) {  
 Check();  
 }  
 });  
  
 **priceMenCheckBox**.addActionListener(**new** ActionListener() {  
 **public void** actionPerformed(ActionEvent e) {  
 Check\_Price();  
 }  
 });  
 **directorCheckBox**.addActionListener(**new** ActionListener() {  
 **public void** actionPerformed(ActionEvent e) {  
 Check\_Director();  
 }  
 });  
  
 **inputButton**.addActionListener(**new** ActionListener() {  
 **public void** actionPerformed(ActionEvent e) {  
 onOK();  
 }  
 });  
  
 **inputPriceButton**.addActionListener(**new** ActionListener() {  
 **public void** actionPerformed(ActionEvent e) {  
 onPrice();  
 }  
 });  
  
 **outputListButton**.addActionListener(**new** ActionListener() {  
 **public void** actionPerformed(ActionEvent e) {  
 **try** {  
 onOutput();  
 } **catch** (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
 });  
 **button1**.addActionListener(**new** ActionListener() {  
 **public void** actionPerformed(ActionEvent e) {  
 **try** {  
 onOutputText();  
 } **catch** (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
 });  
  
 setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 }  
  
 **private void** Check() { *// Активировать полномочия Работника* **if** (**workerCheckBox**.isSelected()) {  
 JOptionPane.*showMessageDialog*(**null**, **"Your can input all names"**);  
 **inputButton**.setEnabled(**true**);  
 **priceMenCheckBox**.setEnabled(**true**);  
 **priceMenCheckBox**.setSelected(**false**);  
 **directorCheckBox**.setSelected(**false**);  
 **inputPriceButton**.setEnabled(**false**);  
 **outputListButton**.setEnabled(**false**);  
  
 **button1**.setEnabled(**false**);  
 }  
 **if** (**workerCheckBox**.isSelected() == **false**) {  
 JOptionPane.*showMessageDialog*(**null**, **"You can not input all names"**);  
 **inputButton**.setEnabled(**false**);  
 }  
 }  
  
 **private void** Check\_Price() {*// Активировать полномочия Бухгалтера* **if** (**priceMenCheckBox**.isSelected()) {  
 JOptionPane.*showMessageDialog*(**null**, **"Your can input all price = you should input "** + **p**.**n** + **"price"**);  
 **inputButton**.setEnabled(**false**);  
 **workerCheckBox**.setSelected(**false**);  
 **priceMenCheckBox**.setEnabled(**true**);  
 **directorCheckBox**.setSelected(**false**);  
 **directorCheckBox**.setEnabled(**true**);  
 **inputPriceButton**.setEnabled(**true**);  
 **outputListButton**.setEnabled(**false**);  
  
 **button1**.setEnabled(**false**);  
 }  
 **if** (**priceMenCheckBox**.isSelected() == **false**) {  
 JOptionPane.*showMessageDialog*(**null**, **"You can not input all price= "** + **p**.**n**);  
 **inputPriceButton**.setEnabled(**false**);  
 }  
 }  
  
  
 **private void** Check\_Director() {*// Активировать полномочия Директора* **if** (**directorCheckBox**.isSelected()) {  
 JOptionPane.*showMessageDialog*(**null**, **"Your can output your journal"**);  
 **outputListButton**.setEnabled(**true**);  
 **priceMenCheckBox**.setSelected(**false**);  
 **workerCheckBox**.setSelected(**false**);  
 **inputPriceButton**.setEnabled(**false**);  
 **outputListButton**.setEnabled(**true**);  
 **inputButton**.setEnabled(**false**);  
 **button1**.setEnabled(**true**);  
 }  
 **if** (**directorCheckBox**.isSelected() == **false**) {  
 JOptionPane.*showMessageDialog*(**null**, **"You can not input all price= "** + **p**.**n**);  
 **inputPriceButton**.setEnabled(**false**);  
 }  
 }  
  
 **private void** onOK() { *// Заполнение названия товара и количества товара* **if** (**formattedTextField2**.getText().equalsIgnoreCase((**""**))) {  
 JOptionPane.*showMessageDialog*(**null**, **"You should write what are count of products"**);  
  
 }  
 **if** (**Text1**.getText().equalsIgnoreCase((**""**))) {  
 JOptionPane.*showMessageDialog*(**null**, **"You should write the names of products"**);  
 } **else** {  
 **p**.**n** = Integer.*parseInt*(**formattedTextField2**.getText());  
 String[] parts = **Text1**.getText().split(**" "**);  
 **for** (**int** i = 0; i < **p**.**n**; i++) {  
 **p**.**name**.add(parts[i]);  
 }  
 JOptionPane.*showMessageDialog*(**null**, **"Name of products done this is your list: "** + **p**.**name**.toString());  
 }  
*// Окно сообщения  
  
// add your code here* }  
  
 **private void** onPrice() { *// Ввод уены товара* **if** (**p**.**n** == 0 && **p**.**name** == **null**) {  
 JOptionPane.*showMessageDialog*(**null**, **"Name of products not input"**);  
 }  
 String[] parts = **Text2**.getText().split(**" "**);  
 **for** (**int** i = 0; i < **p**.**n**; i++) {  
 **p**.**price**.add(parts[i]);  
  
 }  
 JOptionPane.*showMessageDialog*(**null**, **"Price of products done = "** + **p**.**price**.toString());  
 }  
  
 **private void** onOutput() **throws** IOException { *// Вывод всех даныхх о товаре в файл* FileWriter writer = **new** FileWriter(**"Lib.txt"**);  
  
 **for** (**int** i = 0; i < **p**.**n**; i++) {  
 writer.write(**p**.**name**.get(i) + **"-"** + **p**.**price**.get(i));  
 writer.write(System.*getProperty*(**"line.separator"**));  
  
 }  
  
 writer.close();  
 JOptionPane.*showMessageDialog*(**null**, **"Your journal is ready"**);  
 }  
  
  
 **private void** onOutputText() **throws** IOException { *// Вывод всех даныхх о товаре в textArea* **textArea1**.setText(**"Name"** + **"\t"** + **"Price"** + **"\n"**);  
 **for** (**int** i = 0; i < **p**.**n**; i++) {  
 **textArea1**.setText(**textArea1**.getText() + **p**.**name**.get(i) + **"\t"** + **p**.**price**.get(i) + **"\n"**);  
 }  
  
 }  
  
  
 {  
*// GUI initializer generated by IntelliJ IDEA GUI Designer  
// >>> IMPORTANT!! <<<  
// DO NOT EDIT OR ADD ANY CODE HERE!* $$$setupUI$$$();  
 }  
  
 */\*\*  
 \* Method generated by IntelliJ IDEA GUI Designer  
 \* >>> IMPORTANT!! <<<  
 \* DO NOT edit this method OR call it in your code!  
 \*  
 \** ***@noinspection*** *ALL  
 \*/* **private void** $$$setupUI$$$() {  
 **root** = **new** JPanel();  
 **root**.setLayout(**new** com.intellij.uiDesigner.core.GridLayoutManager(11, 5, **new** Insets(10, 10, 10, 10), -1, -1));  
 **inputPriceButton** = **new** JButton();  
 **inputPriceButton**.setEnabled(**false**);  
 **inputPriceButton**.setText(**"Input Price"**);  
 **root**.add(**inputPriceButton**, **new** com.intellij.uiDesigner.core.GridConstraints(3, 4, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **inputButton** = **new** JButton();  
 **inputButton**.setEnabled(**false**);  
 **inputButton**.setText(**"Input"**);  
 **root**.add(**inputButton**, **new** com.intellij.uiDesigner.core.GridConstraints(10, 4, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **Text1** = **new** JFormattedTextField();  
 **Text1**.setEnabled(**true**);  
 **root**.add(**Text1**, **new** com.intellij.uiDesigner.core.GridConstraints(9, 1, 1, 3, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_HORIZONTAL, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_WANT\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **new** Dimension(150, 40), **null**, 0, **false**));  
 **workerCheckBox** = **new** JCheckBox();  
 **workerCheckBox**.setEnabled(**true**);  
 **workerCheckBox**.setText(**"Worker"**);  
 **root**.add(**workerCheckBox**, **new** com.intellij.uiDesigner.core.GridConstraints(4, 0, 5, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_WEST, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **Text2** = **new** JFormattedTextField();  
 **root**.add(**Text2**, **new** com.intellij.uiDesigner.core.GridConstraints(3, 1, 1, 3, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_HORIZONTAL, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_WANT\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **new** Dimension(150, 40), **null**, 0, **false**));  
 **final** JLabel label1 = **new** JLabel();  
 label1.setText(**"Input names of all products"**);  
 **root**.add(label1, **new** com.intellij.uiDesigner.core.GridConstraints(5, 1, 3, 2, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **final** JLabel label2 = **new** JLabel();  
 label2.setText(**"Input price of all products"**);  
 **root**.add(label2, **new** com.intellij.uiDesigner.core.GridConstraints(1, 1, 2, 2, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **final** JLabel label3 = **new** JLabel();  
 label3.setText(**"Input count of products"**);  
 **root**.add(label3, **new** com.intellij.uiDesigner.core.GridConstraints(7, 4, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **outputListButton** = **new** JButton();  
 **outputListButton**.setEnabled(**false**);  
 **outputListButton**.setText(**"Output List "**);  
 **root**.add(**outputListButton**, **new** com.intellij.uiDesigner.core.GridConstraints(0, 1, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **directorCheckBox** = **new** JCheckBox();  
 **directorCheckBox**.setEnabled(**false**);  
 **directorCheckBox**.setText(**"Director"**);  
 **root**.add(**directorCheckBox**, **new** com.intellij.uiDesigner.core.GridConstraints(0, 0, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_WEST, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **formattedTextField2** = **new** JFormattedTextField();  
 **root**.add(**formattedTextField2**, **new** com.intellij.uiDesigner.core.GridConstraints(9, 4, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_WANT\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **new** Dimension(150, 40), **null**, 0, **false**));  
 **priceMenCheckBox** = **new** JCheckBox();  
 **priceMenCheckBox**.setEnabled(**false**);  
 **priceMenCheckBox**.setText(**"PriceMen"**);  
 **root**.add(**priceMenCheckBox**, **new** com.intellij.uiDesigner.core.GridConstraints(3, 0, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_WEST, com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 **textArea1** = **new** JTextArea();  
 **root**.add(**textArea1**, **new** com.intellij.uiDesigner.core.GridConstraints(0, 4, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_WEST, com.intellij.uiDesigner.core.GridConstraints.FILL\_VERTICAL, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_WANT\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_WANT\_GROW, **null**, **new** Dimension(150, 50), **null**, 0, **false**));  
 **button1** = **new** JButton();  
 **button1**.setEnabled(**false**);  
 **button1**.setText(**"OutputInTExtArea"**);  
 **root**.add(**button1**, **new** com.intellij.uiDesigner.core.GridConstraints(0, 2, 1, 1, com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER, com.intellij.uiDesigner.core.GridConstraints.FILL\_HORIZONTAL, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW, com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_FIXED, **null**, **null**, **null**, 0, **false**));  
 }  
  
 */\*\*  
 \** ***@noinspection*** *ALL  
 \*/* **public** JComponent $$$getRootComponent$$$() {  
 **return root**;

План тестирования по задачи.

|  |  |  |  |
| --- | --- | --- | --- |
| N | Ввод данных работника | Ввод данных бухгалтера | Вывод информации |
| 5 | Number1 Number2 Number3 Number4 Number5 | 34 23 123 345 23 | Number1-34 Number2-23 Number3-123 Number4-345 Number5-23 |
| 4 | N1  N2  N3  N4 | 23  23 23 23 | N1-23 N2-23 N3-23 N4-23 |
| 3 | Hor Vor Lor | 241 23 12 | Hor-241 Vor-23 Lor-12 |
| 3 | Hor Vor Lor | 241 23 12 | Name Price  Hor 241 Vor 23 Lor 12 |

Снимок экрана, отражающего работу программы по задаче.

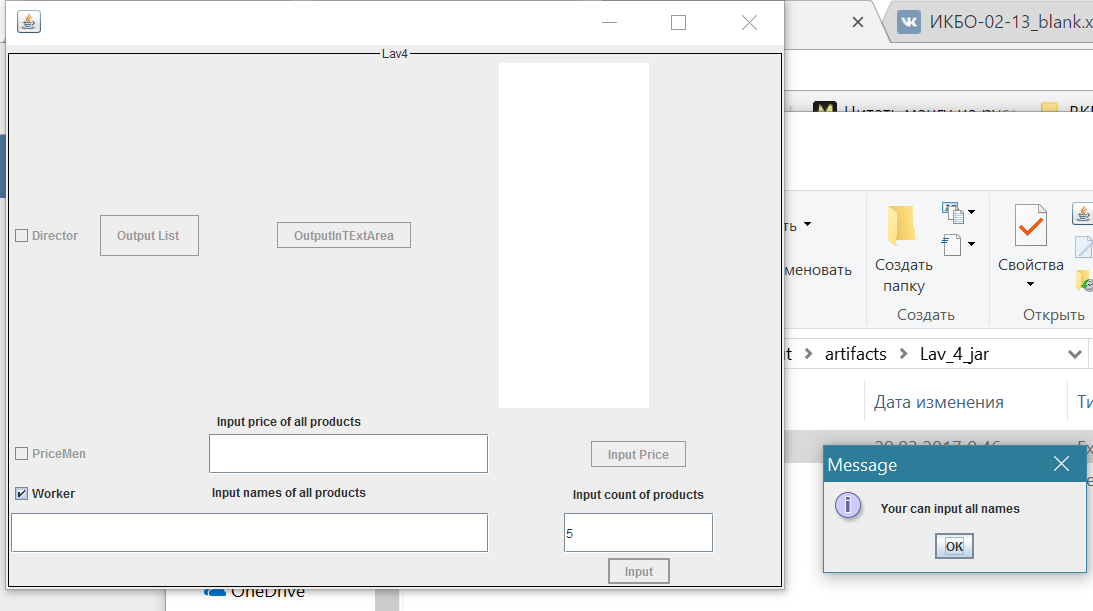


Рис. 1

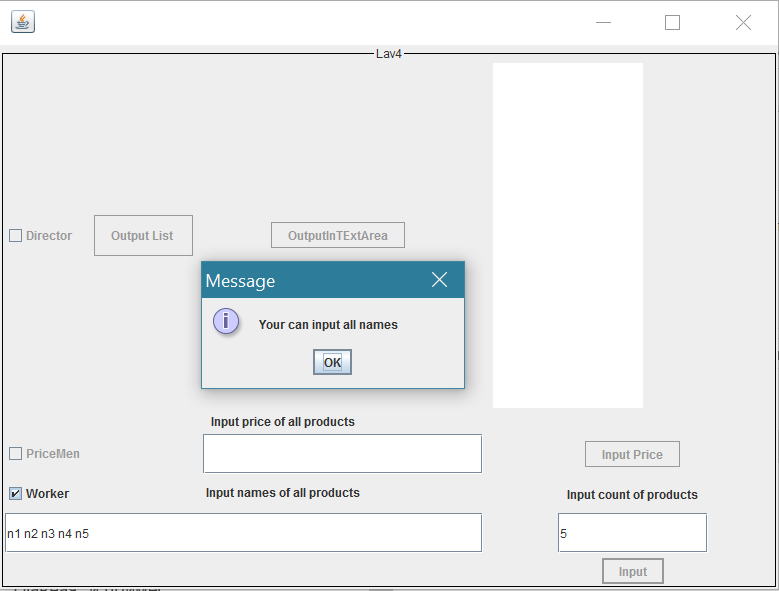


Рис. 2

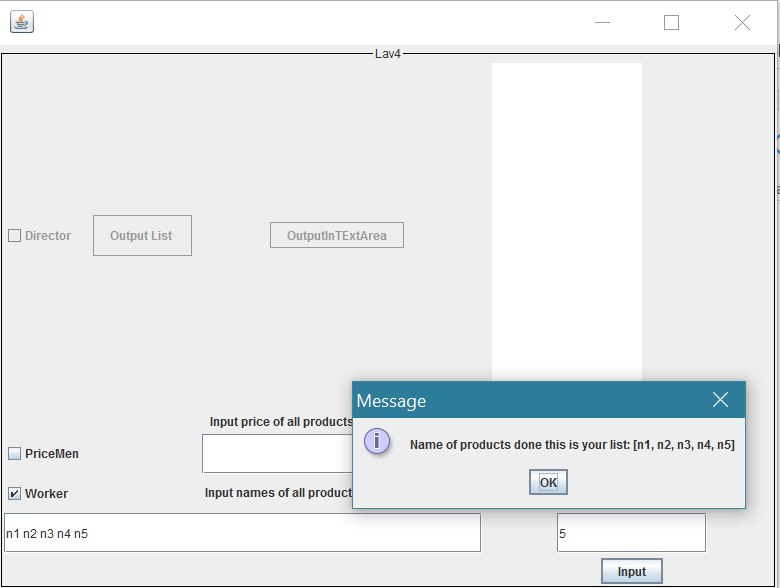


Рис. 3

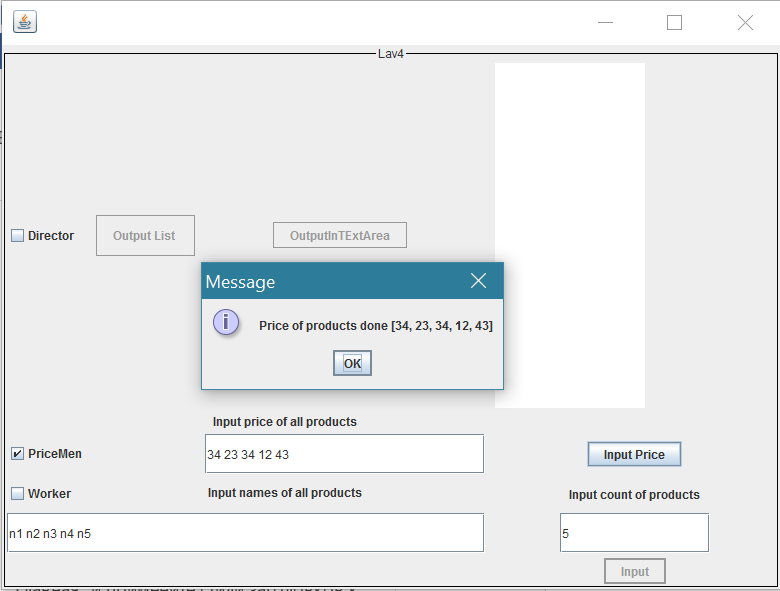


Рис. 4

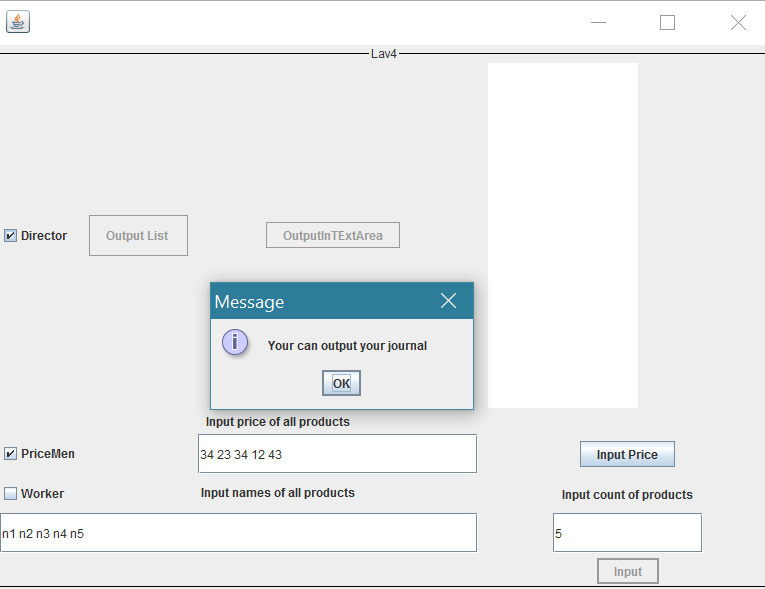


Рис. 5

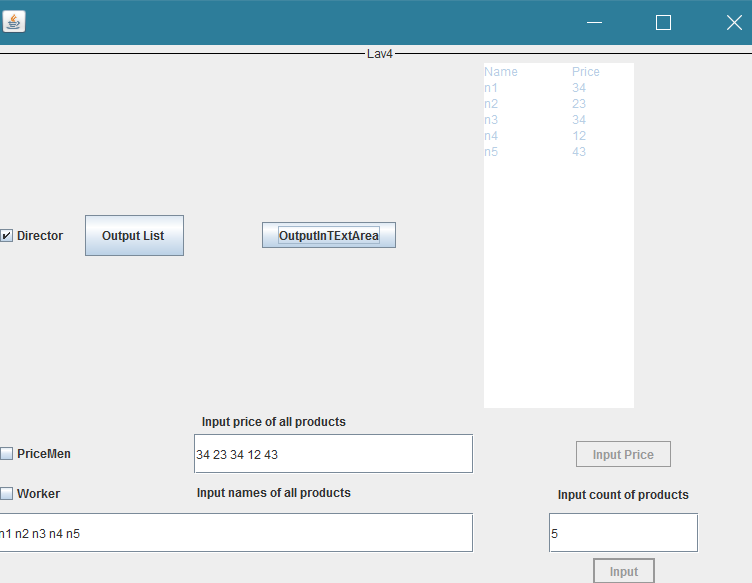


Рис. 6

Вывод: были получены знания по написанию графических приложений на языке java.