МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**КУРСОВАЯ РАБОТА**

по дисциплине «Программирование на языке Java»

на тему «**Научный калькулятор**»

|  |  |
| --- | --- |
| Выполнил:  студент группы 10702120 | Королёв П. И. |
| Руководитель: | Сидорик В. В. |

Минск 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по дисциплине «Программирование на языке Java»

на тему «**Научный калькулятор**»

|  |  |
| --- | --- |
| Выполнил:  студент группы 10702120 | Королёв П. И. |
| Руководитель: | Сидорик В. В. |

Минск 2022

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc121264423)

[Постановка и описание задачи 5](#_Toc121264424)

[1 Теоретический раздел 6](#_Toc121264425)

[2 Практический раздел 9](#_Toc121264429)

[2.1 Построение UML-диаграммы классов 10](#_Toc121264431)

[2.2 Реализация графических окон 17](#_Toc121264432)

[3 Описание структуры проекта 9](#_Toc121264429)

[ЗАКЛЮЧЕНИЕ 22](#_Toc121264435)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 23](#_Toc121264436)

[ПРИЛОЖЕНИЕ А 24](#_Toc121264437)

# ВВЕДЕНИЕ

Java — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle.

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года. Занимает высокие места в рейтингах популярности языков программирования (2-е место в рейтингах IEEE Spectrum (2020) и TIOBE (2021)).

На сегодняшний день платформа Java привлекла более 9 миллионов разработчиков программного обеспечения. Она используется во всех главных сегментах индустрии, а также в широком диапазоне устройств, компьютеров и сетей.

Универсальность, эффективность, портативность платформ и безопасность технологии Java делают эту технологию идеальным выбором для сетевых вычислений. От портативных компьютеров до центров сбора данных, от игровых консолей до суперкомпьютеров, используемых для научных разработок, от сотовых телефонов до сети Интернет - Java используется повсюду!

• Java используется на 1.1 млрд. настольных ПК;

• Следующие успешные проекты реализованы с привлечением Java (J2EE) технологий: RuneScape, Amazon, eBay, Yandex, LinkedIn;

• Ежегодно регистрируется 930 миллионов загрузок среды выполнения Java;

• Java используется в 3 млрд. мобильных телефонов;

• Java используется в 100% всех проигрывателей Blu-ray;

• Ежегодно выпускается 1.4 млрд. платформ Java Card;

• На основе технологий Java работают приставки, принтеры, веб-камеры, игры, навигационные системы для автомобилей, терминалы для проведения лотерей, медицинские устройства, автоматы для оплаты парковки и многое другое.

# Постановка и описание задачи

Цель программы: разработать приложение на языке программирования Java и реализовать функционал в соответствии с бизнес-логикой выбранной темы. Программный код документировать. Сформировать с помощью утилиты javadoc html-файл, документирующий проект. Приложение скомпоновать в единый jar-файл.

Детализация: написать программу, реализующая принцип работы научного калькулятора.

Технологии, используемые в работе:

* Объектно-ориентированный язык программирования Java;
* Платформа для создания приложений с графическим интерфейсом Swing;
* Операционная система Windows 10;
* Среда разработки Intelij IDEA 2022.2.3.

Исходные данные: данные, вводимые в основном окне.

Выходные данные: результат решения примера.

# Теоретический раздел

Для создания графического интерфейса была выбрана библиотека Swing. Swing предоставляет более гибкие интерфейсные компоненты, чем более ранняя библиотека AWT. Для отрисовки используется 2D, что приносит с собой сразу несколько преимуществ. Набор стандартных компонентов значительно превосходит AWT по разнообразию и функциональности. Swing позволяет легко создавать новые компоненты, наследуясь от существующих, и поддерживает различные стили и скины. Важнейшим отличием Swing от AWT является то, что компоненты Swing вообще не связаны с операционной системой и поэтому гораздо более стабильны и быстры. Такие компоненты в Java называются легковесными (lightweight), и понимание основных принципов их работы во многом объяснит работу Swing. Для создания графического интерфейса приложения необходимо использовать специальные компоненты библиотеки Swing, называемые контейнерами высшего уровня (top level containers). Они представляют собой окна операционной системы, в которых размещаются компоненты пользовательского интерфейса. К контейнерам высшего уровня относятся окна JFrame и JWindow, диалоговое окно JDialog, а также апплет JApplet (который не является окном, но тоже предназначен для вывода интерфейса в браузере, запускающем этот апплет). Контейнеры высшего уровня Swing представляют собой тяжеловесные компоненты и являются исключением из общего правила. Все остальные компоненты Swing являются легковесными.

# Практический раздел

## Построение UML диаграммы классов

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

При решении сложных задач моделирование значительно упрощают программирование. Также стоит отметить, что вносить изменения в диаграммы классов легче, чем в исходный код.

Таким образом, первым шагом к реализации курсового проекта были построены UML диаграммы (см. рисунок 2.1).

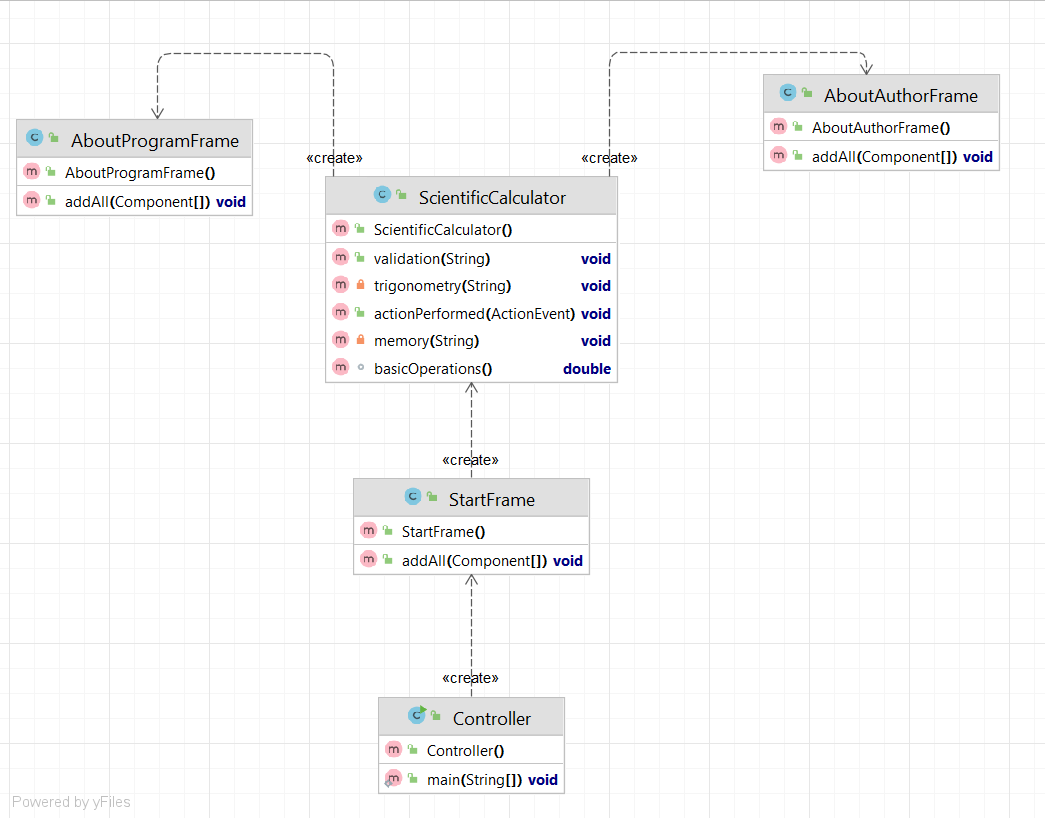


Рисунок 2.1– UML диаграмма окон и контроллера

## Построение блок-схемы алгоритма

Наглядным способом представления алгоритмической последовательности является блочная схема. Она выглядит как последовательность функциональных блоков, связанных между собой. Каждый из этих элементов отвечает за выполнение одного либо нескольких действий. Так как возможны разные действия, то каждому типу действия соответствует геометрическая фигура.

Очередность выполнения действий определяется линиями, которые соединяют блочные элементы с данными. Принято, чтобы компоненты схемы соединялись слева направо и сверху вниз. В случае наличия какой-либо иной последовательности, используют линии направленного вида (со стрелками).

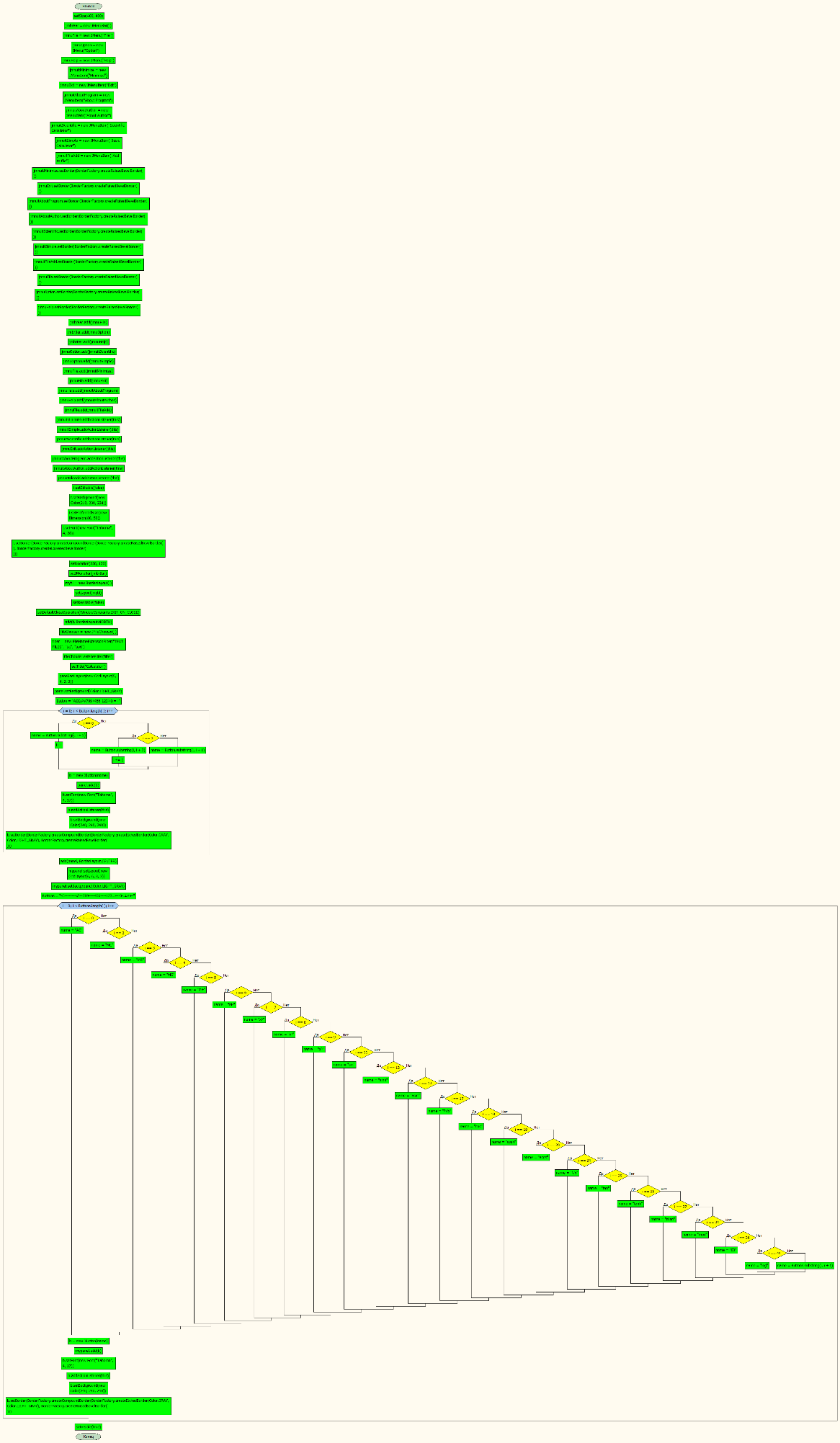


Рисунок 2.2 – Блок-схема класса ScientificCalculator

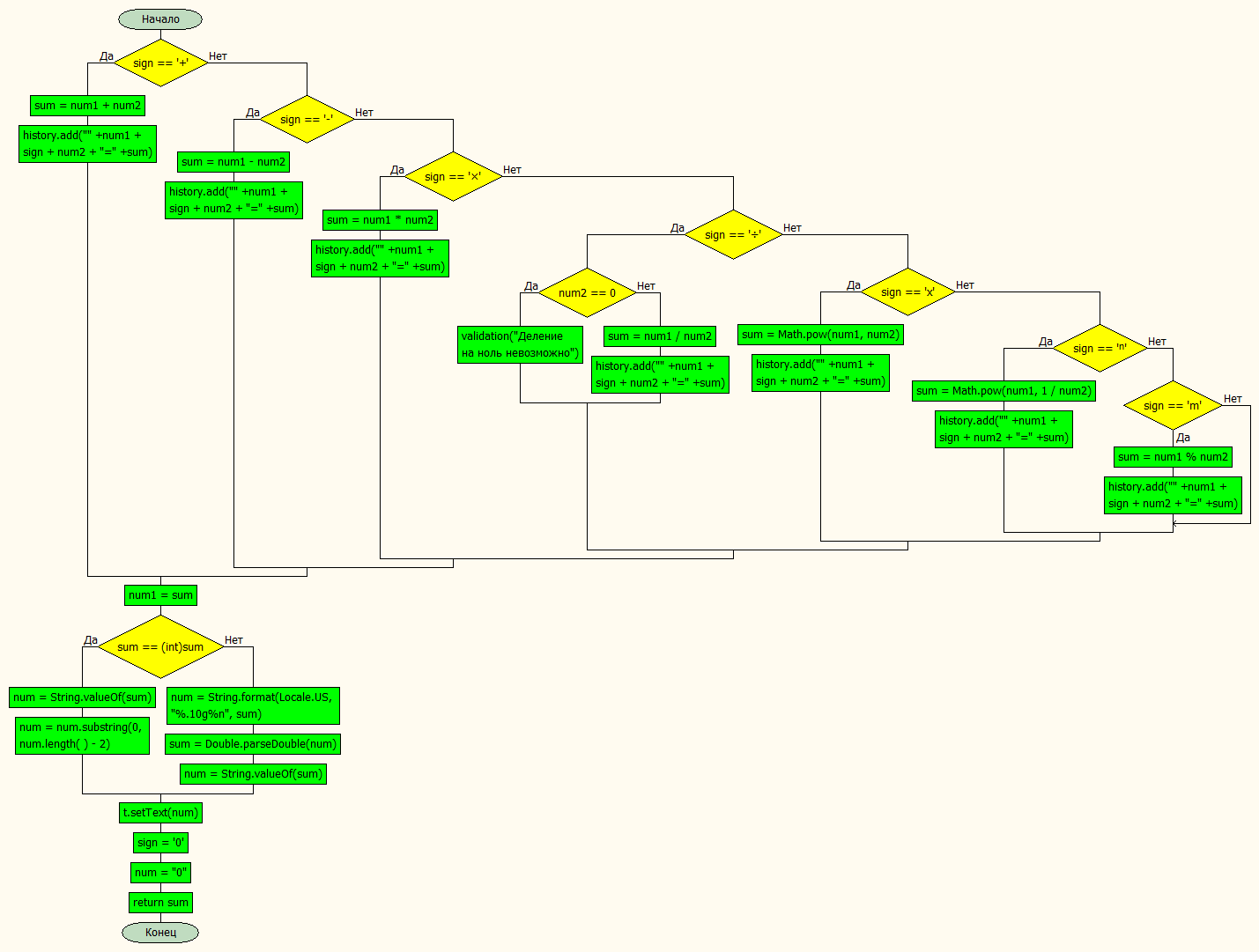


Рисунок 2.3 – Блок-схема класса ScientificCalculator

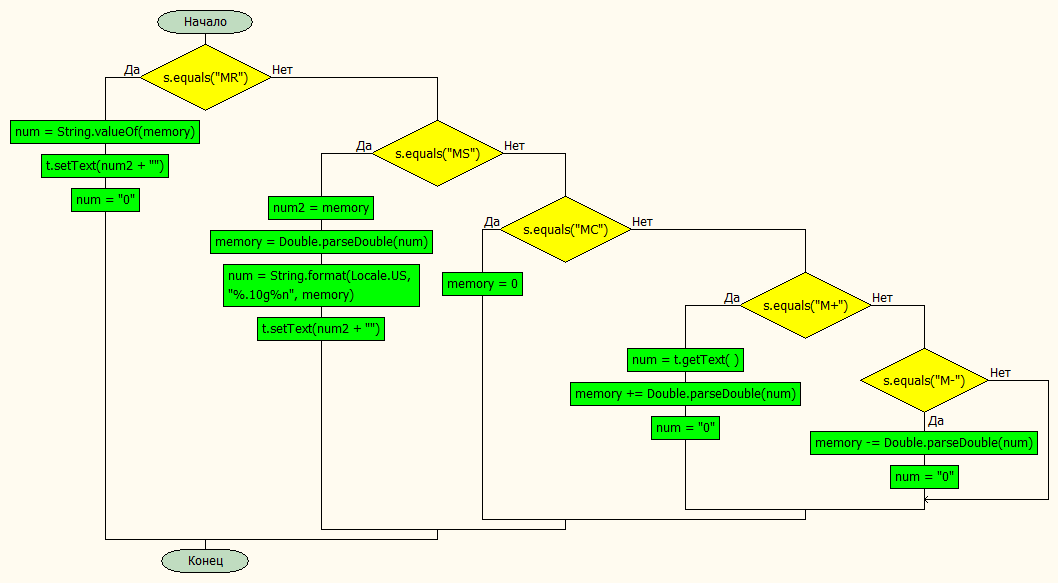


Рисунок 2.4 – Блок-схема класса ScientificCalculator

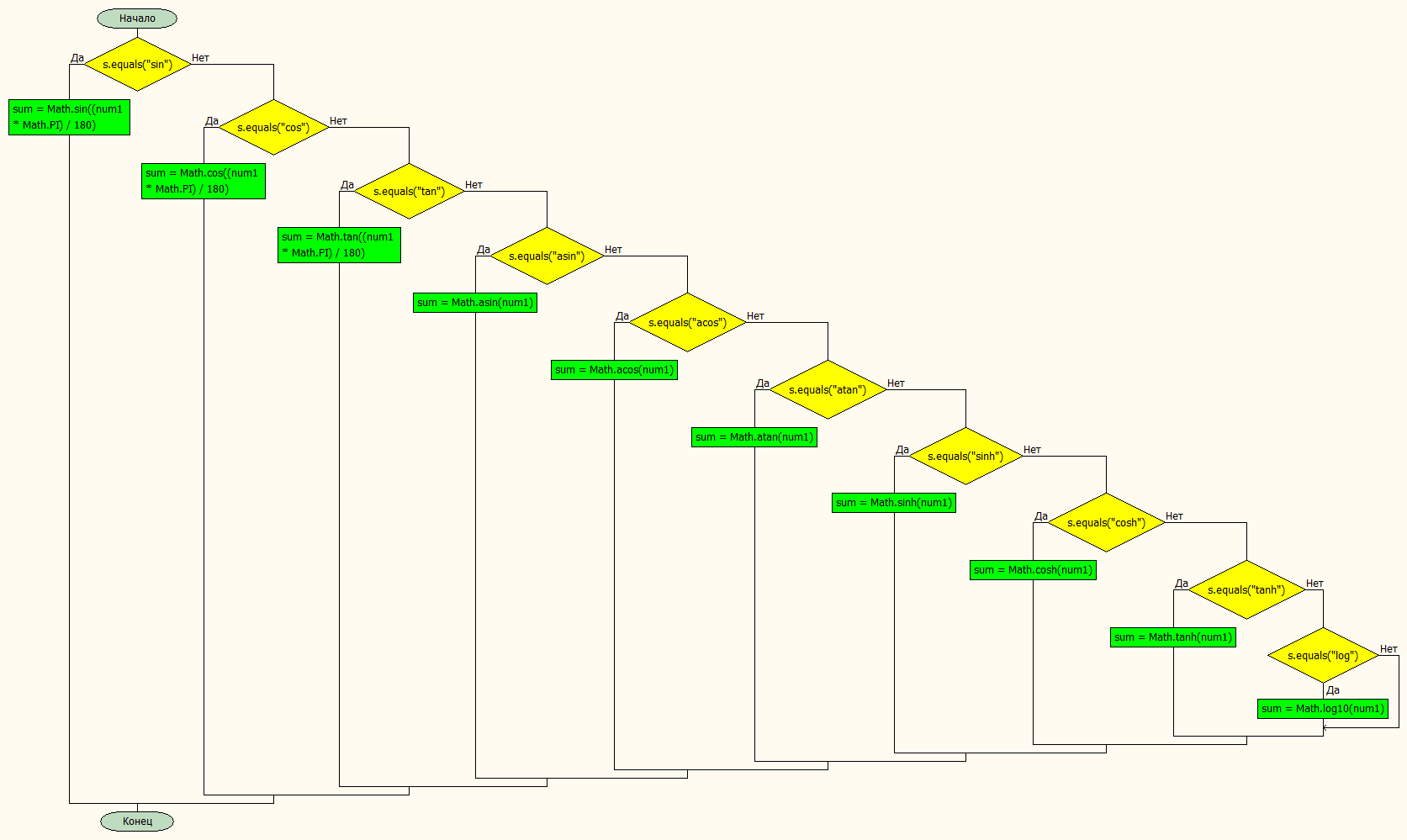


Рисунок 2.5 – Блок-схема класса ScientificCalculator

## Реализация графических окон

Приложение состоит из 5 окон взаимосвязанных между собой: Splash Screen (появляется на 1 секунду при запуске приложения), Main Window (основное окно для работы с приложением), About program (о программе), About Author (об авторе), Help (помощь).

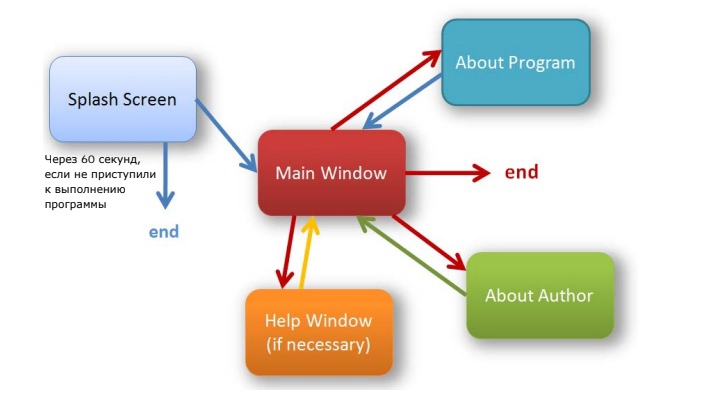


Рисунок 2.6 – Бизнес-логика переключения графических окон

**Стартовое окно**

Данное окно является приветственным, содержит элементы управления для взаимодействия с пользователем. Оно отображается, пока пользователь не нажмет кнопку “Далее” в течение 60 секунд, иначе оно автоматически закроется. Пользователь может выйти из приложения при помощи кнопки “Выход”.

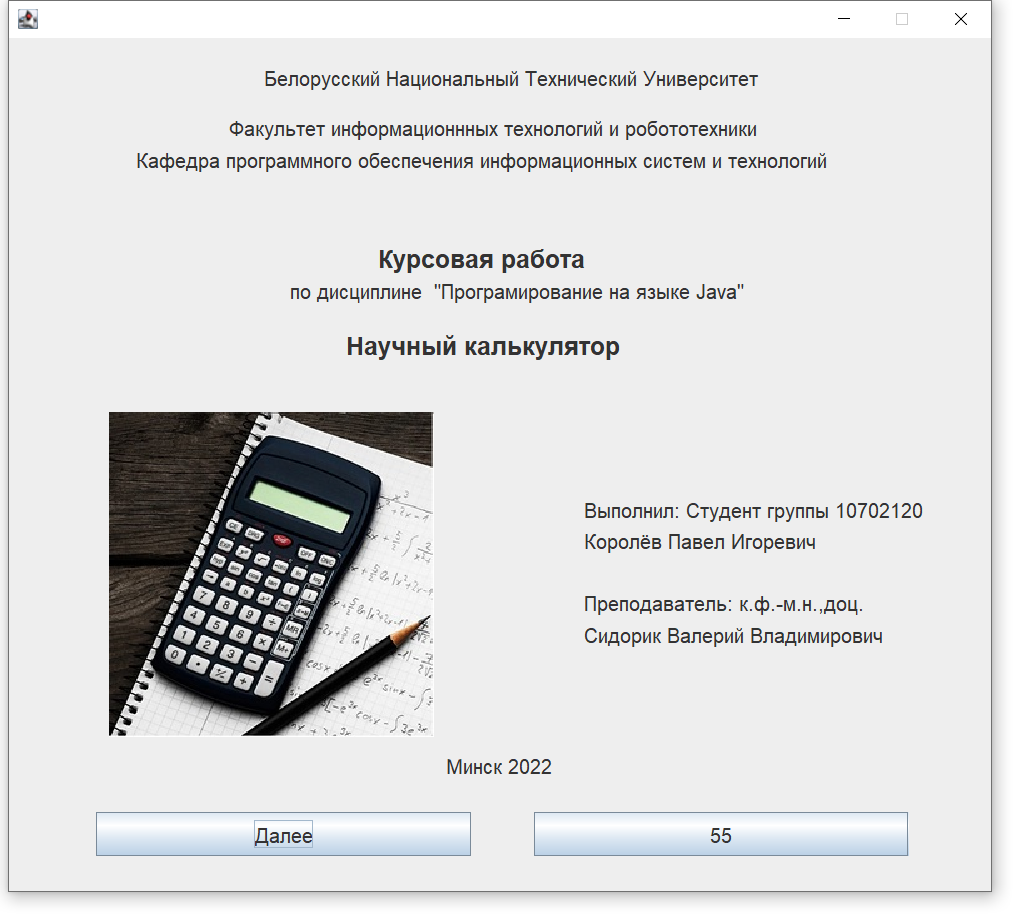


Рисунок 2.7 – Бизнес-логика переключения графических окон

**Главное окно**

На главном окне находятся элементы, для взаимодействия с вводом пользователя. Так же есть 3 меню, в которых можно выбрать режим работы калькулятора, вкладки «О программе», «Об авторе», «Добавить в файл», «Минимизировать», «Выход».

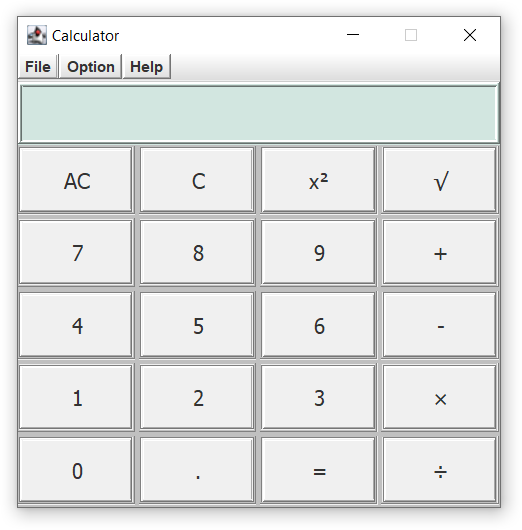


Рисунок 2.8 – Вид главного окна приложения

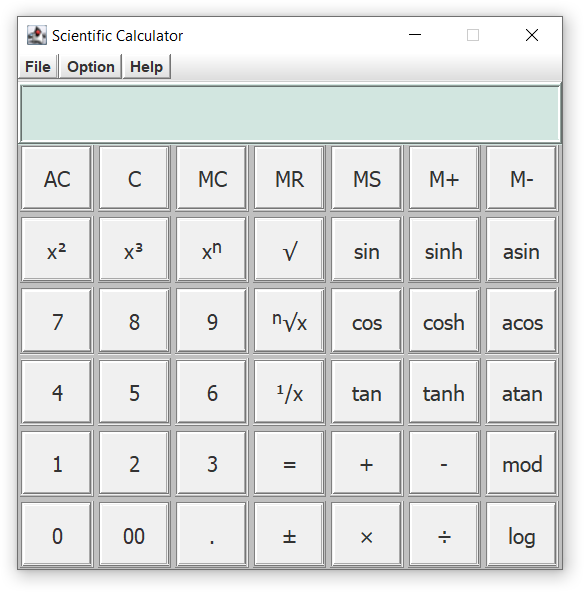


Рисунок 2.9 – Вид главного окна приложения

**Главное окно с решением**

На окне выводиться ответ решения примера.

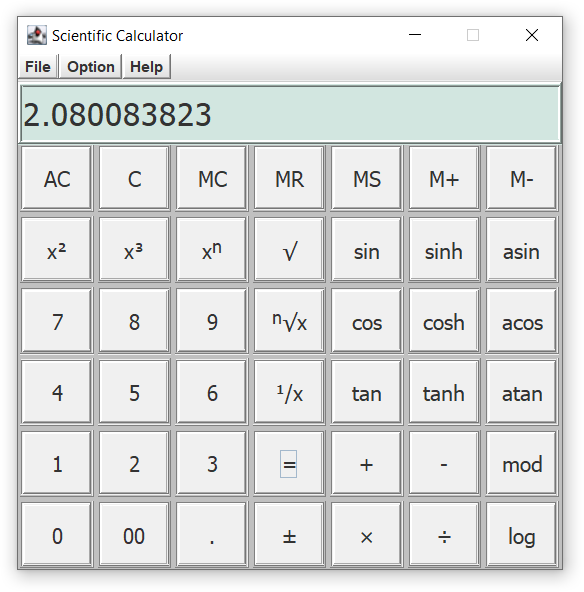


Рисунок 2.10 – Вид окна решения примера

**Об авторе**

Окно об авторе содержит основную информацию об авторе проекта: номер группы в университете, фамилию, имя, отчество, электронную почту, фотографию.

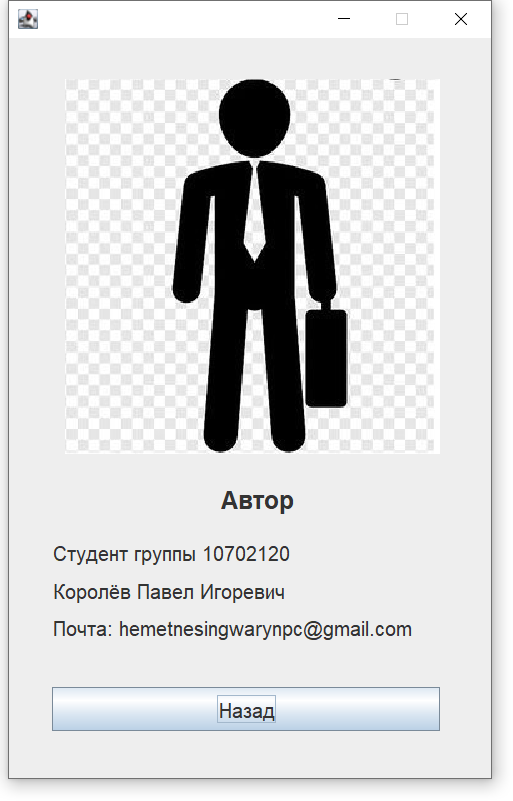


Рисунок 2.11 – Окно об авторе

**О программе**

Окно о программе отображает краткую информацию о приложении. Содержит название приложения, возможности приложения и краткое руководство.

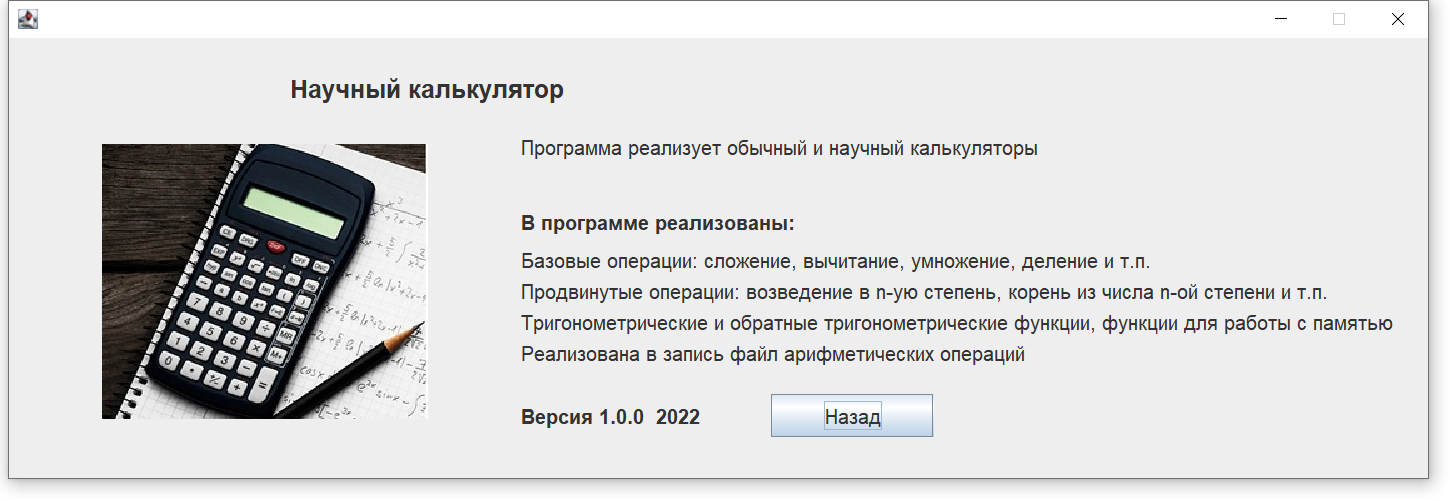
****

Рисунок 2.12 – Окно о программе

# Описание структуры программы

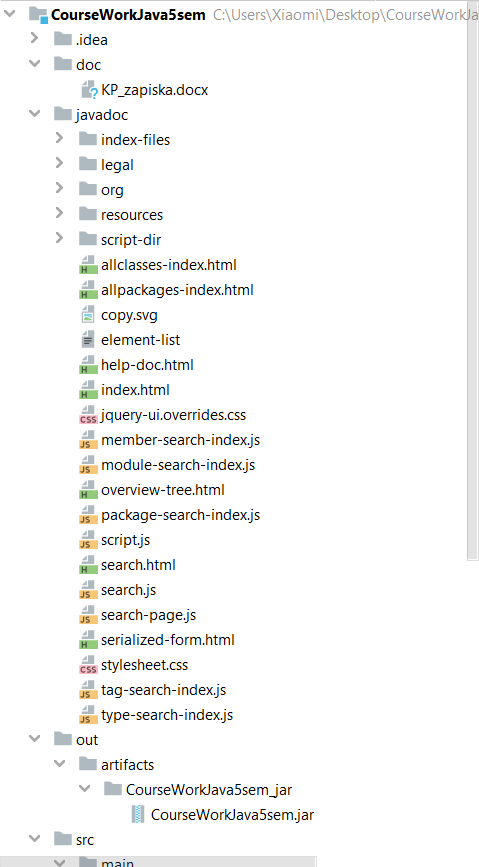
Структура проекта была спроектирована в соответствии с требованиями к курсовому проекту.

Основные папки проекта:

1. uml – папка, в которой находится UML-диаграмма;
2. doc – папка, в которой находится пояснительная записка в формате docx;
3. zip – папка, в которой находится весь проект сжатый в формате zip;
4. Java – папка со всеми файлами проекта Java.

В папке со всеми файлами проекта Java следующие основные папки:

1. scr – папка с файлами источниками .java;
2. out – папка, в которой находится исполняемый файл;
3. javadoc – папка с файлами Javadoc.



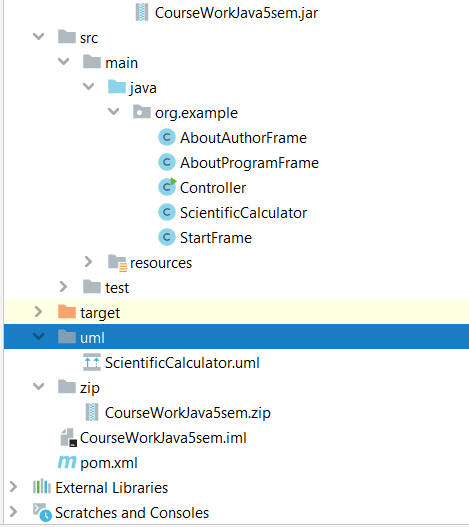


Рисунок 3.1 – структура папок в проекте

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте был реализован науный калькулятор.

В результате выполнения курсовой работы было разработано приложение, написанное на языке программирования Java.

В процессе выполнения курсовой работы был получен практический опыт работы с библиотекой Swing и разработки многооконного приложения, изучено взаимодействие и обмен информацией между окнами приложения.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Шилдт, Г. Java: руководство для начинающих, 7-е изд. / Г. Шилдт. – СПб : ООО «Диалектика», 2019. – 816 с.
2. Эккель, Б. Философия Java. 4-е полное изд. / Б. Эккель. – СПб : Питер, 2015. – 1168 с.
3. Буч, Гради Введение в UML от создателей языка / Гради Буч , Джеймс Рамбо , Ивар Якобсон. - М.: ДМК Пресс, 2015. - 496 c.
4. Классы в языке программирования Java : учебно-методическое пособие для студентов и слушателей системы повышения квалификации и переподготовки / О. М. Болбот, В. В. Сидорик ; под общ. ред. В. В. Сидорика. – Минск : БНТУ, 2020. – 76 с.
5. Сидорик, В. В. Методические указания и требования к оформлению, представлению и защите курсовой работы по дисциплине «Программирование на языке Java» / В. В. Сидорик. – Минск : БНТУ, 2021. – 9 с.
6. ГОСТ 19.701-90 Единая система программной документации. Схемы алгоритмов, программ данных и систем. Условные обозначения и правила выполнения. – Москва: Государственный стандарт союза ССР. – 35 с.
7. ГОСТ 7.1–2003. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – Минск: Госстандарт Респ. Беларусь, 2001. – 52 с.
8. Swing — java-online [Электронный ресурс]. – Режим доступа: https://java-online.ru/libs-swing.xhtml.

# ПРИЛОЖЕНИЕ А

### Файл Controller.java

**package** org.example;  
*/\*\*  
 \** ***@author*** *Koroliov P.I.  
 \** ***@version*** *1.0  
 \*  
 \* Class that run StartFrame.  
 \*/***public class** Controller {  
 **public static void** main(String args[]) {  
 **new** StartFrame();  
 }  
}

### Файл ScientificCalculator.java

**package** course.project;  
  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
**import** java.util.Locale;  
**import** javax.swing.\*;  
**import** javax.swing.filechooser.FileNameExtensionFilter;  
**import** java.util.List;  
*/\*\*  
 \** ***@author*** *Koroliov P.I.  
 \** ***@version*** *1.0  
 \*  
 \* Class for finding the answer.  
 \*/***public class** ScientificCalculator **extends** JFrame **implements** ActionListener {  
 **private int flag** = 0;  
 **private char sign**;  
 **private** String **name** = **"0"**;  
 **private** String **num** = **"0"**;  
 **private double sum** = 0;  
 **private double num1** = 0;  
 **private double num2** = 0;  
 **private double memory** = 0;  
 */\*\*  
 \* Динамический массив для хранения истории операций.  
 \*/* List<String> **history** = **new** ArrayList<String>();  
  
 JTextField **t** = **new** JTextField();  
 JPanel **mypanel** = **new** JPanel();  
 JPanel **panel** = **new** JPanel();  
  
 JFrame **frame** = **this**;  
 JFileChooser **fileChooser**;  
 */\*\*  
 \* Set properties of Calculator.  
 \*/* **public** ScientificCalculator() {  
 *//Menubar: ==================* setSize(400, 400);  
 JMenuBar jmbrBar = **new** JMenuBar();  
 JMenu jmnuFile = **new** JMenu(**"File"**);  
 JMenu jmnuOption = **new** JMenu(**"Option"**);  
 JMenu jmnuHelp = **new** JMenu(**"Help"**);  
 JMenuItem jmnuitMinimize = **new** JMenuItem(**"Minimize"**);  
 JMenuItem jmnuExit = **new** JMenuItem(**"Exit"**);  
 JMenuItem jmnuitAboutProgram = **new** JMenuItem(**"About Program"**);  
 JMenuItem jmnuitAboutAuthor = **new** JMenuItem(**"About Author"**);  
 JMenuItem jmnuitScientific = **new** JMenuItem(**"Scientific Calculator"**);  
 JMenuItem jmnuitSimple = **new** JMenuItem(**"Basic Calculator"**);  
 JMenuItem jmnuitFileAdd = **new** JMenuItem(**"Add to file"**);  
  
 jmnuitMinimize.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuExit.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuitAboutProgram.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuitAboutAuthor.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuitScientific.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuitSimple.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuitFileAdd.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuFile.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuOption.setBorder(BorderFactory.*createRaisedBevelBorder*());  
 jmnuHelp.setBorder(BorderFactory.*createRaisedBevelBorder*());  
  
 jmbrBar.add(jmnuFile);  
 jmbrBar.add(jmnuOption);  
 jmbrBar.add(jmnuHelp);  
 jmnuOption.add(jmnuitScientific);  
 jmnuOption.add(jmnuitSimple);  
 jmnuFile.add(jmnuitMinimize);  
 jmnuFile.add(jmnuExit);  
 jmnuHelp.add(jmnuitAboutProgram);  
 jmnuHelp.add(jmnuitAboutAuthor);  
 jmnuFile.add(jmnuitFileAdd);  
  
 jmnuitMinimize.addActionListener(**this**);  
 jmnuitSimple.addActionListener(**this**);  
 jmnuitScientific.addActionListener(**this**);  
 jmnuExit.addActionListener(**this**);  
 jmnuitAboutProgram.addActionListener(**this**);  
 jmnuitAboutAuthor.addActionListener(**this**);  
 jmnuitFileAdd.addActionListener(**this**);  
  
 *//TextField: ===============================* **t**.setEditable(**false**);  
 **t**.setBackground(**new** Color(210, 230, 224));  
 **t**.setPreferredSize(**new** Dimension(80, 50));  
 **t**.setFont(**new** Font(**"Tahoma"**, 4, 26));  
 **t**.setBorder(BorderFactory.*createCompoundBorder*(BorderFactory.*createRaisedBevelBorder*(), BorderFactory.*createLoweredBevelBorder*()));  
  
 *// ============================* setLocation(350, 150);  
 setJMenuBar(jmbrBar);  
 BorderLayout mybl = **new** BorderLayout();  
 setLayout(mybl);  
 setResizable(**false**);  
 setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 add(**t**, BorderLayout.***NORTH***);  
  
 **fileChooser** = **new** JFileChooser();  
 FileNameExtensionFilter filter = **new** FileNameExtensionFilter(**"TEXT FILES"**, **"txt"**, **"text"**);  
 **fileChooser**.setFileFilter(filter);  
  
 setTitle(**"Calculator"**);  
 *//Panel: ================* **panel**.setLayout(**new** GridLayout(5, 4, 3, 3));  
 **panel**.setBackground(Color.***LIGHT\_GRAY***);  
 String Button = **"ACCx²√789+456-123×0.=÷"**;  
 **int** i;  
 **for** (i = 0; i < Button.length(); i++) {  
 **if** (i == 0) {  
 **name** = Button.substring(i, i + 2);  
 i++;  
 } **else if** (i == 3) {  
 **name** = Button.substring(i, i + 2);  
 i += 1;  
 } **else** {  
 **name** = Button.substring(i, i + 1);  
 }  
 JButton b = **new** JButton(**name**);  
 **panel**.add(b);  
 b.setFont(**new** Font(**"Tahoma"**, 4, 17));  
 b.addActionListener(**this**);  
 b.setBackground(**new** Color(240, 240, 240));  
 b.setBorder(BorderFactory.*createCompoundBorder*(BorderFactory.*createEtchedBorder*(Color.***GRAY***, Color.***LIGHT\_GRAY***), BorderFactory.*createRaisedBevelBorder*()));  
 }  
 add(**panel**, BorderLayout.***CENTER***);  
  
 *//MyPanel: =====================* **mypanel**.setLayout(**new** GridLayout(6, 6, 3, 3));  
 **mypanel**.setBackground(Color.***LIGHT\_GRAY***);  
 String Buttons = **"-C--------√---789----456----123=+--0-.±×÷-"**;  
 **for** (i = 0; i < Buttons.length(); i++) {  
 **if** (i == 0)  
 **name** = **"AC"**;  
 **else if** (i == 2)  
 **name** = **"MC"**;  
 **else if** (i == 3)  
 **name** = **"MR"**;  
 **else if** (i == 4)  
 **name** = **"MS"**;  
 **else if** (i == 5)  
 **name** = **"M+"**;  
 **else if** (i == 6)  
 **name** = **"M-"**;  
 **else if** (i == 7)  
 **name** = **"x²"**;  
 **else if** (i == 8)  
 **name** = **"x³"**;  
 **else if** (i == 9)  
 **name** = **"xⁿ"**;  
 **else if** (i == 11)  
 **name** = **"sin"**;  
 **else if** (i == 12)  
 **name** = **"sinh"**;  
 **else if** (i == 13)  
 **name** = **"asin"**;  
 **else if** (i == 17)  
 **name** = **"ⁿ√x"**;  
 **else if** (i == 18)  
 **name** = **"cos"**;  
 **else if** (i == 19)  
 **name** = **"cosh"**;  
 **else if** (i == 20)  
 **name** = **"acos"**;  
 **else if** (i == 24)  
 **name** = **"¹/x"**;  
 **else if** (i == 25)  
 **name** = **"tan"**;  
 **else if** (i == 26)  
 **name** = **"tanh"**;  
 **else if** (i == 27)  
 **name** = **"atan"**;  
 **else if** (i == 34)  
 **name** = **"mod"**;  
 **else if** (i == 36)  
 **name** = **"00"**;  
 **else if** (i == 41)  
 **name** = **"log"**;  
 **else  
 name** = Buttons.substring(i, i + 1);  
 JButton b = **new** JButton(**name**);  
 **mypanel**.add(b);  
 b.setFont(**new** Font(**"Tahoma"**, 4, 17));  
 b.addActionListener(**this**);  
 b.setBackground(**new** Color(240, 240, 240));  
 b.setBorder(BorderFactory.*createCompoundBorder*(BorderFactory.*createEtchedBorder*(Color.***GRAY***, Color.***LIGHT\_GRAY***), BorderFactory.*createRaisedBevelBorder*()));  
 }  
 setVisible(**true**);  
 }  
 */\*  
 \* Method for messaging about Exception  
 \* @param String messages - message about exception  
 \*/* **public void** validation(String messages) {  
 JOptionPane.*showMessageDialog*(**null**, messages);  
 }  
  
  
 **private void** trigonometry(String s) {  
 **if** (s.equals(**"sin"**))  
 **sum** = Math.*sin*((**num1** \* Math.***PI***) / 180);  
 **else if** (s.equals(**"cos"**))  
 **sum** = Math.*cos*((**num1** \* Math.***PI***) / 180);  
 **else if** (s.equals(**"tan"**))  
 **sum** = Math.*tan*((**num1** \* Math.***PI***) / 180);  
 **else if** (s.equals(**"asin"**))  
 **sum** = Math.*asin*(**num1**);  
 **else if** (s.equals(**"acos"**))  
 **sum** = Math.*acos*(**num1**);  
 **else if** (s.equals(**"atan"**))  
 **sum** = Math.*atan*(**num1**);  
 **else if** (s.equals(**"sinh"**))  
 **sum** = Math.*sinh*(**num1**);  
 **else if** (s.equals(**"cosh"**))  
 **sum** = Math.*cosh*(**num1**);  
 **else if** (s.equals(**"tanh"**))  
 **sum** = Math.*tanh*(**num1**);  
 **else if** (s.equals(**"log"**))  
 **sum** = Math.*log10*(**num1**);  
 }  
  
 **private void** memory(String s) {  
 **if** (s.equals(**"MR"**)) {  
 **num** = String.*valueOf*(**memory**);  
 **t**.setText(**num2** + **""**);  
 **num** = **"0"**;  
 } **else if** (s.equals(**"MS"**)) {  
 **num2** = **memory**;  
 *//num=t.getText();* **memory** = Double.*parseDouble*(**num**);  
 **num** = String.*format*(Locale.***US***, **"%.10g%n"**, **memory**);  
 **t**.setText(**num2** + **""**);  
 } **else if** (s.equals(**"MC"**)) {  
 **memory** = 0;  
 } **else if** (s.equals(**"M+"**)) {  
 **num** = **t**.getText();  
 **memory** += Double.*parseDouble*(**num**);  
 **num** = **"0"**;  
 } **else if** (s.equals(**"M-"**)) {  
 *//num = t.getText();* **memory** -= Double.*parseDouble*(**num**);  
 **num** = **"0"**;  
 }  
 }  
 */\*\*  
 \* Method for basic operations  
 \*/* **double** basicOperations() {  
 **if** (**sign** == **'+'**) {  
 **sum** = **num1** + **num2**;  
 **history**.add(**""** + **num1** + **sign** + **num2** + **"="** + **sum**);  
 } **else if** (**sign** == **'-'**) {  
 **sum** = **num1** - **num2**;  
 **history**.add(**""** + **num1** + **sign** + **num2** + **"="** + **sum**);  
 } **else if** (**sign** == **'×'**) {  
 **sum** = **num1** \* **num2**;  
 **history**.add(**""** + **num1** + **sign** + **num2** + **"="** + **sum**);  
 } **else if** (**sign** == **'÷'**) {  
 **if** (**num2** == 0) {  
 validation(**"Деление на ноль невозможно"**);  
 } **else** {  
 **sum** = **num1** / **num2**;  
 **history**.add(**""** + **num1** + **sign** + **num2** + **"="** + **sum**);  
 }  
 } **else if** (**sign** == **'x'**) {  
 **sum** = Math.*pow*(**num1**, **num2**);  
 **history**.add(**""** + **num1** + **sign** + **num2** + **"="** + **sum**);  
 } **else if** (**sign** == **'ⁿ'**) {  
 **sum** = Math.*pow*(**num1**, 1 / **num2**);  
 **history**.add(**""** + **num1** + **sign** + **num2** + **"="** + **sum**);  
 } **else if** (**sign** == **'m'**) {  
 **sum** = **num1** % **num2**;  
 **history**.add(**""** + **num1** + **sign** + **num2** + **"="** + **sum**);  
 }  
 **num1** = **sum**;  
 **if** (**sum** == (**int**) **sum**) {  
 **num** = String.*valueOf*(**sum**);  
 **num** = **num**.substring(0, **num**.length() - 2);  
 } **else** {  
 **num** = String.*format*(Locale.***US***, **"%.10g%n"**, **sum**);  
 **sum** = Double.*parseDouble*(**num**);  
 **num** = String.*valueOf*(**sum**);  
 }  
 **t**.setText(**num**);  
 **sign** = **'0'**;  
 **num** = **"0"**;  
 **return sum**;  
  
 }  
 */\*\*  
 \* Переопределяем метод интерфейса слушателя.  
 \*/* **public void** actionPerformed(ActionEvent e) {  
 String s = e.getActionCommand();  
 **char** ch = s.charAt(0);  
 **if** (s.equals(**"Exit"**)) {  
 System.*exit*(0);  
 } **else if** (s.equals(**"Minimize"**)) {  
 setState(JFrame.***ICONIFIED***);  
 */\*\*  
 \* Реализация записи в файл.  
 \*/* } **else if** (s.equals(**"Add to file"**)) {  
 **if** (**fileChooser**.showSaveDialog(**frame**) == JFileChooser.***APPROVE\_OPTION***) {  
 **try** {  
 FileWriter fileWriter = **new** FileWriter(**fileChooser**.getSelectedFile());  
 **for** (**int** i = 0; i < **history**.size(); i++) {  
 fileWriter.write(**history**.get(i) + **"\n"**);  
 }  
 fileWriter.close();  
 } **catch** (IOException ex) {  
 **throw new** RuntimeException(ex);  
 }  
 }  
 } **else if** (s.equals(**"About Program"**)) {  
 **new** AboutProgramFrame();  
 } **else if** (s.equals(**"About Author"**)) {  
 **new** AboutAuthorFrame();  
 } **else if** (s.equals(**"Scientific Calculator"**)) {  
 setBounds(getLocation().**x**, getLocation().**y**, 360, 320);  
 remove(**panel**);  
 setTitle(**"Scientific Calculator"**);  
 add(**mypanel**, BorderLayout.***CENTER***);  
 } **else if** (s.equals(**"Basic Calculator"**)) {  
 setBounds(getLocation().**x**, getLocation().**y**, 220, 290);  
 remove(**mypanel**);  
 setTitle(**"Calculator"**);  
 add(**panel**, BorderLayout.***CENTER***);  
 } **else if** (s.charAt(0) == **'M'**) {  
 memory(s);  
 } **else if** (s.equals(**"x²"**) || ch == **'√'** || s.equals(**"x³"**) || ch == **'¹'** || s.equals(**"sin"**) || s.equals(**"cos"**) || s.equals(**"tan"**) || s.equals(**"sinh"**) || s.equals(**"cosh"**) || s.equals(**"tanh"**) || s.equals(**"asin"**) || s.equals(**"acos"**) || s.equals(**"atan"**) || s.equals(**"log"**)) {  
 **num** = **t**.getText();  
 **num1** = Double.*parseDouble*(**num**);  
 **if** (s.equals(**"x²"**))  
 **sum** = Math.*pow*(**num1**, 2);  
 **else if** (ch == **'¹'**)  
 **sum** = 1 / **num1**;  
 **else if** (s.equals(**"x³"**))  
 **sum** = Math.*pow*(**num1**, 3);  
 **else if** (s.charAt(0) == **'c'** || s.charAt(0) == **'t'** || s.charAt(0) == **'s'** || s.charAt(0) == **'a'** || s.charAt(0) == **'l'**) {  
 trigonometry(s);  
 } **else if** (ch == **'√'**)  
 **if** (**num1** > 0) **sum** = Math.*sqrt*(**num1**);  
 **else** validation(**"Неверный ввод"**);  
 **flag** = 0;  
 **if** (**sum** == (**int**) **sum**) {  
 **num** = String.*valueOf*(**sum**);  
 **num** = **num**.substring(0, **num**.length() - 2);  
 } **else** {  
 **num** = String.*format*(Locale.***US***, **"%.10g%n"**, **sum**);  
 **sum** = Double.*parseDouble*(**num**);  
 **num** = String.*valueOf*(**sum**);  
 }  
 **t**.setText(**num**);  
 **num** = **"0"**;  
 **num1** = 0;  
 } **else if** (ch == **'±'**) {  
 **num** = **t**.getText();  
 **if** (**num**.charAt(0) != **'-'**)  
 **num** = **"-"** + **num**;  
 **else if** (**num**.charAt(0) == **'-'**)  
 **num** = **num**.substring(1, **num**.length());  
 **t**.setText(**num**);  
 } **else if** (ch == **'C'**) {  
 **if** (**num**.length() > 1)  
 **num** = **num**.substring(0, **num**.length() - 1);  
 **else  
 num** = **"0"**;  
 **t**.setText(**num**);  
 } **else if** (ch == **'A'**) {  
 **sign** = **'0'**;  
 **num** = **"0"**;  
 **sum** = 0;  
 **num1** = 0;  
 **num2** = 0;  
 **flag** = 0;  
 **t**.setText(**num**);  
 } **else if** (ch >= **'0'** && ch <= **'9'** || ch == **'.'**) {  
 **if** (ch == **'.'** && **num**.charAt(**num**.length() - 1) == **'.'**)  
 s = **""**;  
 **if** (ch == **'.'**)  
 **num** += s;  
 **else if** (**num**.equals(**"0"**))  
 **num** = s;  
 **else  
 num** = **num** + s;  
 **t**.setText(**num**);  
 } **else if** (ch == **'='**) {  
 **if** (**flag** == 1) {  
 **num2** = Double.*parseDouble*(**num**);  
 basicOperations();  
 } **else** {  
 **num** = **t**.getText();  
 **t**.setText(**num**);  
 **num** = **"0"**;  
 }  
 **flag** = 0;  
 } **else** {  
 **num** = **t**.getText();  
 **if** (**flag** == 0)  
 **num1** = Double.*parseDouble*(**num**);  
 **else** {  
 **if** (**num**.charAt(**num**.length() - 1) >= **'0'** && **num**.charAt(**num**.length() - 1) <= **'9'** && **num**.charAt(**num**.length() - 1) == **' '**)  
 **num2** = Double.*parseDouble*(**num**);  
 **num1** = basicOperations();  
 }  
 **num** = **"0"**;  
 **for** (**int** i = 0; i < 23; i++)  
 **num** += **" "**;  
 **num** += s;  
 **t**.setText(**num**);  
 **num** = **"0"**;  
 **sign** = ch;  
 **flag** = 1;  
 }  
 setSize(450, 450);  
 }  
}

### Файл StartFrame.java

**package** course.project;  
  
**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.awt.image.BufferedImage;  
**import** java.io.File;  
**import** java.io.IOException;  
*/\*\*  
 \** ***@author*** *Koroliov P.I.  
 \** ***@version*** *1.0  
 \*  
 \* Main Frame of application.  
 \*/***public class** StartFrame **extends** JFrame {  
 */\*\*  
 \* Set properties of Frame.  
 \*/* **public** StartFrame() **throws** HeadlessException {  
 setSize(800, 720);  
 setResizable(**false**);  
 setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 JFrame frame = **this**;  
  
 JLabel jlblUniversity = **new** JLabel(**"Белорусский Национальный Технический Университет"**);  
 JLabel jlblFaculty = **new** JLabel(**"Факультет информационнных технологий и робототехники"**);  
 JLabel jlblDepartment = **new** JLabel(**"Кафедра программного обеспечения информационных систем и технологий"**);  
 JLabel jlblCourseWork = **new** JLabel(**"Курсовая работа"**);  
 JLabel jlblDiscipline = **new** JLabel(**"по дисциплине \"Програмирование на языке Java\""**);  
 JLabel jlblScientificCalculator = **new** JLabel(**"Научный калькулятор"**);  
 JLabel jlblStudent = **new** JLabel(**"Выполнил: Студент группы 10702120"**);  
 JLabel jlblName = **new** JLabel(**"Королёв Павел Игоревич"**);  
 JLabel jlblProfessor = **new** JLabel(**"Преподаватель: к.ф.-м.н.,доц."**);  
 JLabel jlblProfessorsName = **new** JLabel(**"Сидорик Валерий Владимирович"**);  
 JLabel jlblYear = **new** JLabel(**"Минск 2022"**);  
  
 Font fontmediumsize = **new** Font(**"System"**, Font.***PLAIN***, 16);  
 Font fonthighsize = **new** Font(**"System"**, Font.***BOLD***, 20);  
  
 jlblUniversity.setFont(fontmediumsize);  
 jlblFaculty.setFont(fontmediumsize);  
 jlblDepartment.setFont(fontmediumsize);  
 jlblCourseWork.setFont(fonthighsize);  
 jlblDiscipline.setFont(fontmediumsize);  
 jlblScientificCalculator.setFont(fonthighsize);  
 jlblStudent.setFont(fontmediumsize);  
 jlblName.setFont(fontmediumsize);  
 jlblProfessor.setFont(fontmediumsize);  
 jlblProfessorsName.setFont(fontmediumsize);  
 jlblYear.setFont(fontmediumsize);  
  
 setLayout(**null**);  
 jlblUniversity.setBounds(204, 20, 400, 25);  
 jlblFaculty.setBounds(176, 60, 440, 25);  
 jlblDepartment.setBounds(102, 85, 570, 25);  
 jlblCourseWork.setBounds(295, 160, 190, 32);  
 jlblDiscipline.setBounds(225, 190, 365, 25);  
 jlblScientificCalculator.setBounds(270, 230, 400, 32);  
 jlblStudent.setBounds(460, 365, 280, 25);  
 jlblName.setBounds(460, 390, 280, 25);  
 jlblProfessor.setBounds(460, 440, 280, 25);  
 jlblProfessorsName.setBounds(460, 465, 280, 25);  
 jlblYear.setBounds(350, 570, 90, 25);  
  
 BufferedImage myPicture;  
 **try** {  
 myPicture = ImageIO.*read*(**new** File(**"src/main/java/resources/images/calc.jpg"**));  
 } **catch** (IOException e) {  
 **throw new** RuntimeException(e);  
 }  
 JLabel jlblPicLabel = **new** JLabel(**new** ImageIcon(myPicture));  
 jlblPicLabel.setBounds(80, 300, 260, 260);  
  
 JButton jbtButtonNext = **new** JButton(**"Далее"**);  
 JButton jbtButtonExit = **new** JButton(**"Выход"**);  
 */\*\*  
 \* Переопределяем метод интерфейса слушателя.  
 \*/* jbtButtonExit.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 System.*exit*(0);  
 }  
 });  
  
 jbtButtonNext.setFont(fontmediumsize);  
 jbtButtonExit.setFont(fontmediumsize);  
 jbtButtonNext.setBounds(70, 620, 300, 35);  
 jbtButtonExit.setBounds(420, 620, 300, 35);  
  
 addAll(jlblUniversity, jlblFaculty, jlblDepartment, jlblCourseWork, jlblDiscipline, jlblScientificCalculator, jlblStudent, jlblName, jlblProfessor, jlblProfessorsName, jlblYear);  
 addAll(jlblPicLabel, jbtButtonNext, jbtButtonExit);  
  
 setVisible(**true**);  
 */\*\*  
 \* Переопределяем метод интерфейса слушателя.  
 \* Create and run Main Frame of application.  
 \*/* jbtButtonNext.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ScientificCalculator();  
 setVisible(**false**);  
 }  
 });  
 */\*\*  
 \* Timer for closing.  
 \*/* **new** Timer(1000, **new** ActionListener() {  
 **int timeInSeconds** = 59;  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **timeInSeconds** -= 1;  
 jbtButtonExit.setText(String.*valueOf*(**timeInSeconds** + 1));  
  
 **if** (**timeInSeconds** < 0) {  
 frame.dispose();  
 }  
 }  
 }).start();  
 }  
 */\*\*  
 \* Method for add components.  
 \*/* **public void** addAll(Component... array) {  
 **for** (Component component : array) {  
 add(component);  
 }  
 }  
}

### Файл AboutAuthorFrame.java

**package** course.project;  
  
**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.awt.image.BufferedImage;  
**import** java.io.File;  
**import** java.io.IOException;  
*/\*\*  
 \** ***@author*** *Koroliov P.I.  
 \** ***@version*** *1.0  
 \*  
 \* Frame about Author.  
 \*/***public class** AboutAuthorFrame **extends** JFrame {  
 */\*\*  
 \* Set properties of Frame.  
 \*/* **public** AboutAuthorFrame() {  
 setSize(400, 630);  
 setResizable(**false**);  
  
 JLabel label1 = **new** JLabel(**"Автор"**);  
 JLabel label2 = **new** JLabel(**"Студент группы 10702120"**);  
 JLabel label3 = **new** JLabel(**"Королёв Павел Игоревич"**);  
 JLabel label4 = **new** JLabel(**"Почта: hemetnesingwarynpc@gmail.com"**);  
  
 Font font1 = **new** Font(**"System"**, Font.***PLAIN***, 16);  
 Font font2 = **new** Font(**"System"**, Font.***BOLD***, 20);  
 label1.setFont(font2);  
 label2.setFont(font1);  
 label3.setFont(font1);  
 label4.setFont(font1);  
  
 setLayout(**null**);  
 label1.setBounds(170, 354, 60, 30);  
 label2.setBounds(35, 400, 250, 25);  
 label3.setBounds(35, 430, 250, 25);  
 label4.setBounds(35,460,350,25);  
  
 BufferedImage myPicture = **null**;  
 **try** {  
 myPicture = ImageIO.*read*(**new** File(**"src/main/java/resources/images/iam.jpg"**));  
 } **catch** (IOException e) {  
 **throw new** RuntimeException(e);  
 }  
 JLabel picLabel = **new** JLabel(**new** ImageIcon(myPicture));  
 picLabel.setBounds(45, 33, 300, 300);  
  
 JButton buttonNext = **new** JButton(**"Назад"**);  
 */\*\*  
 \* Переопределяем метод интерфейса слушателя.  
 \* Create and run Main Frame of application.  
 \*/* buttonNext.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 setVisible(**false**);  
 }  
 });  
 buttonNext.setFont(font1);  
 buttonNext.setBounds(35, 520, 310, 35);  
  
  
 addAll(label1, label2, label3, label4, picLabel, buttonNext);  
  
 setVisible(**true**);  
 }  
 */\*\*  
 \* Method for add components.  
 \*/* **public void** addAll(Component... array) {  
 **for** (Component component : array) {  
 add(component);  
 }  
 }  
}

### Файл AboutProgramFrame.java

**package** course.project;  
  
**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.awt.image.BufferedImage;  
**import** java.io.File;  
**import** java.io.IOException;  
*/\*\*  
 \** ***@author*** *Koroliov P.I.  
 \** ***@version*** *1.0  
 \*  
 \* Frame about Program.  
 \*/***public class** AboutProgramFrame **extends** JFrame {  
 */\*\*  
 \* Set properties of Frame.  
 \*/* **public** AboutProgramFrame() {  
 setSize(1150, 390);  
 setResizable(**false**);  
  
 JLabel label1 = **new** JLabel(**"Научный калькулятор"**);  
 JLabel label2 = **new** JLabel(**"Программа реализует обычный и научный калькуляторы"**);  
 JLabel label3 = **new** JLabel(**"В программе реализованы:"**);  
 JLabel label4 = **new** JLabel(**"Базовые операции: сложение, вычитание, умножение, деление и т.п."**);  
 JLabel label5 = **new** JLabel(**"Продвинутые операции: возведение в n-ую степень, корень из числа n-ой степени и т.п."**);  
 JLabel label6 = **new** JLabel(**"Тригонометрические и обратные тригонометрические функции, функции для работы с памятью"**);  
 JLabel label7 = **new** JLabel(**"Реализована в запись файл арифметических операций"**);  
 JLabel label8 = **new** JLabel(**"Версия 1.0.0 2022"**);  
  
  
 Font font1 = **new** Font(**"System"**, Font.***PLAIN***, 16);  
 Font font2 = **new** Font(**"System"**, Font.***BOLD***, 20);  
 Font font3 = **new** Font(**"System"**, Font.***BOLD***, 16);  
 label1.setFont(font2);  
 label2.setFont(font1);  
 label3.setFont(font3);  
 label4.setFont(font1);  
 label5.setFont(font1);  
 label6.setFont(font1);  
 label7.setFont(font1);  
 label8.setFont(font3);  
  
  
 setLayout(**null**);  
 label1.setBounds(225, 25, 660, 30);  
 label2.setBounds(410, 75, 635, 25);  
 label3.setBounds(410, 135, 635, 25);  
 label4.setBounds(410, 165, 635, 25);  
 label5.setBounds(410, 190, 735, 25);  
 label6.setBounds(410, 215, 750, 25);  
 label7.setBounds(410, 240, 635, 25);  
 label8.setBounds(410, 290, 250, 25);  
  
 BufferedImage myPicture = **null**;  
 **try** {  
 myPicture = ImageIO.*read*(**new** File(**"src/main/java/resources/images/calc.jpg"**));  
 } **catch** (IOException e) {  
 **throw new** RuntimeException(e);  
 }  
 JLabel picLabel = **new** JLabel(**new** ImageIcon(myPicture));  
 picLabel.setBounds(25, 85, 360, 220);  
  
 JButton buttonNext = **new** JButton(**"Назад"**);  
 */\*\*  
 \* Переопределяем метод интерфейса слушателя.  
 \*/* buttonNext.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 setVisible(**false**);  
 }  
 });  
  
 buttonNext.setFont(font1);  
 buttonNext.setBounds(610, 285, 130, 35);  
  
 addAll(label1, label2, label3, label4, label5, label6, label7, label8, picLabel, buttonNext);  
  
 setVisible(**true**);  
 }  
 */\*\*  
 \* Method for add components.  
 \*/* **public void** addAll(Component... array) {  
 **for** (Component component : array) {  
 add(component);  
 }  
 }  
}